Diss. ETH No. 23464

Wearable barcode scanning

Advancements in visual code localization, motion blur compensation, and gesture control

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. Sc. ETH Zurich)

presented by

GÁBOR SÖRÖS

M.Sc. in Electrical Engineering
Budapest University of Technology and Economics
born on 02.06.1986, citizen of Hungary

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Otmar Hilliges, co-examiner
Prof. Dr. Dieter Schmalstieg, co-examiner

2016

to my parents
Rózsa and Géza

Abstract

Visual codes like barcodes and quick response (QR) codes are the most prevalent linking elements between physical objects and digital information. They are found in numerous consumer applications such as shopping, electronic payments, ticketing, or marketing campaigns, but they are also found in logistics and enterprise asset tracking and provide employees access to detailed service records. Reading visual codes is also often the first step for pairing and interaction with physical appliances in various research projects in the area of pervasive computing. While these codes are found almost everywhere, the reading of codes usually requires expensive scanner devices which hinders the original goal of easy access to information about every physical object.

Technological advancements in wearable computing and mobile computer vision may radically expand the adoption of visual codes because smartphones, smartwatches, smartglasses, and other wearables enable instant barcode scanning on the go. The wide accessibility, the computing performance, the intuitive user interface, and the relatively low price make personal wearable computers strong competitors for traditional scanners while they also enable new use cases for visual codes. As these devices are primarily designed for other purposes, there are also a few shortcomings.

In this dissertation, we describe methods to overcome these shortcomings and to add advanced features that can make wearable barcode scanning an attractive alternative to traditional barcode scanning even outside the consumer domain. We present fast and robust solutions to the following problems on computationally restricted unmodified wearable devices:

(i) Fast and robust localization of visual codes: Current smartphone-based barcode scanning solutions require the user to hold and align the tagged object close to the camera. This is especially problematic with smartglasses and leads to lower user acceptance. On the other hand, today's wearable cameras have a high enough resolution to scan visual codes that are further away, once they are segmented in a preprocessing step. We propose a fast algorithm for joint 1D and 2D visual code localization in large digital images. The proposed method outperforms other solutions in terms of accuracy while it is invariant to scale, orientation, code symbology, and is more robust to blur than previous approaches.

We further optimize for speed by exploiting the parallel processing capabilities of mobile graphics hardware for image processing. Fast segmentation allows for scanning multiple codes at the same time and thus helps in application scenarios where interaction with multiple objects is necessary.

(ii) Fast and robust compensation of motion blur: When the camera or the code undergoes slight motion during scanning, in contrast to the performance of laser scanners, wearable cameras often suffer from motion blur that renders the codes unreadable. We build upon existing work in photograph deblurring and develop a fast algorithm for scanning motion-blurred QR codes on mobile devices. We exploit the fact that QR codes do not need to be visually pleasing for decoding and propose a fast restoration-recognition loop that exploits the special structure of QR codes. In our optimization scheme, we interweave blind blur estimation from the edges of the code and image restoration regularized with typical properties of QR code images. Our proposed restoration algorithm is on par with the state of the art in quality while it is about a magnitude faster. We also propose to combine blur estimation from image edges with blur estimation from built-in inertial sensors to make the restoration even faster. Fast blur compensation means that no precise code alignment is required but the user can simply swipe the camera in front of the code.

(iii) Fast and robust recognition of in-air hand gestures: Wearable devices have limited input capabilities; the way of interaction is usually limited to a few buttons or slim touchpads. We add natural hand gesture input to our wearable computers, but indirectly also to other smart appliances in our smart environment that can be automatically recognized through tiny visual codes, and that can "outsource" their own user interface to our wearable devices. We present a machine learning technique to recognize hand gestures with only a single monocular camera as can be found on off-the-shelf mobile devices. The algorithm robustly recognizes a wide range of in-air gestures and runs in real time on unmodified wearable devices. We further show that, with little modification, our method can not only classify the gesture but can also regress the distance of the hand from the camera. 3D in-air gesture control allows hands-free scanning with smartglasses which brings many advantages in enterprise scenarios. Furthermore, through user interface outsourcing, it also enables expressive vision-based gesture control even to those appliances that do not possess a camera by their own.

Along with the dissertation, we created several showcase scenarios and demonstrators of our contributions. All proposed algorithms have been designed and implemented to be compatible with various platforms and device families (e.g., PCs, tablets, smartphones, smartwatches, smartglasses), with their resource constraints in mind. The solutions presented in this dissertation are pushing forward the state of the art in terms of accuracy, speed, and robustness, and thus help to make wearable barcode scanning a promising alternative to traditional barcode scanning.

Kurzfassung

Visuelle Codes, wie beispielsweise Strichcodes oder QR-Codes, werden im Rahmen zahlreicher mobiler Anwendungsszenarien genutzt, sie stellen optisch erkennbare Marker als Verbindungselemente zwischen Objekten der physischen Welt und digitaler Information dar. Vom intelligenten Einkaufsassistenten über elektronische Zahlungen, Tickets, Marketingkampagnen bis hin zu Anwendungen in Logistik und Güteverkehr unterstützen visuelle Codes eine große Zahl an Einsatzmöglichkeiten. Auch bei vielen aktuellen Forschungsvorhaben im Bereich des *pervasive computing* ist das Erfassen visueller Codes der erste Schritt zur Interaktion mit physischen Objekten und Geräten in der näheren Umgebung. Doch obwohl visuelle Codes mittlerweile auf praktisch allen Produkten und vielen anderen Dingen zu finden sind, benötigt das optische Erkennen der Codes und das Auslesen der darin hinterlegten Daten ("scannen") bisher noch dedizierte und teure, meist laserbasierte Lesegeräte. Der intendierte Zweck eines unmittelbaren und ubiquitären Zugangs zur damit verknüpften Information wird dadurch infrage gestellt.

Das zunehmende Interesse an visuellen Codes wird in der jüngeren Zeit auch durch rasante technische Entwicklungen im Bereich des *wearable computing* sowie des mobilen maschinellen Sehens induziert. Smartphones, Smartwatches, Smartglasses und andere tragbare Geräte bringen mit ihren eingebauten kleinen Kameras sowie ihren leistungsfähigen CPUs alle Voraussetzungen für eine effiziente Codeerkennung mit. Aufgrund ihrer breiten Verfügbarkeit, hohe Rechenleistung, intuitiven Mensch-Maschine-Schnittstelle sowie ihres günstigen Preises konkurrieren sie mit den traditionellen Lesegeräten für visuelle Codes und ermöglichen gleichzeitig neue Anwendungsszenarien. Da diese neueren mobilen Geräte aber vornehmlich für andere Verwendungszwecke entwickelt wurden, weisen sie in Hinblick auf die Codeerkennung als spezielle Eigenschaft allerdings einige markante Unzulänglichkeiten auf.

In der vorliegenden Dissertation beschreiben wir Methoden, die diese Defizite nicht nur überwinden, sondern darüber hinaus auch einige neue und interessante Funktionen ermöglichen. Damit wird der traditionellen Strichcodeerkennung eine attraktive Alternative gegenübergestellt, welche auch in vielen Bereichen außerhalb des Konsumentensektors, vor allem im Industrieeinsatz, zusätzlichen Nutzen stiften sollte. Im Wesentlichen stellen wir schnelle und robuste Lösungen für die folgenden Problembereiche bei der Verwendung leistungsbeschränkter (und generell ressourcenlimitierter) *wearable devices* vor:

(i) Schnelle ung robuste Lokalisierung von visuellen Codes: Gängige Smartphone-basierte Scannerlösungen verlangen, dass der Nutzer das Produkt dicht vor die Kamera hält und sorgfältig positioniert. Dies ist insbesondere bei Smartglasses problematisch und führt zu einer geringeren Akzeptanz. Andererseits verfügen die heutigen digitalen Kameras über eine genügend hohe Bildauflösung, um visuelle Codes auch aus großer Entfernung zu lesen, sofern die Codes durch einen vorverarbeitenden Schritt segmentiert werden. Um dieses Potential zu nutzen, entwickeln wir einen schnellen Algorithmus für die gleichzeitige visuelle Lokalisierung mehrerer ein- und zweidimensionalen Codes in großen digitalen Bildern. Die realisierte Methode übertrifft andere Lösungen an Genauigkeit, während sie bezüglich Skalierung, Orientierung und Symbologie der Codes invariant ist. Zusätzlich ist sie robuster gegen Unschärfe (*blur*) als bisherige Ansätze. Unter Ausnutzung der Parallelisierungsfähigkeiten moderner mobiler Graphikhardware zeigen wir, dass die Geschwindigkeit weiter optimiert werden kann. Die schnelle Codesegmentierung erlaubt das gleichzeitige Lesen mehrerer Codes und unterstützt Situationen, bei denen eine Interaktion mit mehreren Objekten erfolgen kann.

(ii) Schnelle ung robuste Kompensation von Bewegungsunschärfe: Wenn Kamera oder Codemarker während des Scannens bewegt werden, verursacht dies im Unterschied zu den traditionellen laserbasierten Scannergeräten oft ein durch Bewegungsunschärfe verzerrtes Bild, das die codierte Information unleserlich macht. Um diesem Problem zu begegnen, bauen wir auf existierenden Ansätzen aus der Bildbearbeitung im Bereich *deblurring* auf und entwickeln einen schnellen Algorithmus, um unscharfe QR-Codes direkt auf mobilen Geräten lesen zu können. Wir profitieren dabei von der Tatsache, dass QR-Codes für ein erfolgreiches Decodieren nicht visuell ansprechend sein müssen und schlagen eine schnelle Restaurierungs- und Erkennungsschleife vor, welche die spezielle Struktur von QR-Codes ausnutzt. In unserem Optimierungsschema verbinden wir eine mittels dominanter Kanten erzielte Unschärfeschätzung des Bildes mit der Bildrestaurierung unter Ausnutzung typischer Eigenschaften von QR-Codes. Die von uns entwickelte Restaurierungsmethode entspricht qualitativ dem aktuellen Stand der Technik, ist aber um eine Größenordnung schneller. Darüber hinaus schlagen wir vor, die kantenbasierte Unschärfeberechnung mit einer Unschärfeschätzung mittels der in mobilen Geräten oft verbauten Trägheitssensoren zu kombinieren, um die Restaurierung weiter zu beschleunigen. Die von uns entwickelte Unschärfekompensation bedeutet, dass keine präzise Positionierung der Codemarkers mehr erforderlich ist und der Nutzer die Kamera einfach über den Code ziehen kann.

(iii) Schnelle ung robuste Erkennung von Handgesten: Kleine Geräte haben oft limitierte Ein- und Ausgabemöglichkeiten, die Interaktion ist dabei auf wenige Tasten oder schmale Touchpads beschränkt. Aus diesem Grund haben wir eine Handgesteneingabe für mobile Geräte entwickelt, welche indirekt sogar von Objekten ohne Kamera und explizi-

te Bedienschnittstellen genutzt werden kann. Automatisch erkennbare kleinste visuelle Codes erlauben den beteiligten Objekten, ihre eigenen “virtuellen” Bedienelemente an andere mit Nutzerschnittstelle ausgestatteten mobilen Geräte auszulagern. Wir stellen ein maschinelles Lernverfahren vor, welches Handgesten mit einer einfachen monokularen Kamera, wie sie in den meisten mobilen Geräten verbaut ist, erkennen kann. Der Algorithmus läuft in Echtzeit auf physisch unmodifizierten Geräten und kann eine breite Auswahl von Handgesten zuverlässig erkennen. Des Weiteren zeigen wir, dass unsere Methode mit kleinen Änderungen nicht nur die Gesten klassifizieren, sondern auch die Distanz der Hand zur Kamera schätzen kann. Die Gestenerkennung erlaubt somit das Lesen von Strichcodes ohne Gebrauch der Hände; ein Vorteil, der besonders im kommerziellen Bereich zur Geltung kommt. Durch Bedienungsauslagerung wird ferner auch Geräten, welche selbst keine Kamera aufweisen, eine ausdrucksvolle Gestensteuerung ermöglicht.

Im Rahmen der Dissertation wurden mehrere Demonstratoren für diverse Anwendungsszenarien erstellt. Von allen beschriebenen Algorithmen liegen Implementierungen für unterschiedliche Plattformen und Geräte (z.B. PCs, Tablets, Smartphones, Smartwatches, Smartglasses) vor, welche die jeweiligen technischen Einschränkungen beachten. Die in der vorliegenden Dissertation präsentierten Verfahren zur mobilen Codeerkennung gehen hinsichtlich Genauigkeit, Geschwindigkeit und Robustheit deutlich über den bisherigen Stand der Technik und Wissenschaft hinaus. Damit stellt die kamerabasierte Codeerkennung mit alltäglichen mobilen Geräten nunmehr eine vielversprechende Alternative zu den traditionell verwendeten spezielleren Lesesystemen dar.

Kivonat

Valós tárgyak és digitális információ között mai világunkban a legelterjedtebb összekötő elemek a vizuális kódok, mint például a vonalkódok és a quick response (QR) kódok. A vizuális kódok számos alkalmazásával találkozhatunk a bevásárlástól az elektronikus fizetéseken és jegyvásárláson át a reklámkampányokig, de mindenütt megtalálhatóak a logisztika és az áruforgalom területén is. A *pervasive computing* területén sok kutatási projektben a fizikai eszközök konfigurálásában és az azokkal való interakcióban is vizuális kódok leolvasása az első lépés. Noha a vizuális kódok majdnem mindenhol jelen vannak, leolvasásuk általában drága olvasóeszközöket *(scannereket)* igényel, és ez meghiúsítja a kódok használatának eredeti célját, hogy bármely tárgyról egyszerűen és gyorsan információhoz juthassunk.

A vizuális kódok további elterjedését jelentősen elősegíthetik a viselhető számítástechnika *(wearable computing)* és a mobil gépi látás *(mobile computer vision)* technikai vívmányai. Az okostelefonok, okosórák, okosszemüvegek és más viselhető eszközök lehetővé teszik, hogy bárhol és bármikor leolvassunk vizuális kódokat. A széleskörű elérhetőségnek, a fejlett számítási teljesítménynek, az intuitív felhasználói felületnek és a viszonylag alacsony árnak köszönhetően a viselhető számítástechnikai eszközök komoly konkurenciát jelenthetnek a hagyományos leolvasóknak, egyúttal utat nyithatnak a kódok számos új alkalmazásának is. Mivel azonban ezeket az eszközöket eredetileg más felhasználási célokra tervezték, kódolvasóként használva őket néhány területen hiányosságokat is mutatnak.

Jelen disszertációban olyan módszereket mutatunk be, amelyek ezeket a hiányosságokat áthidalják, és ezen felül egyéb új funkciókat is lehetővé tesznek. Így a viselhető vonalkódolvasás *(wearable barcode scanning)* új alternatívát kínál a hagyományos leolvasókkal szemben mind a fogyasztók körében, mind pedig ipari alkalmazásokban. A disszertációban gyors és hatékony módszereket mutatunk be a következő problémákra korlátozott számítási teljesítményű, módosítatlan viselhető eszközökön:

(i) Gyors és hatékony kódlokalizáció: A mai okostelefon-alapú leolvasó megoldások megkövetelik, hogy a felhasználó a kódot stabilan, közel a kamera előtt tartsa. Ez gondot okozhat okosszemüvegek esetében, mert a felhasználók körében nemtetszést válthat ki. Másrészről a viselhető kamerák elég magas képfelbontással rendelkeznek ahhoz, hogy a

kamerától távolabb található vizuális kódokat is képesek legyenek leolvasni, ha azokat egy előzetes feldolgozó lépésben szegmántáljuk. Egy olyan gyors és hatékony algoritmust mutatunk be, amely nagy digitális képeken tud egyszerre egy-, és kétdimenziós vizuális kódokat lokalizálni. Az új módszer felülmúlja a korábbiakat pontosságában, miközben a kód méretét, orientációját, és a kód típusát tekintve nem támaszkodik előfeltételekre, valamint kevésbé érzékeny az elmosódottsággal szemben. A modern, mobil grafikus hardverek párhuzamos számítási képességeit kihasználva az algoritmus sebességét tovább lehet növelni. A gyors kódszegmentáció lehetővé teszi egyszerre több kód leolvasását is, amely olyan esetekben különösen hasznos, amikor egyszerre több tárggyal való interakció szükséges.

(ii) Gyors és hatékony elmosódás-helyreállítás: Ha leolvasás közben a kamera vagy a kód kis mértékben elmozdul, a lézerolvasókkal ellentétben a viselhető kamerák gyakran olvashatatlan, elmosódott képet rögzítenek. Ennek a problémának a kiküszöbölésére egy olyan gyors és hatékony algoritmust fejlesztettünk ki, amely képes rendkívül elmosódott QR kódokat is leolvasni. Az algoritmus a fényképretusálásban is alkalmazott módszereken alapul, amelyek minden lépését a QR kódok egyedi tulajdonságaihoz igazítottuk. Azt a tényt használjuk ki, hogy a QR kódoknak nem szükséges vizuálisan esztétikusnak lenniük ahhoz, hogy dekódolhatóak legyenek. A módszerünk alapja egy gyors javító-dekódoló ciklus, amely kihasználja a QR kódok speciális struktúráját. Az optimalizációs ciklusban felváltva becsüljük a kép elmosódottságát a kód éleiből és javítjuk a kép minőségét QR-tipikus regularizációt alkalmazva. Ez a rekonstrukciós módszer minőségben a jelenlegi technikákkal azonos szintet képvisel, ám egy nagyságrenddel gyorsabb azoknál. Az elmosódottságot nemcsak az élekből, hanem a viselhető eszközökbe beépített mozgásszenzorokból is becsülhetjük, hogy a helyreállítási folyamatot még gyorsabbá tegyük. A gyors elmosódás-kompenzáció lehetősége azt jelenti, hogy a leolvasás során nem szükséges többé a kód pontos pozícionálása, a felhasználó egyszerűen elhúzhatja a kamerát a kód előtt.

(iii) Gyors és hatékony gesztusfelismerés: A viselhető eszközök korlátozott be- és kiviteli lehetőségekkel rendelkeznek, az interakció többnyire pár gombra vagy vékony érintőképernyőkre korlátozódik. Ezért egy, a természetes kommunikációhoz sokkal közelebb álló, gesztikuláción alapuló beviteli módszert fejlesztettünk ki, amely indirekt módon akár más, az “okos” környezetünkben található eszközökhöz is használható. Az automatikusan felismerhető apró kódok lehetővé teszik a kommunikációban résztvevő eszközök számára, hogy saját kiszolgáló elemeiket a felhasználó viselhető eszközeire “ruházzák át”. Egy olyan tanuló algoritmust mutatunk be, amely a gesztusokat egy egyszerű, monokuláris kamera segítségével is felismeri, amellyel a legtöbb viselhető eszköz rendelkezik. Az algoritmus a kézi gesztusok széleskörű felismerésére képes, és valós időben fut módosítatlan viselhető eszközökön. Azt is bemutatjuk, hogy módszerünk kis változtatással nemcsak a

gesztusokat tudja felismerni, hanem a kéz és a kamera távolságát is meg tudja becsülni, és ezáltal háromdimenziós gesztusbevitelt tesz lehetővé. Módszerünk lehetővé teszi a vonalkódok leolvasását gombok és érintőképernyő használata nélkül, amely főként ipari alkalmazásokban jelenthet nagy előnyt, valamint a kezelőfelület viselhető eszközökre való átruházásával lehetővé teszi a gesztikuláción alapuló vezérlést olyan eszközökön is, amelyek maguk nem rendelkeznek kamerával.

A disszertációhoz szemléltető példákat és bemutatókat is készítettünk. Az algoritmusokat úgy terveztük és alkottuk meg, hogy különböző platformokon és eszközökön is működjenek, például személyi számítógépeken, okostelefonokon, okosórákon és okosszemüvegeken – azok minden technikai korlátjára gondolva. Az eredmények, amelyeket itt bemutatunk mind pontosság, gyorsaság és megbízhatóság tekintetében jelentős előrelépést jelentenek. Segítségükkel a viselhető vonalkódolvasás ígéretes alternatívát kínálhat a hagyományos vonalkódolvasókkal szemben.

Acknowledgements

The work in this thesis would not have been possible without the help and support of many people. First and foremost, I would like to express my gratitude to Prof. Friedemann Mattern and Prof. Otmar Hilliges for their guidance and motivation throughout the last five years. I greatly appreciate the freedom to choose my research topic and to follow my passion in mobile computer vision. I also thank Prof. Dieter Schmalstieg for his valuable feedback as co-examiner.

I thank all my colleagues and friends at the Institute for Pervasive Computing for the joyful moments inside and outside the office: my first officemates, Wilhelm Kleiminger and Simon Mayer for the stimulating work atmosphere, game nights and grill events. Simon for his infinite positivism and Willi for his endless skepticism. Matthias Kovatsch for the bananas. I appreciate that Christian Beckel will let me win in Unreal next time. All participants of the greatest bachelor party and my first, last, and worst ski experience ever. Furthermore, I thank Jie Song and Fabrizio Pece for all the nights and weekends we spent together in the office, and our explorations in Chinese and Italian cuisine. I thank Mihai Bâce for his moral support. Thanks for the enjoyable moments for the new generation: Subho Basu, Vincent Becker, Hông-Ân Cao, Marian George, Anwar Hithnawi, Leyna Sadamori, Hossein Shafagh, Benjamin Hepp, Christoph Gebhardt, Tobias Nägeli, and for the old generation: Robert Adelmann, Alexander Bernauer, Dominique Guinard, Iulia Ion, Benedikt Ostermaier, Vlad Trifa, Markus Weiss. I thank Silvia Santini and Marc Langheinrich for the Ubicomp experience. Furthermore, I am very grateful to Barbara von Allmen Wilson for all her support.

I thank my former colleagues at Scandit, Christian Flörkemeier, Christof Roduner, Moritz Hartmeier, Viviana Petrescu, and Simon Wenner for the inspiring technical discussions, the hardware support, and for introducing me the world of start-ups.

Thanks to Christian Flörkemeier and Otmar Hilliges for teaching me how to write papers and how to give presentations. Special thanks go to Wilhelm Kleiminger for proofreading my thesis.

During the PhD, I have had the rewarding experience of supervising numerous bright students: Ekansh Anand, Claude Barthels, Carlo Beltrame, David Chettrit, Marc Fischer, Michael Franz, Claudio Gargiulo, Julia Giger, Mauro Guerini, Yassin N. Hassan, Andreas Hess, Moritz Hoffmann, Luc Humair, Pascal Josephy, Thomas Knell, Sandro Lombardi, Severin Münger, Markus Schalch, Bram Scheidegger, Stephan Semmler, Sander Staal, Andy Zimmermann. I thank for your hard work and enthusiasm.

Furthermore, I would like to thank ETH Zurich and all the people behind ETH services for contributing to one of the best scientific work environments in the world.

Above all, I would like to thank my wife Zsófi for her endless support and for constantly inspiring me to aim for the best. I thank the incentive of my daughter Júlia who was born and grew with the pages of this text. I would like to dedicate the thesis to my parents Rózsa and Géza who ignited my passion for mathematics and technology.

Köszönöm.

Contents

Chapter 1

Introduction

1.1 Visual codes and applications

Codes, in particular visual codes, exist since the early history of mankind with the purpose of information transfer to others in space and time. The information contained in a code can be anything the communicating parties agree on, let it be map symbols, numerical systems, languages, musical notes, or others. Of particular importance in our economy are machine-readable visual codes like barcodes and matrix codes for object identification that allow information access on physical items.

The first barcode was patented by Norman Woodland and Bernard Silver in 1949 (US patent no. 2612994 [249]) with the goal of making the inventories in supermarkets easier. The idea of encoding information in bars of different size came from Morse code dots and dashes that the authors extended to long vertical lines. The linear reading method was inspired by the 1920's movie industry: the authors adapted the way of sound encoding at the edge of the film via patterns of varying transparency. The barcode was shifted in front of a strong light bulb and could be read by a photo-multiplier tube behind the code. Because the reading required almost complete darkness, the first barcode readers were not practical and the idea was suspended for a while.

The first large-scale application of barcodes was installed in the railway industry for identifying individual freight cars on the railroad, developed by David J. Collins and standardized US-wide in 1967. The KarTrak encoding scheme used orange and blue bars on reflective material, in principle similar to a barcode. The original Woodland-patent also proposed an encoding scheme with concentric circles that would be readable from any orientation, but the linear version was more practical due to its advantages in contemporary

printing technology [273]. In 1973, the Universal Product Code (UPC) standard got accepted in the US, the European Article Number (EAN) was introduced in Europe three years later, and since then barcodes boosted a whole industry of automatic identification and data capture (short AIDC).

Today, visual codes represent the most widespread object identification technology and have a wide range of applications in both consumer, enterprise, and research domains. There exists a wide variety of mappings between information content and visual appearance, called code symbologies, but all symbologies have a well-defined special structure usually consisting of black and white bars, circles, triangles, or squares that are easy to detect for a machine. A few popular symbologies are collected in Figures 1.1 and 1.2. Visual codes are usually printed on flat surfaces or displayed on flat screens, and are read by a barcode scanner device that converts the black and white optical signal to a digital signal. There are several different technological principles for this conversion, the most versatile one is using a handheld computer with a camera. These scanners, however, are very expensive and in the recent years they are being replaced by off-the-shelf personal mobile computers like smartphones in many scenarios. Despite the many advantages of smartphones in their technological parameters, their availability, and their price, the performance of smartphone-based scanners is still lagging behind professional scanners.

(a) EAN13/UPC-A: 012345678910

(b) Code-128: Gabor Soros ETH Zurich

(c) MSI Plessey: 0123456789

(d) Postnet: 008092

Figure 1.1: Popular 1D barcodes generated by [272]. Code contents are indicated in the captions.

In this thesis, we present various methods to extend the capabilities of smartphone-based barcode scanners beyond traditional barcode scanners and generalize our methods to other personal wearable computers like smartwatches and smartglasses, and combinations of them. We believe that inexpensive wearable barcode scanning is an important step towards the vision of ubiquitous information access on and interaction with physical objects. In the rest of this section, we highlight a number of applications to hint at the ubiquity of visual codes in our economy and to motivate the topic of the thesis.

(a) QR code:
gabor.soros@inf.ethz.ch

(b) Data Matrix:
www.vs.inf.ethz.ch

(c) Data Matrix:
Gabor Soros vCard

(d) Aztec code:
Universitätstrasse 6, CNB H

(e) Code One:
gabor.soros@inf.ethz.ch

(f) Maxicode:
Gabor Soros

(g) PDF417:
Gabor Soros, Institute for Pervasive Computing, ETH Zurich

Figure 1.2: Popular 2D barcodes (matrix codes) generated by [272]. Code contents are indicated in the captions.

1.1.1 Consumer applications

In the consumer domain, visual codes[1] can link unlimited information to a specific product. The widespread GTINs[2] are printed on the packaging of goods in form of 1D barcodes and allow product identification at any point of the supply chain. The GTIN is a combination of a producer identifier and a product identifier, and note that a barcode is only one possible representation of this GTIN identifier. While a standard barcode encodes only the GTIN, other 2D codes can encode more information for item-level identification. This allows the consumer to track back a specific item in the production chain. For instance, we can look up which specific farm the product originates from. Also, by simply encoding a Web address onto the product packaging, we can look up various kinds of information about the product. This is an elegant way for producers to extend the packaging space with detailed nutrition facts, descriptions in multiple languages, advertisements about complementary products, but also user manuals and quick configuration guidelines. Crowd-sourced online databases [102, 175] of product ratings, expert reviews, allergy checks, and price comparisons help customers in making right purchasing decisions[277]. Merchants can scan barcodes to check inventories or to redeem coupons.

1.1.2 Enterprise and governmental applications

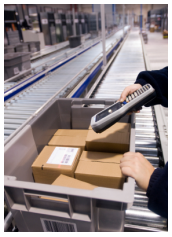
In enterprise scenarios, logistics employees are scanning visual codes for delivering parcels, tracking assets, refilling inventories, checking tickets, etc. When employees

[1] In the document we use the words visual code, visual tag, and barcode interchangeably

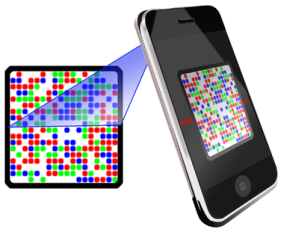
[2] GTIN: Global Trade Item Number, the worldwide identification standard also including Universal Product Code (UPC), European Article Number (EAN), Japanese Article Number (JAN). They are maintained by the non-profit international organization GS1 [293].

(a) Grocery shopping [324]

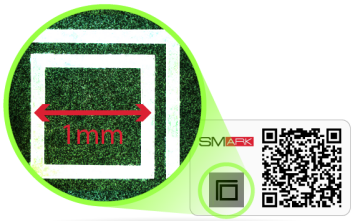
(b) Ticket control [325]



(c) Product picking [274]



(d) Authentication [279]



(e) Originality check [334]

Figure 1.3: Visual codes in consumer and enterprise applications

perform order picking in big logistics warehouses, they scan each item before putting it into the basket. Following the vision of an industrial Internet of Things with highly customizable production lanes, intelligence is embedded in smart factory machines, smart tools, but also partially assembled smart products. A smart machine can recognize a workpiece and can automatically decide what to do with it. Very small visual codes can help realizing this vision. Domain-specific visual codes are applied for instance in the medical industry (Aztec code) or in the car industry (Vehicle Identification Number). QR codes are also widely used on e-documents, tax forms, bills, etc. as links to further information, but also codes on business cards are becoming popular. Some financial institutes already introduced visual transaction authentication methods like the CrontoSign visual cryptogram (Figure 1.3d [279]). For a more secure economy, SMark (Figure 1.3e [334]) and ScanTrust [332] develop irreproducible visual codes for originality check of products. In contrast with the wide-spread consumer use cases, employees and factory workers usually need to carry dedicated enterprise handhelds to scan their own visual tags.

1.1.3 Visual codes in tourism and marketing

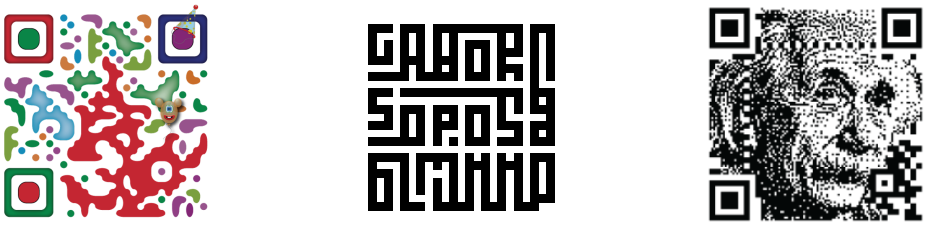
Visual codes are increasingly popular marketing elements in the tourism industry. QR codes on landmarks link to further audio-visual content about historical events in Gibraltar [287] or in 'the first Wikipedia town' Monmouth in Wales (see Figure 1.4a). A code next to a painting in a museum provides information about the painter (Figure 1.4b), scanning the code on a memorial can link to the biography of the famous person. In 2010,

an interactive board game (The World Park) has been installed in the Central Park of New York City (Figure 1.4c) with interactive QR codes (named Parkodes) that reveal the park's iconic history. By scanning a code, visitors can access famous location-related movie scenes, photographs, and historical, scientific, and cultural facts. Visual codes are not only ideal for participants of self-guided walking tours but also for the organizers because they learn how often people use the services and can analyze walking patterns [22]. Hotels can improve customer experience and accelerate customer feedback through QR codes.

(a) A QRpedia landmark [308] (b) Painter biography [310] (c) Interactive park [339]

Figure 1.4: Visual codes in tourism

Since black and white squares are not the most compelling marketing assets, a number of more artistic visual codes have been developed [44] (see Figure 1.5). For instance, Human Readable Quick Response (HRQR) codes [296] consist of specially designed modules that correspond to human-readable characters. Half-tone QR codes [40] combine halftoned images with conventional QR codes. The authors first learn which patterns (QR code modules) are still readable under small transformations and blur, then build a large dictionary of robust patterns, and decompose the image into the dictionary words. The final image is a decodable QR code while it still appears similar to the original image.



(a) QR Artist code [323] (b) HRQR code [296] (c) Halftone QR code [40]

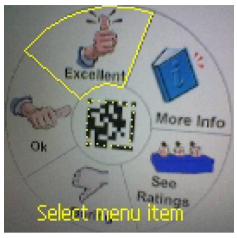
Figure 1.5: Artistic visual codes

1.1.4 Visual codes for interaction and augmented reality

Visual codes are also used in a number of research projects on interaction. In their work, Rohs [182], Rukzio [185], and Roduner [181] investigate various techniques for interaction

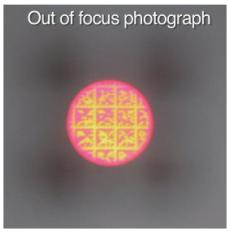
between smart physical objects and mobile phones. The visual codes not only serve as object identifiers and hyperlinks between the real and the virtual worlds, but can also help to determine the position and orientation of an interaction device (smartphone) with respect to the real object of interest. Through such 3D registration, the mobile device can be turned into a magic lens, a window in the world that reveals invisible things. The motion of the device can be translated into a diverse set of interactions with the digital content (see Figure 1.6a). Moving the smartphone has analogies to moving an ordinary magnifying glass in order to change the focused area and the viewing perspective. This is called an embodied user interface because the smartphone embodies a well-known real-world object and the user can find analogies from experiences with the real-world object to explore the virtual functionalities [182].

As opposed to communication where high data capacity is required, augmented reality (AR) registration markers (Figure 1.6b) contain less information and must be easily detectable and trackable under various lighting conditions. While the code itself serves as a registration marker, it is also possible to encode a URL[3] to a model database where we can read a 3D model and metric scales of the object for initialization of markerless tracking [236, 237]. While polluting the environment with fiducial markers seems less attractive, in some scenarios it is advantageous to explicitly control the users' attention to the AR-enabled objects. When the user knows there is AR content for an object, he or she can explicitly enable or disable camera processing. This way, there is no need for constant visual processing so the application can save energy. Clever design also helps to hide visual codes into the environment, for example a Bokode (Figure 1.6c) is a simple light dot for the human eye but it reveals lots of information for a defocused camera.



(a) A visual code widget [182]

(b) AR wheel marker [344]



(c) Bokode [149]

Figure 1.6: Visual codes for interaction and augmented reality

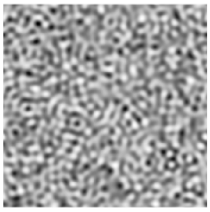
[3]URL: Uniform Resource Locator, an address to a resource on the Internet

1.1.5 Visual codes for communication

Visual codes can also be used to transmit information on wireless screen-camera links which attracted great research interest in recent years. Visible light communication has many advantages over radio communication including no multi-path interference, no licence fees, and privacy due to the short range and required line of sight. Earlier attempts like motion QR codes simply display a sequence of QR codes after each other, 4D Barcode [127] extends this to different colors, COBRA [74] improves on robustness against blur, RainBar [240] improves on robustness against segmentation errors. When compared to the all-or-nothing decodability of previous symbologies, the multi-level Strata [87] and the OFDM[4]-based PixNet [165], Focus [81], and the method of [150] allow the receiver to decode less or more information by getting further or closer to the code. VRCode [248], InFrame [238], ImplicitCode [198], and HiLight [135] can be hidden in a regular video stream and are invisible to the human eye. Such codes can be used for unidirectional communication in cinemas or at public displays, but also as watermarks in copyrighted content without disturbing the audience.

(a) Video response code [248]

(b) Strata code [87]



(c) Focus code [81]

Figure 1.7: Visual codes for communication

1.1.6 Other object identification technologies

Besides visual codes, a number of other other object identification techniques have been developed. One may expect that radio-frequency identification (RFID) or near field communication (NFC) will soon replace visual codes. The main advantages of RFID are the possibility of simultaneous scanning of multiple tags and no required line of sight. We argue that visual codes will remain an indispensable object identification method because of the limitations of radio waves in certain scenarios (e.g., metallic parts or water content, radio interference, tag collisions). Furthermore, the high price (passive $0.07 - 0.15$ USD, active $25 - 100$ USD in 2015 [320]) compared to visual codes and the lack of handling electronic waste make radio chips less attractive.

[4]OFDM: Orthogonal Frequency-Division Multiplexing

In the recent years, several commercial product recognition methods appeared based on advanced image recognition techniques. Image recognition engines like Google Goggles [291], Microsoft Bing Vision [306], Amazon Firefly [270], CamFind [276], or Kooaba [301] are able to recognize book and CD/DVD covers, posters, wine labels, etc. that link directly to specific products. One step further in this direction is 3D product recognition by companies like Moodstocks [309], Slyce [333], Aipoly[269], or Fashwell [284]. Visual product recognition engines, however, cannot provide us with item-level information because all items of the product have the same appearance. The recognition engines may also have problems with different appearance of the same product in different countries, or very similar appearance of different products of the same brand. As such, we argue that visual codes, and in particular 2D codes, are and will remain important in many applications.

1.2 A short history of barcode scanners

In the beginning, barcodes were read by simply swiping a photodiode over the code. Old scanner devices had a *wand* or pen-like form with a light source and the photodiode at the tip [89, 184]. To avoid the tedious and often imprecise manual swipe movement[5], *laser scanners* have been developed in the 1970s that swipe a laser beam over the code and read a slice of it by measuring the reflection [16, 221]. Swiping the laser beam can be implemented by a rotating mirror. Stacked (extended) 1D symbologies can also be read by making multiple passes over the code. The main advantage of laser scanners even today is the high speed that allows scanning even if the code or the scanner is (slowly) moving. Furthermore, some laser scanners are able to read codes at a range up to 15 meters. Aiming with the laser beam is also relatively easy for the user.

In the late 1980s, the moving laser had been replaced by a photodiode array [98] or by a 1D charge coupled device (CCD) image sensor (US patent 4874933 [186] in 1989). These so called *linear imaging scanners* consist of a line of light sources and light sensors and capture a single scanline across the code. Linear imaging is a cost-effective solution when long range and scanning 2D codes are not required.

The need for extending the data capacity of codes and the reducing cost of 2D imaging technology pushed the emergence of 2D symbologies (for instance, QR code [157]) and 2D scanners called *area imaging scanners* (US patent 5378883 [18] in 1995). An area imager is essentially a low-resolution high-speed camera and a light source. The main advantage of area imagers is their ability to read any symbology while laser scanners cannot read 2D

[5]Note that varying swipe speed over the code might lead to false decoding

codes as there is no swipe pattern that would work from any orientation. Therefore also producers of traditional scanning hardware started to shift towards 2D sensors [244], and we also focus on this general domain. A 2D imager can be also easily extended with photo capture and document scanning functionality. However, compared to laser scanners, area imagers have a limited, typically $5 - 30$ cm working range that stems from the relatively low resolution of the image sensor. The camera's typical spatial resolution is in the range of 640×480 to 800×600 pixels, and the typical temporal resolution is up to 60 Hz with a global shutter [295]. Today, area imagers built into handheld computers represent the most versatile tools for scanning visual tags. Examples of such handheld scanner devices are illustrated in Figure 1.8.

(a) Intermec SG20 handheld wireless scanner, $252.94 [299]

(b) Symbol DS6878-SR cordless 2D imager and base station, $620.00 [338]

(c) Intermec CN70 heavy-duty mobile computer, $1,290.00 [298]

Figure 1.8: A few current handheld area imager models. Prices are from Amazon.com in February 2016

In the late 2000s, programmable smartphones and tablets with built-in cameras appeared and boosted the adoption of visual tags and consequently many new application areas opened in the consumer domain [104]. Scanning products by simply taking our smartphone out of our pocket enables convenient mobile interaction and a wide range of services, which lead to an increasing interest in robust barcode scanning engines for smartphone apps. In 2011, Adelmann [2] presented a method for robust and fast scanning of defocused barcodes. Before 2010, smartphone cameras had no autofocus (AF) and were usually not optimized for capturing close objects. Even with more expensive models with a macro mode available, the user had to precisely align the code at a certain distance. Compared to a handheld laser scanner, this heavily deteriorated the user experience. The new blurry decoder algorithm made scanning without AF possible, and enabled very fast scanning when AF-cameras appeared, because the algorithm was able to decode images even before focusing actually succeeded. Blurry barcode scanning made companies like RedLaser (now owned by eBay) [327] and Mirasense (now Scandit) [331] a commercial success.

The professional barcode scanners in enterprises are expensive (cf. Figure 1.8) and therefore available to a limited number of employees only. The proprietary protocols of

dedicated scanners are also rather difficult to integrate into business applications. On the other hand, almost every employee has a smartphone today with extensive processing and sensing capabilities, with a wide variety of applications, and with an intuitive user interface. The camera of a smartphone is often much better than any area imager on the market, and advanced features like character recognition [267] and high-quality document scanning [281] can also be easily added as software features. Also, in contrast to area imagers, the resolution of today's smartphone cameras is high enough for scanning multiple codes at a distance. This, and other advantages, encourage the use of off-the-shelf smartphones as a replacement for traditional scanner devices.

In summary, visual codes that are traditionally rooted at enterprise applications transitioned to the consumer domain, where the fast technological advancements in consumer hardware allowed wide adoption and opened up possibilities for numerous new applications. Bringing the cheap and versatile smartphone scanners from consumers now back to the enterprises offers many advantages over the use of traditional scanners. Putting the smartphone in a ruggedized case can also protect it from environmental conditions. From a business process perspective, there lies great potential in a ubiquitous, smartphone-based scanning solution because it allows every employee to have a programmable barcode scanner and to access information on every item across the value chain.

1.3 Wearable barcode scanning

We argue that for an even broader adoption of visual tags, scanners must be very easy to use and must be available to everyone, everywhere, everytime. We propose to bring barcode scanning functionality not only to tablets and smartphones but also to smartwatches, smartglasses, and other personal wearable devices (see Figure 1.9). These emerging devices – and most importantly their combinations as we will see later – bear the potential to unify the advantages of long-range laser scanners with the versatility of area imagers, in addition to their original features. In the rest of the thesis, we will refer to barcode scanning with personal wearable computers, i.e., with devices that were not primarily designed for barcode scanning, as *wearable barcode scanning*. Figure 1.12 illustrates this transition from traditional scanner devices to the emerging class of new devices.

The idea of a wearable barcode scanner is not new, there exist several specialized devices for this purpose. The first hands-free body mounted laser scanner was already patented in 1993 [117]. The main problem with these devices is that they are very expensive with only limited applicability. Figure 1.10a illustrates the i.d.Mate Quest hand-mounted barcode reader device for visually impaired people. The built-in mini computer can access a database of products and read out loud the name of the scanned product. The Zebra

(a) Smartphones [342] (b) Smartwatches [343] (c) Smartglasses [341]

Figure 1.9: A few current personal wearable computers.

CS1504 keychain scanner is a very small laser scanner with internal memory that stores the scanned IDs. As the device has no radio unit, the IDs can be transferred to a computer via cable only. The Honeywell 8650 scanner ring is worn on the finger and is connected to a Bluetooth unit on the wrist. The Bluetooth unit then transmits the scanned IDs to a mobile computer or a fixed external computer. In turn, smartwatches and smartglasses offer a much wider applicability and we believe they can become powerful alternatives of dedicated wearable scanners.

(a) i.d. Mate Quest 1D product scanner for visually impaired people, $1,299.00 [297]

(b) Zebra Technologies CS1504 keychain laser barcode scanner, $129.99 [350]

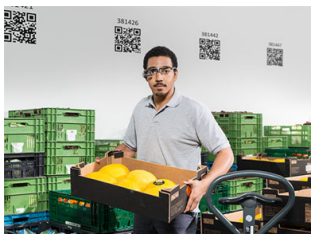
(c) Honeywell 8650 Bluetooth Ring Scanner, $1,148.00 (without computer) [294]

Figure 1.10: Wearable dedicated barcode scanners. Prices are from Amazon.com in February 2016

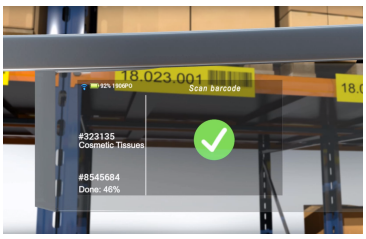
We are inspired by the success of smartphone-based scanning in customer scenarios and the success of head-mounted displays in order picking scenarios. Order picking is the task of collecting specific items into a basket in a large warehouse. It is an important and labour-intensive step of logistics and supply chain operations. Order picking is a monotone task that is prone to mistakes in the collected items which leads to customer dissatisfaction. Therefore, various systems have been developed for assisting employees in doing the task: pick-by-paper (list), pick-by-voice (audio guide), pick-by-light (notification lamps), pick-by-HMD (head-mounted display), etc. [60, 67, 174, 242].

Head-mounted displays are particularly useful in providing instant visual feedback such as navigation hints [112], an attention tunnel [174], or directly highlighting the next object [67] in form of augmented reality. It has also been shown that HMD-based picking solutions outperform other approaches in speed [250]. They are also preferred by employees due to lower mental workload [250]. Scanning barcodes on each picked item can ensure a low pick error rate, but current solutions require explicit scanning with a laser scanner. Our vision of wearable barcode scanning allows implicit scanning with smartglasses while putting an item into the basket. Concurrently, the first head-mounted code scanner was patented in 2014 [220] and the first commercial solutions (Ubimax [340], iTiZZiMO [300], Picavi [319], etc., see Figure 1.11) appeared. Wearable computers can also extend our physical and mental capabilities. For example, smartglasses enable vision enhancement [91], vision augmentation [21, 92, 143], automatic task logging and reporting [189], but also instant object recognition and interaction [145]. Live video transfer with smartglasses enables "eye swapping" between a specialist and a technician in the field [321]. The specialist in the service center can guide the local technician and even send documents to the technician's wearable display.

(a) Maintenance [275]



(b) Logistics [319]



(c) Order picking [319]

Figure 1.11: A few industrial use cases for smartglasses. Image credits: Brückner Servtec Callisto [275], Logcom GmbH Picavi [319])

Device	CPU	Memory	Display	Connectivity	Camera	Other
Intermec CN70 (2015)	1 GHz	512 MB	640 × 480 touch screen	WLAN, WAN (UMTS), Bluetooth	5 MP	keyboard, laser aimer (no laser scanner), GPS
Motorola Nexus 6 (2014)	quad-core 2.7 GHz	3 GB	2560 × 1440 touch-screen	WLAN, WAN (LTE), Bluetooth	13 MP	GPS
Osterhout R-7 (2016)	quad-core 2.7 GHz	3 GB	1280 × 720 see-through (2×)	WLAN, Bluetooth	5 MP still, 2.1 MP video @ 120 Hz	GPS

Table 1.1: Hardware comparison of current scanner, smartphone, and smartglasses models.

If we compare computing aspects, a current-generation smartphone is actually much more powerful than a current-generation handheld scanner computer. Table 1.1 compares the top-of-the-line *Intermec CN70* heavy duty mobile computer (Figure 1.8c) with a *Motorola Nexus 6* smartphone. Except the laser aimer, the large battery, and the protective casing, a smartphone beats the scanner in almost all parameters. Smartglasses and smartwatches are also quickly following up in computing power, the hardware specifications of the recent *Osterhout R-7* [317] smartglasses are close to that of the smartphone two years ago.

Personal wearable computers are becoming important parts of our daily lives and hence it is a straightforward idea to apply them as barcode scanners (see Figure 1.12) in addition to their intended usecases. Barcode scanning with personal wearable computers is attractive for many reasons:

- good processing power (CPU, GPU, DSP – driven by mobile entertainment)
- very good cameras (compared to area imagers) and other sensors
- high-resolution display
- constant wireless connectivity
- seamless connectivity *between* the devices
- multimodal input and output, intuitive user interface
- convenient, ergonomic form factor
- easy application development and business process integration
- flexible update with new functionality as software plugins
- wide accessibility
- barcode scanner applications are available for free [331, 349, 351]

However, combining wearable devices unlocks even more advantages. The smartglasses share the user's viewpoint, the smartwatches provide a convenient touch screen and pedometer, while the smartphone has advanced sensors and fast Internet connectivity. The camera of any device can be applied to scan visual codes and to provide instant navigation hints. The smartphone can take the role of the main computation hub and the other devices can assist the user and enable cross-device interaction with other smart objects in the environment [145].

Overall, personal wearable computers can become strong competitors of traditional scanners and enable new use cases for barcode scanning. The unifying theme of this thesis is barcode scanning with off-the-shelf mobile and wearable devices, which bears the potential of widely extending the possibilities of information access on the go. Our long-term goal is to bring the comfort, the productivity, and the business opportunities of wearable barcode scanning also to the professional users who are accustomed to the performance of laser scanners.

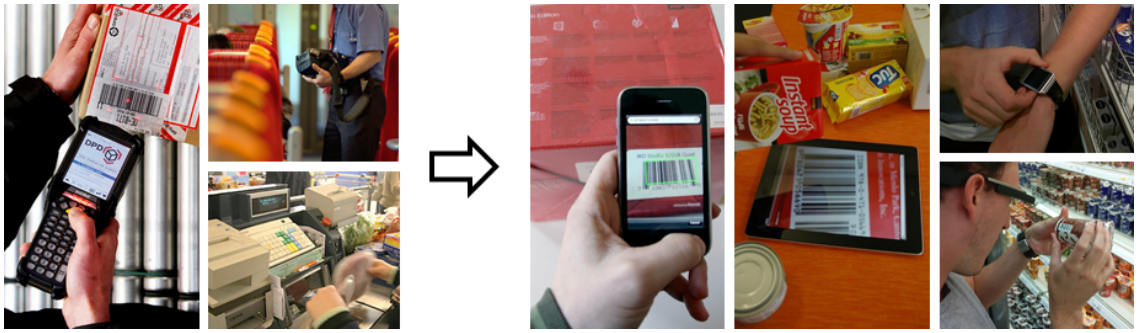


Figure 1.12: Traditional barcode scanning (left three photos) vs. wearable barcode scanning (right four photos). Typical wearable scanning scenarios: 1.) 'shopping' scan with a handheld camera, 2.) 'checkout' swipe past a static camera, 3.) 'shopping' scan with a watch 4.) 'glasses' scan with a head-worn camera.

1.4 Research problems and contributions

As discussed above, visual codes represent important physical-virtual links in our digitally augmented environments, and wearable computers offer plenty of advantages over traditional barcode scanners in addition to their great original features. However, we also have to overcome a few difficulties that arise when using wearable cameras instead of laser scanners (see Figure 1.13). For example, wearable computers come without a laser aimer which makes code positioning more difficult. Even though the spatial resolution of wearable cameras is often high enough to capture multiple codes, the temporal resolution is usually lower than that of area imagers which might lead to motion blur in the captured images. Furthermore, wearable cameras apply a rolling shutter instead of a global shutter that may introduce severe distortions in the image when the code is moving (on a conveyor belt, for instance). Even if cameras in smartglasses offer perhaps the most advantages, smartglasses have very limited input capabilities compared to handheld devices.

In this dissertation, we describe methods that overcome these limitations and add advanced features that can make wearable barcode scanning an attractive alternative to traditional barcode scanning even outside the consumer domain. Our algorithmic contributions can be divided into three areas: *visual code localization*, *blur compensation*, and *gesture control*. We propose fast and robust solutions to these problems that leverage the capabilities of the latest generation of wearable computers, in particular their cameras, sensors, and GPUs.

Figure 1.13: Challenges in wearable barcode scanning: lack of laser beam for localization, (multiple) small codes, defocus and motion blur, lack of robust and efficient user input (image credit: [319])

1.4.1 Fast and robust visual code localization

In many existing smartphone-based barcode scanning solutions, a possible way to reduce the computational cost is to limit the scanning algorithm into a search window and along a predefined scanline over the camera image. The user is required to hold the product close to the camera and align the code into the search window. This is especially problematic with smartglasses and leads to lower user acceptance. A related issue is the lack of a laser beam that shows where the scanner is currently reading. The need for continuous code re-alignment in the image frame can easily lead to user frustration. Therefore, we must be able detect codes in the full image area and provide immediate visual feedback to the user.

The resolution of wearable cameras is typically high enough to scan visual tags that are further away (about 1 m range) once the tags are segmented in a preprocessing step. Scanning multiple codes will become straightforward if a localization method can produce and store code probability maps (i.e., a map of possible code candidates in a large image) in an intermediate step. The ability to scan multiple codes simultaneously is one of the key advantages over laser scanners and traditional low-resolution area imagers. Hence, our first challenge is visual tag localization in large digital images, especially in case of small codes, distant codes, and blurry codes.

We propose a fast algorithm for joint 1D and 2D visual tag localization in large digital images. We focus on two prevalent examples: 1D EAN/UPC barcodes and 2D QR-codes that have different characteristics. Our proposed localization method outperforms existing methods in terms of accuracy while it is invariant to scale, orientation, code symbology, and is more robust to blur than previous approaches. We further optimize for speed by exploiting the parallel processing capabilities of mobile graphics hardware for image processing. Fast code segmentation enables scanning multiple codes at the same time and

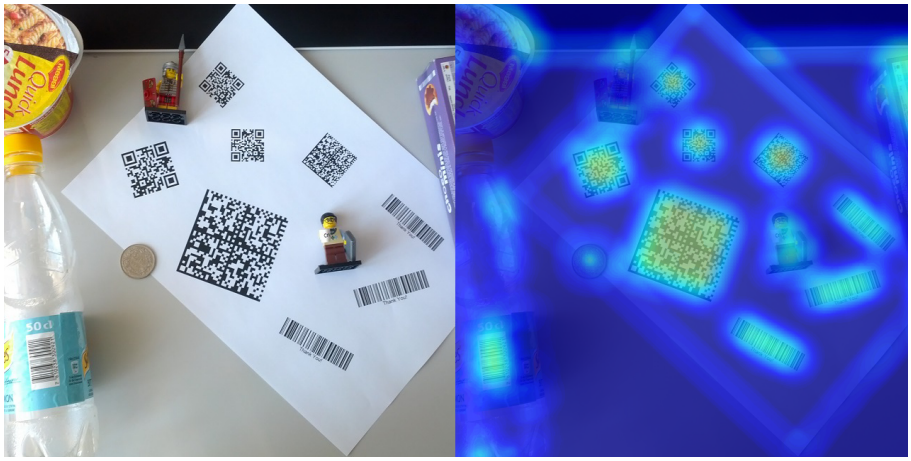


Figure 1.14: Our localization method is able to detect 1D and 2D symbologies at various scales. Left: input image. Right: code probability map overlaid on the image

thus helps for instance in assembly scenarios where interaction with multiple objects is necessary.

1.4.2 Fast and robust motion blur compensation

In contrast to the performance of laser scanners, smartphones and wearable cameras suffer from various types of image degradation such as *defocus blur* or *motion blur*. Blur can make the scanning difficult or even impossible without image enhancements. These limitations make the use of wearable computers less attractive in enterprise applications where scanning speed and reliability are of great importance. Professional users are accustomed to the performance of laser scanners and are unwilling to hold the devices steady or to carefully align the barcode with a wearable camera.

Defocus blur can already be successfully compensated in commercial applications. Motion blur is a more general and mathematically more complex form of image degradation caused by slight motion of the camera or the code during scanning (see Figure 1.15). The problem of removing motion blur from photographs has been widely studied in the past decade but besides being slow, the existing algorithms are optimized for natural images and typically fail on artificial visual tags.

We build upon existing work in photograph deblurring and develop a fast algorithm for scanning motion-blurred QR codes on mobile devices. We exploit the fact that QR codes do not need to be visually pleasing for decoding and propose a fast restoration-recognition loop that exploits the special structure of QR codes. In our optimization scheme we interweave blind blur estimation from the edges of the code and image restoration

Figure 1.15: Top: When the camera or the scanned object move even slightly during scanning, the captured visual codes become distorted by motion blur and conventional decoder algorithms cannot decode them. Bottom: We propose a fast blur removal algorithm that allows the scanning of motion-blurred QR codes.

regularized with typical properties of QR code images. Our proposed restoration algorithm is on par with the state of the art in quality while it is about a magnitude faster. We also propose to combine blur estimation from image edges with blur estimation from built-in inertial sensors to make the restoration even faster. Fast blur compensation means that no precise code alignment is required but the user can simply swipe the camera in front of the code.

1.4.3 Fast and robust gesture control

The input capabilities of traditional barcode scanners are usually limited to a set of buttons. Smartphones offer convenient touch screens but are hard to operate when wearing gloves. Hands-free operation is certainly one of the biggest advantages of using smartglasses over using any handheld device. In our vision, the user just needs to look in the direction of a visual tag that automatically gets localized and scanned, while the user's hands remain free to concentrate on the actual task. As the wearable camera is always on for scanning, we can also use the visual channel for natural hand gesture input to our wearable computers, but also to other smart objects in our environment. In particular, smart appliances and other connected machines in our smart environment can be automatically recognized based on their tiny visual tags, and these machines can "outsource" their own user interface to

our wearable devices. Such a mechanism brings expressive vision-based gesture control even to those appliances that do not posses an own camera.

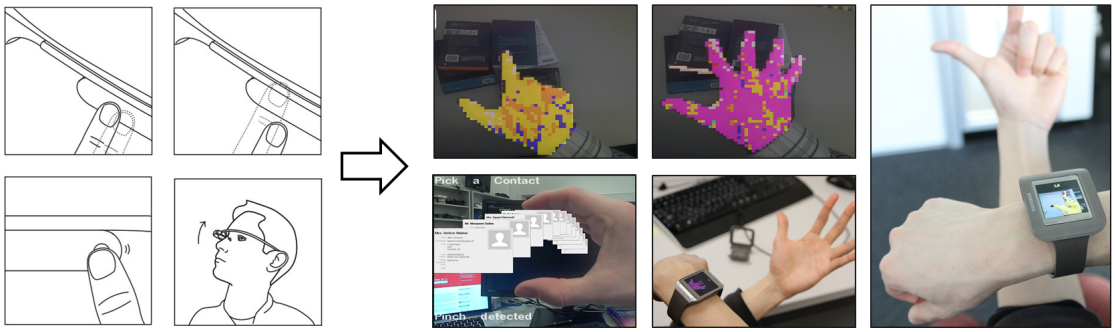


Figure 1.16: Left: The small buttons and slim touchpads of personal wearable computers are hard to operate when wearing gloves. Images from [288]. Right: We enable 3D in-air hand gesture control using the simple built-in RGB camera of the wearable devices.

Gesture input so far has required either special hardware instrumentation, special gloves, or energy-consuming depth cameras. Our goal is to enable simple gesture control with the built-in camera of smartglasses. We present a machine learning technique to recognize hand gestures with only a single camera now commonplace on off-the-shelf devices. The algorithm robustly recognizes a wide range of in-air gestures and runs in real time on commercially available wearable devices. We further show that with little modification, our method can not only classify the hand gestures but can also regress the distance of the hand from the camera. Such 3D in-air gesture control allows hands-free scanning with smartglasses and brings many advantages in smart environments. A large part of the gesture recognition results stem from joint work with the Advanced Interactive Technologies group of ETH Zurich, which is marked at the respective parts in the text.

1.5 Positioning and structure of the thesis

This thesis can be positioned at the intersection of several computer science disciplines. Linking the physical and virtual worlds – via visual codes or via other means – is a long-standing problem in *ubiquitous computing* research. We primarily deal with visual signals, i.e., signals of *mobile computer vision* all over the thesis. We learn from well-established methods in texture detection and blur removal from the vast literature on *image processing*. Furthermore, we combine various sensor modalities to improve our images which is has been long investigated in the field of *computational photography*. We also propose new techniques for *human computer interaction*. Finally, in the whole thesis, we specifically

target a new class of emerging devices that originate from decades of research in *wearable computing*. We will see how generally very difficult visual computing problems such as

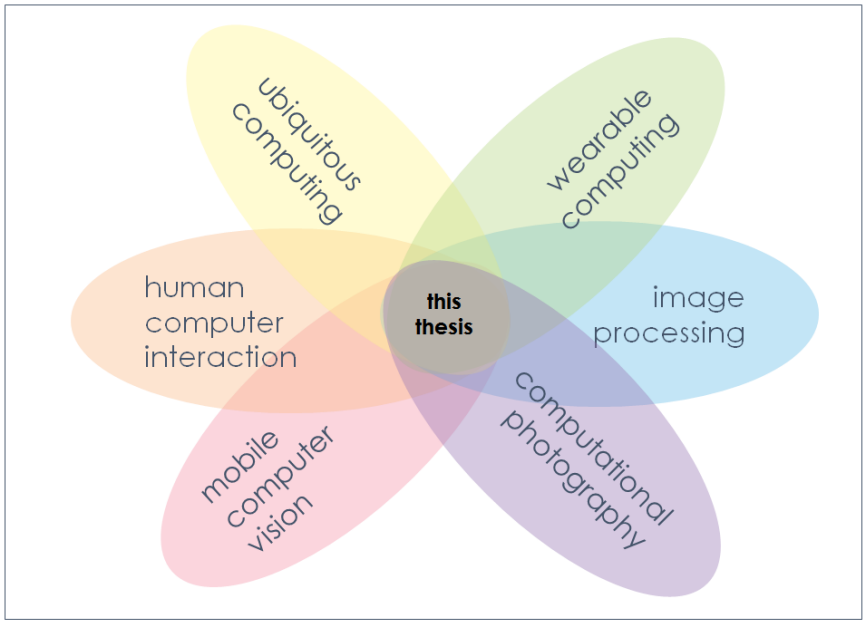


Figure 1.17: This thesis can be positioned at the intersection of several computer science disciplines.

object detection, *image restoration*, and *shape classification* can still have fast and robust solutions even on resource-constrained wearable devices when we restrict ourselves to a specific domain of binary images – visual codes and hand silhouettes.

The thesis does not describe a single line of research but rather describes a selection of techniques related to wearable computers and visual codes. The main technical chapters form self-contained individual entities and can be read separately. Parts of the contributions stem from collaboration with other researchers and students, therefore the document is formulated in the more inclusive scientific plural. Contributions of other people are explicitly marked in the text, and summarized in a contribution statement in Section 5.1.5.

The structure follows the order of our contributions outlined above. Chapter 2 deals with fast and robust code localization, Chapter 3 is about fast and robust blur compensation, while Chapter 4 is devoted to fast and robust in-air gesture control. Finally in Chapter 5, we summarize our findings and propose directions for future research. A review of previous work and the description of necessary mathematical tools are included in the respective chapters and in the Appendix. A large part of the results described in the thesis has been peer-reviewed and presented at conferences, these publications are also listed below (Section 1.6) and referenced at the corresponding chapters. All our algorithms are

implemented on mobile and wearable devices and have been demonstrated along with the publications. The appendix describes the content of accompanying videos recorded during the demonstrations.

1.6 Publications on parts of the thesis

This dissertation presents research that has also been published in the following papers.

[208] Gábor Sörös, Christian Floerkemeier – Poster: Towards Next Generation Barcode Scanning, *In Proceedings of the* 11^{th} *ACM International Conference on Mobile and Ubiquitous Multimedia (MUM 2012), Ulm, Germany, December 2012*

[209] Gábor Sörös, Christian Flörkemeier – Blur-Resistant Joint 1D and 2D Barcode Localization for Smartphones, *In Proceedings of the* 12^{th} *ACM International Conference on Mobile and Ubiquitous Multimedia (MUM 2013), Lulea, Sweden, December 2013*

[206] Gábor Sörös – GPU-Accelerated Joint 1D and 2D Barcode Localization on Smartphones, *In Proceedings of the* 39^{th} *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014), Florence, Italy, May 2014*

[145] Simon Mayer, Gábor Sörös – User Interface Beaming - Seamless Interaction with Smart Things using Personal Wearable Computers, *In Proceedings of the* 11^{th} *International Conference on Wearable and Implantable Body Sensor Networks (BSN 2014) Workshop on Glass & Eyewear Computers, Zurich, Switzerland, June 2014*

[204] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Fanello, Shahram Izadi, Cem Keskin, Otmar Hilliges – In-air Gestures Around Unmodified Mobile Devices, *In Proceedings of the* 27^{th} *ACM User Interface Software and Technology Symposium (UIST 2014), Honolulu, Hawaii, USA, October 2014*

[203] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, Otmar Hilliges – Joint Estimation of 3D Hand Position and Gestures from Monocular Video for Mobile Interaction, *In Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2015), Seoul, South Korea, April 2015*

[210] Gábor Sörös, Severin Münger, Carlo Beltrame, Luc Humair – Multiframe Visual-Inertial Blur Estimation and Removal for Unmodified Smartphones, *In Proceedings of the*

23rd International Conference on Computer Graphics, Visualization and Computer Vision (WSCG 2015), Plzen, Czech Republic, June 2015

[205] Jie Song, Gábor Sörös, Fabrizio Pece, Otmar Hilliges – Extended Abstract: Estimation of 3D Hand Position and Gestures on Unmodified Wearable Devices, *In Proceedings of the IEEE Workshop on Observing and Understanding Hands in Action (HANDS 2015), in conjunction with the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015), Boston, MA, USA, June 2015*

[212] Gábor Sörös, Stephan Semmler, Luc Humair, Otmar Hilliges – Fast Blur Removal for Wearable QR Code Scanners, *In Proceedings of the 19th International Symposium on Wearable Computers (ISWC 2015), Osaka, Japan, September 2015*

Chapter 2

Visual code localization

This chapter presents a new method for finding 1D and 2D visual codes in large digital images captured by smartphones. Most parts of the results have been published in [206] and [209].

2.1 Introduction

In wearable barcode scanning, code localization is an important preprocessing step that quickly scans the entire camera image and passes code candidates to the actual decoder. The ideal code localization algorithm is scale-, orientation-, symbology- and blur-invariant (see Figure 2.1) and operates in real time on the entire camera image. Scale-invariance is important because visual codes are printed in different form factors and the distance to the smartphone camera can vary. Orientation invariance is necessary because the user will not always place the code exactly horizontally in the camera image. There are also a number of different code symbologies that need to be successfully detected by the code localization algorithm. These include linear 1D barcodes such as those found on consumer products and on logistical labels like EAN13, UPC12, CODE39, CODE128 (see Figure 1.1) and 2D codes such as the popular QR codes, but also DataMatrix and PDF417 codes (see Figure 1.2). Enterprise applications may require scanning 1D and 2D symbologies by the same algorithm without explicit switching by the user. Since the latest commercial barcode decoders can also decode blurry barcodes, it is also important to localize blurry barcodes in camera images. Blurry barcode images can result from a fixed focus camera, a barcode being placed too close to the camera or the camera focusing on the distant background when the barcode is small (cf. Figure 2.1 right). Previous work focused only

on selected aspects of the requirements mentioned above such as orientation invariance and speed for 1D codes or scale invariance for 2D codes.

Figure 2.1: Visual code localization in digital images poses *four main challenges*: different scales, different orientations, different symbologies, blur.

In this chapter, we present a code localization approach that is orientation, scale, and symbology (1D and 2D) invariant and shows better blur invariance than existing approaches while it operates in real time on a smartphone. Our probability maps of code locations over the image are derived from the structure matrix and the saturation in the HSV color system. The comparison with three other real-time code localization algorithms that represent the state of the art shows that our approach outperforms existing ones with respect to symbology and blur invariance at the expense of a reduced speed. Besides the efficient CPU version, we present the parallel implementation details of the algorithm on mobile GPUs. The different steps of the localization algorithm can be formulated as OpenGL ES 2.0 fragment shaders, and both 1D and 2D barcode saliency maps can be computed directly on the graphics hardware. The fragment shaders are portable between all tested wearable platforms. The presented method can detect barcodes and QR codes at various scales and orientations at 22 frames per second in HD resolution images on a current generation smartphone. In the next section, we give an overview of previous work on 1D and 2D tag localization and summarize what we can learn from previous methods.

2.2 Related work

The wearable barcode scanning pipeline consists of several steps. In the initial step, video frames are acquired from the camera. The frame rate is usually between 5 and

30 frames per second and the resolution in video mode is typically between 640×480 and 1280×720 pixels depending on the required quality and speed of processing. In practice, the scanning algorithms must be limited to a small search window in the image because (blurry) decoding is computationally complex. Due to the lack of a laser beam, the users do not see where the scanner is currently reading. Thus they often struggle with continuously aligning the barcode at a dedicated position and angle within the search window. To maximize ease of use, barcodes need to be scanned in the entire camera image. For that a quick barcode localization algorithm is required that selects possible barcode candidates before a particular scanline through the code is decoded using approaches such as Scandit [331], RedLaser [327], or ZXing [351].

Despite the fact that barcodes have an apparent structure for the human eye, the fast and robust localization in digital images is still an active research area. The different approaches presented previously can be categorized by the image processing technique used. Previous work on 1D localization includes simple image filters, orientation histograms, line detection approaches, morphology operators, Gabor filters, and harmonic analysis. 2D localization has been previously implemented via image thresholding and scanning, via orientation maps, or via combinations of line/corner/square detection.

2.2.1 Localization of 1D barcodes

Simple image filters combine various low-level image features (e.g., image gradients, intensity variance, etc.), and try to find bar-like structures [11, 63, 234, 257]. They offer a good compromise when accuracy is not crucial but localization speed is an important aspect in the application. Ando and Hontani [11] detected sharp 1D barcodes by classifying image regions to plain, uni-directional, and omni-directional areas and by inspecting the transitions between those areas. Gallo et al. [63] presented a very fast barcode localization technique that despite its simplicity outperforms many complex methods. However, it works only with a single sharp code that lies horizontal (less than 30° rotated) in the image. For each pixel, the algorithm calculates the measure $m = |I_x| - |I_y|$ where I_x and I_y stand for image derivatives in x and y directions, respectively, and $|X|$ denotes the absolute value of X. The temporary image m is then blurred with a box filter and binarized by Otsu's method [159]. The idea behind this approach is that horizontally oriented barcodes have strong gradients in the x direction but no gradients in the y direction, so m is very high at bars in a barcode. The box filtering connects the bars into a region. To find the axis-aligned bounding box of the code, Gallo et al. first find the maximum value in m (assuming this is within the barcode region) and search in four directions on the binarized image for the region boundaries. Finally, a rectangle is fitted on the four endpoints. We also apply some of these ideas in our approach.

Orientation histograms are used for 1D barcode localization by Tekin and Coughlan [230]. Their algorithm is part of the BLaDE barcode scanning aid for visually impaired users who have difficulties with aligning codes in a search window. The algorithm builds a histogram of gradient orientations for each 20×20 patch in the image, calculates the entropy (peakedness) of the histogram, and finds its dominant orientations. The patches are then clustered according to their orientation. The resulting regions are tested for alternating edges in their dominant orientation and the most likely barcode candidate is selected. This approach provides orientation invariance but the patch size has to be tuned carefully to the expected code scales. Also many other algorithms look at the distribution of the gradient orientations over image patches; sharp 1D codes have a single peak and 2D codes have two peaks 90° apart. However, when the image gets blurred, many of the thin bars disappear and the code breaks into many disconnected parts. The blur also flattens the orientation histogram of a 2D code, it "smears" the gradients in practically all directions making the common localization algorithms fail.

Line detection with Hough transform has also been used to estimate the orientation of a single barcode close to the camera with little background clutter [1]. Line detection methods are also useful for finding 2D codes. Dubska et al. [49, 222] presented a fast localization algorithm that successfully distinguishes 2D codes from text. However, to detect a sufficiently large number of lines, the code must be fairly prominent in the camera image.

Mathematical morphology is another tool for detecting bar structures. Combinations of image erosions and dilations enhance barcode areas but these algorithms tend to produce frequent false positives and will always be constrained by the proper choice of the structuring elements. Detection at different scales requires a search with multiple structuring elements which can be a rather slow process. An overview on morphological localization can be found in the work of Katona et al. [105]. The authors also propose an alternative approach that uses bottom-hat filtering and a distance map and achieve over 90% detection rates in their simulations. Their method can be extended to combine 1D and 2D localization but it often classifies text areas as part of a code.

Gabor filters are combinations of different Gaussian and sinusoid functions that model the edge-sensitivity of the human primary visual system. They can be used to detect stripe patterns at any orientation and scale. Concrete examples in localization include [124, 239]. These methods offer scale and rotation invariance but tend to be slow.

Harmonic analysis approaches transform the image into the Fourier, DCT, or Wavelet domain and look for barcode-specific features. Examples include [72, 119]. We consider these methods to be too complex for real-time barcode scanning on smartphones.

2.2.2 Localization of 2D barcodes

2D-specific methods for codes like DataMatrix, QR, etc., rely on the codes' apparent black and white rectangular structures and special finder patterns with given black-white signal ratios. The typical localization pipeline consists of image thresholding and line-by-line search in the binary image for the finder patterns. Obviously, if the finder patterns are degraded by blur the black-white proportions do not follow the standards anymore.

Alfthan's work [6] is devoted to localizing QR codes in blurry images with three different approaches. First, the author trains an SVM on color and intensity variance features extracted from QR codes. This approach can successfully separate the code area from text but is not scale-invariant. The second, morphology-based method requires precise parameter tuning. Third, the author tries to find the white square support of the code in Hough-space, which is inherently influenced by the success of the underlying edge detection.

Xu et al. [255] presented the first method to localize and also deblur linearly blurred 2D codes using a special camera. They localize the blurry code based on intensity variance, Harris corner density [75] and background substraction. The authors rely on the fact that corners are less sensitive to motion blur than edges so the many corners of a 2D code can still be found in a blurry image.

In summary, there is a vast amount of literature on localizing barcodes in digital images, but none of the existing approaches focuses on all four identified challenges. The existing algorithms usually need to trade accuracy and robustness for speed and make certain assumptions about code orientation, code scale, or code symbology. Existing algorithms often have difficulties with defocused or motion-blurred images because blur distorts the barcode structures. We developed a combined approach that addresses orientation, scale, symbology, and blur invariance together.

2.3 Fast joint 1D and 2D code localization

In this section, we present a new localization algorithm for 1D and 2D codes that also works with blurry images. We rely on the facts that 1D codes consist of black and white parallel bars while 2D codes have a black and white grid structure. The grid corners of 2D codes are less sensitive to blur and stay apparent also in blurry images. In our combined algorithm, we search for areas with high concentration of edge structures as well as for areas with high concentration of corner structures. We derive two separate barcode probability maps from the structure matrix for 1D and 2D codes. The two maps can be

calculated very efficiently at the same time. To further speed up the sharp localization and to allow blurry localization, we also rely on information from the HSV (hue, saturation, value) color channels, an approach that – to the best of our knowledge – has not been proposed before.

2.3.1 Structure matrix

Our goal is to find image areas with many edges and corners. We combine the derivations of Harris [75] and Ando [10] who both defined edge and corner measures based on the structure matrix (also called second moment matrix). As we will see, using the structure information instead of simply the gradient like other methods can enhance the robustness against blur.

The intuition behind the structure matrix is that when we shift a patch around within its small neighborhood and compare it with the underlying pixels, we can distinguish between flat, edge, and corner areas (see Figure 2.2). In a flat image area, the image intensities are

Figure 2.2: Flat, edge, and corner image areas can be distinguished by shifting a patch and comparing it with its small neighborhood. Figure adapted from [75].

the same in any direction, in an edge area the change is small in one direction while large in a perpendicular direction, while in a corner area the change is large in any direction. These changes can be captured in mathematical form in various similar ways. While Harris approached the problem from approximating the local autocorrelation function of the image, Ando looked at the covariance of the x- and y-gradients. In fact, both methods calculate the structure matrix in the first step.

The E window-averaged change in patch intensity for the shift (u,v) can be written as:

$$E(u,v) = \sum_{x,y} w(u,v)\,[I(x+u,y+v) - I(x,y)]^2$$

Now, if we consider the second-order Taylor series expansion of $E(u,v)$ for small shifts, we arrive at the bilinear approximation

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

The 2×2 structure matrix M is calculated for each pixel p as:

$$M = \begin{bmatrix} < I_x^2 > & < I_x I_y > \\ < I_x I_y > & < I_y^2 > \end{bmatrix} = \begin{bmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{bmatrix}$$

where I_x and I_y are the image derivatives in x and y direction. The entries C_{ij} are calculated over the D neighborhood of the pixel p with a box or Gaussion window function w:

$$C_{i,j} = \sum_{(x,y) \in D} w(x,y) I_i(x,y) I_j(x,y)$$

The structure matrix M has a number of important properties. It is a Hermitian (self-adjoint matrix) matrix, i.e., $M = transp(conj(M))$, so its eigenvalues λ_1 and λ_2 are always real, and its eigenvectors form an orthonormal basis. The two eigenvectors are in fact the principal components of the ellipse that M describes (ellipse equation $E = c$ constant), and show the directions of the fastest and slowest change in E. The two eigenvalues show the rate of change, i.e., are proportional to the principal curvatures of the local autocorrelation function [75], and are also proportional to the variances of the two principal components of the (I_x,I_y) distribution [10].

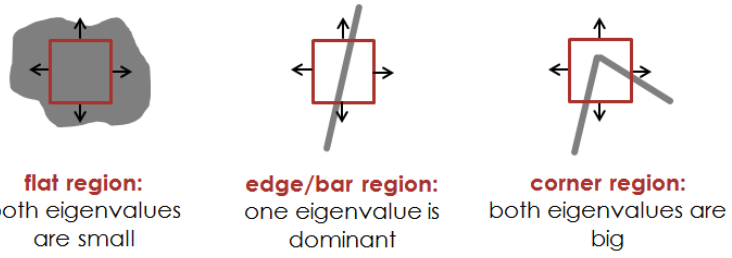


Figure 2.3: Flat, edge, and corner image areas can be distinguished by inspecting the eigenvalues of the structure matrix. Figure adapted from [75].

We found that the calculation of the derivatives of the discrete image is indeed an important factor for robustness to barcode orientation. Because calculating the image gradient via the common $[1,-1]$ and $[1,-1]^T$ filters produces relatively large errors at slanted edges, we chose for the optimized derivative filters of Farid and Simoncelli [57] that improved our detection rates.

2.3.2 Edge and corner maps

We can now distinguish between flat, edge, and corner image areas by looking at the eigenvalues of the structure matrix (see Figure 2.3). The λ eigenvalues of the structure matrix are the solutions of the quadratic equation

$$\begin{vmatrix} C_{xx}-\lambda & C_{xy} \\ C_{xy} & C_{yy}-\lambda \end{vmatrix} = (C_{xx}-\lambda)(C_{yy}-\lambda) - C_{xy}^2 =$$

$$= \lambda^2 - (C_{xx}+C_{yy})\lambda + C_{xx}C_{yy} - C_{xy}^2 = 0.$$

Using Vieta's formulas, the two solutions λ_1 and λ_2 satisfy

$$\lambda_1 + \lambda_2 = C_{xx} + C_{yy} = trace(M) > 0$$

$$\lambda_1 \lambda_2 = C_{xx}C_{yy} - C_{xy}^2 = det(M) > 0$$

The discriminant of the quadratic equation is always non-negative:

$$(C_{xx}+C_{yy})^2 - 4(C_{xx}C_{yy} - C_{xy}^2) = (C_{xx}-C_{yy})^2 + 4C_{xy}^2 \geq 0.$$

This means that both λ_1 and λ_2 are real and non-negative and therefore the inequality of their additive and multiplicative average must hold. By taking the ratio of their multiplicative average to their additive average, one can define the homogeneity measure m_2:

$$m_2 = \Big(\frac{\sqrt{\lambda_1\lambda_2}}{(\lambda_1+\lambda_2)/2}\Big)^2 = \frac{4(C_{xx}C_{yy}-C_{xy}^2)}{(C_{xx}+C_{yy})^2}$$

Since if $\lambda_1, \lambda_2 > 0$, we have

$$0 \leq \sqrt{\lambda_1\lambda_2} \leq (\lambda_1+\lambda_2)/2$$

the measure m_2 is dimensionless and is normalized such that $0 \leq m_2 \leq 1$. Let us define a complementary measure m_1 as

$$m_1 = 1 - m_2 = \frac{(C_{xx}-C_{yy})^2 + 4C_{xy}^2}{(C_{xx}+C_{yy})^2}$$

which is also dimensionless and is normalized such that $0 \leq m_1 \leq 1$. To avoid division by zero in flat image regions, we add a small number ε to the denominators. Ando proves the following properties: (1) m_1 reaches 1 where the image intensity varies one-dimensionally (edges and ridges), (2) m_2 reaches 1 where the image intensity changes with circular symmetry or with rotational periodicity with a period $\pi/2$ (checkerboard

corners). We also note here that Harris defined the cornerness as $det(M) - k * trace(M)^2 = (C_{xx}C_{yy} - C_{xy}^2) - k * (C_{xx} + C_{yy})^2$ with $k = 0.05$, which is similar to the form of m_2. For more mathematical details, we refer to [10] and [75].

2.3.3 Block filtering

So far, we have defined the measure m_1 which is strong at edge structures in any orientation and the measure m_2 which is strong at corners.

$$m_1 = \frac{(C_{xx} - C_{yy})^2 + 4C_{xy}^2}{(C_{xx} + C_{yy})^2 + \varepsilon}, \quad m_2 = \frac{4(C_{xx}C_{yy} - C_{xy}^2)}{(C_{xx} + C_{yy})^2 + \varepsilon}$$

Next, we blur these two maps by a block filter to connect areas where there is high line density or high corner density. Note that while the applied edge/corner detection is invariant to illumination changes and to rotation, it is not invariant to scale changes. To achieve invariance over large barcode scale changes, the calculations of M have to be repeated on multiple image scales (bigger and bigger image patches D) and selecting extrema over the scales (see for example [147]). As one of our main concerns is speed, we calculate the m_1 and m_2 maps only on a single scale (HD input resolution, see experiments in Section 2.4) which also gives fair scale invariance over typical barcode sizes within 1 m range. Thanks to the fact that visual codes contain plenty of edge and corner structures of different size, there will be lots of edges and corners detected even using a fix neighborhood size $D = 7$ independent of the distance between the code and the camera. We achieve good scale invariance by connecting those edges/corners via a big block filter. In our experiments we used a separable box filter of size 30 pixels. This not only connects high-density areas but also removes separated edges/corners of background clutter. There is, however, often text in the vicinity of visual codes that also exhibits high edge and corner density, so a further step is necessary to remove those areas. The next steps are illustrated in Figure 2.4.

2.3.4 Barcode saliency maps

The two box-filtered maps are linearly combined to get our final barcode saliency maps s_1 and s_2. The idea behind this is that 1D codes must not contain corner areas (note that this is how text gets removed from 1D codes) and sharp 2D codes must also contain edge areas. The barcode saliency maps are defined as $s_1 = m_1 - \alpha m_2$ and $s_2 = \beta m_2 + (1 - \beta)m_1$. We empirically found that $\alpha = 0.25$ and $\beta = 0.65$ give good results. The resulting "barcodeness" images are thresholded and barcode borders are found by tracing the

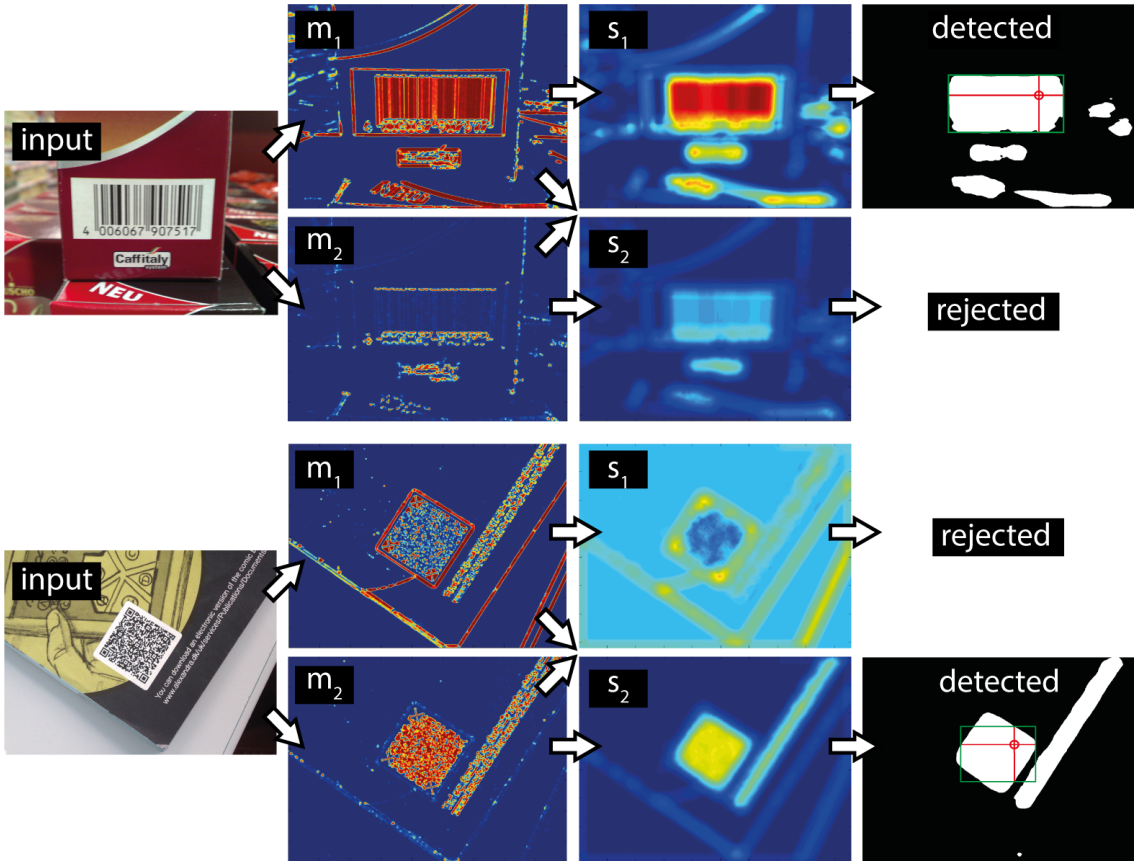


Figure 2.4: Algorithm outline: m_1: edge density map, m_2: corner density map, s_1: 1D saliency, s_2: 2D saliency, bounding box detection. The color coding is normalized to the range $[0, 1]$.

binary image in four directions starting from the pixel with maximal strength (cf. Gallo's method [63]). Note that using a more sophisticated method for finding the peak areas in the barcodeness map would allow for detecting multiple codes even with different symbologies.

The above described method works reliably with sharp 1D, sharp 2D, and blurry 2D codes. However, blurry 1D codes are challenging as the blur leaves little structure information in the code area (see Figure 2.6). If the camera focuses on a background that contains strong parallel lines, the algorithm finds that area more likely to be a barcode. To overcome this limitation we propose an extension to our algorithm using information from the saturation channel.

2.3.5 HSV color information

So far we have not considered that barcodes are almost always printed black and white and in most cases also in a rectangular white support. This is a very important clue because if we look at a sharp and a blurry code in the HSV (hue, saturation, value) color system, their saturation values are very low in both cases. Saturation is a normalized measure of color content, it is defined as $s = (\max(R,G,B) - \min(R,G,B))/\max(R,G,B)$, and can have values between 0.0 and 1.0. Black, white, and all gray shades inbetween have saturation 0.0 while the true rainbow colors have saturation 1.0 (see Figure 2.5). Although blur distorts the intensity structure of a code, smearing black and white together still results in gray and hence low saturation. Looking at the saturation of each pixel can be a fast first

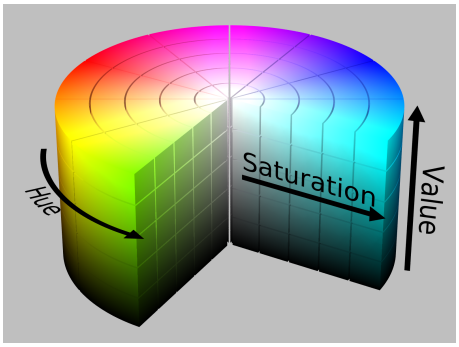


Figure 2.5: HSV color system [348]

test to reject non-barcode pixels in general. We also augmented our edge/corner detector with a color conversion module and we set all pixels with saturation above 0.25 to zero in both m_1 and m_2 to remove background clutter.

In very blurry cases where only little structure information is left, code localization may be still possible by finding rectangles in the saturation channel (see Figure 2.6). If both m_1 and m_2 are very low, our algorithm switches to rectangle detection mode. Of course, this works only with the mentioned rectangular white support but it is the case with most consumer products.

We note here that our original algorithm uses the saturation values, but in our smartphone experiments we noticed that under low illumination, the saturation is unstable in dark and noisy regions of the image. In fact, colors close to black may have small or large saturation values depending on the little difference in RGB values which makes the thresholding unreliable. Therefore in our smartphone implementation, we replace the saturation with the chroma value $c = \max(R,G,B) - \min(R,G,B)$, which is guaranteed to be close to zero for all shades close to black. This problem does not affect white or the other shades of

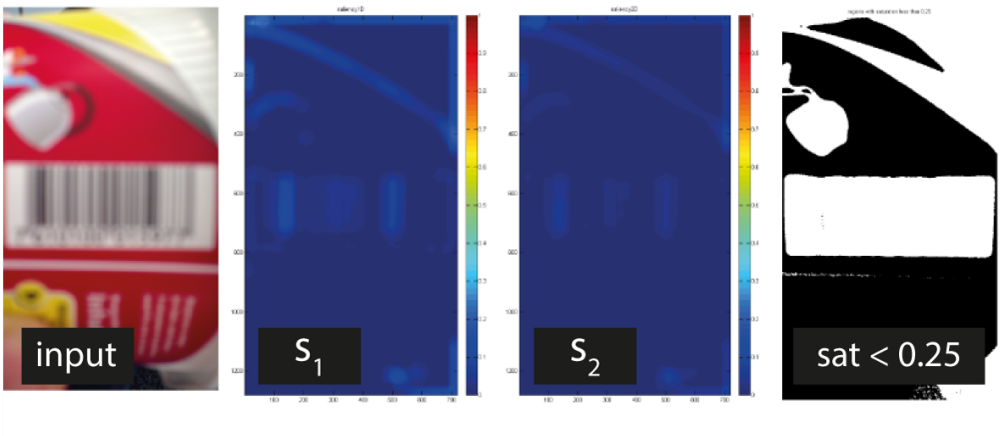


Figure 2.6: A very blurry 1D code results in low 1D saliency (s_1) and low 2D saliency (s_2) but if wee look at areas with saturation below 0.25, the code rectangle is clearly visible.

gray. As the chroma value is not normalized, the threshold value requires little manual adjustment depending on the overall scene brightness.

2.4 Evaluation

2.4.1 Implementation

We have developed our algorithm in Matlab and ported it to C++ using the OpenCV [313] framework. The native OpenCV code can also be executed on iOS and Android with little modifications. As speed is crucial in mobile barcode scanning, we compare our method to three others (Gallo2011 [63], Tekin2012 [230], Katona2013 [105]) that represent the state of the art and allow real-time operation on smartphones. To only compare the quality of the saliency maps of the different algorithms, we apply Gallo's method (described in Section 2.2) in each algorithm to generate a bounding box from the saliency map except for Tekin2012 that already returns a single scanline.

2.4.2 Test environment

Mobile barcode scanner applications read video preview images with high frame rate rather than still images to speed up the scanning. Recent smartphone cameras are able to

deliver preview frames in 720×960 or 720×1280 (HD) resolution, so we focus on this image size in all our tests.

We test the 1D performance on the Muenster BarcodeDB[1] data set by Wachenfeld et al. [234]. We did not include their localization approach in our comparison because they assume the user already positioned the code to the center of the image, which is too restrictive for our purposes. The algorithms we test do not need to make this assumption. The 1050 images were taken with a Nokia N95 phone with autofocus (AF). Additionally, we also recorded 200 blurry images with an iPhone 5 with its AF turned off.

For our tests with 2D codes, we use the QR code dataset of Dubská et al.[2] which consists of about 400 images with perspective distortions, illumination variations, blur, and surrounding text. The images were taken by a mobile phone in (high-resolution) photo mode. We also recorded 120 new images of QR codes with an iPhone 4S in video mode with and without AF. The images contain codes in various scale, orientation and blurriness. Our images are also publicly available[3].

We hand-labeled the code corners in all images and stored the bounding boxes as ground truth. We compared the accuracy of the four algorithms by measuring the overlap of their output with the ground truth. We calculate the Jaccard coefficient $J(A,B) = |A \cap B| / |A \cup B|$ as the overlap measure where A is the of the ground truth bounding box and B is the detected bounding box. A coefficient above 0.5 represents a visually good match when the returned bounding box is bigger than the ground truth, but actually a smaller coefficient would also be acceptable as decoders can deal with codes that cover one third of the search window. When the returned bounding box is smaller than the ground truth we accept only 5% loss. Because the algorithm of Tekin returns a single scanline instead of a bounding box, we calculate its Jaccard coefficient in one dimension along the code axis.

2.4.3 1D code localization performance

In our first test, we took 1000 original sharp images from the Muenster dataset and looked at the average Jaccard coffecient of the four algorithms. The results are summarized in Table 2.1. On sharp images, we achieved an average bounding box overlap $\bar{J}_{ours} = 0.665$ with standard deviation 0.278, so an average overlap well above the 50% requirement while the standard deviation is also lower than that of Gallo2011. Overall, our algorithm accurately detects 82.5% of the barcodes out of 1000 images. False positives are mostly caused by a very dominant sharp edge in the background or selecting the wrong code

[1] http://cvpr.uni-muenster.de/research/barcode/
[2] http://medusa.fit.vutbr.cz/pclines/?p=86
[3] http://people.inf.ethz.ch/soeroesg/

among multiple ones (in case of multiple codes in the image, we always labeled the one which is closest to the image center). False negatives are caused by colorful codes (rejected by the saturation threshold) and non-rectangular (skewed) codes. Our algorithm detected more blurry codes accurately than the other algorithms. Example images are shown in Figure 2.7.

Figure 2.7: Positive and negative examples from the Muenster dataset. A green rectangle indicates positive detection ($J \geq 0.5$), a red rectangle indicates false detection ($J < 0.5$), blue rectangles represent the hand-clicked ground truth. Bottom row: The failures are caused by (1) parallel lines in the background, (2) glare, (3) dominant text, (4) object lines (5), colored code.

It is important to note that on sharp images, we achieve about the same detection rate as the other algorithms but we make no assumptions about code size (like Katona2013) nor code orientation (like Gallo2011) nor code position (like Wachenfeld2010). Although Katona2013 [105] reports over 90% detection rate in simulations, the method performs weakly in our tests. The method almost always returns a bounding box that is too big and contains text and other objects around the code. Gallo2011 is fast and works well even with fairly blurry codes, but often returns a too small bounding box because a blurry code breaks into many parts. Its high detection rate is based on the fact that most codes in the data set are horizontal. Tekin2012 works with multiple orientations, however, on blurry images it tends to give too short scanlines and returns many false positives.

Algorithm	Gallo	Tekin	Katona	Ours
Average of $\overline{J}$	70.89%	81.22%	19.94%	66.47%
Std. dev. of $\overline{J}$	35.42%	25.62%	11.56%	27.77%
Success rate ($J \geq 0.5$)	79.45%	88.04%	2.34%	82.5%

Table 2.1: Average Jaccard coefficients on 1000 sharp images. We defined every $J \geq 0.5$ a good match. (Recall that for Tekin2012 J is calculated only in 1D therefore results in better overlap).

In our second test, we investigated the robustness of the algorithm against Gaussian (defocus) blur. We added artificial Gaussian blur to the original images with standard deviation $\sigma = 1, 2, 5, 7, 9, 11, 13$ pixels. Our results are summarized in Figure 2.8.

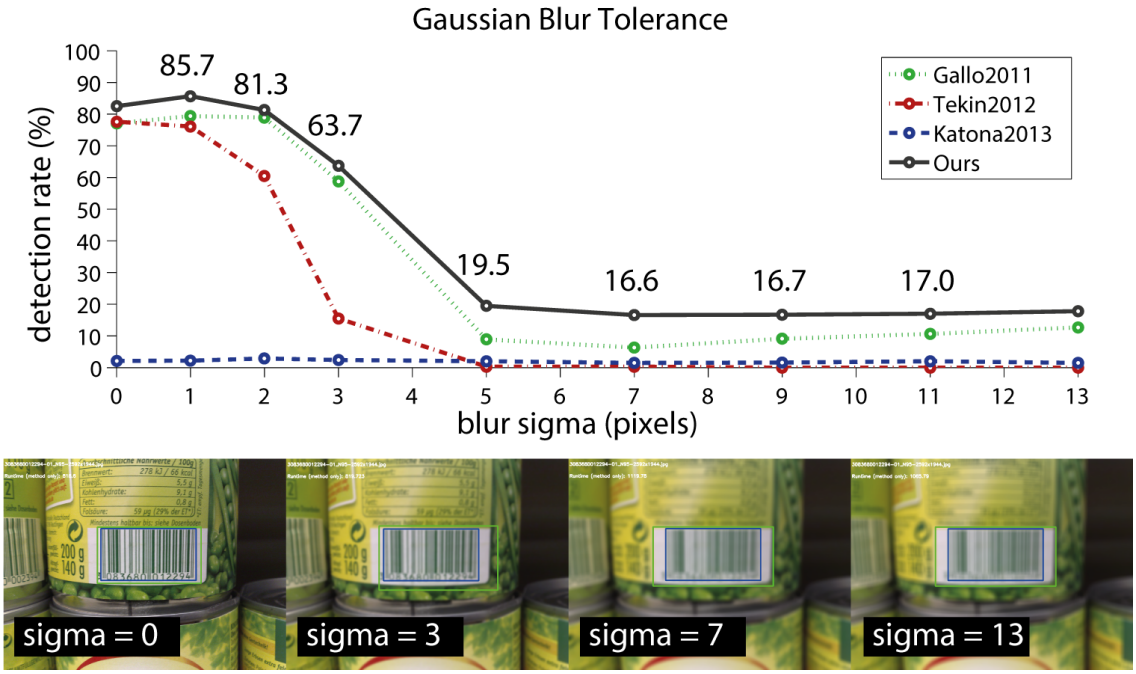


Figure 2.8: Successful 1D detection rates ($J \geq 0.5$) on 1000 images from the Muenster data set with various degrees of artificial Gaussian blur. The bottom row gives an impression on the effect of the blur parameter sigma.

In our third test, we added artificial motion blur to the 1000 sharp images. We chose motion blurs with length $3, 5, 7, 9, 11, 13, 15$ pixels at 135° orientation to make sure we destroy some of the bars. Figure 2.10 shows that our algorithm is more robust to motion blur than the others.

The detection rates on the images we recorded with real defocus and motion blur are 5.2% (Gallo2011), 0.0% (Tekin2012), 8.1% (Katona2013), and 29.8% (ours). Example images are shown in Figure 2.9. Our method has difficulties with blurry codes without

Figure 2.9: Positive and negative examples from our no-AF dataset. Left: Our algorithm can localize blurry codes with rectangular white support. Right: difficult cases with all-gray background and sharp background (no codes found).

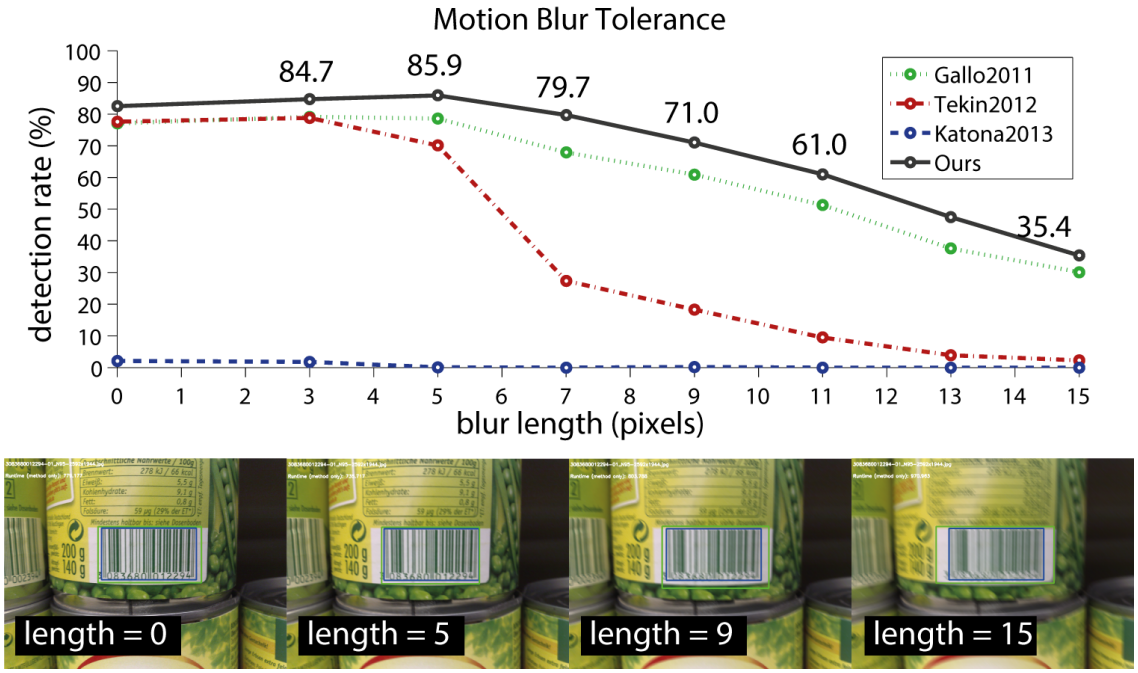


Figure 2.10: Successful 1D detection rates ($J \geq 0.5$) on 1000 images from the Muenster data set with various degrees of artificial motion blur. The bottom row gives an impression on the effect of the blur length.

a rectangular frame and sometimes sharp gray stripes in the background are found more likely to be a barcode than the actual blurry one. We can still conclude that our algorithm outperforms the previous 1D localization approaches in blur resistance while it is also scale and rotation invariant and can detect 2D codes as well.

2.4.4 2D code localization performance

We tested the 2D performance of the algorithm on the QR code dataset of Dubská et al. While their line detection algorithm performs well on big sharp codes of the dataset, our algorithm returns oversized bounding boxes also including the surrounding text. However, on the motion blurred images of the dataset, the corner measure is more reliable than line detection (see Figure 2.11). Due to our oversized bounding boxes, we achieve an overall detection ratio of only 42.6%.

On our QR data set with real sharp, defocused, and motion blurred images, our algorithm can reliably detect 81% of the codes. However, if a code is rotated close to 45°, our simple axis-aligned bounding box detection tends to return a bounding box that is too small because it reaches the code border too early in the barcodeness map. Examples are shown in Figure 2.12

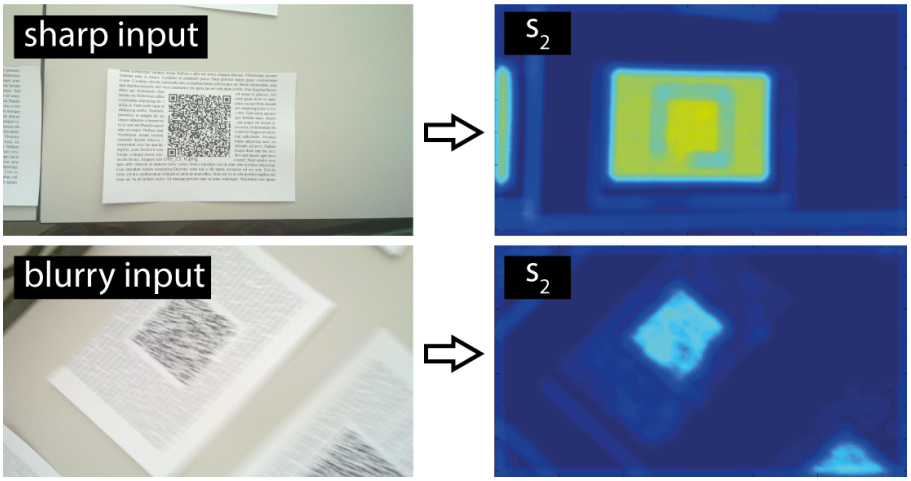


Figure 2.11: Sharp text around the QR code misleads our algorithm and it returns an oversized bounding box. In motion blurred images the text disappears but the high corner density in the code area makes it still clearly distinguishable. Input images are from Dubská et al. [49]

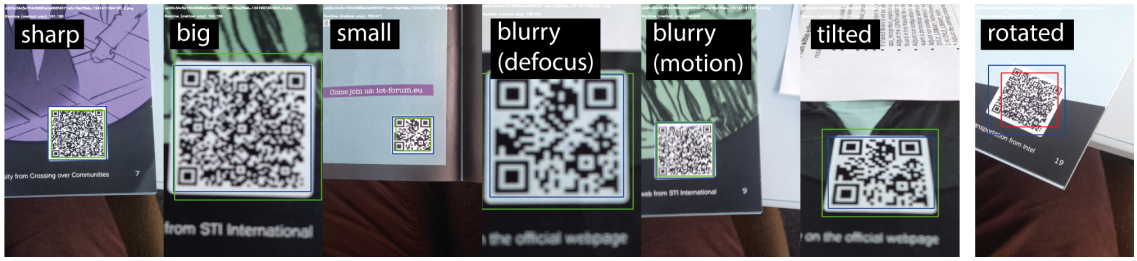


Figure 2.12: Positive and negative examples from our QR code dataset. Note that the misalignment on the right is caused by the simple axis-aligned bounding box detection step. The code area would be still clearly distinguishable in the s_2 map.

2.4.5 Multi code performance

All described methods focus on localizing only one code in the image but they all build a barcodeness map in an intermediate step. Therefore, they all could be extended to localize multiple codes and provide a list of ranked candidates. Figure 2.13 shows two examples with multiple 1D and 2D codes together with our barcodeness maps. If both code types are present in the image, the linear combination of m_1 and m_2 causes crosstalk in s_1 and s_2 which may lead to false classifications. Note that the codes are still visible but the code type might be uncertain. We leave the detailed analysis of this effect for future work.

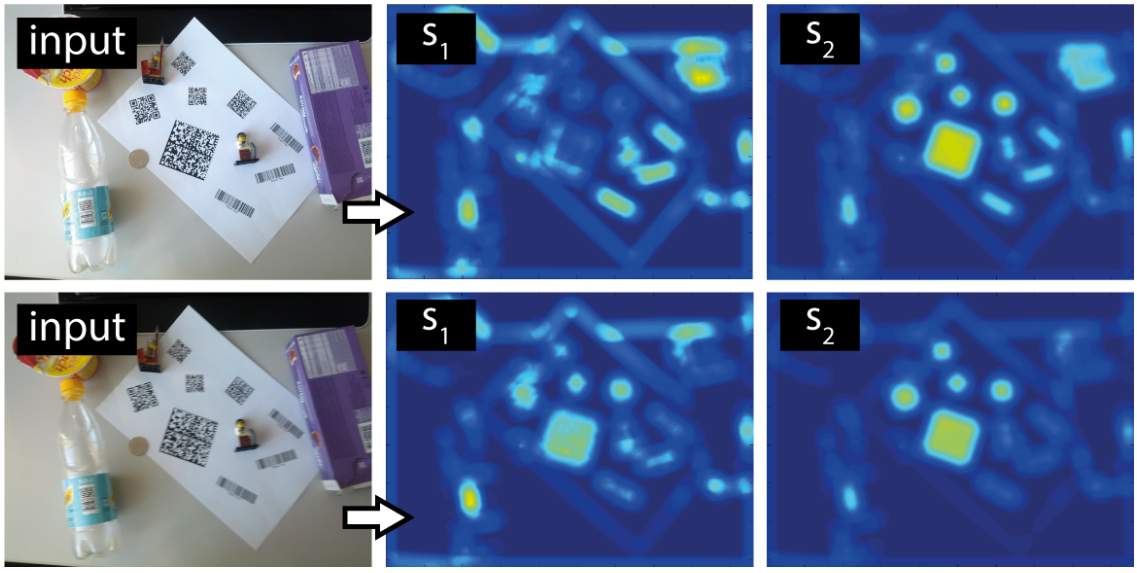


Figure 2.13: Multiple 1D and 2D codes in a sharp image (top row) and in a blurry image (bottom row). Note that the measures s_1 and s_2 clearly show the barcode areas in both the sharp and blurry images. Also note that the characters on the chocolate box are so small that the text lines are detected as a barcode candidate.

2.4.6 Other symbologies

Our method can be easily extended for localization of other 1D and 2D symbologies that mainly consist of black and white line and corner structures. As the method is also fairly robust to blur, we can employ it for detecting noise-like 2D patterns such as the FOCUS codes [81] for communication. The gray noise-like pattern of the code generates a dominant m_2 signal. Figure 2.14 shows a couple of examples with FOCUS codes.

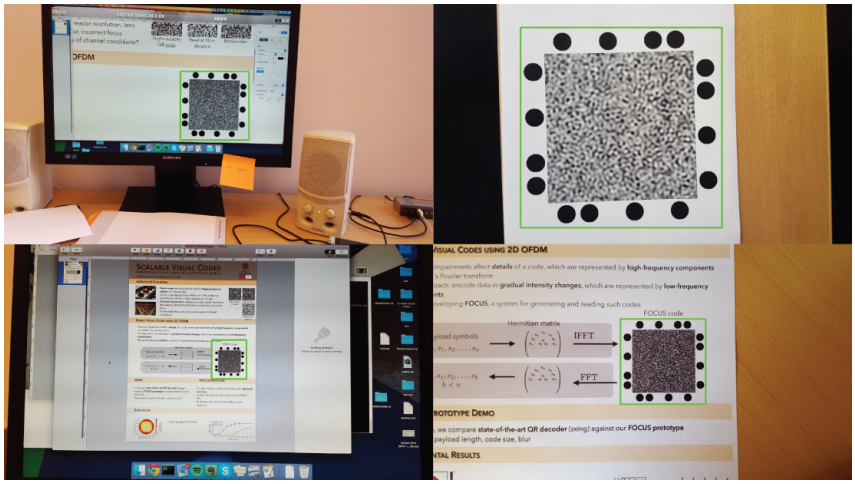


Figure 2.14: Our method can be easily extended to detect other visual codes. We detect noise-like FOCUS codes [81] in these examples.

2.4.7 Discussion

In our 1D tests, we achieve similar sharp detection rates as the other approaches but without their limiting assumptions, while the algorithm also detects close to 17% of the blurry codes. This means in 17% of the cases a blurry decoder can already start decoding the barcode even before the AF was triggered leading to a better user experience.

The average runtime of our algorithm with images of resolution 720×960 pixels on a notebook PC (Intel Core i7 M620 CPU with 2.67 GHz clock frequency) is 73 ms (see Table 2.2) that adds up as follows: HSV conversion (19.90 ms); gradient calculation (9.19 ms); calculation of C_{ij} (14.32 ms); calculation of saliency (6.19 ms); box filtering (7.51 ms); bounding box detection/rectangle detection (15.87 ms). We can clearly see that the most time is spent on color space conversion. Without the HSV information, the algorithm still finds sharp codes but it becomes more sensitive to clutter. Thanks to the saturation information, our algorithm returns significantly less false positives than the others in our tests. It is important to note that current smartphone models are also

equipped with a graphical processing unit, and not just color conversion but every step of our algorithm can be implemented as OpenGL ES fragment shaders. We achieved a speedup of factor 3.2× by calculating the barcode probability maps on the mobile GPU. We present the implementation details on the GPU in the next section.

Algorithm	Gallo2011	Tekin2012	Katona2013	Ours
Runtime PC	27 ms	49 ms	63 ms	73 ms
Runtime phone	85 ms	26 ms†	173 ms	380 ms/118 ms‡

Table 2.2: Average runtime of the four algorithms on PC (960×720 pixels) and on a smartphone (640×480 pixels). † Tekin is optimized C++ code, the other algorithms are written in OpenCV. Tekin is slower in our PC simulations because of image format conversions. ‡CPU only / calculation of s_1 and s_2 on GPU and reading back to CPU. All phone measurements were taken with a Samsung Galaxy Nexus (2011-model).

While our algorithm is fully orientation invariant, it is not fully scale invariant. The size of the box filter is chosen to connect barcode areas of a fairly wide scale range, but it would not work with tiny codes (would include clutter) or huge codes (would break apart).

Recall that for comparison, we applied Gallo's method to generate a bounding box from the saliency map, a method that assumes axis-aligned barcodes in the image. While also rotated barcodes are well visible in our saliency maps, the final bounding boxes are sometimes too small ($J < 0.5$). A more sophisticated bounding box algorithm should significantly improve our detection rates.

As mentioned before, our algorithm consists of per-pixel operation and is hence well suited for parallelization. In the next section, we show how to implement the algorithm on mobile graphics hardware to make it even faster.

2.5 Fast implementation on mobile GPUs

2.5.1 Overview

This section presents the implementation steps of the above barcode localization algorithm on embedded graphics hardware. By reformulating the distinct steps as fragment shader programs, one can compute the barcode probability maps directly on the graphics hardware which brings significant speedup (e.g., of factor 3.2 in our tests with 4 shader cores).

The presented GPU-assisted approach allows real-time operation without the limiting assumptions of the concurrent methods.

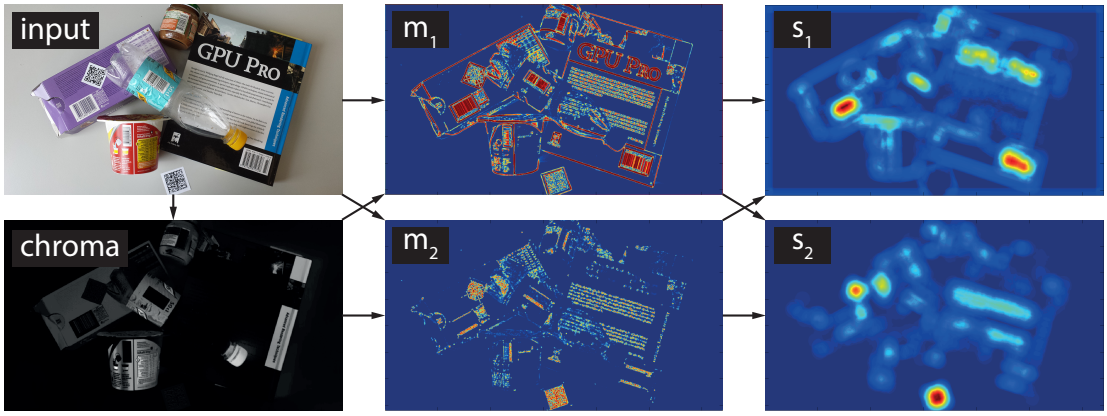


Figure 2.15: Algorithm outline: input image and its chroma map, m_1: edge density map, m_2: corner density map, s_1: 1D saliency, s_2: 2D saliency. The color coding is normalized to the range $[0,1]$.

We shortly repeat and illustrate the necessary stages of our barcode localization algorithm in Figure 2.15. The captured color camera image is first converted to grayscale and the chroma value $c = \max(R,G,B) - \min(R,G,B)$ of each pixel is also stored separately for later steps. Because 1D barcodes contain many edges and 2D codes contain many corners, the algorithm searches for areas with high concentration of edge structures as well as for areas with high concentration of corner structures. Assuming that the codes are printed black and white, the chroma mask can be used as a fast test to reject no-code areas in an early stage of the pipeline. The next step is to calculate the elements of the structure matrix $M = \begin{pmatrix} C_{xx} & C_{xy} \\ C_{xy} & C_{yy} \end{pmatrix}$ for each pixel. Next, an edge map m_1 and a corner map m_2 are derived from the structure matrix. The two maps are then blurred with a big block filter to connect areas where there is high line density or high corner density and to remove background clutter. Finally, the two blurred maps are linearly combined to give the 1D and 2D barcode probability maps s_1 and s_2, respectively. Note that all steps from color conversion through filtering to linear combination are data-parallel operations that can be efficiently implemented in fragment shader programs.

2.5.2 Image processing on mobile GPUs

OpenGL ES [314] is the de facto standard software-hardware interface for rendering 3D graphics on embedded devices. Version 2.0 of OpenGL ES introduced the programmable graphics pipeline to mobile graphics hardware opening the doors for general purpose GPU

computations on smartphones. There are certain limitations though compared to proprietary high-level parallel computing frameworks such as CUDA [280], OpenCL [312], or RenderScript [328] because OpenGL was primarily designed for rendering 3D scenes. This means one has to reformulate computing problems as rendering problems and carefully tune the algorithms to graphics features. Nevertheless, OpenGL ES provides portability across all smartphone platforms and version 2.0 is widely available already in middle-class consumer devices.

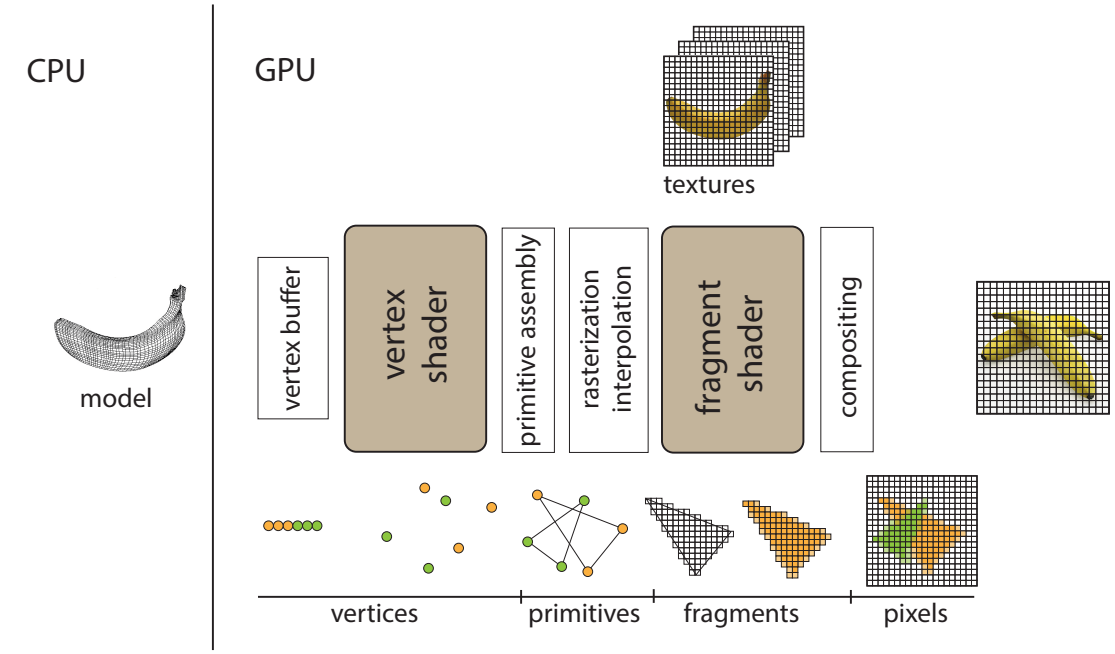


Figure 2.16: The OpenGL ES 2.0 rendering pipeline.

The programmable graphics pipeline consists of several steps as shown in Figure 2.16. The 3D scene is defined as a set of triangles formed by vertices. The vertices have various attributes such as 3D coordinates, normal vectors, primary color, texture coordinates, etc., that are passed to a programmable SIMD processor called vertex shader. The vertex shader program calculates the screen coordinates of each vertex and passes those to the primitive assembly unit that determines the points, lines, and triangles to be rendered on the screen. The rasterizer decomposes the primitives into individual fragments and for each fragment it interpolates the attributes from those of the corresponding vertices. Each fragment is passed to another programmable SIMD processor called fragment shader which calculates the color of the fragment based on the attributes and optional texture inputs. Textures are general 2D data containers with hardware-accelerated interpolation. The individual fragments run through several tests until they are finally combined in a pixel of the frame buffer. When all primitives are processed, the content of the frame buffer gets presented on the screen. OpenGL ES 2.0 also features off-screen render targets called frame buffer objects (FBOs). Using an FBO, the output can be directed into a texture.

Image filtering can be reformulated as a multi-pass rendering problem in the following setting (see Figure 2.17): The input image is stored in a texture and an output FBO is set up to have the same size as the input texture (1-to-1 mapping between texels and pixels). The 3D scene consists of only two triangles (a quad) that together cover the whole screen (or FBO). The virtual camera looks fronto-parallel to this quad and orthographic projection is applied. Once the scene is drawn, the full-screen quad generates a single fragment for each output pixel. To determine the color value of a fragment, the fragment shader program is run which contains the actual image processing routine. The fragment shader can read from any input texture position but can write its output only to the fragment index it is currently assigned to. Therefore, different shaders (filters) are loaded in subsequent rendering passes and the role of the input and output textures is always exchanged. This way a whole chain of image filters can be realized in the GPU. The resulting image is presented on the screen or read back to the CPU for further non-parallel processing. For further reading on GPU computing, we refer to the excellent textbooks GPU Gems [152] and GPU Pro [50].

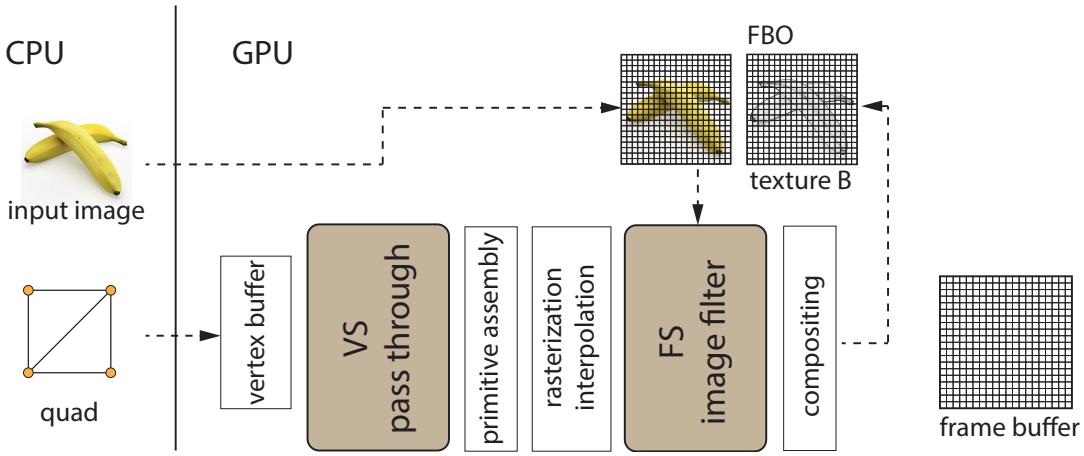


Figure 2.17: Image filtering can be reformulated as a rendering task using textures for input as well as for output.

2.5.3 Constraints on mobile platforms

OpenGL ES is a constrained subset of its desktop counterpart and hence additional considerations need to be taken in our algorithms when targeting mobile devices. Mobile GPUs have significantly less memory and lower clock speed to reduce energy consumption. Also, transferring data from GPU to CPU and back should be avoided during the algorithm due to low memory bandwidth. The texture fetch latency is significant so the number of texture reads should be minimized. The number of texture units varies but is at least 8

while the maximum texture size is 2048. Although the shaders can perform floating-point calculations, the results must be stored in low-precision fixed-point textures and there is only one color render target (with 4 color channels, R, G, B, and A). However, for practical applications the wide availability compensates for the API's limitations. Next, we present how our localization algorithm can be reformulated and optimized for the mobile GPU. We target a wider range of devices so we apply only OpenGL ES 2.0 features with one extension that allows copying the camera image directly to the GPU memory bypassing the CPU. One has to keep in mind that shader programs are submitted as a string literals and compiled by the graphics driver prior execution and this may lead to slightly different behaviour on different hardware platforms.

2.5.4 Data formats

The inputs and outputs of the subsequent stages are stored in 32-bit RGBA textures. The colors are represented as 8-bit fixed-point values between 0 and 1. Each stage consists of one or two render passes in which a simple quad gets textured with the result of the previous step. Texture filtering is set to `NEAREST` to sample one pixel only and coordinate wrapping is set to `CLAMP_TO_EDGE`. We use non-power-of-two textures which prohibits mipmapping. Texture coordinates are measured in the $[0, 1]$ range so the offset between neighboring texels (1.0/width) has to be supplied to all the shaders as a uniform variable. Calculating the coordinates of the neighbors can be shifted to the vertex shader letting the hardware interpolate them for all the fragments.

2.5.5 Streaming camera images to the GPU

The smartphone camera captures the preview frames in NV21 (YUV) format but the textures must be stored in RGBA format. The GLES2 extension `EGL_image_external` allows to bind an external image buffer to a texture unit. The stream output of the camera can be redirected to this image buffer and OpenGL automatically converts its content to RGBA at reading time. This way, the manual conversion of camera frames can be saved and the frames do not need to be loaded into the CPU.

2.5.6 Calculating the structure matrix

In the first stage of the algorithm, the elements of the structure matrix $\sum_D I_x^2$, $\sum_D I_y^2$, and $\sum_D I_x I_y$ need to be calculated. The summation over the small neighborhood D can be

postponed to the following step so that the calculations for each pixel can be decoupled and run in parallel. Gray conversion is easily performed as a dot product of the input RGB vector with the constant vector $[0.3, 0.59, 0.11]$ while the chroma calculation involves only built-in maximum and minimum functions.

We calculate the gradients using the *derivative5* filters of Farid et al. [57] that improve the robustness to barcode orientation. This filter gives an optimal approximation of the gradient with regards to gradient oriantation, i.e., it produces more accurate estimate than the usual $[-1, 1]$ and $[-1, 1]^\intercal$ filters when the gradient is not axis aligned. The *derivative5x* consists of a 5-tap horizontal component d_1 and a 5-tap vertical component p while in *derivative5y* these two components change their roles. Thanks to the commutativity of convolution, we can perform all horizontal filtering in one pass and all vertical filtering in the second pass which requires only 5 texture fetches in each pass: $I_x = I \otimes d_1 \otimes p^T$ while $I_y = I \otimes d_1^T \otimes p = I \otimes p \otimes d_1^T$. In the first pass, we take 5 gray pixels horizontally and multiply-sum them once with d_1 and store the result in R, and once with p and store the result in G. In the second pass, we filter R with p^T and G with d_1^T vertically. The squares I_x^2, I_y^2, and $I_x I_y$ are then stored in the R,G,B channels of the output while the chroma value is always passed in the A channel.

To remove colorful non-barcode areas in this stage, we set all entries in M to zero if the chroma of the pixel is over 0.25. Conditional statements in shaders decrease performance, so we formulate the thresholding without a condition using the built-in *step* function. The code line $c < 0.25?A = 1.0 : A = 0.0;$ is equivalent to $A = 1.0 - step(0.25, c);$

2.5.7 Fast Gaussian filtering

In the second stage, the derivatives are summed up in a small neighborhood to get the final entries of the structure matrix. We apply a 7x7 Gaussian window with standard deviation 2. The 2D Gaussian is the outer product of two 1D Gaussians, so again, the 2D filter can be substituted by a horizontal plus a vertical 1D convolution. The two passes require only $7 + 7 = 14$ texture fetches instead of $7 \times 7 = 49$. Furthermore, the GPU works parallel on the RGB channels of the input so the three different entries of the structure matrix are calculated at the same time.

2.5.8 Edge and corner maps

In the third stage, the "edgeness" m_1 and the "cornerness" m_2 maps are generated over the image based on the entries of the structure matrix which involves only pixel-wise

arithmetic operations. Intermediate steps of the algorithm are shown in Figure 2.18.

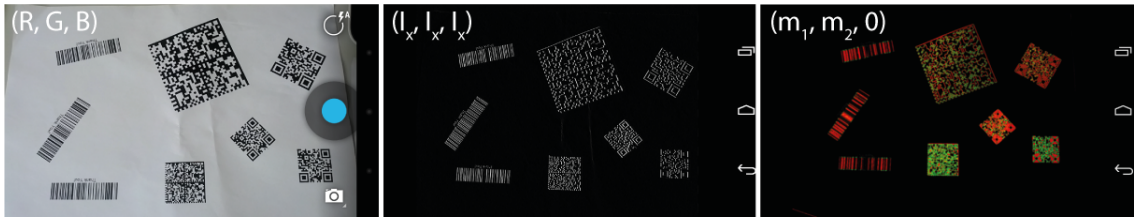


Figure 2.18: Intermediate results rendered on the screen: original image; x-derivatives; edges (red) and corners (green)

2.5.9 Box filtering and saliency maps

Box filtering with a large kernel would require a great number of texture fetches for each fragment. However, if we set the texture filtering to `LINEAR` and sample between the pixels, thanks to hardware interpolation we get the average of the two values in one texture fetch. We use a horizontal and a vertical 32-tap box filter with only 16 texture fetches each between texels (i.e., in total 32 instead of 32^2). In the future, we plan to implement box filtering via hardware-accelerated mipmaps.

In the final stage, the blurred m_1 and m_2 maps are combined to 1D and 2D barcodeness maps, respectively. Linear combination of fragment values can also be done very efficiently in GPU using the *mix* command. Finally, the barcode probability maps are transferred back to the CPU where barcode region candidates are selected. This step is identical to [63, 209].

2.5.10 Running on wearable devices

First, we tested our algorithm on three smartphones with different hardware: a *Galaxy Nexus* (1.2 GHz dual-core ARM Cortex-A9 CPU, Imagination PowerVR SGX540 GPU), a *Galaxy S3* (1.4 GHz quad-core ARM Cortex-A9 CPU and ARM Mali-400 MP4 GPU) and a *Galaxy S4* (1.9 GHz quad-core Qualcomm Krait 300 CPU and Qualcomm Adreno 320 GPU). The algorithm has been implemented using OpenCV 2.4.6 (native C++) for the CPU and using OpenGL ES 2.0 for the GPU. Table 2.3 summarizes the runtimes from image capture to the barcode maps including GPU-CPU transfer. The GPU implementation achieves a speedup of factor 3.2 on the Nexus phone with four shader cores and factor 2.0 on the S3 which we believe has only two shader cores. The S4 allows 6 frames per second even in HD resolution. In that case, the 152 ms(‡) runtime adds up as follows: derivatives

and color conversion (22 ms), Gaussian blur (23 ms), edge/corner maps (7 ms), box filter (41 ms), reading from GPU (20 ms), and rendering to the screen (39 ms). We conclude that OpenGL ES 2.0 brings a significant speedup to data-parallel image processing algorithms and offers code portability across a wide range of hardware platforms.

Frame size	Device	Galaxy Nexus (2011)	Galaxy S3 (2012)	Galaxy S4 (2013)
640×480 307200 pixels (X)	CPU CPU GPU	380 ms 421 ms 118 ms (3.22×)	291 ms 308 ms 144 ms (2.02×)	n.a.† 299 ms 49 ms (6.10×)
960×720 691200 pixels ($2.25X$)	CPU CPU GPU	797 ms 970 ms 259 ms (3.08×)	634 ms 652 ms 322 ms (1.96×)	n.a.† 570 ms 104 ms (5.48×)
1280×720 921600 pixels ($3.0X$)	CPU CPU GPU	1127 ms 1434 ms 349 ms (3.23×)	842 ms 968 ms 414 ms (2.03×)	n.a.† 822 ms 152 ms (5.41×)‡

Table 2.3: Average runtime of the GPU implementation and speedup compared to the CPU implementation on different smartphone models from image capture to the barcode maps including GPU-CPU transfer. The first CPU line is with using OpenCV's native camera, the second CPU line is with using OpenCV's Java camera † OpenCV 2.4.6 does not support the S4 camera.

The approach presented here can be easily extended to run on any OpenGL ES-compliant wearable computer with little modifications Figure 2.19 shows the same algorithm running on a Samsung Gear v1 smartwatch and on Google Glass v1 smartglasses. Also, while the number of CPU cores in wearable devices is increasing slowly, the number of GPU cores is expected to increase much faster in the future which means our algorithm becomes faster with time without modifying the code. For instance, we repeated the experiment with a 2015-model Motorola Nexus 6 smartphone (Qualcomm 2.7 GHz quad-core Krait 450 CPU, Adreno 420 GPU with 128 shader cores) using 1280×720 resolution, and achieved 45.05 ms runtime including readback which corresponds to a processing frequency of about 22 frames per second. Furthermore, the GPU with more cores at lower frequency requires much less energy than performing the same calculations on the CPU as heat dissipation increases super-linearly with clock frequency, so it is beneficial for battery-powered devices to outsource data-intensive computations to the graphics hardware.

Figure 2.19: Parallelization in OpenGL allows portability across a wide range of devices including wearable computers such as the Google Glass (left) or the Samsung Galaxy Gear watch (right).

2.6 Conclusions

We have presented a combined 1D and 2D barcode localization algorithm that addresses orientation, scale, and symbology invariance and is also more robust to blur than previous methods. We compared the speed and accuracy of our approach to three recent localization algorithms on publicly available datasets and achieved higher detection ratio even with defocused images where previous methods perform weakly. Taking our algorithm as a preprocessing step, blurry barcode scanning can be extended to full-resolution images in the future. Extending the simple bounding box detection approach for instance using mean shift segmentation immediately allows for detecting multiple codes in an image.

Also, we collected a new public dataset that contains hundreds of challenging barcodes and QR codes recorded by smartphone cameras to facilitate further research in the area. The images are degraded by various types of real blur, noise, shadows, glare, and scratches that represent the problems in real-world scanning situations.

We have presented the implementation, optimization, and evaluation of our barcode localization algorithm on embedded GPU using the widely adopted OpenGL ES 2.0. The GPU code is portable between all existing smartphone platforms (we tested on PowerVR, Mali, Adreno), can run even in middle-class smartphone models and even on a first-generation smartwatch and a smartglass. We carried out several optimizations in the processing pipeline including streaming camera images directly to texture memory, resampling on graphics hardware, reshuffling vertical and horizontal filter operations, applying separable filters to reduce the number the texture fetches, reading multiple texels simultaneously by leveraging hardware interpolation, and creating branch-free code using the built-in step function.

We expect further optimizations in the future once the OpenGL ES 3.0 [314] standard

becomes widely available in embedded graphics hardware. The new standard brings several additional features compared to v2.0, including multiple render targets, GPU-CPU synchronization, and floating point textures that allow more sophisticated image processing with less rendering passes. Furthermore, GPU-CPU parallelism would speed up barcode scanning even more; instead of simply waiting for the results, the CPU can process the barcode candidates of the previous frame. The OpenGL ES 3.1 standard further introduces the compute shaders. New, general purpose GPU programming APIs are also becoming available in high-end smart devices. The Open Computing Language (OpenCL) [312] API (which is unfortunately not available on Android) introduces a new programming model that is more suitable for computations on heterogenous processors. The tasks are executed transparently either on the CPU, the GPU, or an embedded DSP[4]. Also, the Vulkan [345] combined compute and graphics API is under standardization. It is considered a ground-up redesign of the OpenGL standard with more direct access to the GPU and unified programming model for desktop and mobile (embedded) applications. We leave the implementation of our algorithm using the new APIs for future work.

[4]DSP: digital signal processor

Chapter 3

Motion blur compensation

This chapter presents a fast blur removal algorithm that allows the scanning of motion-blurred QR codes. The blur removal process is tailored to the special properties of QR codes images, and also makes use of the built-in inertial sensors of wearable devices. Most parts of the methods and results presented in this chapter have been published in [210, 212].

3.1 Introduction

Current mobile barcode scanning solutions work well only if the camera is held still and close to the scanned object. Commercial scanner engines by companies such as RedLaser [327] and Scandit [331] can successfully compensate for *defocus and lens blur* and can decode also blurry codes. This makes barcode scanning possible on devices without autofocus and accelerates the scanning on devices with autofocus. The existing solutions, however, still cannot compensate for more complex blurs like *motion blur* (caused by the slight relative movements between the camera and the scanned object), *zoom blur* (caused by the autofocus), or *shake blur* (caused by the user's hand). Unfortunately, with wearable cameras, slight motion during scanning is almost unavoidable which can easily render the captured codes unreadable, greatly deteriorating the user experience and utility (see Figure 1.15). Although the various blur effects can be modeled in a similar way mathematically, motion blur is a far more complex problem than the others.

High-end cameras reduce motion blur with optical image stabilization, where motion sensors control mechanical actuators that shift the sensor or the lens during exposure to compensate for motions of the camera. Unfortunately, image stabilization neither prevents

motion blur caused by the movements of the object, nor extreme translational movements of the camera, nor rotation around its optical axis. We expect the spread of wearable cameras with hardware image stabilization in the future, but the mentioned limitations demand software-based enhancements.

The problem of removing blur from photographs has been widely studied in the past. However, existing algorithms typically fail on artificial black and white visual tags because they look very different from natural images. We will show concrete examples of this problem later in this chapter. Furthermore, the existing restoration algorithms are computationally too demanding to be carried out on mobile devices – even if they make strong assumptions about the image type. We are not aware of any published method that achieves near real time blur removal performance (even on PCs). However, the special structure of visual codes compared to general photographs allows for optimizations in terms of restoration speed and, to a lesser extent, quality.

In this chapter, we present an algorithm that can robustly decode QR codes degraded by motion blur. We exploit the fact that QR codes do not need to be visually pleasing for decoding and propose a fast restoration-recognition loop that removes blur using the special structure of QR codes. In our optimization scheme, we interweave blind blur estimation from the edges of the code and image restoration regularized with typical properties of QR code images. We show in experiments with synthetic and real blurred images that our proposed restoration algorithm is on par with the state of the art in quality while it is about a magnitude faster. In addition, we propose to combine blur estimation from image edges with blur estimation from built-in inertial sensors to make the restoration even faster.

3.2 Related work

Restoring images degraded by blur has been an active research area for a long time because of its military, forensic, and astronomical applications. The topic gained considerable interest again in the past decade due to the large demand of improving photographs captured by consumer point-and-shoot cameras and smartphones. In the recent years, a large number of algorithms (algorithm families) have been proposed for restoring blurred images [171]. In this section, we give a brief overview of the existing methods and summarize what we can learn from them for the efficient restoration of motion-blurred visual codes.

3.2.1 Blur removal in general

Deconvolution The formation of a blurred image B can be modeled by a convolution of a sharp image I with a blur kernel k as also illustrated in Figure 3.1, therefore blur removal is also termed deconvolution in the literature. Formally,

$$B = k * I + N \tag{3.1}$$

where the convolution of two-dimensional signals f and g is defined as

$$f(x,y) * g(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i,j) g(x-i, y-j) \tag{3.2}$$

We distinguish between non-blind deconvolution, where the blur kernel is known in advance, and blind deconvolution, where the blur kernel is unknown and needs to be estimated first.

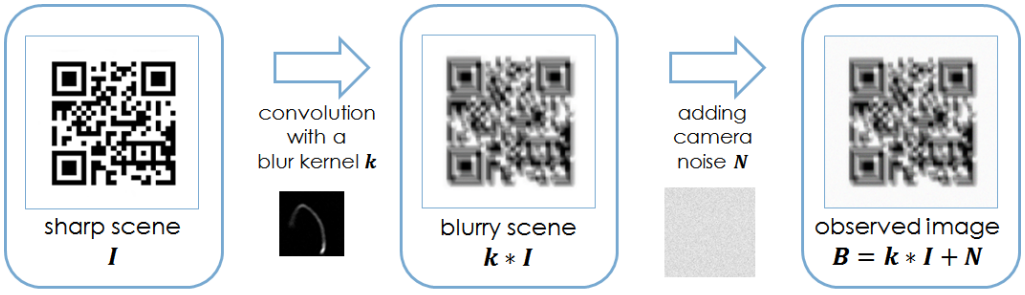


Figure 3.1: Blurry image formation modeled by convolution.

The blur kernel can either be measured in some (very limited) settings, or it can be estimated for instance from salient edges in the image [35, 97, 219, 254], from the frequency or cepstral domain [66, 162], from an auxiliary or modified camera [37, 225, 256], or from motion sensors [96, 164]. Unfortunately, an infinite number of combinations from latent sharp images and blurs may result in the same blurry image (see Figure 3.2), which makes blind deconvolution and blur kernel estimation very hard. Even known blur is difficult to invert because many false images can also satisfy the equations. Intuitively, non-blind deconvolution can be expressed by the equation $B = k * ? + N$, and blind deconvolution corresponds to $B = ? * ? + N$, which is very under-determined. Despite the significant progress in this research area, a method that inverts the effect of arbitrary blur is not known.

Blurry barcode scanning is a blind deconvolution problem since the blur kernel needs to be estimated first. We target unmodified wearable devices, which limits our available

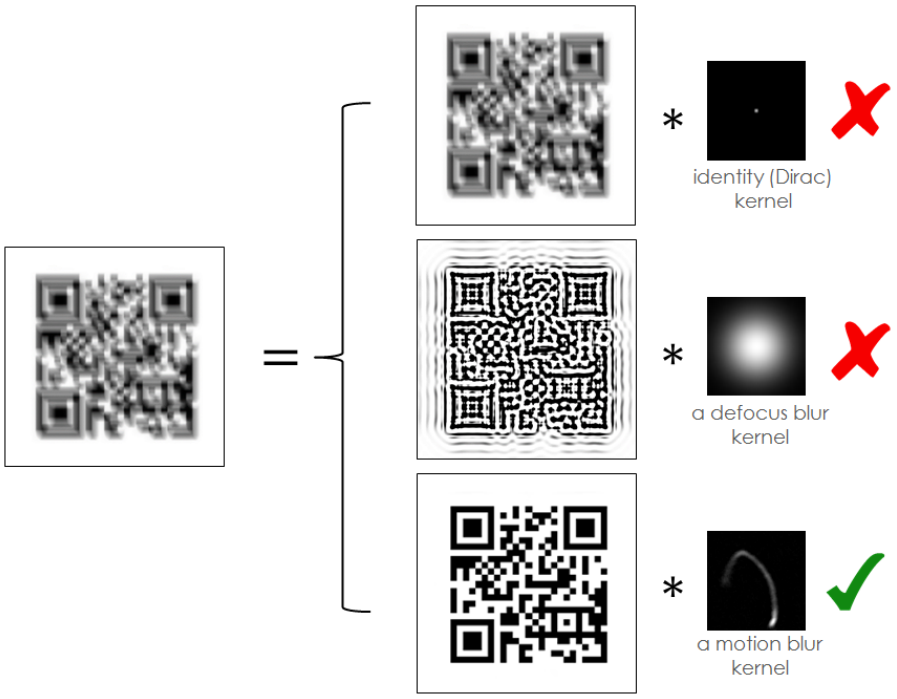


Figure 3.2: The ambiguity in blind deconvolution: many blur kernels and latent images can result in the same blurred image, so finding the right solution is hard. Figure adapted from Robert Fergus [58].

computational resources, but we can also benefit from the specific setting of barcode scanning. Our images always contain a large number of black and white edges, so our attention is on the family of edge-based methods, and only the fastest of those. Also, we can exploit the fact that most of today's wearable devices possess built-in motion sensors, and these sensors can hint at the expected motion blur in the image. Before coming to these two method families, we quickly summarize the whole landscape of blur removal.

Uniform vs. non-uniform blur Both defocus blur and motion blur can be modeled as a convolution with a *uniform* (shift-invariant) blur kernel, under the assumptions of a planar fronto-parallel scene and only translational motions. Defocus blur is caused by the fact that the image of a single object point not in focus is spread over many pixels on the sensor (hence the blur kernel is also called point-spread function or PSF). A defocus kernel is well approximated by a circle or a 2D Gaussian shape, but actually the exact camera aperture shape should be used. It might also slightly vary over the image based on lens characteristics, but this effect is only apparent at very high image resolutions. However, as a motion blur kernel corresponds to some camera motion, it can have practically arbitrary form which renders the latter problem significantly more difficult. Still, a motion blur kernel is usually very sparse and contains a thin continuous path that corresponds to the

motion the camera underwent during image capture. Figure 3.2 shows typical blur kernel shapes: Dirac delta function is the identity of the convolution operator, a typical defocus kernel that has circular shape, and a motion blur kernel that is a sparse continuous motion path. If there is no in-plane camera rotation and the scene is planar or far away and static, or if we concentrate only on a small part of the image, the uniform kernel is valid also in case of motion blur. This assumption simplifies the mathematical models and allows for much faster restoration algorithms [33, 35, 122, 160, 161, 167, 219, 252].

The uniform blur model is usually violated in many real scenarios that can lead the restoration to fail, often even lowering the quality of the processed image [133]. In particular, the kernel may be *non-uniform* (position-dependent) over the image when the camera undergoes large rotations, when the scene is not planar and the camera captures parallax effects, or if objects are moving in the scene. Models for non-uniform blur do exist (we discuss some of them later at the sensor-based methods) but handling different blur at each pixel of the image is computationally too demanding for a live application.

The estimation of complex motion blur can actually be treated as an image registration problem using a projective motion path model [34, 226] (see Figure 3.3). The uniform convolution can also be interpreted as a summation of translated and weighted versions of the same image. The new, generalized model allows any projective transformation during exposure, but of course the number of variables to estimate increases drastically. Whyte [245] limits the allowed motions to x, y, z rotations while Gupta [68] restricts the model to two translations and one in-plane rotation. The assumption of a planar scene is still necessary though. Cho [32] proves that the effects of out-of-plane rotations can also be well approximated by the uniform blur model, however, in-plane rotations need to be handled differently.

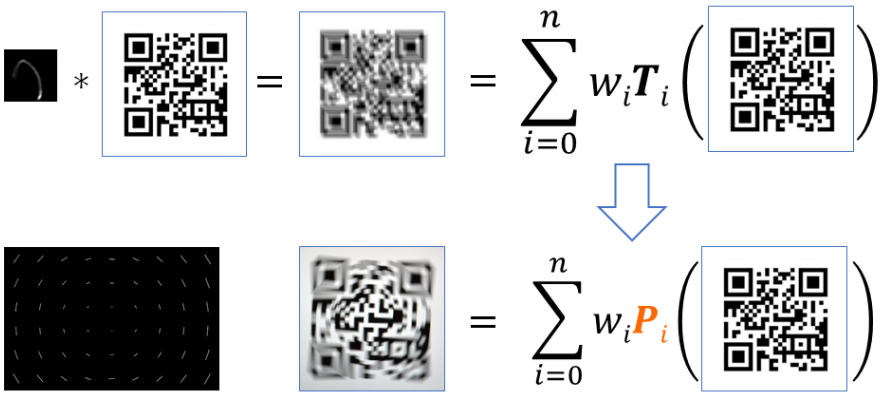


Figure 3.3: The convolution model can be generalized to the projective motion path model for modeling non-uniform blur. Instead of translations T, we plug general planar homographies P into the summation. Figure adapted from Yu-Wing Tai [226].

For our smartphone and wearable applications, a semi-non-uniform approach might be still viable that divides the image into smaller overlapping regions where uniform blur can be assumed. The final sharp image can then be re-assembled from the independently restored regions. If the visual code is only a fraction of the image, then as the cut-out region is usually a planar surface, the assumption of uniform blur is valid in that image region. In the rest of this thesis we therefore focus only on uniform blurs.

Non-blind deconvolution Non-blind deconvolution refers to deconvolution in applications where the kernel is known or can be measured in advance. Non-blind deconvolution is a mathematically ill-posed inverse problem because multiple solutions satisfy the equations. As convolution in the spatial domain is equivalent to a multiplication in the Fourier domain, deconvolution is possible by inverse filtering (division) in the Fourier domain, in theory. In practice, high-frequency image noise gets amplified by the division with small filter values. This problem was addressed by regularization in Wiener filtering [247], or by Richardson-Lucy filtering [178]. For the Wiener filter, the signal-to-noise ratio needs to be known in advance, and the method is prone to ringing artifacts at edges. The Richardson-Lucy filter is well suited for astronomical images because it is based on a Poisson-distributed image formation model. Newer deconvolution algorithms usually constrain the solution space to images that follow certain properties of natural images [58, 122, 132]. These new methods enforce natural look of the reconstructed image by for instance shaping the distribution of the image gradients. This way they can effectively reduce ringing artifacts that were common with the older methods. Of particular interest to us is the algorithm of Krishnan and Fergus [122] because it produces high-quality results, is very fast, and is suitable for parallelization.

Blind deconvolution Blind deconvolution, i.e., finding the original sharp image and the blur function from a single given blurry image is a severely ill-posed problem and hence needs to be treated with some kind of regularization. Earlier attempts posed regularization constraints on the blur kernel and assumed it has a parametric form [36, 259]. They estimated its parameters for instance in the Fourier domain. Recent methods usually follow a common pattern of Bayesian energy minimization, and constrain both the kernel and the latent sharp image, e.g., by shaping the distribution of its gradients according to natural image statistics. The unknown latent sharp image and blur kernel are estimated successively in a multi-scale iterative optimization scheme [58, 123, 133, 194]. All these techniques assume a uniform blur kernel over the image and iteratively estimate the sharp image in one step and the kernel in another step using complex optimization methods. While they enable impressive results, they have their own limitations, and in particular their long processing time is not acceptable for a mobile barcode scanning application.

For example, the time required for blindly deconvolving an image of 640×480 pixels that was originally blurred by a kernel of size 25×25 takes about an hour with the method of Fergus (2006) [58], about 5 minutes with the methods Shan (2008) [194], and about 1 to 5 seconds with the method of Cho (2009) [35] depending on whether we use the GPU or the CPU. This example shows that while there have been tremendous advancements in the past years, we are still far from real-time blind deblurring.

Multiframe deblurring Others have shown how significant improvements can be achieved when multiple blurry images of the same object are available as this can help to reduce ambiguities in the restoration [17, 26, 173, 260, 264]. The subsequent images contain differently degraded forms of the original scene which can be exploited in joint multi-frame deblurring. The use of multiple blurry/noisy images for restoring a single sharp frame has been proposed by Rav-Acha and Peleg [173], applied for sharp panorama generation by Lie et al. [136], and recently for HDR and low-light photography by Ringaby et al. [179]. Methods to jointly restore multiple blurred images are for instance [47, 263, 264] but those are too slow for smartphone applications. The key insight of these approaches is that the image alignment problem and the camera shake problem are actually very similar in that both result from camera movement, just the former comes from inter-frame movements and the latter comes from intra-frame movements. The cost function for multi-image deblurring is the sum of the cost functions of single image deblurring, because each other image is also just a sum of transformed versions of the original sharp image. The work in [264] extends multi-shot imaging even further into a unified model for image alignment, blur removal, and super resolution. Deblurring and super resolution are inherently connected problems as every super resolution approach needs to invert a downsampling operator, hence a deconvolution is a necessary step in super resolution [195].

Schechtman [197] and Takeda [227] present the theory of joint space-time super resolution and deblurring. The key insight is that motion blur is actually the result of a temporal effect, it is caused by too low temporal resolution of the camera, hence it should be treated temporally. Instead of spatially deconvolving single images with complex and even varying motion blur kernels, we should try to deconvolve a video along the time axis. The temporal blur kernel is the same simple rectangular function in the whole image, independently of whether the motion blur was caused by object motion or camera motion. The obvious advantage of this method is that deconvolution becomes much simpler and uniform, but our experiments in this direction turned out to be too slow for a practical live application. Nevertheless, the idea is still very interesting for offline video enhancement.

Although multiple images are available when scanning a barcode with a wearable camera, the joint multiframe optimization scheme of the above algorithms involves excessive

computations, therefore they are not suitable for a live scanner application. We have performed experiments in reducing the problem to only black and white images but we have not found a reliable image registration method that could robustly align our blurred images.

Deblurring with special hardware Alternative approaches use dedicated hardware to aid the deblurring. Ben-Ezra et al. [20] combine a high frame rate video camera with a still camera to help in recording the blur kernel. Optical flow from the high-speed camera helps in computing the motion blur in the captured images of the high-resolution camera. Raskar et al. proposed solutions to overcome the loss of high frequency components using a fluttered camera shutter [172] or by varying the exposure time of subsequent frames [4]. However, these methods rely on precise motion segmentation and the knowledge of object velocities, and cannot handle blur caused by camera rotations. Cho et al. [37] built a camera that moves on a parabolic path during exposure. Under this special movement, the motion blur caused in the image is independent of the speed of moving objects in the scene, hence all blur can be removed by deconvolving the whole image with the same blur kernel. Wang et al. [241] design a new imaging sensor that is able to capture a blurry colour image and a noisy grayscale image at the same time, thereby avoiding the need for a registration step between the two. Image restoration from a blurry/noisy image pair can be done by the method of [260]. As our goal is deblurring on unmodified wearable devices, we do not further discuss this family of methods.

3.2.2 Deblurring with edge enhancement

The way an edge gets blurred in an image gives information about the cross-section of the blur kernel perpendicular to the edge [38, 95, 97]. The edge-based algorithms in a first step try to hallucinate sharp edges, and in a second step find the blur kernel that causes the observed blurry edges (see Figure 3.4). The two steps are alternated in an iterative optimization, often also across multiple scales to aid the convergence. Edge-based methods are often much faster than any other approaches and they are proven to be effective in practice [268] but they are difficult to analyze because of heuristic steps.

Cho and Lee [35] presented the first fast algorithm for blind motion deblurring. They iterate between an image restoration step and a kernel estimation step, and use only simple image filters between the two. The main idea is that for image reconstruction, a low-quality but fast step is sufficient if the errors are suppressed and edges are boosted by edge-aware image filters, before the kernel estimation step. This image-kernel refinement loop is repeated several times from smaller to larger image scales, improving the overall conver-

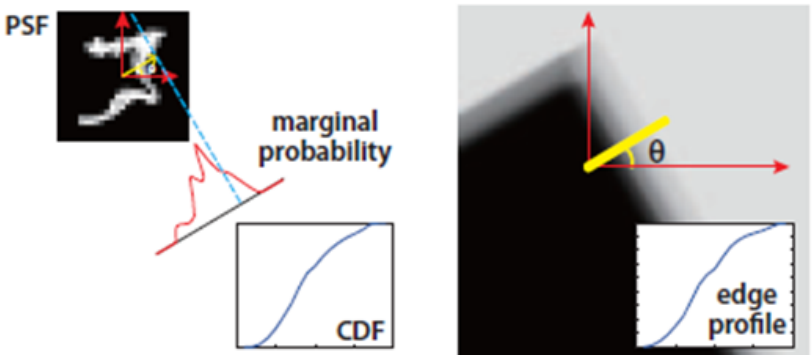


Figure 3.4: The blur kernel (PSF) can be estimated from the profile of dominant edges. The way an ideal step edge gets blurred gives information about the cross-section of the blur kernel perpendicular to the edge. Figure from Sunyeong Kim [224].

gence of kernel estimation. The authors report to achieve results of quality comparable to previous attempts within a few seconds (up to two magnitudes faster than others), opening the doors for interactive use. Their multi-scale loop structure forms the basis of many subsequent algorithms and is also fundamental to our work.

Xu and Jia [252] analyzed which edges are actually useful for kernel estimation and showed that structures smaller than the kernel size (like thin lines of a barcode or modules of a QR code) may mislead the optimization as shown in Figure 3.5. They proposed a filtering approach that selects strong edges in the image and uses only those for kernel estimation. Other methods like [38] and [97] recover the blur kernel by explicitly inspecting how sharp edges of the scene get blurred and reconstruct the kernel from its cross-sections. In [254], the authors exchange the parametric edge enhancement filters of [35] with a new regularization term that approximates the L_0 norm of the gradients, i.e., it keeps the number of non-zero gradients minimal (see details in the Appendix). With this new energy formulation, the algorithm no longer relies on ad-hoc edge selection and filtering methods which means that no parameter tuning is required. [167] revisits an older method called total variation deconvolution, and analyzes in details why – although it is simpler than natural image priors – it can still find the correct sharp solution.

3.2.3 Deblurring with inertial sensors

Kernel estimation from the image content alone often involves iterative optimization schemes that might be computationally too complex to perform on a smartphone within acceptable time. Auxiliary hardware might be expensive and difficult to mount, so kernel estimation from built-in motion sensors seems the most appealing for wearable

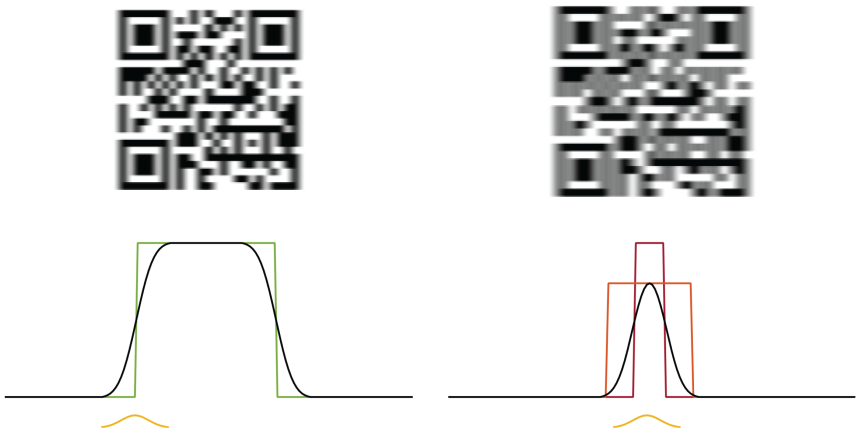


Figure 3.5: Left: At large structural edges (structures larger than the blur kernel), the blurry pixels are easy to match with the edges, hence it is easy to estimate the blur kernel. Right: However, at small-scale structures (structures smaller than the blur kernel), it is hard to tell which blurry pixel originates from which edge, and most blur estimation methods fail. Therefore we must limit our blur estimation algorithms to large-scale structures. Figure adapted from Li Xu [252].

applications. Combining camera frames and inertial measurement units (IMU) has been successfully used for video stabilization [19, 59, 73, 101], for denoising [90, 179] and also for blur removal [96, 164, 202].

Joshi et al. [96] presented the first IMU-based deblurring algorithm with a DSLR camera[1] and external gyroscope and accelerometer. In their approach, the camera and the sensors are precisely synchronized through the flash trigger. The DSLR camera has a global shutter which makes the problem easier to handle than the case of smartphones. To keep the problem tractable, they assume a constant uniform scene depth, which they find together with the sharp image by solving a complex optimization problem. Bae et al. [13] extend this method to depth-dependent blur by attaching a depth sensor to the camera. Ringaby and Forssen [179] develop a virtual tripod for smartphones by taking a series of noisy photographs, aligning them using gyroscope data, and averaging them to get a clear image, but not targeting blur removal. Köhler [118] and Whyte [246] show in their single-frame deblurring experiments that three rotations are enough to model real camera shakes well. Karpenko [101], Ringaby [59], and Bell [19] developed methods for video stabilization and rolling shutter correction specifically for smartphones. Without a depth sensor, they all have to assume a planar scene far from the camera. Recently, a new model for joint deblurring and rolling shutter de-warping has been presented in [217], but this

[1] DSLR camera: Digital single-lens reflex camera

method is too complex for our purposes, the authors report a runtime of 18 min on a single 800×600 image.

Park and Levoy [164] compared the effectiveness of jointly deconvolving multiple images degraded by small blur versus deconvolving a single image degraded by large blur, and versus averaging a set of blur-free but noisy images. They record a series of images together with gyroscope measurements on a modified tablet with advanced camera controls (e.g., exposure control, RAW[2] data access) through the FCam API [163], and attach an external 750 Hz gyroscope to the tablet. They estimate the rotation axis, the gyroscope drift, and the time delay between the frames and the gyroscope measurements in a non-linear optimization scheme over multiple image segments. Their optimization scheme is based on the fact that applying two different kernels to an image patch is commutative. This means in the case of true parameters, applying the generated kernels in different order results in the same blurry patch. The rolling shutter parameter is calculated off-line with the method of Karpenko [101]. They report the runtime of the algorithm to be 24.5 seconds using the Wiener filter and 20 minutes using a sparsity prior for deconvolution, not mentioning whether on the tablet or on a PC.

Closest to our goals and constraints is the series of work by Sindelar et al. [200, 201, 202] who also reconstruct the blur kernels from the sensor readings of an unmodified smartphone in order to make the deconvolution non-blind. The first approach [200] considers only x and y rotations and generates a single kernel as weighted line segments using Bresenham's line drawing algorithm. Unfortunately, their time calibration method is not portable to different phones. They find the exact beginning of the exposure by inspecting the logging output of the camera driver, which might be different for each smartphone model. They read the exposure time from the EXIF[3] tag of the captured photo, however, this information is not available for the preview frames we intend to use for a live deblurring system. Extending this work in [202] the authors generate piecewise uniform blur kernels and deblur overlapping patches of the image using the Wiener filter. They also account for the rolling shutter by shifting the time window of gyroscope measurements when generating blur kernels for different rows of the image. This last approach is appealing for kernel estimation in wearable devices, however, we need to deal with various calibration issues.

[2] A RAW image file is also called digital negative as it is not a viewable image but contains all sensor information required to create an image

[3] EXIF: Exchangeable image file format

3.2.4 Deblurring text and visual codes

Intuitively, blur removal from text images appears to be a similar problem to ours. Previous work addressed defocus and motion blur removal from text and also barcode images. While the shape of a defocus kernel is given by the lens characteristics and can be measured accurately (in document imaging applications), a motion blur kernel can have very different shapes depending on the object or camera motion, hence compensating the latter is significantly more difficult. Furthermore, text deblurring methods expect a few thin black lines on dominant white background. Visual codes, in contrast, usually have an equal distribution of dark and light areas.

Compensating *defocus blur* in 1D barcodes can be considered solved [2, 30, 39, 52, 62, 139] and fast algorithms exist in commercial barcode scanner applications [327, 331]. Liu et al. [137] extended a standard deconvolution method with a bi-level image histogram constraint for quickly removing defocus blur from 2D DataMatrix codes, but this method would not work with motion blur.

Compensating *motion blur* in 1D barcodes is addressed in Yahyanejad et al. [257] who present a blind deconvolution algorithm to remove translational motion blur. They reduce the problem to 1D by averaging over several lines of the barcode, which renders the method inapplicable for 2D QR codes. Guo et al. [67] showed how to scan barcodes using a light field camera [305] that has extended depth of field so the distance of the camera and the barcode can vary in a wider range. Xu and McCloskey [256] presented a motion deblurring method using a camera with a fluttered shutter, a hardware modification that makes the blur easier to invert. We target off-the-shelf wearable devices without such modifications. Gennip et al. [232] presented an algorithm for blind deblurring of QR codes by explicitly making use of the known finder patterns to estimate the kernel. The algorithm is evaluated only with synthetic blurs, and with the assumption that the location of the blurry finder pattern is known, which might be difficult to determine in real images.

Our work is closely related to recent research on blind text deblurring. The method of Cho et al. [33] can successfully remove motion blur from text images that have a lot in common with visual tags, but the algorithm relies on precise text segmentation and is too slow for our purposes. The text deblurring method of Chen et al. [31] is specifically developed for binary text images and is unlikely to work well with cluttered images. The algorithm of Pan et al. [160] applies L_0-minimization not only to image gradients (see [254] for natural images) but also to the pixel values which means the algorithm enforces the image to have a few black pixels among many white pixels, which is typical for documents. Our experiments have shown that this method only works for QR codes if its parameters are carefully tuned. Furthermore, the algorithm is too slow for mobile applications.

3.2.5 Towards fast visual code deblurring

Overall, the general photograph deblurring algorithms today have rather high computational requirements and are optimized for natural scenes. Our experiments have shown that they often fail on artificial image content such as visual tags due to the different appearance. Even if the blur function is known, non-blind deconvolution of visual tag structures (step edges, thin lines, etc.) differs from that of photographs because the assumptions (e.g., natural image priors) do not hold. Also, many existing methods have difficulties with sharp discontinuities and produce ringing artifacts. Therefore, directly applying a natural image deblurring method usually fails with visual code images as also illustrated in Figure 3.6.

Figure 3.6: A blurry visual tag deblurred by the methods of Cho2009 [35], Sun2013 [219], and Xu2010 [252]. Existing blind deconvolution methods are optimized for natural scenes and often fail with artificial image content.

However, we can exploit two fundamental differences between restoring general photographs and restoring visual codes. First, the codes do not need to *look* perfectly for decoding. Second, the 1D codes usually have error detection (checksum), and 2D codes have both error detection and error correction (Reed-Solomon encoding, etc.) capabilities. These properties allow early termination of the restoration algorithm; we may arrive at an acceptable image within a few iterations only. Furthermore, there is no need for a complicated visual quality measure for automated termination.

The approach of inspecting blurry edges in the image to recover the blur function is especially appealing since it is relatively fast, and visual tags originally contain a large number of sharp edges. Moreover, 2D codes are also rich in corners that are more distinctive under blur and make localization and tracking of blurry codes over multiple video frames feasible. In today's smartphones, we have access to built-in inertial sensors that can aid the kernel estimation. The code region itself is planar, so a uniform kernel assumption is valid in the code area.

Unfortunately, the thin lines of 1D barcodes often become indistinguishable after motion blur because the distance between the bars is often smaller than the blur kernel (cf. Xu's conclusions about natural image deblurring [252]), therefore we cannot robustly estimate the blur solely from a barcode's edges. Also, a barcode contains only edges in one direction

so only one cross-section of the blur kernel can be recovered. This cross-section, however, should be enough for restoring the barcode information, in theory. The strong rectangular region around the code (that we also exploited for localization in Chapter 2) might help in recovering more information about the kernel, but our experiments were unsuccessful in this direction. We therefore turned towards 2D codes that have apparent finder patterns (e.g., the corner areas of QR codes). In the rest of the thesis, we focus on QR codes, but generalization of our methods to other matrix codes containing some thick lines should be easily possible.

In addition to the findings made in the general blind deconvolution techniques, we can make the following observations. (i) QR codes contain many black and white corners that are easy to localize even in a blurred image. (ii) QR codes include a checksum, so the algorithm can terminate when the checksum is correct. False positives are hence practically impossible. (iii) QR codes contain strong error correction, so even partially restored codes may be decoded. This is especially important when the blur is slightly non-uniform in the code area. (iv) QR codes consist of sharp edges, which is advantageous for blur estimation, but disadvantageous for blur removal. (v) QR codes also contain small structures that may mislead the blur estimation process and therefore their influence needs to be suppressed.

In the rest of this chapter, we present two methods to tackle motion blur in visual codes on unmodified wearable devices. Our first contribution is a new "blurry QR decoder" algorithm that iteratively estimates the blur kernel from the edges in the image in a restoration loop with the goal of fast decoding instead of visually pleasing deblurring. Our second contribution is a quick blur estimation method from built-in inertial sensors that can be used as a fast initialization step for the QR restoration loop.

3.3 Fast blind deblurring of QR codes

In this section, we present an algorithm that is particularly suitable for fast restoration of a single QR image (see Figure 3.7). Instead of high-quality deblurring, we rather focus on high-speed decoding. We apply and combine well-established techniques from the photograph deblurring literature, our contributions are the adaptations that make the algorithm particularly suitable for fast QR restoration.

Figure 3.7: We recover the information from severely motion-blurred QR codes

3.3.1 Method overview

We apply the uniform blur model which describes the blurred image B as a convolution of a latent sharp image I and a blur kernel k, with additive Gaussian noise N:

$$B = k * I + N \tag{3.3}$$

Uniform blur is a valid assumption if the blur is caused mainly by translational motion. Cropping the image to a small search region in practice reduces blur to mostly this kind.

The algorithm consists of four main components that work in a tightly coupled restoration-recognition loop, illustrated in Figure 3.8. In the first step, we run a standard *decoder* algorithm; if an image is sharp enough for decoding, the loop terminates. Otherwise, the contrast is increased in the second step and *edge-aware filtering* is applied. This removes noise from the image, flattens small structures while it keeps strong edges. Next, the blur *kernel estimation* step finds the most likely kernel which transforms the sharpened image into the blurry one. In the fourth step the blurry image is deconvolved with the currently estimated kernel to get a sharper *image estimate*, which again enters the decoder. The process is iterated up to five times to attain improved kernel estimates and sharper image estimates. Performing the loop over multiple image scales ensures fast convergence to the correct solution.

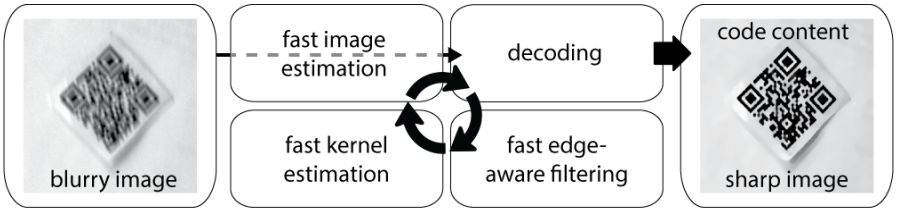


Figure 3.8: Method overview

Similar to other blur removal techniques, we formulate the QR deblurring problem as an energy minimization scheme over I and k and iteratively optimize for one while keeping the other constant. The total energy function to be minimized consists of a data fitting term and regularization terms $\rho_I(I)$ on the image and $\rho_k(k)$ on the kernel with regularization weights λ and γ, respectively. The derivation of the whole energy function can be found in the Appendix. For the data fitting term, the L_2 norm is commonly used, and different blind deconvolution methods apply different regularization terms. The common form of the energy function is

$$\underset{I,k}{\operatorname{argmin}} \, ||B - k * I||_2^2 + \lambda \rho_I(I) + \gamma \rho_k(k) \tag{3.4}$$

Our goal is to find a good compromise between restoration quality and restoration speed, also taking into account the typical black and white structure of QR codes. We chose a sparsity prior on the image gradients ∇I_i and an L_2 sparsity prior on the kernel values k_i:

$$\rho_I(I) = \sum_{\forall i} |\nabla I_i|^\alpha \qquad \rho_k(k) = ||k||_2^2 = \sum_{\forall j} k_j^2 \tag{3.5}$$

In our notation, i and j index image pixels and kernel pixels, respectively. The optimization of the total energy function can be separated into I- and k-subproblems that are presented in the following.

3.3.2 Fast image estimation

Given the currently estimated kernel k, we apply the fast *non-blind* deconvolution method of Krishnan [122] that in general enforces a sparse hyper-Laplacian distribution on the gradients of the sharp image. With the exponent $\alpha = 1$, the enforced Laplacian distribution of the gradients in turn corresponds to a total variation regularization of the image I. This is well suited for QR-codes as it favors flat image regions while it also allows sharp edges.

$$\underset{I}{\operatorname{argmin}} \, ||B - k * I||_2^2 + \lambda ||\nabla I||^\alpha \tag{3.6}$$

For $\alpha = 1$, the solution is particularly simple and fast, relying only on FFTs and thresholding operations [122].

To reduce boundary artifacts in the FFT-based restoration, we wrap the image boundaries using the method of Liu and Jia [138] prior to deconvolution. This wrapping method is suitable for not only symmetric but general kernels, is reasonably fast, and can be used in any FFT-based restoration method.

3.3.3 Fast edge-aware filtering

Due to an imperfect kernel, the fast deconvolution produces unwanted ringing artifacts and noise that need to be suppressed while the main structure of the image must be kept unchanged before kernel estimation. At this stage other deblurring methods usually apply a combination of bilateral and shock filters [35] which require parameter tuning, or L_0-smoothing [160, 254] which is slower. We propose to use the joint weighted median filter (WMF) [265] to remove small variations in the structure. This new filter produces an output similar to other edge-aware smoothing filters but is significantly faster. The filter replaces each pixel with a weighted median of its neighborhood of radius r. The weights depend on a measure of similarity with the current pixel (intensity, color, etc. – we use intensity). The neighbors of the pixel are ranked according to their weight and the middle value is chosen. The WMF has computational complexity $O(r)$ so it is very fast. The filter radius r is set $1/5$ proportional with the current kernel size. The blurry B and the sharpened I image pair is passed to the next stage for blur kernel estimation.

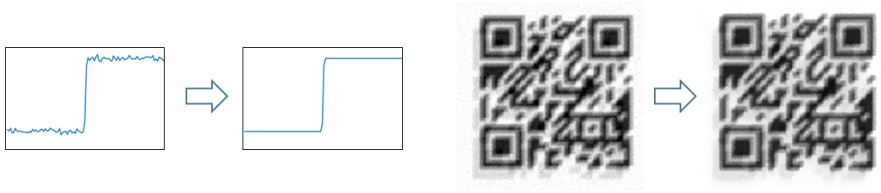


Figure 3.9: The weighted median filter smooths high-frequency components but keeps sharp edges.

3.3.4 Fast kernel estimation

Given an image estimate I, we solve for the kernel k in the gradient space using the efficient method of Cho and Lee [35]. The energy function to be minimized here is

$$\underset{k}{\operatorname{argmin}} ||\nabla B - k * \nabla I||_2^2 + \gamma ||k||_2^2 \tag{3.7}$$

which can be solved with the conjugate gradient method [35] in the Fourier domain where convolution turns into multiplication. The L_2-norm on k favors sparse solutions which is desirable because a motion blur kernel consists of a thin continuous motion path. Working with image gradients instead of image intensities is crucial here because it allows to ignore boundary artifacts which drastically reduces the number of FFTs required [35].

Next, potentially disconnected small components in the kernel are discarded, small values are thresholded, and the kernel is shifted to its geometrical center (see Figure 3.10). Other methods usually shift the kernel to its mass center but that might clip long thin tails at the boundaries. Finally, the kernel is normalized to sum to 1 so that the convolution does not reduce or increase the energy of the image.

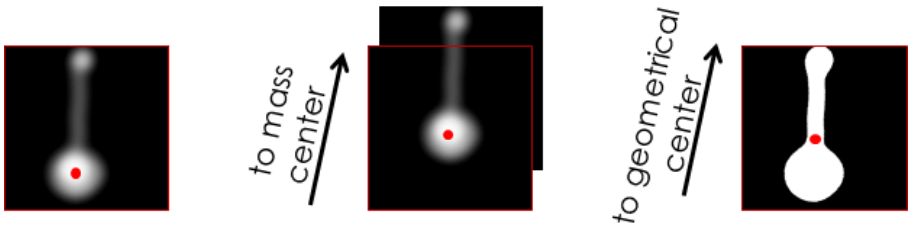


Figure 3.10: The kernel estimation step may produce off-centered kernels but thin motion blur kernels may contain important entries at borders. Because mass centering might clip off important parts, we shift the geometrical center to the middle instead.

3.3.5 Decoding in a restoration-recognition loop

After each iteration, we let a common QR detector and decoder algorithm process the image. The error correction in QR codes practically disables false decoding while it can guarantee that the algorithm converged to the right solution. The first successful decoding allows early termination.

We perform our calculations over multiple image scales, starting with a kernel size of 5×5 up to 33×33 pixels. The image pyramid is built with scale factor $1/\sqrt{2}$. On each scale level the algorithm performs up to 5 iterations. Between scale levels, the kernel is bilinearly upscaled. Figure 3.11 illustrates how the kernel gets refined over the iterations.

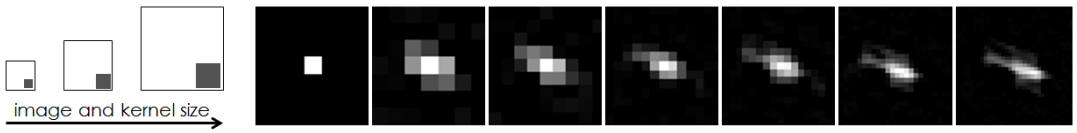


Figure 3.11: Illustration of kernel refinement over 7 scales using a single peak as starting kernel.

3.3.6 Initialization

One of our main contributions is a grid-shaped starting kernel. Other methods usually initialize the kernel either with a Dirac delta function (identity blur) or with a small 2D Gaussian (small defocus blur). In contrast, we initialize the kernel with a grid of Dirac functions (see Figure 3.12) which corresponds to multiple shifted copies of the sharp image after convolution. The proposed kernel has, to our knowledge, not been reported in the literature but makes an important difference, in particular for heavily blurred images. While the delta kernel works well in simulations, we found in experiments with real

Figure 3.12: Illustration of image and kernel refinement over 8 iterations using a grid of peaks as starting kernel. The grid kernel converges slower but is able to restore very large blurs that are otherwise unsuccessful with the peak kernel. The example also illustrates how disconnected kernel noise gets removed during the process. The image is 300×300, the kernel is 33×33 pixels, the decoding took 3.10 s. The blurry image was taken with a smartphone.

smartphone images that using a grid as initial kernel greatly improves the robustness of the algorithm. With this initialization, our algorithm is able to invert very large blurs. We provide an analysis on the impact of the new kernel in the evaluation section. However, this gain in deblurring performance comes at a small cost as in general the grid kernel converges slower than the delta kernel. Fortunately, one could overcome this tradeoff by reading the inertial sensors during image capture and thus selecting the right initial kernel based on a first estimate of the blur size, as we will show in Section 3.4.

3.3.7 Implementation and test environment

We have implemented the algorithm on both PC and Android using the open-source OpenCV [313], FFTW [285], and ZBar [349] libraries. We have found the constants $\lambda = 0.002$, $\gamma = 2$, and WMF standard deviation 55 to work well in all our experiments. The algorithm requires no further parameter tuning. Our scanner is built as a standard native Android application which makes it portable to any Android device.

Our linear blur model does not take into account the manufacturer-dependent non-linear image enhancements and other effects in the camera that may cause significant impact on the performance of blur removal algorithms [224]. It is therefore vital that we set the tone map curve to linear and switch off the automatic image enhancement functions in our camera (this is the reason why our real input images appear dark). In our experiments, this is possible through the Android *Camera2* API. This fine camera control is available since Android 5.0 if the camera driver also supports it. This fact, unfortunately, limits our live tests to high-end smartphones; however, we expect that with rapid technological advancement the smartphones' capabilities will soon be available in smartwatches and smartglasses as well. Until then, the smartglasses implementation can be tested using synthetic images, which is sufficient for speed and memory analysis. The graphical user interface of our smartphone application is also shown in Figure 3.25, and the interface on smartglasses is shown in the accompanying video (see Appendix). Once the QR code is recognized, depending on its content, a URL is opened or the contact details are shown to the user.

We evaluate the effectiveness of our algorithm on a series of QR code images contaminated by synthetic and real blur, and also compare our algorithm with the state of the art. For convenience, the comparisons were performed on a notebook with a 2.40 GHz Core i7-4700MQ CPU. Our algorithm currently uses a single thread only. The input images with synthetic blur were created in Matlab, the input images with real blur were captured with a Google Nexus 6 smartphone. We also show qualitative results of our algorithm running directly on the smartphone and on smartglasses.

3.3.8 Removing synthetic motion blur

In the first experiment, we test whether the algorithm can remove synthetic uniform blur, i.e., blur that our mathematical model assumes. Figure 3.13 illustrates the experiment. For repeatability, we use six out of the eight benchmark kernels from Levin et al. [133]. We leave out two kernels that are so small that we can decode the images without deblurring. The input images are of resolution 200×200, the kernels are between 17×17 and 27×27

Figure 3.13: Removing synthetic uniform blurs from images (0.1% synthetic noise). Top: input images and ground truth kernels. Bottom: output images and kernel estimates when first decoded. Kernel size indicates the scale level.

pixels. In this experiment, we use small images because the benchmark kernels are rather small. To simulate imperfections of real images, we also add small 0.1% noise to the blurred images. The top row of Figure 3.13 shows the input images and in the insets the ground-truth blur kernels that were used to create the input images. The bottom row shows each intermediate image and the current estimate of the kernel when the code first could be decoded. The output images and kernels are upscaled for visualization using nearest neighbor interpolation. The form of our estimated kernels resembles that of the ground truth kernels. On the PC, it takes about 0.45 s to restore the first five images while the last one takes 1.6 s because iterations on two more scales are required due to larger blur.

We repeated the experiment after adding large 1% image noise (see Figure 3.14). The algorithm can successfully decode all blurry and noisy examples with the expense of a longer average runtime of 0.66 s.

We conclude that our algorithm is able to quickly restore blurry codes that follow our mathematical model; however, the restoration speed depends on both the image size and blur size. The main bottleneck of real-time operation is the number of single-threaded FFTs, but we expect a speedup with a parallel implementation.

3.3.9 Comparison of blind deblurring methods

For practical applications, the restoration speed is crucial. In this experiment, we compare our algorithm with the state of the art in terms of runtime and readability of the resulting images. All selected methods are available open source or as executables. Methods *(B)-(G)* were developed for general blind deconvolution problems while method *(H)* is specifically

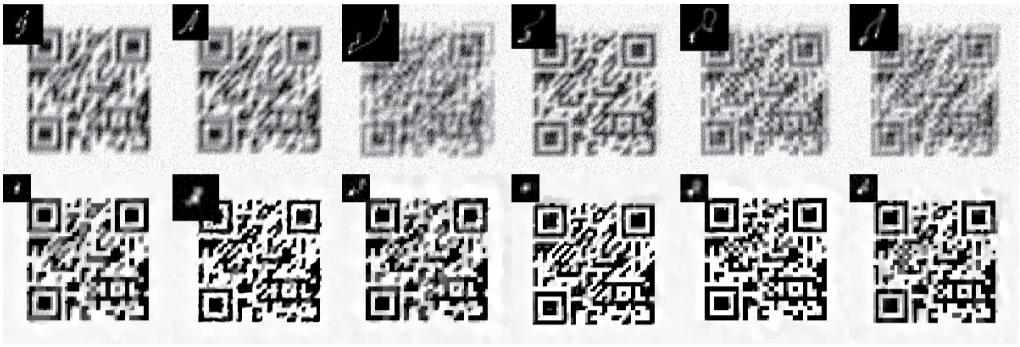


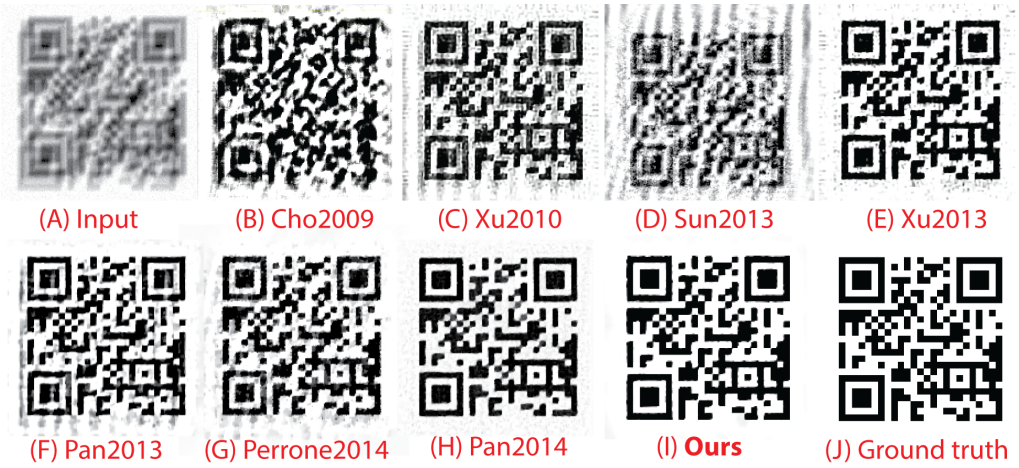
Figure 3.14: Removing synthetic blurs from images (also added 1% synthetic noise). The top row shows the input images and ground truth kernels. The bottom row shows the output images and kernel estimates when first decoded. Kernel size indicates the scale level. The kernels are from the blur test set from Levin et al. [133].

for text image deblurring. A short summary of each method can be found in the related work section. Figure 3.15 summarizes our findings.

Method (B) [35] was the first fast algorithm for natural image deblurring, but the applied gradient statistics cause it to fail on artificial black and white content. Also, this method is sensitive to initialization parameters. Method (C) [252] selects good edges for kernel estimation and reconstructs the sharp edges but the result exhibits significant ringing artifacts that make the code unreadable. Method (D) [219] decomposes the image into a dictionary of sharp edge and corner patches to aid the kernel estimation. The method is very slow, and although the patch prior was expected to fit well with the structure of QR codes, the result is surprisingly unreadable. Method (E) [254] is fast and can successfully restore the code. However, to get good results, we had to supply an estimate for kernel size almost double of the true kernel size which is rather inefficient in terms of computational overhead.

Method (F) [161] also implements an edge selection strategy, but the decoding failed. Method (G) [167] uses TV regularization like our approach. It can successfully restore the code but is very slow. Method (H) is especially interesting as text images share similar properties with visual codes. We tested the authors' Matlab implementation and ported the algorithm to C++ to make it faster (we achieved about 30% speedup, not using the GPU at all). The runtime was 12.73 s in Matlab and 9.69 s in our C++ implementation. The critical bottleneck is the L_0 restoration step. Also, the algorithm requires tuning many parameters, we found that QR codes need different regularization weights than text.

Our method (I) almost perfectly reconstructed the code (cf. ground truth (J)). Further



	A Input	B Cho2009	C Xu2010	D Sun2013	E Xu2013
binary	-	C++/GPU	C++/GPU	Matlab	C++/GPU
runtime	-	0.481 s	0.955 s	217.730 s	1.049 s
decoded	-	no	no	no	**yes**

	F Pan2013	G Perrone2014	H Pan2014	I Ours2015	J Truth
binary	Matlab	Matlab	Matlab (*C++*)	C++	-
runtime	133.8 s	171.898 s	12.736 s (*9.691 s*)	1.765 s (**0.614 s**)	-
decoded	no	**yes**	**yes**	**yes**	-

Figure 3.15: Comparison of blind deconvolution algorithms on a synthetically blurred QR code. (A) Input frame, (B) Cho2009 [35], (C) Xu2010 [252], (D) Sun2013 [219], (E) Xu2013 [254], (F) Pan2013 [161], (G) Perrone2014 [167], (H) Pan2014 [160] original Matlab implementation and in brackets *our C++ port*, (I) Ours, in brackets the time of first decoding, (J) Ground truth

advantages of the proposed method over the reconstruction quality are its low memory footprint and high speed. Our method is almost as fast as (E), and it is important to note that (E) performs the FFT calculations on the GPU while we use one CPU core only. Also note that Matlab performs parallel processing in several built-in functions, so an exact runtime comparison is difficult.

3.3.10 Removing real motion blur

Next, we test the algorithm on codes contaminated by real motion blur. We capture a sequence of images with a smartphone while holding a QR code in front of the camera and deliberately shaking either the code or the smartphone. We used the same QR code with error correction level M in all our experiments. We capture camera preview frames because barcode scanner applications usually work with those instead of still images. We set the frame size to 720×480 pixels. We tested 340 images in total, out of this 217 frames were decoded without deblurring (63.8%), 83 frames were decoded after deblurring (24.4%), and only 40 frames were unsuccessful (11.8%), so our algorithm significantly increased the number of codes that could be scanned. Figure 3.16 shows examples of the restored codes. The decoding of these blurry examples took on average 1.4 s on the PC, but the number of iterations and thus the speed depends on the size of the blur kernel.

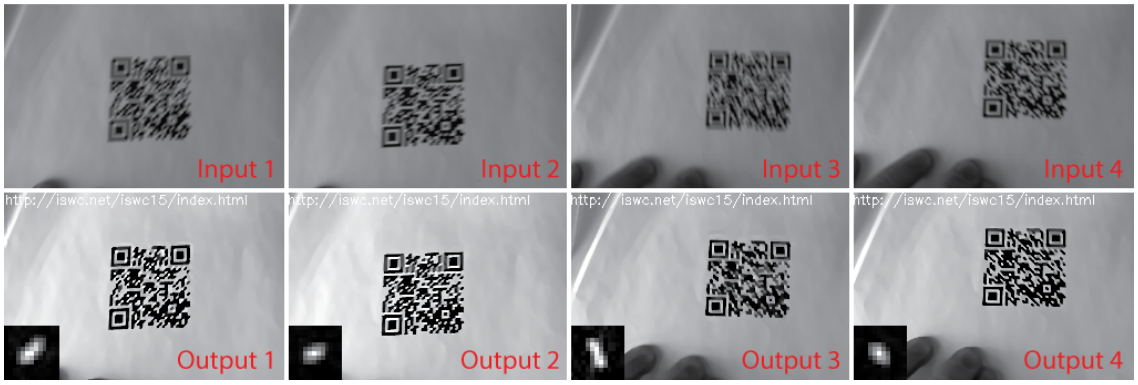


Figure 3.16: Removing real motion blur from QR code images. The decoded content `http://iswc.net/iswc15/index.html` is written in the top of the images.

3.3.11 Impact of initial kernel choice

As discussed before, one of our main contributions is the proposed initialization scheme. In experiments with real data we have experienced that in case of heavy blur and when

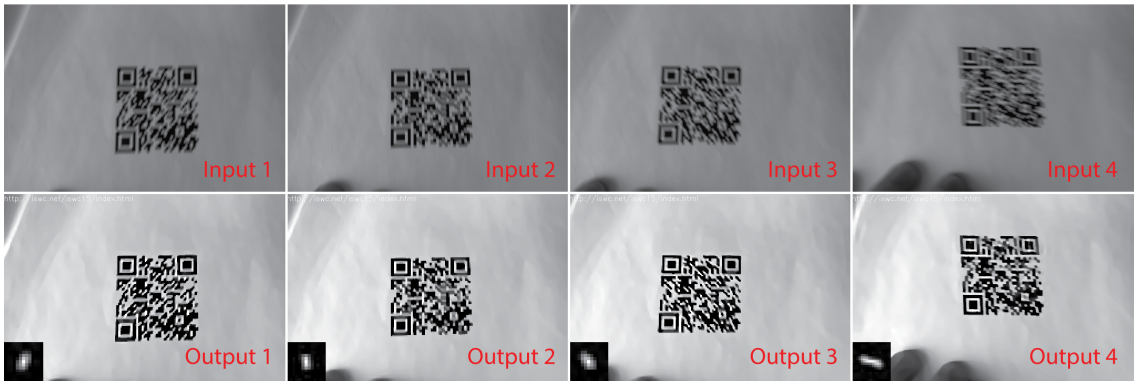


Figure 3.17: Further results of removing real motion blur from QR code images.

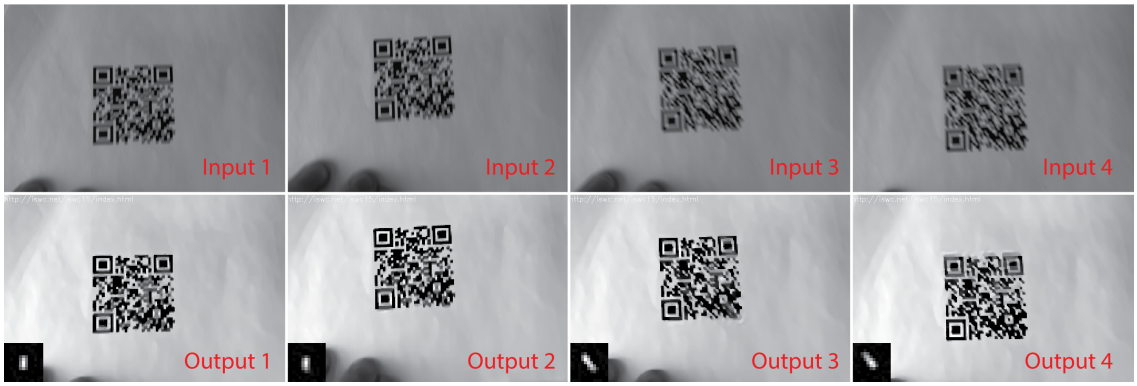


Figure 3.18: Further results of removing real motion blur from QR code images.

initialized with the Dirac kernel, the algorithm often did not converge to the right solution. In her seminal work, Levin [133] analyzed why blind deconvolution algorithms may converge to a blurry solution, however, the problem is different here, because we observed sharp but unreadable output images. We then introduced a new grid-shaped starting kernel that works better with large blurs than the Dirac kernel. In Figure 3.19 and Figure 3.20, we show several example results of our experiments that verify our claims. The example also illustrates how disconnected kernel noise gets removed during the process. We assume the better performance of the grid kernel is connected to the regular structure of the underlying sharp QR code, but the detailed analysis is left for future work.

3.3.12 Other visual codes

DataMatrix codes The algorithm also performs well on DataMatrix codes (see Figure 3.21), but the blur estimation phase is less robust as the modules (the black and

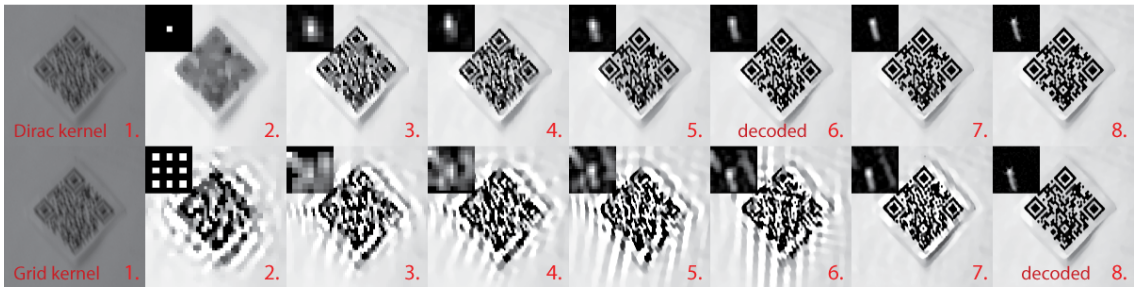


Figure 3.19: Illustration of image and kernel refinement over 8 iterations using a single peak or a grid of peaks as starting kernel. In this example, both initial kernels lead to a correct solution, but our grid kernel converges slower. However, the grid kernel is able to restore very large blurs that are otherwise unsuccessful with the peak kernel (see next figures). The image is 300×300, the kernel is 33×33 pixels, the decoding took 0.72 s and 3.10 s, respectively. The blurry image was taken with a smartphone.

Figure 3.20: Further examples illustrating image and kernel refinement using a single peak or a grid of peaks as starting kernel. In these examples, codes were not recognized using the peak initial kernel, but our grid kernel is successful in removing large blurs.

white squares) of a DataMatrix are much smaller than the modules of a QR code. We performed this experiment directly on the smartphone and we did not change anything in our algorithm and as it still expected a QR code, it took the maximum amount of allowed time. The first three images became decodable with a commercial decoder.

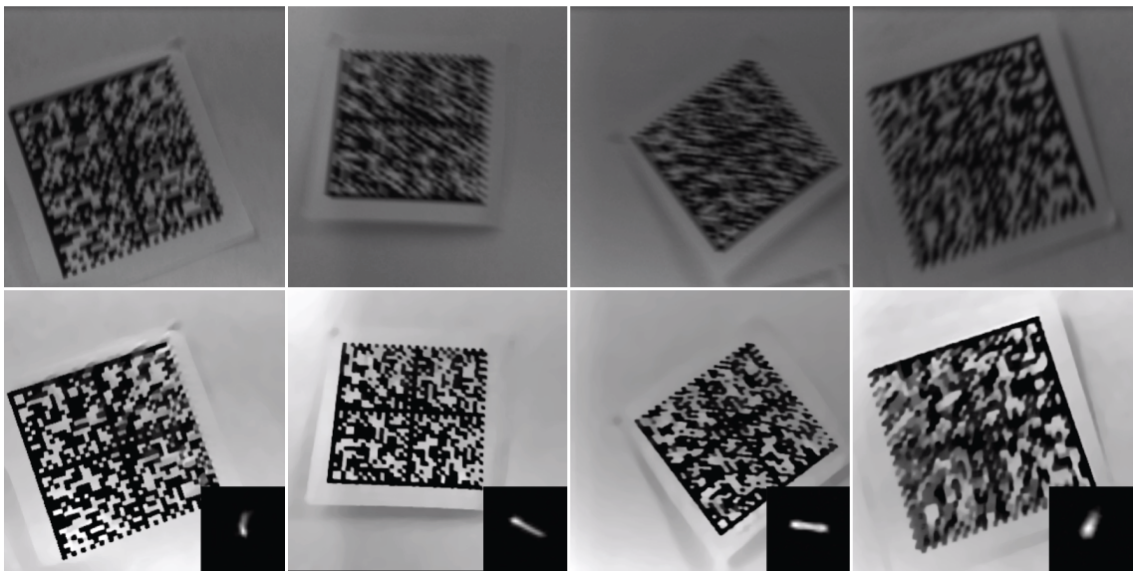


Figure 3.21: Running our algorithm on DataMatrix input without modifications. The top row shows the input images and the bottom row shows the output images and estimated kernels. The first three results can be decoded by the Scandit decoder [331] while the rightmost example fails due to non-uniform blur.

EAN barcodes Figure 3.22 and Figure 3.23 summarize the results of our experiments with a barcode and synthetic motion blurs. The blur kernels are the same as in the synthetic QR experiment, originally from Levin et al. [133]. The kernels are of size from 17×17 to 27×27 pixels, the input images of Figure 3.22 are of size 200×200 pixels, and the input images of Figure 3.23 are of size 300×300 pixels. This means the same kernel causes visually larger blur in Figure 3.22 than in Figure 3.23. Unfortunately, our method performs poorly when the distance between the thin bars is smaller than the blur kernel because the thin lines can mislead the kernel estimation. Our restored barcodes were often less visually appealing than the input images. Nevertheless, the thick bars and the numbers below the bars got sharper. While none of the input images is readable with the ZXing reader [351], some of our restored barcodes became decodable. We note here that the ZXing reader is open source and applies no image enhancement steps other than simple thresholding operations. It is also apparent from the experiments that if the blur is small or if most of the kernel values are concentrated around a dominant spot, the very efficient defocus blur model of the Scandit [331] scanner can decode the barcode (recall the Gaussian shape of a

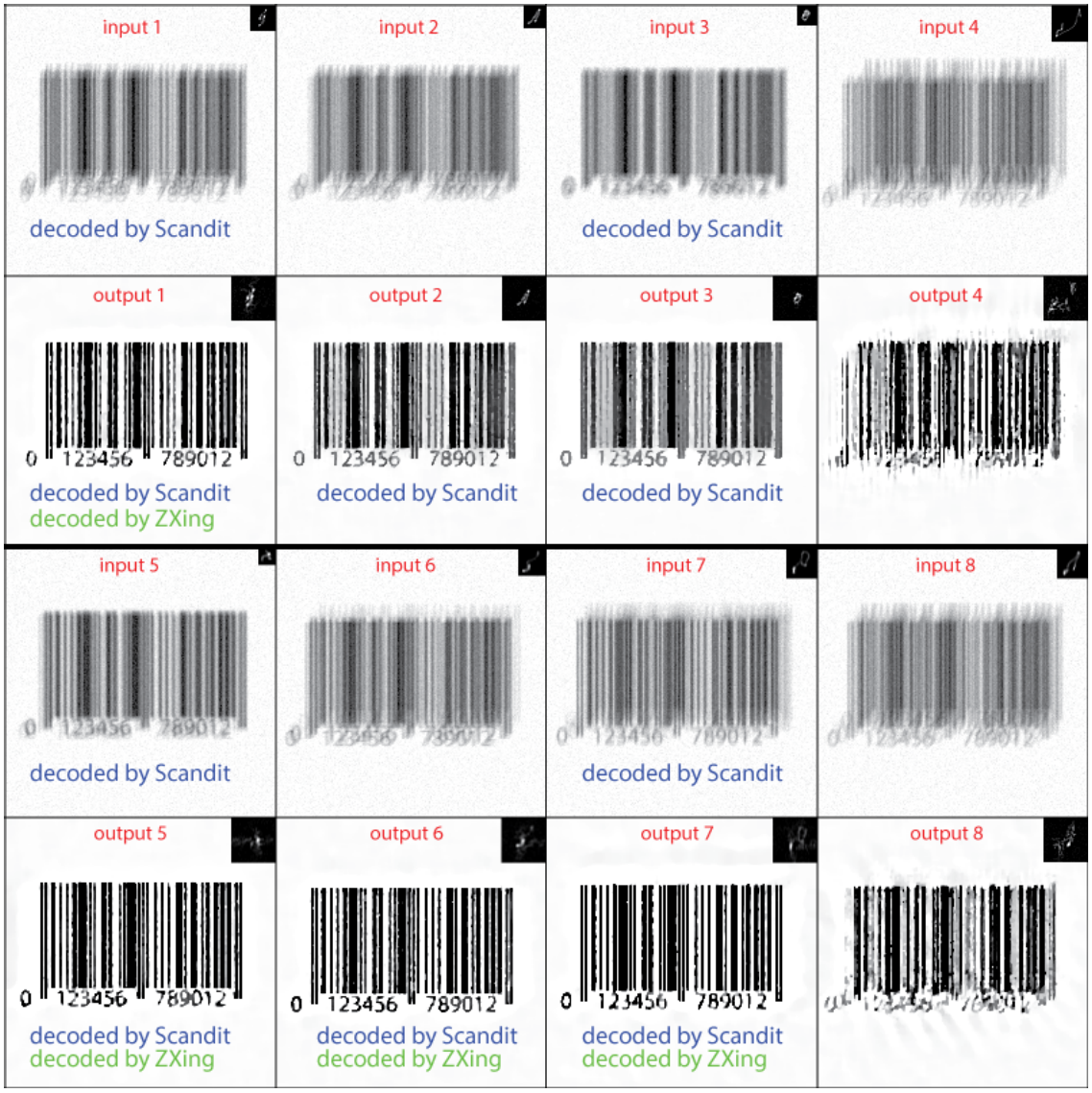


Figure 3.22: Experiment with a 200×200 barcode image and synthetic blurs with the kernels of Levin [133]. Our method can make 4 out of 8 images decodable with the simple ZXing decoder. However, some codes degraded by small (or concentrated) motion blur can be decoded with the Scandit scanner that applies a very efficient defocus blur model.

typical defocus blur kernel). Hence there is actually no need for expensive blind motion blur estimation in most cases when the blur is small, but we can apply one of the fast defocus models (see related work section) instead. A quick estimate on the size of the motion blur can be derived from the inertial sensors as we will describe in Section 3.4. 1D

Figure 3.23: Experiment with a 300×300 barcode image and synthetic blurs with the kernels of Levin [133]. Here the blur is visually smaller than in Figure 3.22. Again, our method makes some barcodes decodable with ZXing, but some codes become even worse than the input, and the Scandit scanner can actually decode the ones where the motion blur can be approximated by a defocus blur model.

barcodes are also vulnerable to false positive detections because they contain only weak error detection by a single checksum character. Therefore, the restoration errors might

result in a false barcode candidate. In contrast, the combined error detection and error correction in QR codes can avoid false detections.

We can conclude that our method is best suited for 2D visual codes with larger modules (w.r.t. typical blur size) and is less robust when applied to codes with smaller modules. In particular, QR codes and AR markers are good representatives while DataMatrix codes and barcodes are not. This behaviour is actually expected from the analysis of Xu and Jia [252] in natural image deblurring. If the code contains only small-scale structures, the kernel estimation is not guaranteed to converge.

3.3.13 Other types of blur

So far, we have focused on decoding motion-blurred codes only, but the QR properties remain the same under other types for blur as well. In our future work, we will investigate the adaptations required in kernel regularization to allow different non-sparse shapes. Here, we briefly show promising preliminary results of our experiments in removing synthetic defocus blur and synthetic upscaling blur.

Removing synthetic defocus blur In theory, it is possible to restore slightly defocused codes if the blur is smaller than the module size in the code. Figure 3.24 left shows a synthetically defocused example using a 178×178 image and a 33×33 Gaussian kernel with standard deviation 3. The image is successfully decoded in 0.32 s; however, some gray artifacts are visible at dense black and white areas of the code. Note the Gaussian-shaped estimated kernel.

Figure 3.24: Left: removing synthetic defocus blur. Note the Gaussian-shaped estimated kernel. Right: Reconstructing a bilinearly 12× upscaled code. Note the square-shaped estimated kernel.

Reading tiny codes Blind deconvolution is also an important step in super resolution algorithms where a downsampling filter needs to be estimated and inverted. The close

connection in the mathematical models suggest that our algorithm might be suitable for super resolving tiny QR codes. We have performed a simulation to justify this (see Figure 3.24 right). In a photo editing software we downscaled a QR code with nearest neighbor interpolation so that one symbol corresponds to only one pixel, and upscaled it to 300×300 pixels ($12\times$) again, using bilinear interpolation. The upscaled code is blurry and not readable. Our algorithm is able to restore and read the code after only two iterations. Note the square shape of the estimated blur kernel in the bottom right corner.

3.3.14 Experiments on smartphones and smartglasses

After successful offline experiments with real motion-blurred images, we ported our algorithm to Android for live experiments. Figure 3.25 illustrates a concrete example with a large QR code and challenging blur restored directly on the Nexus 6 smartphone (Qualcomm Snapdragon 805 SoC with 2.7 GHz Krait 450 CPU and 3 GB RAM). The Android GUI consists of the camera preview and three image views for the input, output, and kernel images, respectively. The camera resolution is set to 720×480, and we added a 300×300 search window on top of the preview and constrain the algorithm to this area. Additionally, there are buttons to capture an image and start the deblurring, and a textbox for logging output. The challenging image is decoded in about 13 s.

We also implemented the algorithm on more constrained Google Glass smartglasses (TI OMAP 4430 SoC with 1.2 GHz CPU and 2 GB RAM). As the tonemap curve cannot be controlled, the recorded images on the Glass do not follow our linear blur model. Due to this limitation of the camera driver, we can report the performance on the Glass only using synthetically blurred images. Deblurring a 300×300 image on the Glass takes 8.54 s (using a single CPU core), which is about $2.5\times$ slower than deblurring the same image on the smartphone. The Android debugger reports the use of 34.9 MB of memory during deblurring, which we find acceptable on a wearable device. More experiments with smartphones and smartglasses can be found in Figure 3.26 and in the supplementary video (see Appendix).

In summary, we have shown that our proposed algorithm achieves good reconstruction quality even on severely blurred QR codes and outperforms existing methods in terms of speed. Our method requires no parameter tuning except a rough estimate of the kernel size which we have set to 33×33 pixels in all our experiments. In the following section, we provide a simple yet effective method to quickly estimate the initial size of the blur from the inertial sensors of a wearable scanner device.

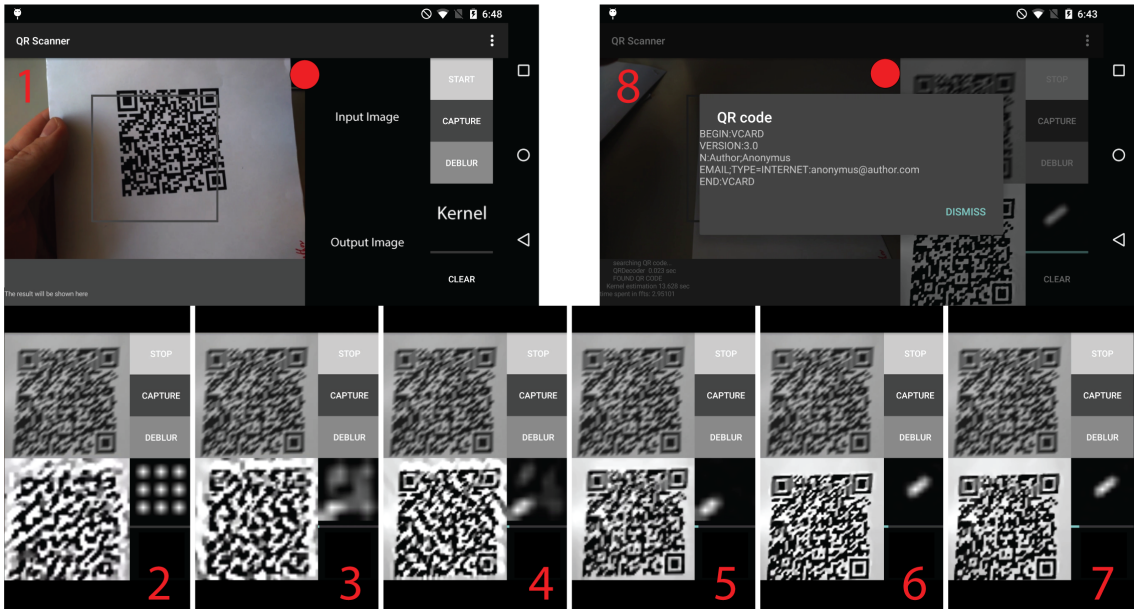


Figure 3.25: Android user interface: This example shows removing large motion blur from a QR code in 13.638 s directly on a smartphone. The input is replicated in the top row and intermediate steps are shown in the bottom row.

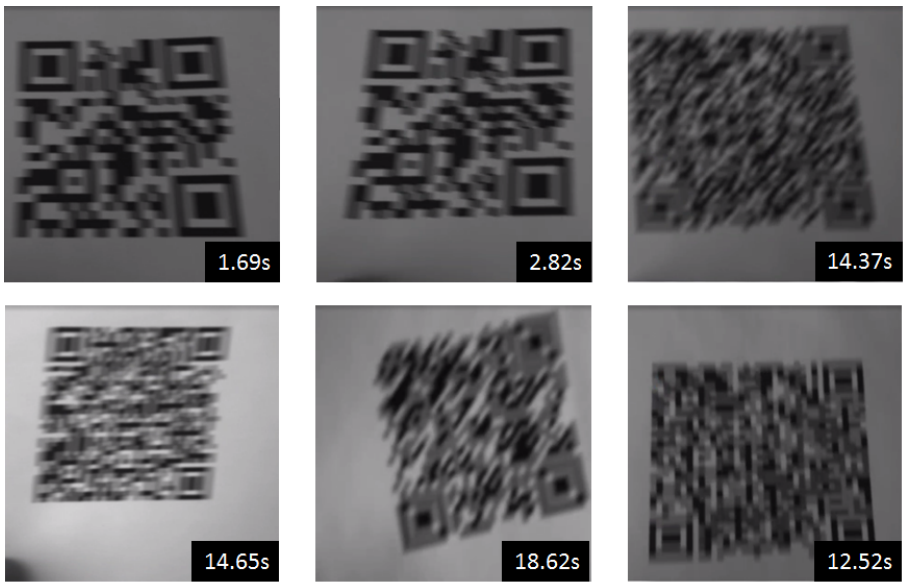


Figure 3.26: Challenging real blurred examples that our algorithm is able to decode directly on a smartphone. The number of iterations depends on the blur size, the total runtime is indicated in the respective image corners. While our method is currently too slow for real-time applications, it is significantly faster than other existing approaches.

3.4 Fast blur estimation from inertial sensors

Further speedup in the deblurring can be achieved with inertial sensors that are common in wearable devices. The sensor stream captured together with the camera stream allows for reconstructing the camera motion during the exposure time which in turn relates to the (non-uniform) blur in an image. This idea is inspired by high-end cameras that compensate for handshake blur via optical image stabilization techniques: Vibration Reduction (Nikon), Image Stabilization (Canon), Optical Stabilization (Sigma), etc., by moving the lens or the image sensor based on inertial sensor measurements. As those hardware technologies are not common for small wearable cameras, we need to perform the enhancements in software.

In Section 3.2, we have discussed deblurring approaches that aid the kernel estimation with motion sensors in custom hardware. With unmodified wearable devices, however, there are a number of practical issues that prohibit accurate blur estimation from the sensors alone. We aim at interactive frame rates which implies a trade-off between restoration quality and speed, and limits the applicability of many approaches. The required hardware modifications and the long restoration time are the two main reasons why none of the existing gyro-based methods fits our purposes. Instead of relying solely on the inertial sensors for deblurring as previous work, we propose to quickly initialize the blind deblurring algorithm from the built-in sensors, while the initial errors are compensated in our edge-based blind kernel estimation loop.

3.4.1 Method overview

We can make the following observations illustrated in Figure 3.27: *i)* The blurry image of a point light source in the scene (such as distant street lights at night) gives the motion blur kernel at that point of the image *ii)* smartphones today also provide a variety of sensors such as gyroscopes and accelerometers that allow reconstructing the full camera motion during camera exposure thereby giving information about the blur process. *iii)* We can get the blur kernel at any point in the image by placing a virtual point light source in front of a virtual camera and replaying the camera motion.

Figure 3.28 gives an overview of our sensor-aided (non-uniform) deblurring approach. First, piecewise uniform blur kernels $\mathbf{K_{ij}}$ along rows i and columns j of the image are estimated from the sensor measurements. The generated kernel(s) are used as initialization in our restoration-recognition loop which speeds up the blind blur estimation process. Within the loop, individual patches are restored from using visual code image priors. Finally, the sharp output $\mathbf{I}$ is assembled from the deblurred patches. When creating

Figure 3.27: Left: The blur is "encoded" in the image of point light sources. Middle: We can reconstruct the camera motion from inertial sensor measurements. Right: We can get the blur kernel by replaying the motion with a virtual camera looking at a virtual point light source.

Figure 3.28, we performed only a single iteration of the image estimation step to illustrate the immediate effectiveness of sensor-based kernel estimation.

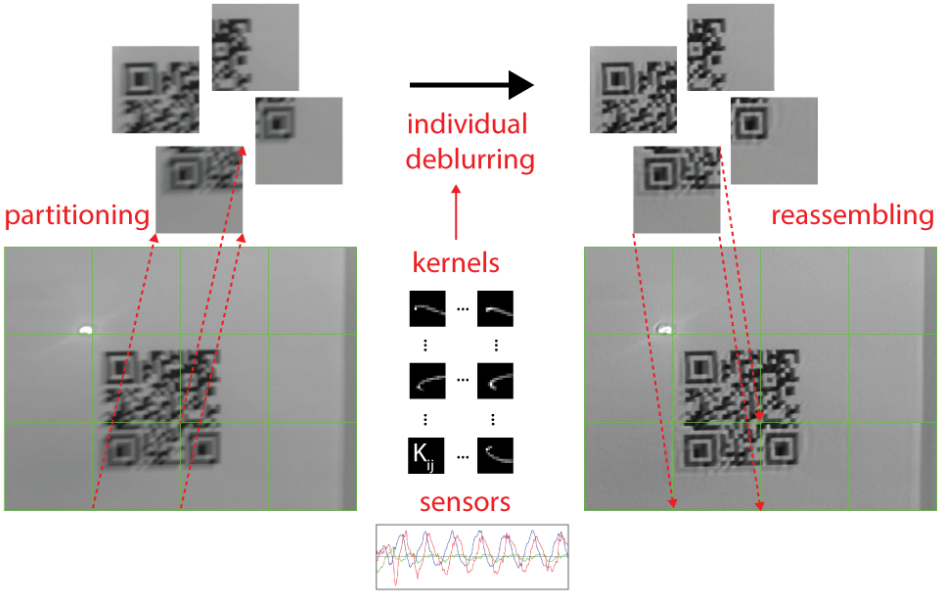


Figure 3.28: Illustration of our gyroscope-aided (non-uniform) blur removal approach.

Typically, wearable computers are equipped with both an accelerometer and a gyroscope. There are, however, a couple of difficulties in using the accelerometer for blur estimation. First, the accelerometer measures linear acceleration, so in order to get the translation of the camera, a double integration is necessary which amplifies the sensor noise. The amplified noise can lead to large errors in kernel estimation. Second, gravity compensation is difficult with complex motions. Third, and most important, is the fact that translational blur also depends on the scene depth so we cannot get the blur kernels only from the sensors. In

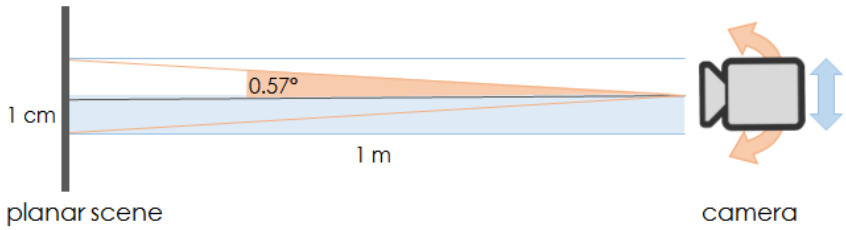


Figure 3.29: Given a scene at 1 m distance, a slight rotation of 0.57° is enough to produce the same blur as 1 cm translation. If the scene is 3 m away, then already 0.19° rotation has the same effect. From this simple example, we see why the blur effect of camera rotation is much more dominant than of camera translation.

contrast, gyroscopes measure rotational velocity and the values need to be integrated only once to get the rotation of the camera. Moreover, rotational blur is independent of scene depth so there is no need for difficult depth estimation or energy-consuming depth sensors. As we are only interested in the camera motion within the shutter interval, the gyroscope bias can be neglected because we only integrate a couple of measurements. As also illustrated in Figure 3.29, rotational blur is dominant in hand shake, especially when the scene is further away. This allows to restrict ourselves only to the gyroscope measurements for blur estimation.

3.4.2 Modeling camera motion and blur

The traditional convolution model for uniform blur is written in matrix-vector form as

$$\vec{\mathbf{B}} = \mathbf{A}\vec{\mathbf{I}} + \vec{\mathbf{N}} \tag{3.8}$$

where $\vec{\mathbf{B}}$,$\vec{\mathbf{I}}$,$\vec{\mathbf{N}}$ denote the vectorized blurry image, sharp image, and noise term, respectively, and $\mathbf{A}$ is the sparse blur matrix. Camera shake causes non-uniform blur over the image, i.e., different parts of the image are blurred differently. We assume piecewise uniform blur and use different uniform kernels for each image region which is a good compromise between model accuracy and model complexity.

The blur kernels across the image can be found by reconstructing the camera movement, which is a path in the six-dimensional space of 3 rotations and 3 translations. A point in this space corresponds to a particular camera pose, and a trajectory in this space corresponds to the camera movement during the exposure. From the motion of the camera and the depth of the scene, the blur kernel at any image point can be derived. We target unmodified

smartphones without depth sensors, so we need to make further assumptions about the scene and the motion.

Similar to Joshi et al. [96], we assume the scene to be static and planar (or sufficiently far away from the camera) so that the blurred image can be modeled as a sum of transformations of a sharp image. The transformations of a planar scene due to camera movements can be described by a time-dependent homography matrix $\mathbf{H_t} \in \mathbb{R}^{3\times3}$. We apply the pinhole camera model with square pixels and zero skew for which the intrinsics matrix $\mathbf{K}$ is defined as

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f is the focal length and $[c_x, c_y]^T$ is the principal point of the camera.

Given the rotation matrix $\mathbf{R}_t$ and the translation vector $\mathbf{T_t}$ of the camera at a given time t, the homography matrix is defined as

$$\mathbf{H}_t(d) = \mathbf{K}(\mathbf{R}_t + \frac{1}{d}\mathbf{T}_t\vec{n}^T)\mathbf{K}^{-1} \tag{3.9}$$

where $\vec{n}$ is the normal vector of the latent image plane and d is the distance of the image plane from the camera center. The homography $\mathbf{H}_t(d)$ maps homogenous pixel coordinates from the latent image $\mathbf{I}_0$ to the transformed image $\mathbf{I}_t$ at time t:

$$\mathbf{I}_t\left((u_t, v_t, 1)^T\right) = \mathbf{I}_0\left(\mathbf{H}_t(d)(u_0, v_0, 1)^T\right) \tag{3.10}$$

The transformed coordinates in general are not integer valued, so the pixel value has to be calculated via bilinear interpolation, which can also be rewritten as a matrix multiplication of a sparse sampling matrix $\mathbf{A}_t(d)$ with the latent sharp image $\vec{\mathbf{I}}_0$ as $\vec{\mathbf{I}}_t = \mathbf{A}_t(d)\vec{\mathbf{I}}_0$ in vector form. Then, assuming constant illumination in the scene during exposure time, we can describe the blurry image as the integration of all transformed images during the exposure time plus noise:

$$\vec{\mathbf{B}} = \int_{t_{open}}^{t_{close}} \mathbf{A}_t \cdot \vec{\mathbf{I}} dt + \vec{\mathbf{N}} = \mathbf{A} \cdot \vec{\mathbf{I}} + \vec{\mathbf{N}} \tag{3.11}$$

Note how this expression resembles the form of Equation 3.8. While for this formula the depth of the scene is required, a common simplification is to assume zero translation [59, 73, 101] because rotation has a significantly larger impact on shake blur [19, 118, 246]. With only rotational motion, Equation 3.9 is no longer dependent on the depth:

$$\mathbf{H}_t = \mathbf{K}\mathbf{R}_t\mathbf{K}^{-1} \tag{3.12}$$

Handling pixel-wise spatially varying blur is computationally too complex to perform on a smartphone, so we adopt a semi-non-uniform approach. We split the images into $R \times C$ overlapping regions (R and C are chosen so that we have regions of size 30×30 pixels) where we assume uniform blur and handle these regions individually. We reconstruct the motion of the camera during the exposure time from the gyroscope measurements and from the motion we reconstruct the blur kernels for each image region by transforming the image of a point light source with the above formulas. Once we have the blur kernels, fast uniform deblurring algorithms can be applied in each region, and the final result can be reassembled from the deblurred regions (recall Figure 3.28).

3.4.3 Camera-IMU calibration

Reconstructing the motion of the mobile phone during camera exposure of a given frame i with timestamp t_i requires the exact time window of sensor measurements during that frame. This is challenging to find on unmodified smartphones given that current smartphone APIs allow rather limited hardware control. We denote with t_d the delay between the recorded timestamps of sensor measurements and camera frames which we need to estimate prior to deblurring.

Synchronization of the camera and the IMU data is an important open issue in sensor-based blur estimation because the mobile operating systems do not provide precise timestamps. Existing methods estimate the unknown parameters (time delay, frame rate, rolling shutter fill time, gyroscope drift) from a sequence of images off-line via optimization. We have found that the camera parameters might even change over time, for example the smartphones automatically adjust the frame rate depending on whether we capture a bright or a dark scene. This is an important issue because it means we require an online calibration method. Jia and Evans [94] proposed such an online camera-gyroscope calibration method for smartphones based on an extended Kalman filter (EKF). The method tracks point features over a sequence of frames and estimates the time delay, the rolling shutter fill rate, the gyroscope drift, the physical sensor offset, and even the camera intrinsics. However, it requires sharp frames for feature tracking.

Furthermore, the rolling shutter in CMOS image sensors introduces a small time delay in capturing different rows of the image that causes image distortions. The pixel values are read out row-wise from top to bottom which means 'top' pixels in an image will be transformed with 'earlier' motion than 'bottom' pixels, which has to be taken into account in our model (see Figure 3.30. For an image pixel $u = [u_x, u_y]^T$ in frame i the start of the exposure is modeled as

$$t([u_x, u_y]^T, i) = t_i + t_d + t_r \frac{u_y}{h} \tag{3.13}$$

where t_r is the readout time for one frame and h is the total number of rows in one frame. The gyro-camera delay t_d is estimated for the first row of the frame, and the other rows are shifted in time within the range $[0, t_r]$. We set the time of each image region to the time of the center pixel in the region.

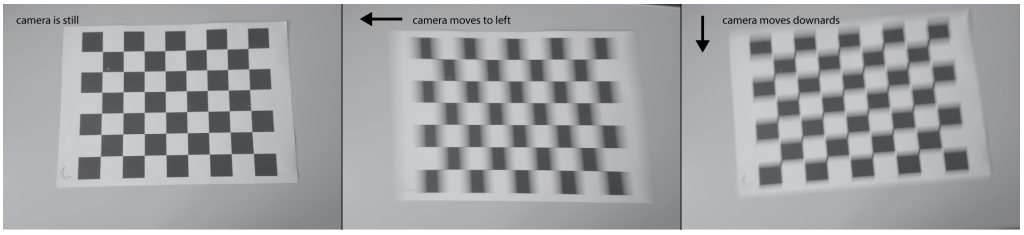


Figure 3.30: The rolling shutter introduces image distortion when the camera (or the object) is moving during exposure because the bottom part of the image is captured later than the top part. When the camera moves to the left, the bottom part of the checkerboard is sheared to the right. When the camera moves downwards, the squares are squeezed vertically. Our motion blur model must take these effects into account.

To find the unknown constants of our model, we apply the Extended Kalman Filter (EKF)-based online gyro-camera calibration method of Jia and Evans [93, 94] once at the beginning of the sequence. This method estimates the rolling shutter parameter t_r, the camera intrinsics f, c_x, c_y, the relative orientation of the camera and IMU, the gyroscope bias, and even the time delay t_d. The intrinsics do not change in our application, and the relative orientation is not important because we are only interested in rotation changes which are the same in both coordinate systems. The gyroscope bias is a small and varying additive term on the measured rotational velocities which slightly influences the kernel estimation when integrated over time. However, for kernel estimation we consider only rotation changes during single camera frames, and in such short time intervals the effect of the bias can be neglected. For example, in case of a 30 Hz camera and a 200 Hz gyroscope we integrate only $200/30 \approx 6$ values during a single frame. We perform the online calibration once at the beginning of our image sequences and we assume the above parameters to be constant for the time of capturing the frames we intend to deblur. The EKF is initialized with the intrinsic values given by the camera calibration method in OpenCV.

The time delay t_d was found to slightly vary over longer sequences, so after an initial guess from the EKF, we continuously re-estimate t_d in a background thread. We continuously calculate the mean pixel shift induced by the movement measured by the gyroscope, and we also observe the mean pixel shifts in the images (see Figure 3.31). The current

t_d is found by correlating the curves in a sliding time window. This calibration method works only with far away scenes so that the camera's translational motion is negligible.

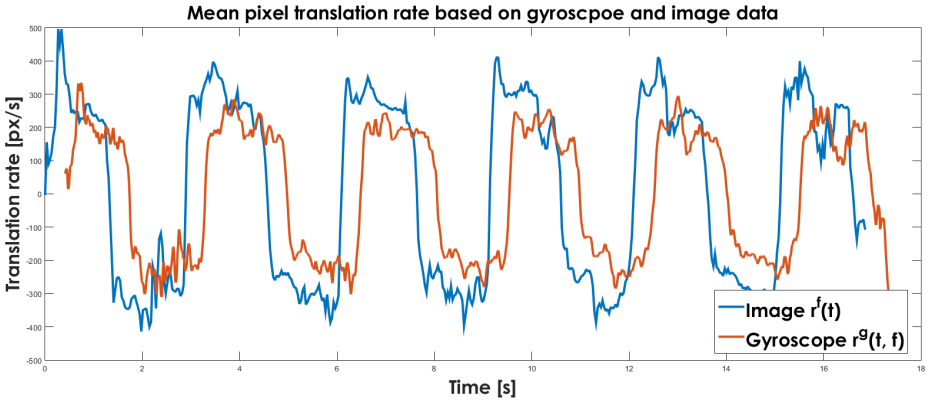


Figure 3.31: Our continuous calibration method for the time delay parameter compares the mean pixel translation rates observed in the images and generated from the gyroscope measurements. The time delay t_d can be found by matching the curves in horizontal direction, and the focal length f can be found by matching the two curves in vertical direction. Figure adapted from Luc Humair.

The last parameter, the exposure time $t_e = t_{close} - t_{open}$ of a frame can be read from the EXIF tag of JPEG images like in [200], but an EXIF tag is not available for live video frames. Therefore, we lock the camera settings at the beginning and capture the first frame as a single image.

3.4.4 Kernel estimation from gyroscope measurements

We generate a synthetic blur kernel at a given point in the image by replaying the camera motion with a virtual camera that is looking at a virtual point light source. For any pixel $u = [u_x, u_y]$ in the image, we place the point light source to $U = [(u_x - c_x)\frac{d}{f}, (u_y - c_y)\frac{d}{f}, d]$ in 3D space. Note that the value of d is irrelevant if we consider rotations only.

First, we need to rotate all sensor samples into a common coordinate system because the raw values are measured relative to the current camera pose. The chosen reference pose is the pose at the shutter opening. Next, the measured angular velocities need to be integrated to get rotations. As described in Section 3.4.3, we neglect the effects of gyroscope bias within the short time of the exposure. In order to get a continuous rendered kernel, we super-resolve the time between discrete camera poses where measurements

exist, using spherical linear interpolation (SLERP). The transformed images of the point light source are blended together with bilinear interpolation and the resulting kernel is normalized to sum to 1. Finally, we crop the kernel to its bounding square in order to reduce the computational effort in the later deblurring step. Note that this method allows to estimate the blur at each pixel of the image. For the sake of restoration speed, we restrict ourselves to the locally uniform blur model and generate only the blur kernels at predefined locations on a grid over the image.

3.4.5 Implementation

We have implemented the proposed method in OpenCV [313]. We recorded grayscale camera preview sequences of resolution 720×480 at 30 Hz together with gyroscope data at 200 Hz on a Google Nexus 4 smartphone. Our tests were conducted offline on a PC but each component of our algorithm is portable to a smartphone with little modifications. The gyro-camera calibration is performed on the first few hundred frames with Jia's implementation [94] in Matlab. We found that this calibration method sometimes converges to incorrect values so we also allow the user to adjust the t_d and t_r values manually. A robust online calibration method is out of the scope of this thesis. Given the rendered piecewise uniform blur kernels, we start the fast blind QR restoration method from Section 3.3. For smoothing discontinuities that may produce ringing artifacts, we perform edge tapering on the overlapping regions before deblurring.

3.4.6 Evaluation of kernel generation

To test the accuracy of kernel generation, we recorded a sequence in front of a point light grid, and also generated the kernels from the corresponding gyroscope data. Ideally, the recorded image and the generated image should look identical, and after (single-frame) deblurring using the generated kernels the resulting image should show a point light grid again. Figure 3.32 illustrates that our kernel estimates are close to the true kernels; however, the bottom part is not matching perfectly because of residual errors in the online calibration. Nevertheless, the accuracy of the generated kernels is enough for initialization of the blind deblurring component.

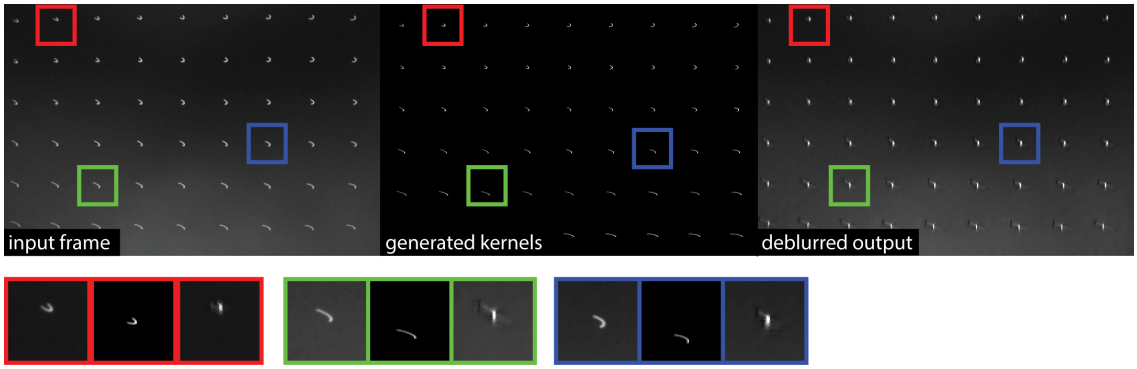


Figure 3.32: Kernel generation example. Left: blurry photo of a point grid. Middle: kernels estimated from gyroscope data. Right: the deblurred image is close to a point grid again.

3.4.7 Removing the blur

We also evaluate the proposed method on images of real blurred QR codes close to the camera. Figure 3.33 illustrates our experiment. We also added a laser spot next to the static code and moved the camera during exposure. The captured laser streak gives a hint about the true blur kernel. Ideally, with perfect deblurring, the laser streak should become a single dot after restoration. We generated local blur kernels from sensor data. The right column illustrates the generated kernels at selected tiles. We then run a *single iteration* of our image estimation step (see details in Section 3.3.2) to get sharper estimates. The generated kernels are close to the true kernels, therefore we can significantly improve the quality of the images within a single iteration only. One of the output images is actually already readable. This means, when incorporating sensor data in our algorithm, the expensive multiscale blind kernel estimation can be skipped and we can immediately continue with kernel estimation on the highest scale. The blind kernel estimation algorithm then quickly finds the solution that gives the sharpest result.

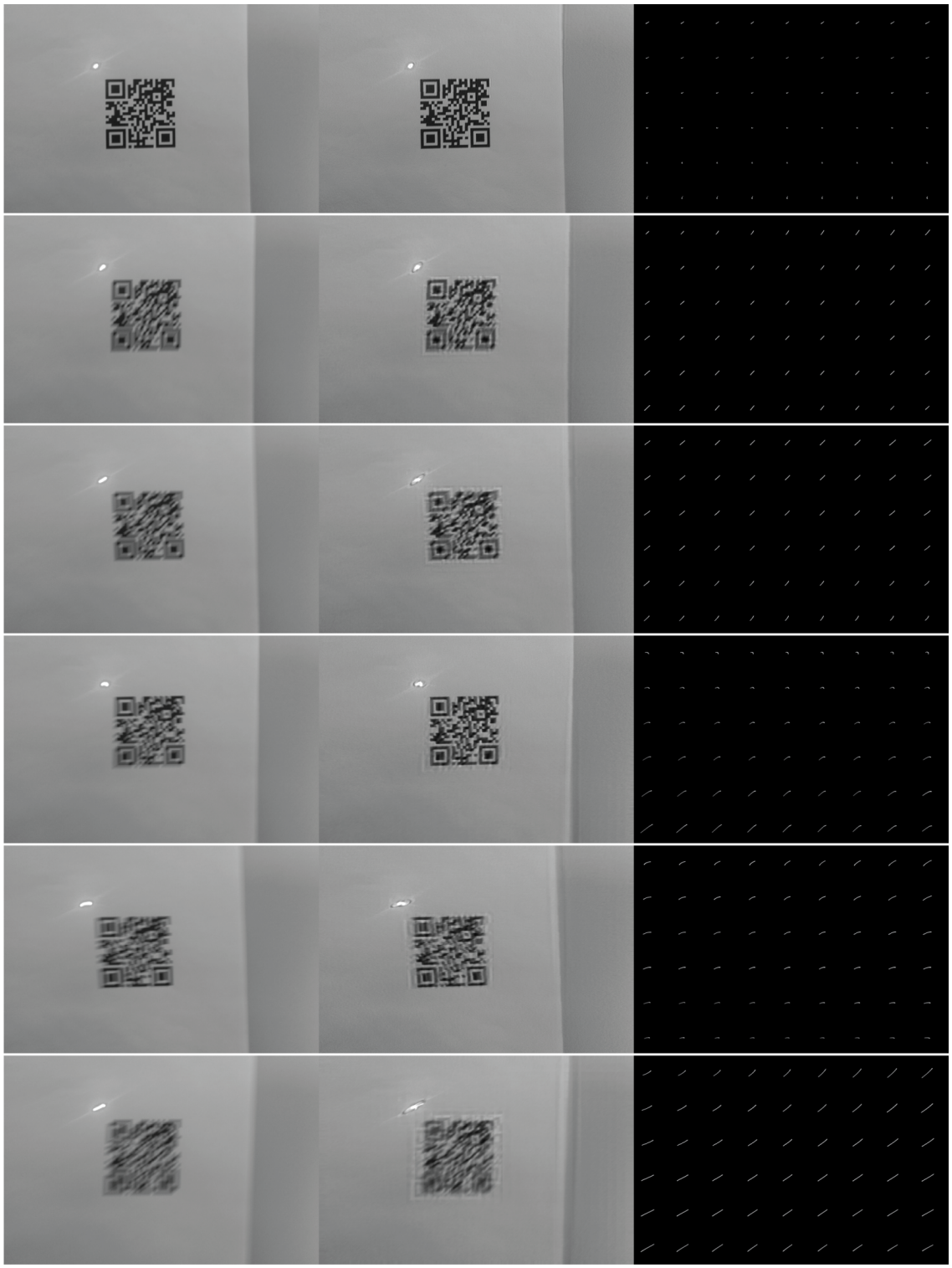


Figure 3.33: Experiment with real blurred images as input (left column). We generate local blur kernels from sensor data (right column) and run one iteration of our image estimation step to get sharper estimates (middle column).

3.5 Discussion

3.5.1 Target devices

As shown in the experiments, our QR restoration algorithm requires a fast processor, moderate amount of memory (less than 40 MB without strict optimizations), and a camera with a suitable driver that allows manual control. As usually more complex applications are built on top of barcode scanning, the high CPU requirements restrict the applicability of our approach to high-end wearable devices. Slower wearable devices (e.g., a smartwatch or a life logger camera) could send the captured image to the user's smartphone which we believe is the most powerful wearable device nowadays. A cross-device barcode scanning solution (i.e., smartwatch camera, smartphone processor, smartglasses display) should be further explored in future research. However, distributing the computation on many individual devices would produce too much communication overhead. In the future, the algorithm could greatly benefit from embedded processors with fast floating point calculations and/or DSP support with hardware FFT features.

3.5.2 Impact of image resolution

An important setting of the algorithm is the image size. We chose a 300×300 search window with the common 720×480 preview resolution. This corresponds to a convenient 15 cm scanning distance for a 5×5 cm code, a typical QR shopping scenario with the smartphone. The Glass camera has a wider field of view, so with the same resolution and search window, the code must be placed closer to the camera. In order to match the search window with a smaller code that the user holds further away, the camera resolution needs to be increased. Doubling the camera resolution means the code is visible from twice the distance, but then a twice higher resolution blur kernel is required to represent the same shake blur. For example with preview resolution 1280×960 on the Google Glass, the 300×300 search window fits a 5×5 cm code in 35 cm distance. As only the search window is processed, the actual camera resolution makes no difference in the image estimation, but the higher resolution makes a big difference in the blur kernel estimation.

It is also important that we pick a tight, but not too tight initial estimate of the kernel size. In the implementation used in our experiment, we represented all kernels by a 33×33 matrix. If the chosen kernel size is too small, parts of the kernel might be lost or clipped (cf. our automatic centering method), and there is little chance to recover the complete image with a truncated kernel. Even worse, the false kernel will cause severe ringing artifacts propagating the errors further in the loop which might lead to a wrong local optimum

or no convergence at all. Choosing the kernel size too big is no problem in the sense of finding the optimum (note that the solution will be filled with zeros around the true kernel) but it has a huge impact on the runtime. A larger kernel implicitly leads to larger matrices and longer processing time because significantly more variables are involved in the optimization. In summary, the image resolution, the blur size in pixels, and thus the kernel resolution are strongly coupled and the latter one needs to be chosen carefully, and therefore our quick initialization scheme from inertial sensors is an important module of an efficient scanner.

3.5.3 Speed optimizations

The runtime of the blind restoration algorithm depends on several factors. Figure 3.34 illustrates the time spent on image estimation and kernel estimation on different scales when the early detection is switched off. The image estimation and the kernel estimation steps contribute similarly to the total time per scale. The complexity of image estimation grows exponentially with the image size, and the complexity of blur estimation grows exponentially with the blur kernel resolution. The complexity of FFT is known, the complexity of conjugate gradients can be estimated, but the number of iterations and scales until decoding depends on the actual blur shape. While our Android tests are still far from real time, we presented some extremely large blurs that might be unrealistic assuming that the user wants to scan the code. Smaller blur is faster to estimate because less iterations are needed. As the total time per scale grows exponentially, the importance of a fixed search window and early QR detection is indisputable.

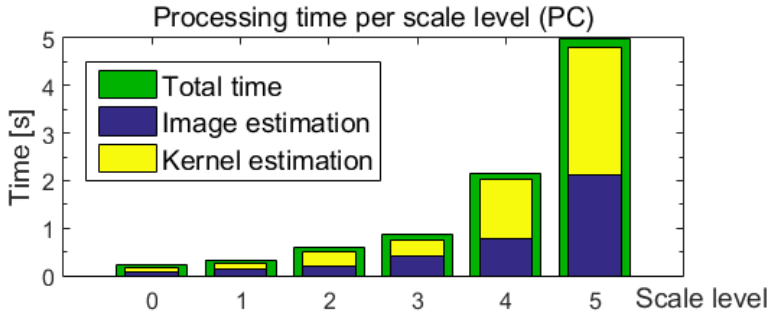


Figure 3.34: Time spent on image and kernel estimation on different scales (PC), with the QR decoder disabled footnoteIn this example, the QR code would be successfully decoded after 3 scales..

The runtime of the patchwise restoration algorithm depends mainly on the size of the input images. In fact, we perform $R \times C$ non-blind deconvolutions on patches but as

the patches are overlapping, we process somewhat more pixels than the image contains. In Figure 3.35, we compare the effect of grid resolution on deblurring quality on a real blurred image. We generate local kernels in 24×36, 12×18, 4×6, and 2×3 locations, respectively. We then run one iteration of the image estimation step. Given the input image resolution of 720×480, there is no significant quality difference when we choose a grid resolution higher than 4×6. The necessary grid resolution also depends on how non-uniform the blur is, but this can also be estimated from the sensors by setting simple thresholds. It seems, however, important that the whole code is deblurred with the same kernel otherwise we introduce new errors in the kernel estimation (see bottom right). This experiment justifies our choice of a 300×300 search window for the blind deblurring algorithm.

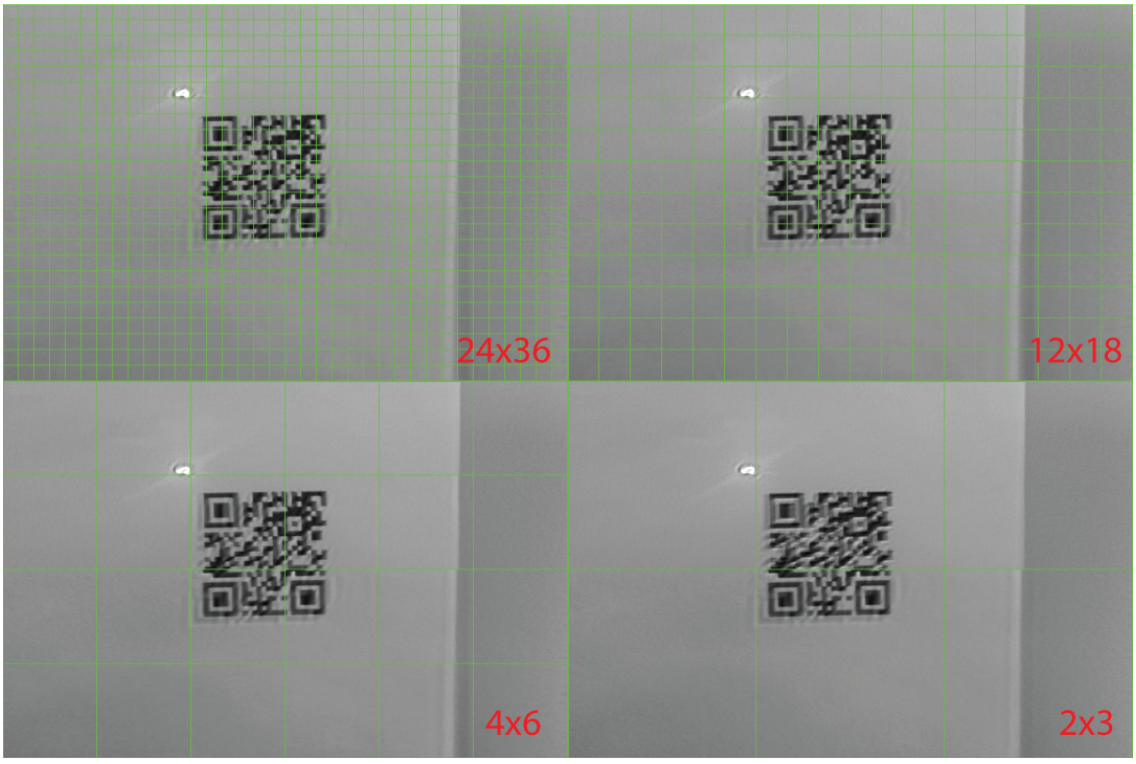


Figure 3.35: Comparison of the effect of grid size. The image resolution is 720×480 and we generate local blur kernels in 24×36, 12×18, 4×6, and 2×3 locations, respectively. There is no significant quality difference when we choose a grid resolution higher than 4×6.

The runtime of the whole algorithm could be further reduced. We provided speed measurements of our single-threaded implementation, but as the algorithm contains a large number of FFTs, it could be greatly accelerated with parallel processing on multiple cores or even mobile GPUs or DSPs.

3.5.4 Failure cases

Our QR restoration algorithm can estimate and remove only uniform blur, therefore it fails in case of significant rotational motion or strong rolling shutter distortions in the image (see video supplement in Appendix). We also assume that the wearable device view is close to perpendicular to the visual code and that the user actually wants to scan the code. Blind non-uniform deblurring is computationally too demanding [245] to perform on current generation mobile devices. However, the uniform blur assumption is usually valid in the search window of our user interface if the code occupies a smaller region of the image.

We assume QR-specific gradient statistics in the whole restoration window which is violated if the code is placed in front of a complex background. To succeed, either the background must be a plain color, or the code must be segmented in a preprocessing step, for which algorithms do exist [209].

3.5.5 Issues with sensor data

As shown in the experiments, the proposed algorithm is able to restore even slightly non-uniformly blurred images. However, there are limitations and assumptions we need to keep in mind. Our image estimation step requires a good kernel estimate so a good calibration is crucial for success. The selected gyro-camera calibration method is sensitive to the initialization values which are not trivial to find for different smartphone models.

We need to assume that a short sequence with detectable feature tracks for calibration before deblurring exists. However, the calibration does not need to be done every time but only when the camera settings change. We expect that future camera and sensor APIs like StreamInput [337] will provide better synchronization capabilities that allow precise calibration.

As the camera is not exactly in the middle of the device, the rotation coordinate frame is not exactly in the middle of the image and the generated kernels are inherently slightly wrong even with perfect synchronization. Therefore it is important that we do not directly use the generated kernel for deblurring but instead as a quick initial estimate for further blind deconvolution.

Our sensor-aided blur model can generate rotational motion blur kernels at any point of the image, but the model is incorrect if the camera undergoes significant translation or if objects are moving in the scene during the exposure time. If not the camera but the code is moving, the sensor readings cannot help in the initialization but the QR restoration should

perform equally good with longer runtime. However, if both the code and the camera are moving, the sensor-based estimate is wrong and may mislead the optimization. In extreme cases, deblurring with the gyro-generated kernels directly may actually degrade the quality of the images. We can rely on user input for this special case.

3.5.6 Camera response function

Cameras internally use a non-linear function that converts the scene irradiance to pixel intensities, which can be modeled by a so called camera response function (CRF) for each color channel. The non-linear CRF has a significant impact on deblurring algorithms because our blur models assume a linear relationship [224]. The reason why manufacturers apply a non-linear function is to compensate the non-linearities of the human eye and to enhance the look of the image. The CRF is different for each camera model and even for different capture modes of the same camera [251]. Some manufacturers disclose their CRFs but for the wide variety of smartphones only few data is available. The CRF of digital cameras can be calibrated for example using exposure bracketing [46] but the current widespread smartphone APIs do not allow exposure control. New camera APIs on iOS since version 6, and on Android since version 5.0, and custom APIs such as the FCam [163] allow more control over the camera but are only available for a very limited set of devices. To overcome this limitation on devices where the linear RAW image data is not accessible, we follow the approach of [136] and assume the CRF to be a simple gamma curve with exponent 2.2. While this indeed improves the quality of our deblurred images, an online photometric calibration algorithm that tracks the automatic capture settings remains an open question.

3.5.7 Color images

Our algorithms were formulated for grayscale images which is sufficient for typical barcode applications, but needs to be extended for colored barcodes. The extension to color images would be possible by solving the grayscale problem for the RGB color channels separately; however, the color optimizations of the smartphone driver may introduce different non-linear CRFs for each channel, which needs to be handled carefully. The calibration of the per-channel CRFs using standard methods will become possible with smartphone APIs that allow low-level exposure control.

3.5.8 Multiple input images

In our proposed methods, we have not made use of the fact that in barcode scanning, we have access to a series of degraded images from the camera stream which could be exploited in multi-frame deblurring algorithms. However, those require precise image alignment which is difficult under arbitrary blur and potential rolling shutter distortions. Our attempts in multi-frame deblurring failed at the blurry image registration step, the registration results were often worse than the original input images. This is indeed a difficult problem and existing multiframe methods work on simulated blurry images, or already assume a perfect alignment of the inputs. As we can also estimate the full camera motion between the frames from the sensors, image alignment could be done with the aid of sensor data instead of pure feature matching but then the gyroscope bias needs to be continuously compensated.

Another possibility is to limit the image alignment and also the kernel estimation solely on the finder patterns in the three corners of the QR code as these parts contain the cleanest edges in the image. We know how a perfect finder pattern must look like, so once a blurred finder pattern is found, comparing the two makes the alignment and kernel estimation simpler (cf. [232]). As convolution is commutative, the kernel is given by non-blind deconvolution of the blurry pattern with the sharp pattern. However, a method for robust localization of the blurry finder pattern under arbitrary blurs remains an open question. A detailed analysis of the challenging image registration and joint deblurring problem is an important aspect of future work.

In some cases, the camera might capture a sharp image during processing, so it might be advantageous to run a simple and fast decoder in parallel with deblurring. In our presented methods, we focus on decoding a single blurry code because often there is no possibility to retake an image. This is the case in low lighting conditions, or when the code is moving. For example, it is impossible to capture a sharp image in the conveyor belt setting due to fast motion, or with a wearable camera in a darker room due to long exposure time.

3.6 Conclusions

We have shown that deblurring visual codes differs from deblurring general photographs and identified potential improvements to bring fast restoration to visual codes. First, we have presented a fast and robust restoration-recognition algorithm for scanning motion-blurred QR codes on mobile and wearable devices. Our method blindly estimates the blur from the salient edges of the code in an iterative optimization scheme, alternating between image sharpening, blur estimation, and decoding. The restored image is constrained to

exploit the properties of QR codes which ensures fast convergence. The checksum of the code allows early termination when the code is first readable and precludes false positive detections. Second, we have rendered piecewise uniform blur kernels from gyroscope measurements of the wearable device to initialize our blur removal algorithm so that less iterations are required in the multiscale restoration loop. We have compared our QR restoration approach with other deblurring methods from the literature. The proposed algorithm achieves good reconstruction quality on QR codes and outperforms existing methods in terms of speed. This speed gain makes our approaches particularly suitable for mobile applications. We have presented PC and Android implementations of a complete QR scanner and evaluated the algorithm on synthetic and real test images. While our method currently does not achieve real-time performance, we believe it has the potential of practical applicability in the close future.

Chapter 4

Gesture control

In this chapter, we describe how wearable computers and tiny visual codes enable universal interaction with smart appliances. We generalize the role of a universal remote control from smartphones to any wearable computer. We then elaborate on what new possibilities the combinations of wearables open, ask what the best way to interact with these devices is, and review the literature on natural ways of input to wearable computers. Then, we present a new method for recognizing in-air hand gestures on unmodified smartphones, smartwatches, and smartglasses. Parts of these contributions result from joint work with others at ETH Zurich. In particular, parts of the text and illustrations are taken from joint publications with Simon Mayer [145], Jie Song, Fabrizio Pece, and Otmar Hilliges [203, 204, 205]. Contributions of others are marked at the respective parts in the text and listed in a detailed contribution statement in Section 5.1.5. The proposed interaction methods are not restricted to barcode scanning but are in general well suited for many wearable applications. We present demonstrator scenarios for interaction both with general digital content and with tagged appliances.

4.1 Introduction

The recent advances in sensing, processing, and display technologies have enabled powerful personal wearable computers to fit in our hands, in our pockets, and even on our head. Smartphones and tablets are with us throughout the day, smartwatches and smartglasses are getting popular. These devices have become our ubiquitous communication devices [243], and most importantly, they are able to enhance our memory [23, 189, 214, 215] and our senses [91, 92] in performing everyday tasks. In the previous chapters, we have shown the

advantages of personal wearable computers over dedicated scanner devices for scanning visual tags, and we have presented advanced methods to improve their scanning reliability. This chapter is devoted to an application area of visual codes and wearables: identifying and controlling smart appliances.

Smartphones and tablets are applied in various scenarios for interaction with tagged physical objects and smart appliances [24, 82, 83, 143, 144, 177, 181, 183, 185]. We consider an appliance smart if *(1)* it contains an embedded computer, *(2)* it is connected to the Internet and is able to communicate wirelessly with people and with other smart objects, *(3)* it is potentially able to explore its context, as defined by Mattern [141]. Smart appliances can thereby expose some of their features to other appliances through standardized protocols. For instance, an increasing number of traditional appliances allow remote control over the Internet besides having their own physical user interface (knobs, sliders, LCD screens, etc.). We can indeed turn any physical object smart by adding a small embedded computer and Internet connectivity. It is, however, not straightforward how to configure and pair new devices in a smart environment, and this is where visual tags and wearable barcode scanning offer several advantages. The tags in form of visual codes serve as unique object identifiers or "hyperlinks" between real objects and their digital counterparts [114, 182]. We note here that other identification techniques (see Section 1.1.6) might be also applied, but with the help of visual codes, determining the position and orientation of an interaction device with respect to the real object becomes straightforward. These properties allow for seamless input and output for appliances in form of handheld augmented reality [82, 143, 177, 182, 235]. A few application examples of a smartphone as *universal interaction device* are illustrated in Figure 4.1.

We can generalize the concept of the universal interaction device from the smartphone to other wearable computers and most importantly, combinations of them. In our vision, the user just needs to look in the direction of a visual code on an appliance that automatically gets localized and scanned, while the user's hands remain free. When it comes to other functionality, it is still advantageous to have touch-free operation. Our wearable devices together form a universal user interface that always accompanies us. Then, we can interact with appliances via automatically generated appropriate user interface primitives [146] on our devices. In the rest of the chapter, we primarily focus on smartglasses as interaction devices but we make no restrictive assumptions about the wearable device type.

Current smartglasses such as Google Glass [289], Vuzix M100 [346], Epson Moverio [282], Recon (now owned by Intel) Jet [326], Osterhout R-7 [317], or other research prototypes [121] commonly leverage smartphone technologies such as touchscreens or acceleration sensors for user input to detect and interpret simple touch or motion gestures. This mostly limits interaction to manipulations of *2D* content – rather than *3D* information embedded into the real world. There are also a large number of research approaches how

to extend and add more expressive natural interaction forms to wearables but they often require special instrumentation of the user or altering the device. This can be a major barrier to adoption, and limits seamless unencumbered user interaction.

We argue that the current touchpad- and button-centric wearable interaction can be effectively complemented by natural hand gestures around the devices. In-air gesture interaction is especially advantageous for smartglasses that share the user's point of view and allow interactions with virtual objects in augmented reality applications. Our research is inspired by recent results on interaction methods for head-mounted displays, in particular the work on in-air gestures and 3D mobile interaction. Similar technologies have appeared in commercial hardware with the recently announced Microsoft HoloLens [307], to a certain degree justifying our objectives.

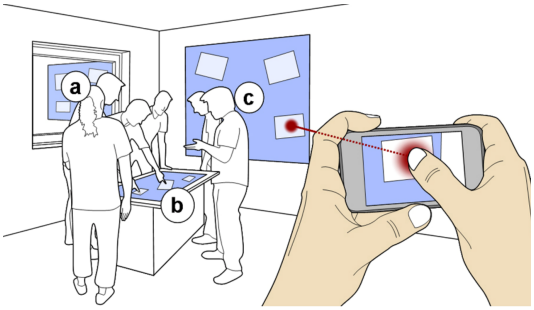
Interaction with machines using in-air gestures has been investigated for a long time [116], but robust real-time techniques have only been available with special hardware so far (a summary of related work follows in Section 4.3). Our main interest is how we can bring rich 3D interaction to wearable barcode scanning and in general mobile AR and interaction without requiring external sensors or depth cameras. Our work builds upon and extends the large body of previous research, but uses only the commonly available features of wearable devices.

Our contributions in this chapter are threefold. First, we generalize the notion of the smartphone as a universal interaction device to smartglasses and a combination of wearable devices. Starting from the concept of simply *outsourcing user interfaces* of smart objects to our wearable devices, we describe cross-device *user interface beaming*, and the opposite direction of *insourcing user interface* elements from wearables to smart objects. Second, we present a fast and robust method for in-air hand gesture recognition on unmodified personal wearable devices. The technique uses only a single color camera now commonplace on wearable devices. Our algorithm robustly recognizes a wide range of discrete in-air gestures. Third, we show how to extend our method to also estimate continuous 3D hand position for natural 3D interaction. The first contribution results from joint work with Simon Mayer, and the second and third contributions stem from joint work with Jie Song, Fabrizio Pece, and Otmar Hilliges. In particular, theoretical and architectural considerations stem from common discussions, the training and evaluation of the learning algorithms was performed by Jie Song, while the wearable implementation and optimization is sole work of the author. Parts of the text and figures are taken from joint publications [145, 203, 204, 205].

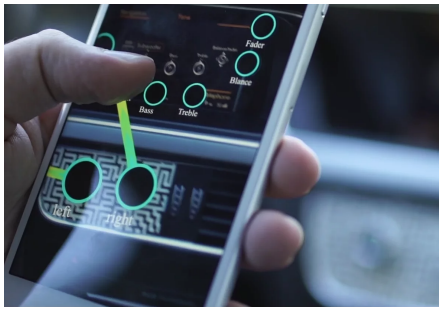
Camera-based gesture interaction is not limited to but is especially suitable for barcode scanners because their cameras are always on for scanning. However, our method is general and versatile enough for a wide range of applications. We present prototype

implementations and demonstrate that our algorithm runs in real time on unmodified smartphones, smartwatches, and smartglasses. Throughout this chapter, we focus on use cases in the smart home domain, but the presented methods can also be applied in other domains, for instance the interaction of workers with industrial robots in a smart factory context. Before coming to our gesture recognition method, we discuss the relevant literature in the areas of universal interaction devices in Section 4.2 and wearable interaction techniques in Section 4.3.

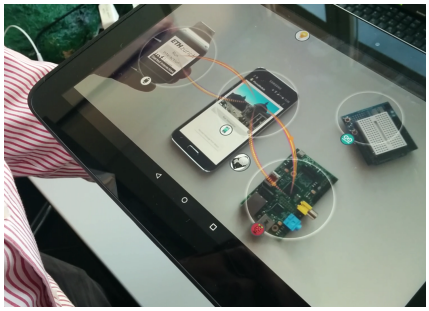
(a) Visual Code Widget [183]



(b) TouchProjector [24]



(c) RealityEditor [82]



(d) Magic Lens Network Visualizer [143]

Figure 4.1: Handheld devices as universal remote controllers in smart environments. The smartphone allows end users to intuitively control and configure their smart appliances.

4.2 Towards universal user interfaces

4.2.1 User interface outsourcing

Connected everyday objects are able to *outsource* their input and output modalities to the smartphone through various apps (for example Peel Smart Remote [318]) that can act as a graphical user interface (GUI) for anything. Note that the same is possible even

for disconnected objects that are associated with a virtual proxy [115]. Although the virtual proxy transmits information to the display device in reality, the user can have the impression that he or she is communicating with the object itself [141]. With such a smartphone-based universal remote control, we can simply switch a lamp [303], control a thermostat [311], or even control a coffee machine [335]. There are also a large number of sport accessories [286, 347] that offer only outsourced, smartphone-based user interfaces for size and cost reasons. The GUI primitives can be predefined or even automatically generated from a user interface description language which is embedded in the object's Web representation [146]. This mechanism gives high flexibility for changing and adding controls later, and also makes the hardware cheaper because new functions are added as software plug-ins.

The smartphone fulfills the role of a universal interaction device [14] as it is small, portable, always connected, and universally configurable in form of apps. While using our smartphone to control a coffee machine seems to be rather over-engineered [180, 181], there are a number of scenarios where an extended UI with special functions is useful (e.g., maintenance, parts replacement, social network connection, etc.). We argue that the true advantage of the universal interaction paradigm is not for appliances that already possess well-engineered traditional interfaces, but it allows us to interact with things that originally do not possess any handles/buttons, things that originally do not offer any human-readable output capabilities. This makes it possible to digitally interact with things that were impossible to interact with before (e.g., sensor nodes, a piece of furniture [85], or even flower pots [302]), and to easily configure smart environments as non-technical end users as also illustrated in Figure 4.1.

4.2.2 User interface beaming

Smartphones have visual, audio, and tactile modalities, but it might be cumbersome to take a phone out from our pocket every time we intend to push a button or read a machine's sensor values. Taking the concept of universal user interfaces to the next step, we envision interfaces in the form of a combination of always-on wearable devices. This is also in line with the long-standing goal of wearable research on reducing the time between the user intention and the triggered action [214].

Smartglasses offer two important advantages over smartphones. First, the cameras of smartglasses perceive the world exactly from the user's viewpoint. Second, smartglasses can visualize information directly in front of the user's eye in form of wearable augmented reality with optical see-through displays. These properties can make smartglasses ideal for information output from digital appliances. However, it is less clear how eyewear

computers alone can be used for interacting with smart things: The primary input modality for current devices is speech, which is cumbersome to use for many interaction tasks in daily life. Additionally, shortcomings in speech recognition algorithms may render the issuing of actuation commands even harder, especially in noisy environments. The slim touchpads and built-in inertial sensors that most smartglasses provide are limited to a few control scenarios. Examples include selecting an item from a list by swiping our fingers, or by tilting our head. While thus not being ideally suited for interaction tasks, smartglasses offer the possibility to know exactly what the user is looking at – thereby allowing the easy and intuitive selection [145, 261] of smart things or services to interact with.

Smartwatches, in contrast, provide an always-available, wrist-worn, touch-sensitive display on which we can render various interaction primitives. We can exploit all these different capabilities of the different personal wearable computers by combining them into a body area network as follows.

In [145], we have presented the concept of *user interface beaming* that allows users to conveniently interact with smart things in their environment via personal wearable computers. We can use smartglasses for appliance recognition, smartwatches for touch control, and smartphones for mediation and main computational hub (see Figure 4.2). Whenever a smart object is recognized in the user's view (via visual tags or other recognition techniques), the smartphone resolves the objects's Web representation and downloads a standardized description [146] of the object's user interface. When the user then looks at his/her smartwatch, the obtained interface description is transmitted – *beamed* – to the smartwatch that renders the required interaction primitives on its screen. The user can then control the target via touch or other sensors of the smartwatch. The interface is continuously updated to reflect state changes of the target object. Combining wearable computers in this way opens up new ways of interaction with everyday appliances: It allows users to discover, understand, and use interfaces of smart things in their surroundings seamlessly without relying on dedicated remote controls.

4.2.3 User interface insourcing

In the beaming concept, smartglasses play the most important role because they share the user's viewpoint and we usually look at the object we want to interact with. Egocentric vision and visual codes attached to appliances allow the user to easily identify the target object[1]. Just-in-time generated interfaces from egocentric requests allow implicit inter-

[1] With today's technology, the visual code must be close enough, but let us assume we can also scan distant codes in the future.

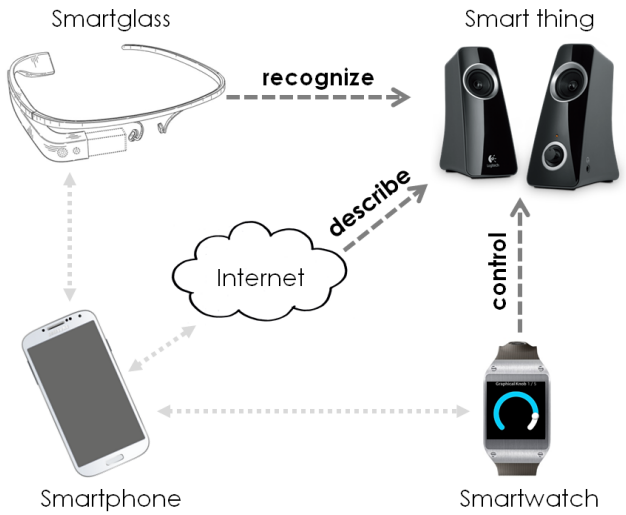


Figure 4.2: User interface beaming from a smart thing to a combination of personal wearable computers [145]. Image credits [290, 304, 329, 330]

action, in contrast with the explicit interaction required with the "universal" smartphone apps.

In some applications, it is also important for the object to identify the user. Contrary to egocentric vision, we can also distinguish object-centric vision when the smart object looks at the user, assuming the object actually has a camera. But in a connected environment, even objects without a built-in camera may take advantage of the user's camera and identify the user in a challenge-response mechanism, in a way *insourcing* the wearable capabilities. Imagine we want to pair two new devices with no camera but only a unique visual code on each of them. An automatic, proximity-based pairing algorithm could take advantage of the end-user's wearable camera and connect the devices by photographing them next to each other. This would be a complementary method to wireless proximity-based authentication [193] or in general common sensory context based [86] solutions. Similarly, we can imagine a smart supermarket shelf that must authenticate the user when he or she wants to buy products with an age limit. The authentication functionality can be insourced from the smartphone that knows the user's age by other reasons. Capability insourcing differs from simple *remote sensing* because in our vision, insourcing is opportunistic and automatic, and is not explicitly configured by the user. In future applications, there might be no need for remote control, because the appliances might be able to control themselves in a distributed control loop using insourced sensors of other devices. We illustrate GUI outsourcing and capability insourcing in Figure 4.3.

The mechanism of capability insourcing could indeed work similarly with any wearable sensor, for example in the following scenarios. A typical loudspeaker has control knobs

to set the volume but does not have any means of measuring the actual volume the user perceives. However, the user is wearing a microphone, so the loudspeaker might connect to that and can implement a feedback loop to keep the volume constant at a certain level. Today's home cinema systems allow calibration by holding a dedicated microphone when sitting on the couch. This calibration is limited to a single spot in the home. However, the system could opportunistically use the smartphone microphone and recalibrate when the user is sitting somewhere else in the room[2]. Similarly, a smart thermostat might have only a single point of temperature measurement at home, while the smartphone's temperature sensor is carried along in the home and can record a large set of measurements to give a fine-grained picture of the heat distribution. A digital candle might change its color based on people's mood at the table that can be derived from the heart rate measured by smartwatches today. Furthermore, a projection screen might automatically adjust geometric distortion and perform brightness and color correction based on the measurements it can get from the viewer person's smartglasses. This is a very important concept because it means that any functionality, and in particular any interaction paradigm we implement for wearable computers can be re-used and insourced for any smart appliance in the smart environment.

Figure 4.3: Left: Outsourcing the user interface from a smart object to a wearable device. Right: Insourcing the capabilities of a wearable device into a smart object. Image credits [271, 304, 330]

We have seen that UI outsourcing and insourcing with smartphones has many advantages, however, smartglasses are better suited for a variety of tasks where free hands and instant visual feedback are required. Even if smartglasses can take over most of the smartphone's functionalities, it is unclear what the easiest and most natural way to interact with glasses is in order to effectively access and control digital content. The input on current glasses and head-mounted displays is usually limited to a few buttons or slim touchpads (see Figure 1.16). More importantly, the current generation of devices require the direct physical (and even visual) attention of the user instead of focusing on the actual

[2] Although not opportunistically but with explicit user control, the idea has been recently implemented in the Sonos Trueplay [336] technology.

task. Our goal is to realize a more natural and seamless interaction without diverting the user's attention or disrupting the user's actions. In the following sections, we give a short overview of wearable natural user interfaces, and afterwards introduce a new method for efficient hand gesture control.

4.3 Wearable gestural interaction

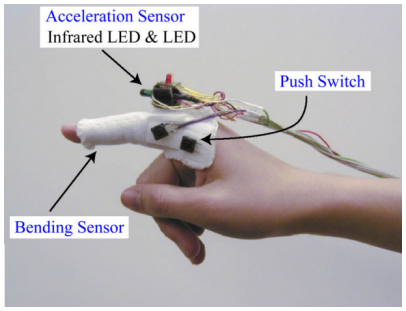
Wearable computers and augmented reality enable always-available user interfaces on our body and in our close environment. We argue that small touchpads and miniature buttons are not in line with how we manipulate the real world, and hence a poor fit for smart device control and AR applications. Here, we review the literature on more advanced interaction techniques that enable gestural input to our devices. From the vast amount of previous work, we focus on a small selection of projects from the last decade that directly or indirectly influence our research.

Ultra-mobile and wearable user interfaces have seen a lot of interest in the past. In particular, researchers have investigated in-air gestural interaction with dedicated wearable sensors, modified smart devices, or external infrastructure. We can categorize the techniques according to the main sensing technologies they use, e.g., infrared sensors [43, 113, 120, 155, 231], capacitive sensors [129, 151, 176, 187], acoustic sensors [7, 79], muscle sensors [154, 158, 188], antennas [3, 106, 169], small magnets [77], wearable cameras [64, 110, 148], or depth sensors [108, 156, 196, 213, 223]. All these different technologies have advantages and disadvantages. Ideally, our goal is to enable in-air gesture control without any hardware modifications of off-the-shelf wearable devices.

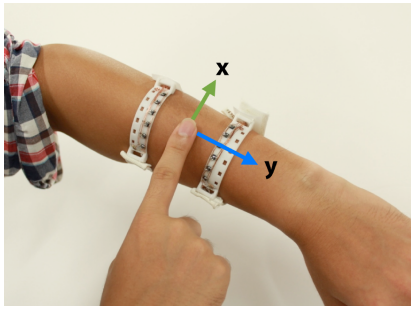
4.3.1 Gesture control with special hardware

Infrared sensors

The *UbiFinger* [231] allows selection and control of Web-enabled appliances by instrumenting the user's finger with a bending sensor, an infrared LED, and an accelerometer. The user can select an appliance by pointing the LED to it which also requires receivers at the appliances. Then, simple gestures of the user are recognized based on accelerometer data and control the selected appliance. *GestureWatch* [113] and *AirTouch* [130] use multiple infrared sensors mounted on the back of the wrist to detect freehand gestures, such as hand swiping along the forearm. *HoverFlow* by Kratz el al. [120] expands the interaction space around mobile devices. The technique recognizes coarse movement-based gestures

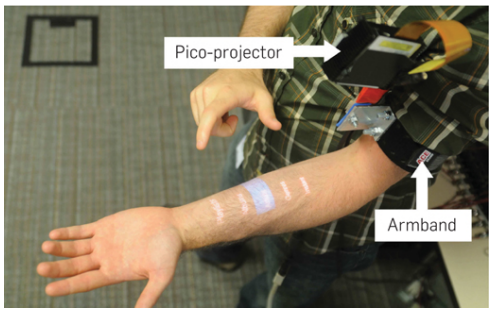


(a) UbiFinger [231]

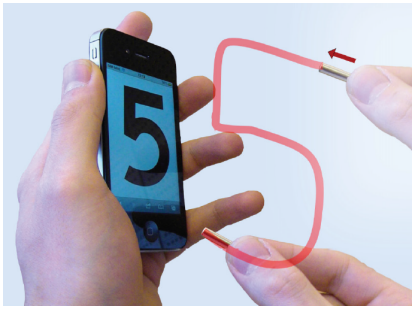


(b) SenSkin [155]

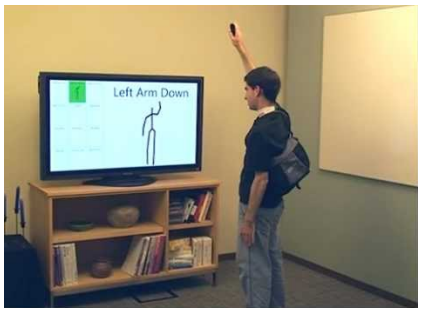
(c) PocketTouch [187]



(d) Skinput [79]



(e) MagiThings [109]



(f) Humantenna [42]

Figure 4.4: A few wearable interaction techniques without cameras.

as well as static position-based gestures, using an array of IR proximity sensors. A similar setup has been used in the motion-sensing mechanical keyboard by Taylor et al. [229]. *SenSkin* [155] uses wristbands of IR proximity sensors for detecting coarse gestures performed on the forearm. The *Mime* by Colaço et al. [43] specifically targets the expansion of input capabilities of smartglasses, by combining low-power time-of-flight triangulation module with an RGB camera. The triangulation module consists of a pulsed infrared LED and a linear array of three photodiodes. Zhang et al. present *HOBS* [261] (head orientation based selection) in which they augment smart glasses with infrared emitters and the environment with receivers for interaction with real-world objects at a distance. Unfortunately, IR approaches suffer from coarse sensing resolution and interference with ambient lighting.

Acoustic sensors

The *sound of the hand* system by Amento et al. [7] is a small wrist-mounted microphone that is able to distinguish finger tapping, rubbing, and flicking. *Skinput* by Harrison et al. [79] is a similar always-available input modality as it uses the user's body as an input surface. With a bio-acoustic sensing armband, they capture mechanical vibrations of finger taps on the forearm that propagate through the body. In contrast, the *SoundWave* system [69] uses only the speaker and microphone found in commodity devices to sense hand motion relying on the Doppler shift of a tone inaudible to humans.

Magnetic sensors

It has been also explored how magnetic field sensing can be used to track finger motion around a device by pairing permanent magnets and magnetometers either by wearing both like *uTrack* [29] or by using the device's built-in inertial measurement unit like *Abracadabra* [77], *Nenya* [12], or *MagiThings* [109]. Just like IR and acoustic sensing solutions, magnetic field sensing can suffer from coarse sensing fidelity and input is limited to tracking a restricted number of discrete points and often with limited degrees of freedom.

Skin and muscle sensors

Saponas et al. [188] shows how forearm electromyography (*EMG*) can be used to detect and decode human muscular movement in real time, enabling interactive finger gesture interaction, creating an always-available input device. Similarly, Amma et al. [9] recognize

finger gestures with a sensor array on the arm. Unfortunately, EMG requires extensive instrumentation of the user, and is prone to calibration issues.

Minput by Harrison and Hudson [78] adds two high-speed optical tracking sensors (like the camera of an optical mouse) on the back of a smartwatch-like mini computer. The device itself can thus react to shifting, twisting, or even more complex motions over the skin surface.

Ortega-Avila [158] captures in the near infrared spectrum how the tendons linked to fingers change slightly in thickness and location when moving the fingers. They train a classifier to recognize six different hand gestures from the measurements of NIR photodiodes attached to the wrist. The *SkinWatch* [154] works with the same principle, it detects pinching and stretching the skin around the smartwatch and translates the skin movement into zooming functionality.

Electric and electromagnetic signals

Rekimoto's *GestureWrist* and *GesturePad* [176] employ capacitive sensing elements that are embedded in a wristband or in clothes, respectively. *PocketTouch* by Saponas et al. [187] is a multi-touch capacitive sensor matrix attached to the back of a smartphone. It allows full alphanumeric text entry without taking the smartphone out of the pocket. Simple resistive 1D position sensors are applied in *WatchIt* [166] which enables simple gesture interaction on a touch sensitive wristwatch or bracelet. More recently, a transparent electric field sensing antenna has been demonstrated that enables sensing of 3D hand and fingertip locations [129]. Other explored the design of an energy harvesting motion sensor by Cohn et al.[41], or the *Humantenna* [42] that leverages the whole body for interaction. *Mirage* by Mujibiya and Rekimoto [151] also senses the human body as an intruder in a static electric field and recognizes human presence, body motion, body gesture, and activity.

AllSee [106] enables gesture classification on passive devices such as RFID tags without batteries using the ambient backscatter principle. The prototype harvests energy from an RFID reader or from ambient television signals. Methods exploiting reflections and Doppler shift of wireless radio signals include *WiSee* [169] and *WiTrack* [3]. Google recently presented *Soli* [292], a tiny radar for interaction. The advantage of radio-based solutions is that they are less privacy-concerning than cameras. It is rarely discussed, however, where the limitations of these methods are. In particular, the interaction fidelity is usually limited to whole-body gestures.

4.3.2 Gesture control with camera systems

A number of camera-based systems have been proposed to extend the interaction space beyond the buttons and touchscreens of our devices. The hands are our primary means for interaction with our surroundings and hence hand gesture recognition has a long history in the HCI and computer vision community. Early approaches explored the possibility to instrument the hands with special sensor gloves or marker gloves (see [48] for a survey). However, gloves tend to rather hinder the applications in mobile scenarios and pure vision-based techniques gained interest. A review of older vision-based hand pose estimation approaches is given by Erol [51] and in the thesis of Kölsch [116]. Early work focused on hand tracking from 2D cameras. However, recognizing complex hand gestures is clearly challenging from such 2D input. Our hands are very expressive because different hand positions, hand orientations, and finger configurations all can have different meanings. The high dexterity also poses a challenge in robustly recognizing the gestures for a machine. Heuristics and pure geometric methods usually fail due to the large variation in gesture execution, variation in hand appearance, variation in lighting conditions, and background clutter.

Wearable cameras

Markerless tracking of the user's outstretched palm for augmented reality has been presented by Lee and Höllerer [131]. Seo et al. [192] developed a system where virtual objects are rendered on top of the non-dominant hand, and the user can interact with the virtual content by simple motions of the hands. However, these methods are not designed for more complex interaction scenarios, but rather serve AR applications where only hand tracking is required.

The *GesturePendant* [64], is a chest-worn IR camera and IR illumination on a necklace for hand sign recognition in front of the body. Similarly, the *Cyclops* by Chan et al. [28] is a chest-worn 235° fisheye camera for full-body gesture input. The device can recognize static and dynamic hand gestures using on motion history images and a randomized decision forest. The *CyclopsRing* [27] is a similar device with a fisheye camera built into a ring that recognizes pinch+motion gestures for controlling tagged smart appliances. The *Surround-see* technique by Yang et al. [258] equips a smartphone camera with an omni-directional mirror that enables peripheral vision around the device. From the omnidirectional image, the system recognizes the user's hand, detects the user's current activity, and also identifies nearby objects. The user's pointing finger is segmented and the fingertip is tracked so that pointing at and interacting with objects in the environment become possible.

Special eye-tracker cameras (for instance Pupil [322]) mounted in smartglasses allow gaze-controlled interaction with objects. The drawback of these approaches is that they require precise calibration whenever the camera was moved with respect to the eyeball. In contrast, *Pursuits* [233] tracks only differential movements of the eye and allows interaction with moving objects on a screen without calibration. It requires specially designed dynamic graphical interfaces though.

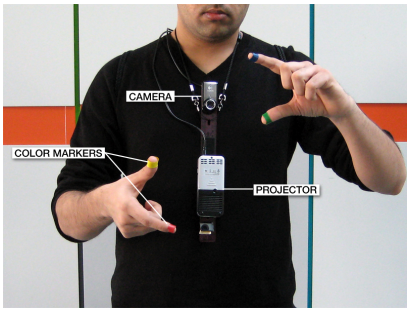
Niikuura et al. [153] leverage IR LEDs and a 2D camera to recognize a single fingertip for *in-air typing*. Higuchi [84] developed an *in-air virtual keyboard* for mobile devices, Jung et al. [99] presented a similar one specifically for smartglasses. However, the vocabulary consists of simple pinch and pointing gestures. [64, 216] classify a wider set of discrete hand postures, e.g., for wearable sign language recognition, but also relying on IR augmentation. The method of Li et al. [134] enables robust pixel-level hand segmentation in ego-centric videos, which is an important first step in every egocentric gesture recognition application, but the authors do not address the actual recognition problem.

Gustafson et al. present the concept of *imaginary interfaces* [70, 71], a method that brings spatial interaction to miniature screen-less mobile devices, using the palm of the non-dominant hand as a universal interaction device. [70] uses a body-worn camera and diffuse IR illumination to recognize 2D pinch gestures. In their prototype of an *ImaginaryPhone* [71], tapping and sliding events in the palm are recognized by a shoulder-worn depth camera, and the input events are transferred to an actual smartphone. Unlike other shoulder-worn system, there is no projector or display. *Digits* [110] reconstructs full 3D hand models in real time by instrumenting the inner side of the wrist with an IR illuminator and IR camera facing towards the palm. 3D hand pose is indeed desirable for many applications, hence researchers turned towards consumer depth cameras.

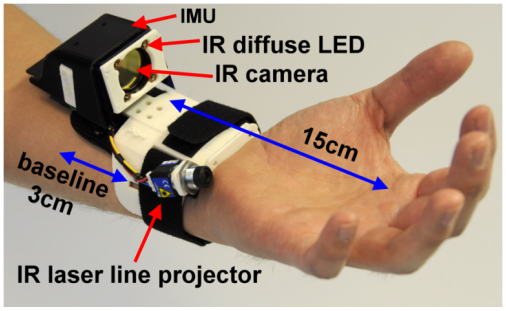
Projector-camera systems

When combined with camera or non-camera input, wearable projectors allow rendering graphical user interface elements, hence the user can tap on virtual buttons and sliders on a surface [148] or on his/her own body [79]. *SixthSense* [148] detects color-marked fingers and simple finger gestures with a body-worn RGB camera, and allows intuitive in-air interaction in front of body while also projecting GUI elements on the wall. *OmniTouch* [76] is a shoulder-worn projector-depth-camera system that can turn any planar surface in the environment into a multitouch display. Pohl et al. [168] generalize this to the concept of *around-device devices*, using any object in the surroundings as a user interface. These projection techniques require the user to wear large additional devices or the environment to be instrumented, which limits seamless interaction and may lower user

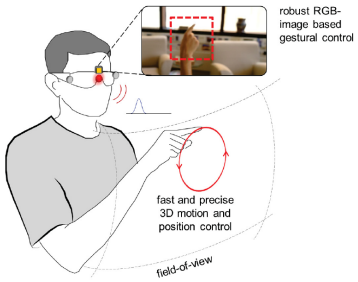
adoption. Instead, we focus on techniques without altering the design of existing devices or requiring any additional hardware.



(a) SixthSense [148]

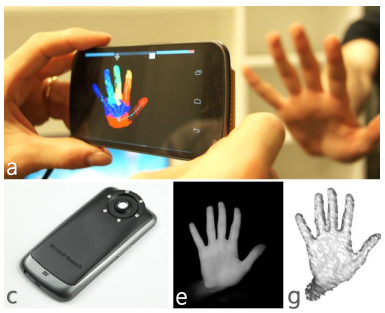


(b) Digits [110]



(c) Mime [43]

(d) CyclopsRing [27]



(e) Learned depth camera [55]

(f) Hand tracker by [196]

Figure 4.5: A few in-air interaction techniques with body-worn cameras

Depth cameras

The emergence of consumer depth cameras has enabled high-fidelity, fine-grained 3D hand pose estimation. However, hand tracking is challenging even with depth cameras because of the many degrees of freedom, self-occlusions, and similarity between the fingers. The current approaches are either model-based that apply 3D hand model fitting and temporal

tracking [156, 196, 213, 223], or data-driven [55, 108, 228] that apply machine learning on hand silhouettes, shading, depth image, or other features.

Keskin et al. [108] introduced multi-layer randomized decision forests for hand pose estimation. Oikonomidis et al. [156] solved the problem of articulated motion capture of two hands, and Ballan et al. [15] proposed a method that allows to capture two hands interacting with an additional object, using eight cameras. Real-time methods were proposed by Ma et al. [140], Qian et al.[170], Tagliasacchi et al. [223], Sharp et al. [196], and Sridhar et al. [213] which represent the state of the art in hand tracking with a depth camera. Most of these methods are optimized to a particular viewpoint where a desktop depth camera is facing the hands, although this requirement has been lifted in Sharp et al. [196]. In wearable scenarios, however, the usage of conventional depth sensors is prohibitive due to power consumption, heat dissipation and size. Recently, Fanello et al. [55] try to overcome these limitations by learning a mapping from IR color to depth using a modified RGB camera surrounded with IR LEDs. They present very accurate depth estimates, however, the method requires hardware modification, and the color-to-depth mapping needs to be trained for each individual.

Positioning our work

While hardware augmentations can largely enhance the interaction capabilities of the mobile devices, they also induce two important problems. First, modifying the original design of devices might make them cumbersome to use. Second, the social acceptance of augmented devices may considerably decrease.

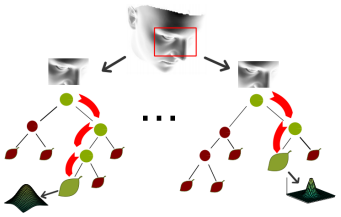
Our contribution builds upon and extends the large body of research on gesture recognition on mobile devices. Our work, though, differs from existing solutions as it leverages only a conventional monocular colour camera, lending itself to run on *unmodified* devices. We propose a randomized decision forest (RF) based algorithm to extend the interaction space around mobile devices by detecting rich gestures performed in front of any wearable camera.

In the next section, we shortly summarize the theory of RFs. Next, we introduce our approach for efficient classification of 2D hand silhouettes. Subsequently, we show how to extend this to a hybrid classification-regression scheme which is capable of successfully learning a direct mapping from 2D color images to 3D hand positions and gestures. The algorithm runs in real time on off-the-shelf mobile devices including resource-constrained smartphones, smartwatches, and smartglasses. Combined with fast visual code localization (Chapter 2), an extension to quickly recognize other objects relevant for interaction is straightforward.

4.4 Randomized decision forests

Our proposed gesture recognition algorithm is based on a powerful machine learning technique named randomized decision forest or short random forests (RF) [8, 25, 45], therefore we give here a short summary of the technique. Random forests have been proven successful in a number of computer vision problems. Their application areas include object detection [61], body pose estimation [199], head pose estimation [53, 54], hand pose and state estimation [108, 196, 228] among others. These approaches rely on depth sensing cameras, but RFs have also found applications using different sensing modalities [111, 129, 229] for human computer interaction. Figure 4.6 illustrates some of these applications. Our approach is inspired by these works in the field, but it presents a much simpler and computationally inexpensive framework which is highly robust against noise.

(a) Body pose estimation [199]



(b) Head pose estimation [53]

(c) Motion gesture recognition [229]

(d) Hand pose estimation [108]

Figure 4.6: Random forests are powerful tools in many computer vision tasks.

A random forest is an ensemble of decision *trees* (see Figure 4.7). Each tree in the ensemble produces a noisy classification result. However, accurate classification results can be attained by averaging the results from multiple, non-biased classifiers together. Each tree consists of *split nodes* and *leaf nodes*. The split nodes themselves can be seen as very primitive classifiers. Each node will evaluate a simple and computationally inexpensive *split function* and as a result forward the currently evaluated datum (in our applications a pixel) to its left or right child until the datum reaches one of the leaf nodes. The split functions can be seen as simple questions where each answer adds a little more information, or in other words reduces the uncertainty about the correct class label for the

particular pixel. The *features* in computer vision applications can be depth values, pixel colors, binary mask values, but also more complex entities. The split functions depend on the actual applications. In a *classification forest*, each leaf node stores a probability distribution describing the likelihood with which a datum belongs to any of the discrete classes in the label space. Based on this distribution, a selected *leaf prediction model* assigns a class label to data points that arrive in the leaf node. In contrast, in a *regression forest*, the leaf nodes store a probability distribution over a continuous variable.

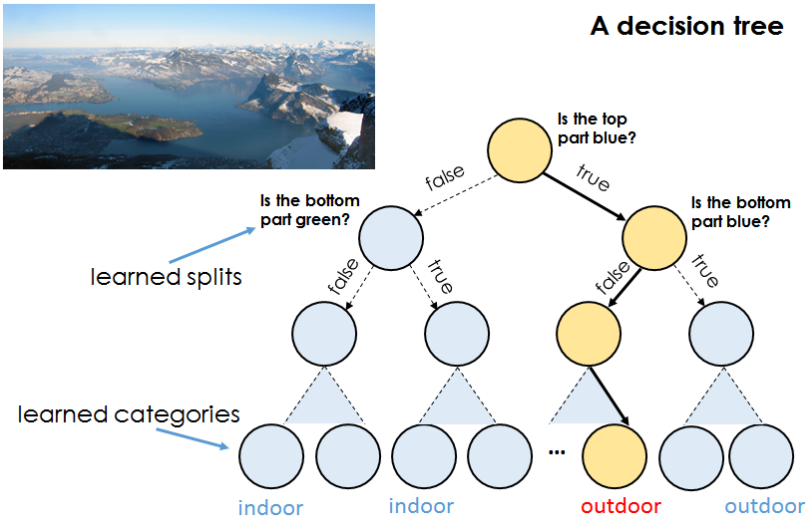


Figure 4.7: A decision tree for classification. Figure adapted from Antonio Criminisi [45]

In the training phase, a large set of pixels and ground truth labels are provided from training data, and each tree is trained independently. Starting from the root, the best split function parameters in each split node are searched that split the data into left and right child sets. The objective for splitting is typically the information gain for classification problems. The training is performed recursively down the tree until the maximum allowed depth is reached or the child sets contain too few data points.

To avoid overfitting, randomness is introduced in the training process [25]. Each tree is trained on a randomly selected subset of training data, and at each split node only a randomly selected subset of possible parameters of the split function are considered. This randomness is injected into the classifier only at training, and the testing phase is usually deterministic, hence the name "randomly trained decision trees" might be closer to the actual concept [45].

Random forests and decision tree classifiers in general trade discriminative power and run-time performance against memory consumption. Given a complex enough problem

and large enough training set, trees will grow exponentially with depth. This is of course a strong limitation for their application on resource constrained wearable platforms.

4.5 Fast gesture recognition on wearable devices

We present a machine learning method that expands the interaction space *around* cameras, enabling in-air gesture interaction on commodity wearable devices. The method allows for robust detection of gestures relying only on RGB camera input without any hardware modifications. The algorithm is based on random forests and runs in real time and can even be used on ultra-mobile devices such as smartwatches. To the best of our knowledge, this is the first real-time implementation of RFs for mobile devices for a pixel labeling task. Compared to most previous work, our algorithm does not rely on highly discriminative depth features but works with only 2D images.

We describe a method to robustly and efficiently classify hand states (i.e., gestures) and salient features (i.e., fingertips) using only binary segmentation masks. We also describe techniques to make the classifier robust to in-plane rotation (i.e., wrist articulation or rotating the device itself) and to variations in depth of the hand. We introduce different techniques, specifically designed for wearable devices, to reduce the size of the forests and in turn reduce the memory consumption drastically while maintaining classification accuracy. These are necessary to fit the classifier into the limited memory of current wearables but generalize to any computing device.

4.5.1 Method overview

We classify single pixels of binary images that correspond to the silhouette of the hand. For the whole image, we decide for the gesture that most pixels vote for (see Figure 4.8). We pass each pixel of the binary image down the decision trees. At an inner node of a tree, we sample two other points from the entire image at locations that are determined by two learnt offset vectors. Depending on the difference of the binary values at the two offset locations and a learnt decision threshold, the pixel is forwarded to the left or right child node (detailed explanation follows at Figure 4.11). With these simple and fast tests at internal nodes, we have a powerful way of determining a per-pixel class label by the time we get to a leaf node. The final class of a pixel is the average of the output of all trees in the forest. We then pool all labels across the image and determine the gesture shown in the image.

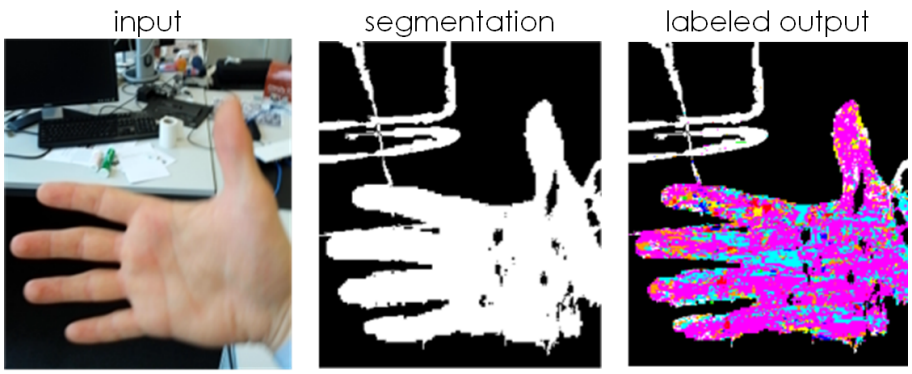


Figure 4.8: From the RGB camera input, we first segment the hand pixels to get a binary mask. Then, every pixel of the binary mask gets classified and votes for the overall gesture of the image. Different colors correspond to different gestures.

In Figure 4.9, we give an overview of the full pipeline of our classification algorithm. All components of the pipeline have been carefully chosen and designed with runtime and memory efficiency in mind, in order to achieve real-time performance even on ultra-mobile and resource restricted devices. The pipeline consists of established image processing steps interwoven with a new, staged classification process. The algorithm processes single camera frames and decides for the most probable gesture shown in the image.

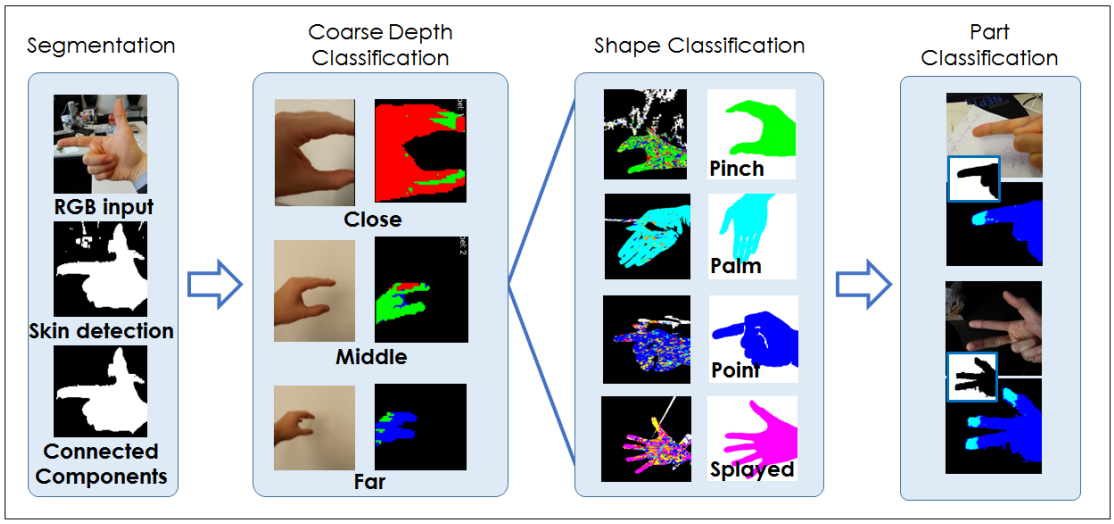


Figure 4.9: Multi-stage classification. (1) Pre-processing segments the hand from background. (2) The DCF coarsely estimates the depth of hand and in-range frames are forwarded. Effective range is between $15 - 50$ cm. (3) The SCF classifies each foreground pixel into six gesture classes plus noise. (4) The PCF classifies location of fingertips for pointing-like gestures.

In the first step, the hand pixels are segmented to get a binary mask. The segmentation is not perfect, so the binary mask still contains background noise which needs to be handled in the later steps. To reduce the large variation caused by hand rotation, we perform principal component analysis (PCA) on the binary mask and rotate the mask to a canonical orientation. The pixels of the binary mask are classified one-be-one in the random forest and the labels are aggregated over the image to decide for the final gesture.

We apply three cascaded random forests (introduced by Keskin et al. [108]) that each solve a sub-problem of the whole task. Instead of training a very large forest that deals with all the variations in the data, the cascaded architecture handles simpler tasks at each stage which effectively reduces the size of the single forests. As the memory requirements are growing exponentially with the depth of the trees, it is crucial to keep the depth minimal. We connect a coarse-depth classification forest (DCF), a shape classification forest (SCF), and a part classification forest (PCF). In this configuration, each classifier is trained on a particular task, and images corresponding to a particular subclass are forwarded in the following stage to the corresponding classifier that was trained on examples from that particular subclass. As we will show, the cascaded architecture with three shallower forests achieves equal classification accuracy at much lower memory requirements than a single deep forest. In the next sections, we describe the components in detail.

4.5.2 Hand segmentation and preprocessing

Our algorithm takes binary images as input, so in the first step we segment the hand from the background. Despite a vast amount of research,this problem remains a challenging task in real time under arbitrary lighting conditions. A good summary of various techniques is provided by [100]. We have experimented with many techniques, including adaptive thresholding in the HSV color system (described in Section 2.3.5), Gaussian Mixture Models (GMM), and probabilistic ways to locate the hand. However, due to the large variation in skin color and the influence of lighting and colorful background, currently none of these methods can provide perfect segmentation under arbitrary conditions.

We opted for a simple RGB color thresholding technique which selects the red-toned areas of the image. This method is easy to implement on the mobile GPU, is computationally cheap, and provides a good compromise between true positives and false negatives. The above complex methods did not significantly improve the segmentation quality compared to the RGB-method but require longer computational time. The little increase in segmentation performance is not worth the increased latency over the simple technique.

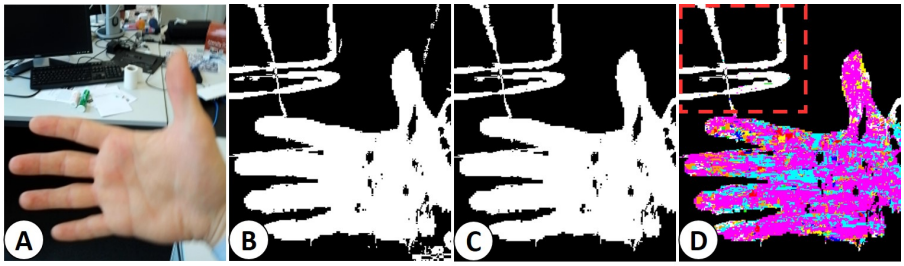


Figure 4.10: Segmentation by classification. A) Input frame. B) Foreground mask after color thresholding. C) Small-scale noise is cleaned up by connected component analysis; larger connected non-hand regions remain. D) Result after classification. Colored pixels indicate class label, white pixels indicate noise class (highlighted by a red box).

Hence, we segment the hand from the background by

$$S_t(p) = \begin{cases} 1, & if \quad min(R_t(p) - G_t(p), R_t(p) - B_t(p)) > \tau_t \\ 0, & otherwise \end{cases} \tag{4.1}$$

where R_t, G_t and B_t denote the RGB-components at frame t and image pixel p, and τ_t is a threshold value and S_t the segmentation. $S_t(p)$ is binary output at pixel p, 1 at foreground and 0 at the background. The threshold $\tau = K/255$, and in our implementation K can be manually set at runtime via a slider. In our experiments, $3 \leq K \leq 8$ worked well in the majority of tested scenes both indoors and outdoors.

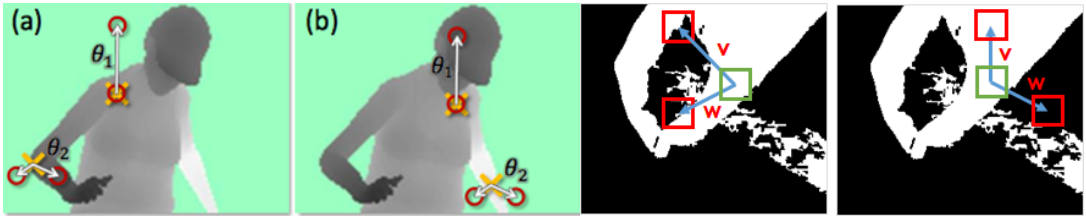
This technique provides reasonable segmentation while it suffers from *false positives* and creates noise outside the hand region. Some of this noise can be suppressed using connected component analysis (CCA) but the regions of noise connected to the hand will remain (see Figure 4.10). However, the subsequent classification algorithm can be made robust to this type of noise as detailed below. At this stage, it is more important to attain a segmentation that does not suffer from *false negatives* so that the whole contour of the hand remains complete for accurate classification results.

In the last preprocessing step, principal component analysis (PCA) finds the in-plane rotation of hands. Rotating the image to a canonical orientation greatly reduces the overall complexity of the classification problem and allows shallower classification trees. In practical implementations, it is faster to rotate the jump vectors and not the image.

4.5.3 Hand shape classification

Recent works with random forests (e.g., [108, 199, 228]) use depth images as input and complementary depth-invariant features for classification. These approaches use continuous-valued features (typically simple depth differences as illustrated in the left side of Figure 4.11), and the split functions compare the feature response against a continuous valued threshold. We do not have depth information as today's wearable cameras offer only a single color camera input. Unfortunately, we cannot rely on color either as the variation in appearance of different hands under arbitrary lighting conditions is too large to cover all variations.

Instead, our method relies only on shape (i.e., binary masks) to infer hand features and states. Within the classification trees, we use split criteria based on binary features. Specifically, for each split node we learn shift vectors and a comparison threshold (see Figure 4.11, right). When evaluating a pixel, we compare the values of the segmented binary mask at these shifted locations and proceed traversing the tree based on the result of the comparison (see Figure 4.12 and Figure 4.13).



(a) Features by Shotton et al. for body pose estimation from depth images [199]

(b) Our features for shape classification from binary images

Figure 4.11: Decisions at inner nodes are made based on comparing the values at two offset locations with a threshold. The offset vectors and the threshold are learned during training.

We use split criteria of the following form in the split nodes:

$$C_j^L(v,w,\Gamma_j) = \{(S,p)|F_{v,w}(S,p) = \Gamma_j\} \tag{4.2}$$

$$C_j^R(v,w,\Gamma_j) = \{(S,p)|F_{v,w}(S,p) \neq \Gamma_j\} \tag{4.3}$$

where F_j is the feature response and Γ_j is a selector value trained at each split node j. Here C_j^L and C_j^R are the mutually exclusive sets of pixels assigned to the left and right children of the split node. For a given segmented image S, we define the feature response at the pixel location p as

$$F_{v,w}(p) = [S(p+v), S(p+w)] \tag{4.4}$$

this is a 1×2 vector where v and w are (randomly) learned offsets and $S(p)$ is the binary pixel value at that location (see Figure 4.12). The feature response is discrete and each

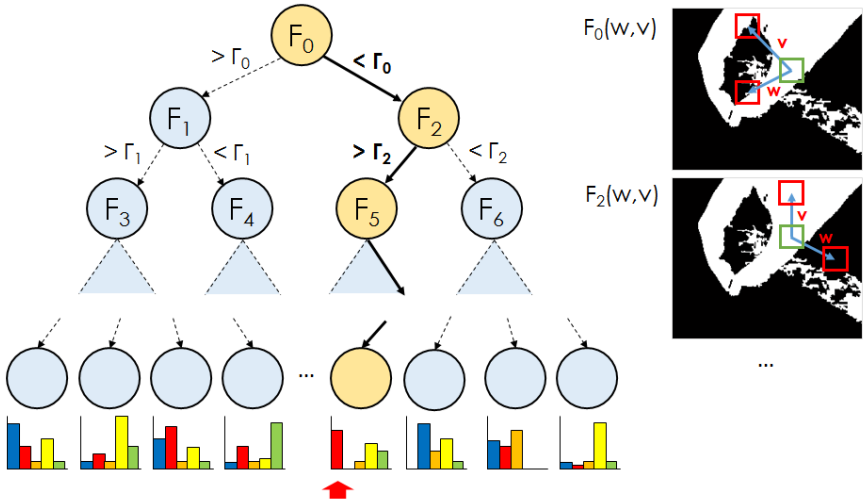


Figure 4.12: Illustration of classifying a single pixel with a single decision tree. At the split nodes, we evaluate the function F_i with the learned parameters v, w, Γ and forward the pixel to the left or the right child node. The leaf nodes store probability distributions over the class labels. Figure adapted from Otmar Hilliges [204].

element of it can take on only two values; 1 for foreground and 0 for background pixels. At evaluation time we simply compare this vector with $\Gamma_j \in \{[0,0],[1,0],[0,1],[1,1]\}$. Later in this chapter, we provide pseudo-code for forest evaluation, and here Figure 4.13 illustrates how an image is processed in the classification forest.

The order in which the split functions are concatenated (the structure of a tree) is very important for efficient classification. Intuitively, we should find the split function at each node that separates the data the most. We can learn the "best" parameters of split function from labeled training data, starting from the root of the tree. Recall from Section 4.4 that this is done by randomly selecting multiple parameter candidates of the split function and and choose the one that splits the data the best. For more details we refer the reader to [25, 199].

As defined in Equation 4.2, our split functions at each node have three parameters: (v, w, Γ). The quality metric for the split thereby is typically defined by the information gain I_j at node j:

$$I_j = H(C_j) - \frac{|C_j^L| H(C_j^L) + |C_j^R| H(C_j^R)}{|C_j|} \tag{4.5}$$

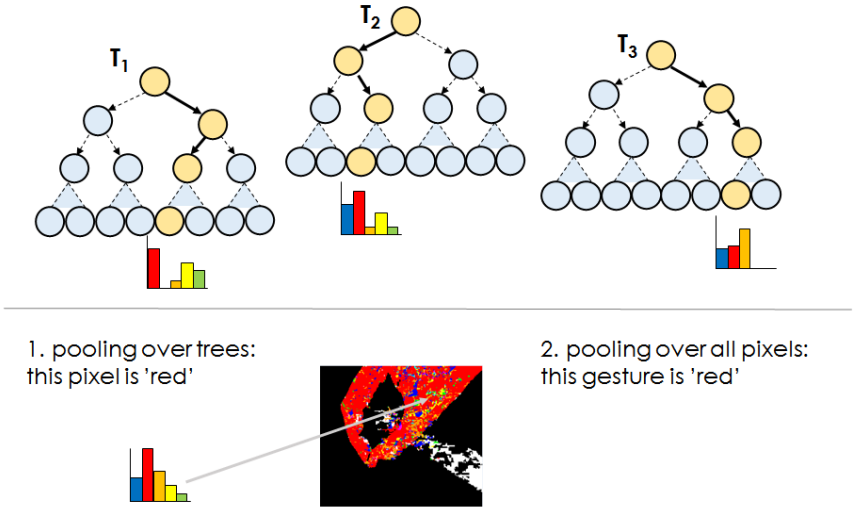


Figure 4.13: Illustration of classifying the whole image with a decision forest. Each pixel is classified by all trees in the forest and the final label is determined by pooling over all trees. The final gesture in the image is determined by majority vote over all pixels. Figure adapted from Otmar Hilliges [204].

where H is the Shannon entropy of the data points in set C. A decrease in entropy (a decrease in uncertainty), denoted by I_j, means increase in information. Weighting the entropy by the cardinality of the child sets avoids splitting off children containing very few data points. Our parameter space is not large compared to other computer vision problems, therefore generating a relatively low number of random splits is enough at training time. At each node j, we generate about 500 possible splits (v_j, w_j, Γ_j) and select the one maximizing I_j.

Note that due to the randomized nature of the training process, the parameters are not guaranteed to be the best at each node of a tree, however, the whole forest overcomes this "noise" of the single tree classifiers. The final probability distributions are also learned from the annotated training data.

The binary features allow us to train relatively complex datasets while keeping the computational time and memory footprint low, without sacrificing precision. Intuitively, the binary values and the shift vectors encode the "hand shape" information and the probability that a given pixel belongs to the hand. In other words, given the offset values, we are able to distinguish between pixels that are inside the hand silhouette and pixels that are outside of it. These features provide a good tradeoff between training efficiency and testing accuracy. Because the features themselves are not depth- or rotation invariant, we cannot constrain the jump distance of the offsets and must allow jumps to any location in the image. This results in many more split candidates. Also note that the number

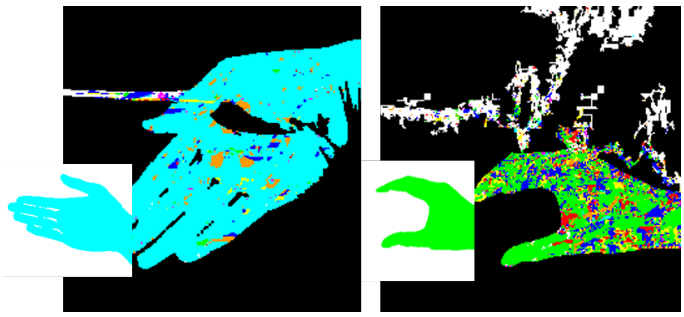


Figure 4.14: Shape classification: each pixel is classified to one of the gestures (colors) or noise (white). The gesture shown in the image is decided by majority vote over the pixels.

of jump locations is also quadratically increasing with image resolution. However, the computational cost for a single feature at training time is not high because a feature can take only a limited number of values, namely four. We could also apply more than two offset vectors in a split function to increase the discriminative power at one node, however, this would strongly affect the training time as the number of offset combinations increases exponentially.

4.5.4 Multi-stage recognition

With the described shape classification procedure, we can recognize a rich set of hand shapes. However, to cover all variations within the training set, we potentially need to grow a very deep forest. While even deep trees can be evaluated in a few milliseconds, the memory footprint of the forest can quickly become an issue, as it grows exponentially with tree depth. Instead, we opt for the above introduced multi-stage approach, where each of the forests then needs to model less variation and hence can be comparatively shallow. We propose to combine multiple forests that are specialized on different tasks and modify the image before they pass it to the next stage. This results in the pipeline of independent but inter-related classifiers illustrated in Figure 4.9.

Stage 1: Coarse depth classification

Given an input image I with associated pixels p and a segmentation S, the depth classification forest (DCF) in the first stage infers a probability distribution $P(d|p,S)$ over the three coarsely quantized depth ranges and an additional noise class (i.e., our labels in the first stage are $d \in \{\texttt{close},\texttt{middle},\texttt{far},\texttt{noise}\}$). The role of the `noise` class is to filter out

most of the noise coming from the simple segmentation. At the end of this stage, we have the coarse depth information and also a cleaner input mask, which greatly decrease the variation in terms of hand size and uncertainty the next stage has to deal with.

The concrete depth ranges depend on the application. For example, for the smartphone camera we show in Section 4.9, we allow a $15-30$ cm range for interaction that corresponds to the `middle` class. For smartglasses, we need a larger interaction range from 9 cm to 39 cm measured from the camera. If the hand is too close or too far from the camera, the pipeline stops at this point. Otherwise, the per-pixel labels d are forwarded to the next stage.

Stage 2: Shape classification

The shape classification forest (SCF) classifies the images into (currently) six gesture classes (see Figure 4.15). In addition, there is one `no_gesture` class (orange in the figures) which is needed to reject non-gesture motion in front of the camera. We recorded "random" other hand gestures in front of the camera to represent this class in the training data. Furthermore, there is one `noise` class (white) to deal with false positives in hand segmentation from the previous step. The classification algorithm is the same as described in Section 4.5.3 above. Figure 4.14 illustrates examples of per-pixel classification results.

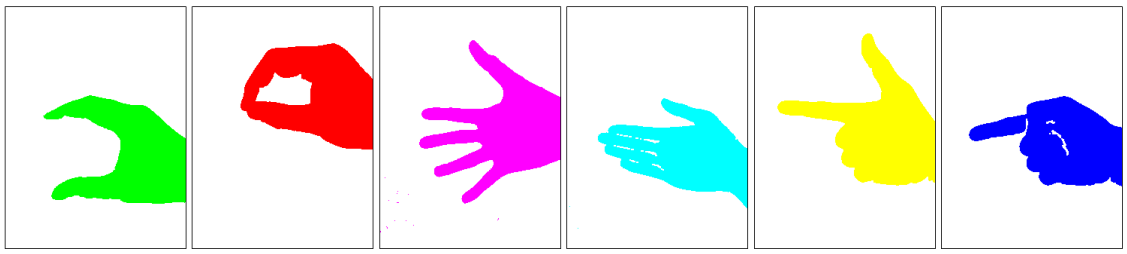


Figure 4.15: The current gesture set recognized by our hand shape classifier: `pinch_open`, `pinch_close`, `splayed_hand`, `flat_hand`, `gun`, `pointing`.

Stage 3: Part classification

Optionally, the pointing-like gestures (i.e., `pointing, gun, splayed_hand`) enter a part classification forest (PCF) that perform per-pixel hand part classification. In particular, we trained this forest to separate the fingertips and the wrist for these hand shapes. These can be used to add a pointing direction to our in-air gestures. For instance, we can detect the number of visible fingertips, which may be mapped to different applications. Without

depth information, we cannot classify all hand parts like [107], but we only select parts of our interest for interaction. As practically all noise is cleaned up in the previous stages, the PCF only needs to process images with well visible fingertips, hence a tree depth of 10 is enough for high accuracy. Example results of this stage are shown in Figure 4.9 on the right side. The three stages are illustrated together in Figure 4.16.

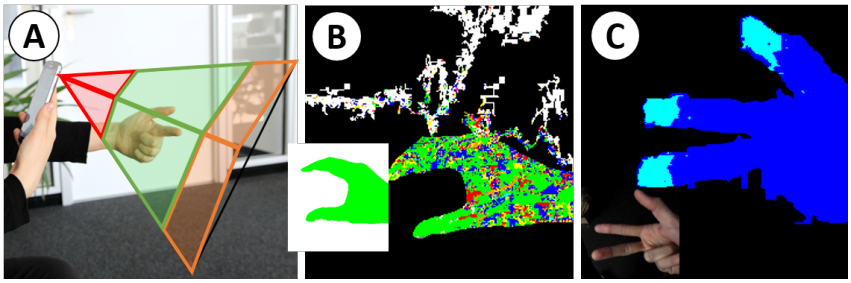


Figure 4.16: Multistage classification: (A) First hand is classified into three different coarse depth bins. (B) Next hand shape is classified into 6 classes. (C) Finally, fingertip parts are detected by PCF. Figure adapted from Fabrizio Pece.

Our multi-stage classification pipeline was inspired by the work of Keskin et al. [107] for fine-grained hand part classification from depth images. Shortly after our classification work [204], Fanello et al. [55] proposed a similar regression architecture that is able to directly map a 2D infrared image to continuous scene depth. In the next section, we show that our method working on binary images from regular cameras can also be extended to depth estimation. Specifically, we extend our framework to a hybrid classification-regression scheme, capable of learning a direct mapping from 2D images to 3D hand positions besides the gestures. In general, the multi-stage forest architecture is appealing for various tasks because it combines the runtime efficiency and discriminative power of regular RFs with a reduced memory footprint, which makes it well suited for off-the-shelf mobile devices while achieving high recognition accuracy.

4.6 Enabling 3D interaction

In the previous section, we have presented a method for recognizing 2D hand gestures from hand silhouette images. In many applications, however, full 3D gesture recognition is also necessary. Unfortunately, depth-sensing cameras in wearable computers are prohibitive at the current stand of the technology due to very high power consumption. Very recently, a few new machine learning based methods have appeared that are able to infer depth from 2D imagery [55, 125, 126]. Although the general problem of reconstructing a 3D object

from a single 2D image is infeasible, we can investigate whether a rough mapping from color images to depth is possible when we restrict ourselves to the specific case of hands.

In this section, we introduce our extended approach for joint *classification* of discrete hand gestures and *regression* of the depth of the hand. We extend our 2D gesture classification technique to a hybrid classification-regression approach that jointly detects hand shapes and regresses the average distance of the hand from the camera. The method is similar to the one presented above but uses hybrid classification-regression forests [25, 45]. The *centroid* and the *average depth* of the hand together allow 3D interaction in the air in front of a wearable camera. The applied gesture set and thus the required training data depends on the location of the wearable camera (e.g., on the arms, chest, head, etc.), here we focus on scenarios with smartglasses only.

Next, we first shortly discuss the literature related to depth estimation with machine learning methods. Afterwards, we describe the details of the 3D extension of our method, and show an evaluation of both together in Section 4.8.

4.6.1 Machine learning for depth estimation

Depth sensing with cameras has always been an important research topic and several complementary approaches have been proposed. These include passive and active stereo, shape-from-shading, structured light, time of flight measurements, and approaches based on structure from motion. For a detailed review, we refer the reader to [128]. These dependencies imply lower bounds on the size, the weight, and the power consumption of such devices. Hence, most depth sensing technologies are currently prohibitive in wearables. We are especially interested in methods that do not require special hardware, but try to estimate depth from images using machine learning.

Estimating depth from a single image is an ill-posed and hard problem. The theory of estimating scene depth from scene intensity images is called shape from shading (SFS), surveyed by [266]. Because the problem involves complex physical and geometrical models, achieving compelling SFS results requires computationally complex algorithms and many assumptions about the scene and the environmental lighting. Therefore, researchers have recently proposed new, data-driven approaches.

Some newer approaches exist that try to estimate surface normals [126] or scene depth [125] from single images. These methods rely on training depth cues and spatial relationships based on associating still images and ground-truth depth pairs, and sometimes ground-truth semantic segmentations to then derive coarse scene depth [103]. These

approaches make use of complex and computationally costly algorithms and are infeasible for interactive scenarios on wearable devices.

Fanello et al. [55] have recently presented a random forest approach that automatically learns to predict per-pixel depth values from pixel intensities in the infrared (IR) range. Relying on IR intensity fall-off, this method requires mounting IR LEDs around the camera and modification of the camera's IR filter. In contrast, the extension of our technique uses *only* the built-in RGB camera, and hence is suitable for off-the-shelf, unmodified wearable devices.

4.6.2 Cascaded classification and regression forests

The pipeline presented in the previous section consists of three cascaded RF classifiers. The first stage decides the coarse depth of the hands (close, middle, or far), the second layer classifies the hand shape (six gestures), and the third layer recognizes hand parts (fingertips). The input to the algorithm is a binary mask of the hand, and the output is a gesture label. We modify the last stage of the pipeline and replace the part classification forest to a depth regression forest (see Figure 4.17). This new last stage is able to map the shape of the binary mask to a single depth value which corresponds to the mean distance of the hand from the camera. This is indeed a highly uncertain problem and naïve heuristic methods based on pixel counting usually fail.

Due to the difficulty of the problem, training a single regression forest for all gestures and all possible depths would result in very deep trees that may exceed the memory limits of wearable devices. Therefore we opt for the same staged architecture, where the role of the first two stages is to reduce the uncertainty the regression forest has to deal with. This means we can use shallower trees (i.e., less memory) to get relatively good accuracy, which will benefit devices like smartwatches and smartglasses. Previous work has also proven that classifying the scene into coarse depth regions greatly improves the fine depth estimation [55, 125]. We can then train separate depth regression forests for each case of coarse depth and gesture. To further simplify the problem, we restrict ourselves to only 3 gestures.

Coarse depth classification

After the hand segmentation, we classify the binary mask into three coarse layers of depth. We are interested in interaction in arms reach. Empirically, we found that 90 mm to 390 mm measured from smartglasses to the center of the palm is a comfortable range for most users. We divide this distance range into three intervals: $90-150$ mm, $150-240$ mm

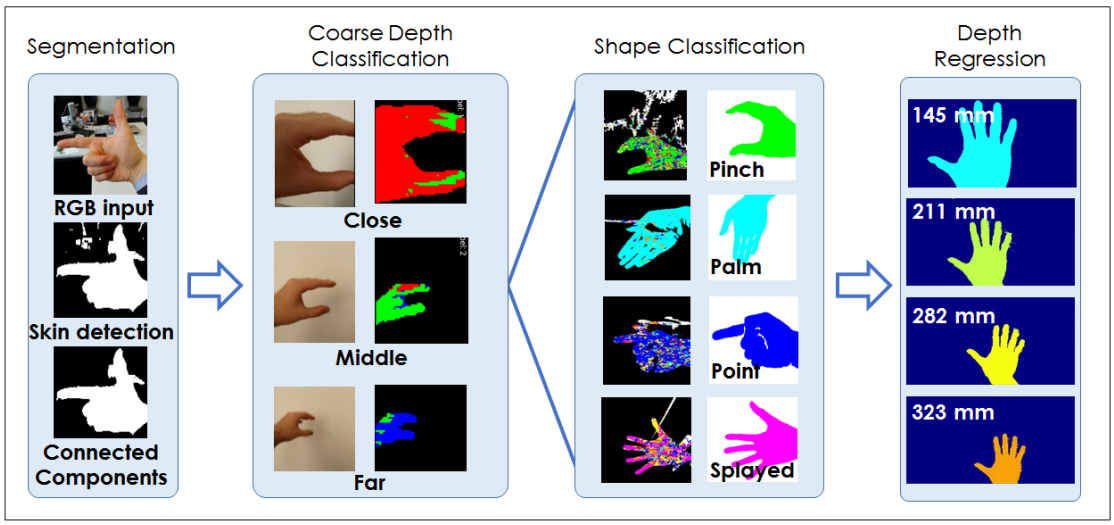


Figure 4.17: Classification-regression pipeline: (1) Hand segmentation. (2) Coarse depth classification. (3) Hand shape classification. (4) Fine-grained depth regression (average hand depth)

and $240-390$ mm as illustrated with color codes on the left size of Figure 4.18. Note that the ranges increase with distance because due to the perspective view, the changes in hand size are bigger in the close range than in the far range. With equal ranges, in the next stage the shape classifier of the close range would have to deal with a much larger variation than the shape classifier of the far range. With unequal ranges, we can balance this effect. Otherwise, the DCF works the same way as in the 2D case. The per-pixel probability distributions over coarse depth layers $d \in \{\texttt{close},\texttt{middle},\texttt{far},\texttt{noise}\}$ are forwarded to the next stage (see Figure 4.18, left).

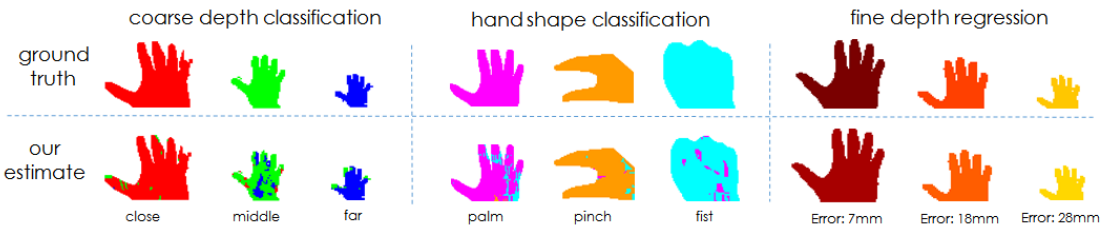


Figure 4.18: Classification-Regression pipeline overview. Top row: ground truth used for training. Bottom row: posteriors outputted at each level by the forests. Left: Coarse depth classification. Middle: Hand-shape classification. Right: Fine-grained depth regression (average depth per hand shown at different depth). The inset shows the error in mm compared to the ground truth. Figure adapted from Jie Song [203].

Shape classification

After finding the per-pixel probability of coarse depth range, the gesture classification is performed in the second stage. We evaluate each pixel p in the foreground mask S again get a per-pixel gesture probability distribution $P(g|p,S)$, where g is the label for different gestures. In our case, $g \in \{\texttt{splayed\_hand}, \texttt{pinch}, \texttt{closed\_hand}\}$. Because we cannot yet know for sure the actual depth range here, we run separate shape classification forests for each coarse depth layer that were trained on images of gestures at the respective distances. The predicted posteriors are then combined:

$$P(g|p,S) = \sum_{l=1}^{L} w_l P_l(g|p,S) \tag{4.6}$$

The output posterior $P(g|p,S)$ is the weighted sum over the estimates P_l of the three SCFs that work on the coarse depth layers. The weights w_l are the posterior probabilities estimated by the first stage. Note that here we need to run one shape classifier for each coarse depth interval simultaneously. In practice, this is not a problem as the trees have only depth 15. Finally, we pool the probabilities across the image to attain the gesture in the image $P(g|S)$ (Figure 4.17, middle).

Depth regression

In the third stage, we switch from classification to regression forests. We want to infer a depth value for each input pixel p. As we already know the gesture g from the previous stage, it is sufficient to run only the regression forest trained on examples of that particular gesture. The depth value $z(p|g,S)$ at pixel p is calculated as

$$z(p|g,S) = \sum_{l=1}^{L} w_l z_l(p|g,S) \tag{4.7}$$

where l denotes the coarse depth level and the weights w_l are again the posteriors from the first stage. Finally, we average the per-pixel depth estimates to get a single average depth value $P(z|S)$ of the hand.

Differences of classification and regression forests

At forest evaluation, we follow the standard method, i.e., we pass individual pixels to several decision trees and pool the results over all trees in the forest. At each split node,

we evaluate a split function, passing the pixel to its left or right child, until it reaches a leaf node. The split features and split criteria are the same as in the original method.

The classification forests in the DCF and SCF stages simply store learned, discrete probability distributions in their leaf nodes. The output of the DCF is the most probable coarse depth level, and the output of the SCF is the most probable gesture. The leafs of the regression forest in the last stage store continuous distributions. As these distributions might be multi-modal, outputting the largest mode may lead to irregular and noisy estimates in a pixel neighborhood[3]. Therefore, we additionally perform median filtering in a small neighborhood of each pixel.

The main difference from classification forest to regression forest is the entropy definition at training. For classification forests, we use the Shannon-entropy of the empirical discrete distributions (of coarse depth ranges or shapes, respectively). For the regression forest, we employ the differential entropy of the empirical continuous density $P(z|S)$. We model the depth distributions as 1D Gaussians. This way, the entropy definition reduces to $H(C) = log(\sigma_C)$, where σ_C is the variance of the 1D Gaussian fitted to the data points in set C. For more details on the training process, we refer to [45]. Except the entropy definition, the training follows the standard method (see Section 4.4) with the objective to maximize information gain at each split node. As the depth regression problem has a larger parameter space, we generate more random splits at training.

In the following section, we present how we acquire training data and elaborate on implementation details for smartphones and smartglasses. Next, we report our evaluations performed on low-level aspects of the algorithm, as well as on its integration with demonstration applications.

4.7 Training data and implementation

Random forests require large amounts of training data for high classification accuracy. Both the tree structures and the final probability distributions are learned from the annotated training data. To get a classifier that is robust to large variations in hand shapes, hand sizes, hand distances, and even to various executions of the same gesture, we need a large but balanced training set. We created demonstration applications of the 2D interaction pipeline on smartphones, and applications of the 3D interaction pipeline on smartglasses. Because the two settings are very different in terms of camera settings and interaction distance from the camera, we recorded different sets of training data for each pipeline.

[3]Note that pixel-labeling random forest models make the assumption that the pixel labels are independent and thus in general fail to enforce spatial smoothness.

4.7.1 Training the classification pipeline

Shape training data

To collect training data for the SCF, we recorded gestures of 20 persons performing 7 gestures under natural variation and recorded short video sequences. We also included one `no_gesture` class where participants casually moved their hands in front of the camera. In total, we recorded 50,000 images with scale and orientation variations and appearance. We split this dataset to 35,000 images for training and 15,000 images for testing in the following section.

We chose simple uniform background for collecting data and used the same RGB-segmentation method for labeling the hand pixels. We also recorded various unconstrained background images to gather noise data. The thresholded artificial noise and the labeled gesture images are combined to build synthetic training data which includes hand gestures and realistic noise. The whole procedure is illustrated on Figure 4.19:

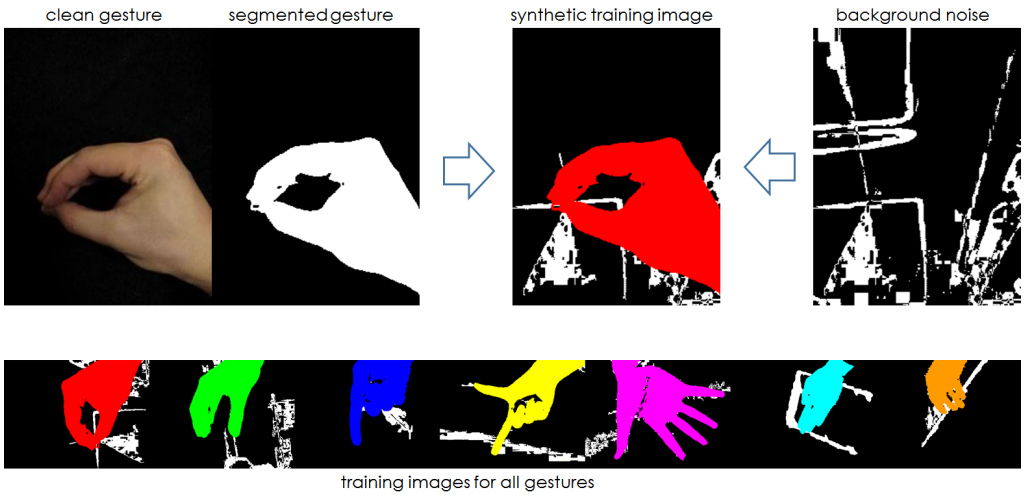


Figure 4.19: Generating training data for shape classification. The orange color represents the `no_gesture` (but hand) class, and white color represents the background noise class. Figure adapted from Jie Song [204].

Coarse depth training data

We collected additional 8,000 images to train the DCF, covering a larger range in camera-hand distance.

Part training data

For fingertip classification with the PCF, we require training data of not only segmented hands but also segmented parts of the hands. We recorded 5,000 images with a similar method as above, but wearing colored gloves, with distinct colors at the fingertips.

Figure 4.20: Generating training data for the fingertip classification. Figure adapted from Jie Song [204].

The number of trees in a forest and the depth of the trees are important parameters affecting classifier performance. Increasing the number of trees may enhance the classification accuracy, but it always increases the computational cost and memory footprint linearly. To ensure that recognition is fast enough for real-time interaction at minimum 30 Hz, we use 3 trees in each forest. The trees of the DCF are of depth 10, the trees of the SCF are of depth 20, and the trees of the PCF are of depth 15 in our implementation.

The described semi-automatic method is reasonably fast in producing a large set of labeled training data for both perfectly segmented hands as well as realistic background noise. We also release the collected training and test data to the public at [283].

4.7.2 Training the classification-regression pipeline

For the 3D interaction pipeline, our ground-truth images are acquired from a Creative Senz3D [278] depth camera attached to the head of the participants. We use simple depth thresholding and connected component analysis to attain clean segmentation of the hand. The binary segmentation masks are used in training the coarse depth and the shape classification forests, and the segmented depth map is used for training the fine depth regression forest. We record training data for the different gestures separately so that the shape labeling can be automated. The coarse depth labels are simply extracted from the

depth data. We also add artificial noise to the binary segmentation in a similar way as in creating the shape training data.

For coarse depth classification, we train a single tree of depth 8 using $8,000$ training images. This tree achieves an average classification accuracy of 90.1% in half-test-half-train cross-validation. For shape classification in the second stage, we use $4,000$ images for training each of the three SCFs at the three coarse depth layers, in total $3 \times 4,000 = 12,000$ images. In each forest, we train 3 trees of depth 15 achieving an average accuracy of 95.3%. For regression, we train one forest per coarse depth layer and per gesture, each consisting of 6 trees. This makes up 3 layers $\times$ 3 gestures $\times$ 6 trees $=$ 54 trees in total. Each tree is trained to depth 16 using $4,000$ training images. As we only execute one of these regression forests at a time, the algorithm still runs in real time. The training of the forests was performed by Jie Song.

Next, we give high-level implementation details of the algorithm described in the main paper, as well as information on its Android implementation. To the best of our knowledge, we present the first real-time implementation of random forests for pixel-labelling tasks on mobile devices.

4.7.3 Pseudo-code of random forest evaluation

Given a random forest $\mathcal{F}$ (trained offline) and a binary foreground segmentation mask S, the probabilities of the trained gestures can be efficiently computed using the following algorithm.

Each tree is stored in a two-dimensional array as shown in Figure 4.21. For each tree, we store the split parameters in split nodes in a `signed char` array[4] (note that jump vectors can be negative), and we represent the gesture probabilities in leaf nodes in an `unsigned char` array. The leaf nodes of the regression forest store a `float` depth value. For a tree with depth k, the split node indices run from 1 to $2^k - 1$ and the leaf node indices run from 0 to $2^k - 1$. The left and right children of node j are stored at $2j$ and $2j+1$, respectively. If a child is a leaf node, we subtract $2^k - 1$ from its index and load the corresponding probability distribution from the leaf array.

We pass each foreground pixel p from the binary mask through each tree t of the forest $\mathcal{F}$. Starting at the root node we traverse down the tree left or right based on binary decisions at each node. A test means comparing two offset mask values with the threshold $\Gamma_j \in \{[0,0],[0,1],[1,0],[1,1]\}$. The two offset positions are defined by the jump vectors v_j and w_j. Leaf nodes contain learnt probability distributions (PDFs) over the set of class

[4]Note that this data type must be explicitly enabled in the Android C++ compiler.

Data: random forest $\mathcal{F}$, binary segmentation mask S
Result: gesture probability distribution function (PDF)
forall pixels $p \in S$ **do**
  forall trees $t \in \mathcal{F}$ **do**
    $j = 1$;
    while *isSplitNode(j)* **do**
      load v,w,Γ from $t_{split}(j)$;
      $F_{v,w} = [S(p+v), S(p+w)]$;
      if *testFeature*($F_{v,w}$,Γ) **then**
        $j = leftChild(j)$;
      else
        $j = rightChild(j)$;
      end
    end
    load PDF from $t_{leaf}(j)$;
  end
  average the PDFs from all trees;
end
pick most probable gesture based on all pixel labels;

Algorithm 1: Pseudo-code of forest evaluation for shape classification.

split nodes

node	wx	wy	vx	vy	Γ
0	-	-	-	-	-
1	x	x	x	x	x
2	x	x	x	x	x
...	x	x	x	x	x
2^k-1	x	x	x	x	x

leaf nodes

node	p1	p2	p3	p4	p5	p6	p7	pN
0	x	x	x	x	x	x	x	x
1	x	x	x	x	x	x	x	x
2	x	x	x	x	x	x	x	x
...	x	x	x	x	x	x	x	x
2^k-1	x	x	x	x	x	x	x	x

Figure 4.21: Trees are stored in 2D byte arrays. Split nodes store the decision parameters w, v, Γ, while leaf nodes store the gesture probabilities and the noise probability.

labels. The final label of a pixel is determined by averaging the PDFs from all trees, and the performed gesture is decided by a majority vote over all foreground pixels in the image.

4.7.4 Android implementation overview

Figure 4.22 gives an overview of our implementation on Android devices. Here, we briefly summarize the most important aspects. Our recognition pipeline is written in native C++ and handles the camera access, performs background segmentation, random forest evaluation, gesture filtering and visualization of the different stages.

At runtime, we read images from the camera directly into an OpenGL ES texture and perform skin detection (Equation 4.1) and bilinear downsampling on the GPU. Next, we download the segmentation mask to the CPU and perform connected component analysis (CC) on the binary image, and we compensate for transformations via principal component analysis (PCA). The resulting binary mask is then classified by the DCF, SCF and, if applicable, by the PCF. Each forest updates a label image as well as the input mask. After classification, we upload the gesture labels and the cleaned segmentation mask back to the GPU for display. The per-pixel label probabilities are pooled across the image to get the overall gesture label. Finally, a low-pass filter stabilizes the class labels temporally and a Kalman filter stabilizes the part labels spatially. The filtered gestures are injected as events to the Java-based Android apps. Camera capture, GPU computations, CPU computations, and user interaction run in four different synchronized threads.

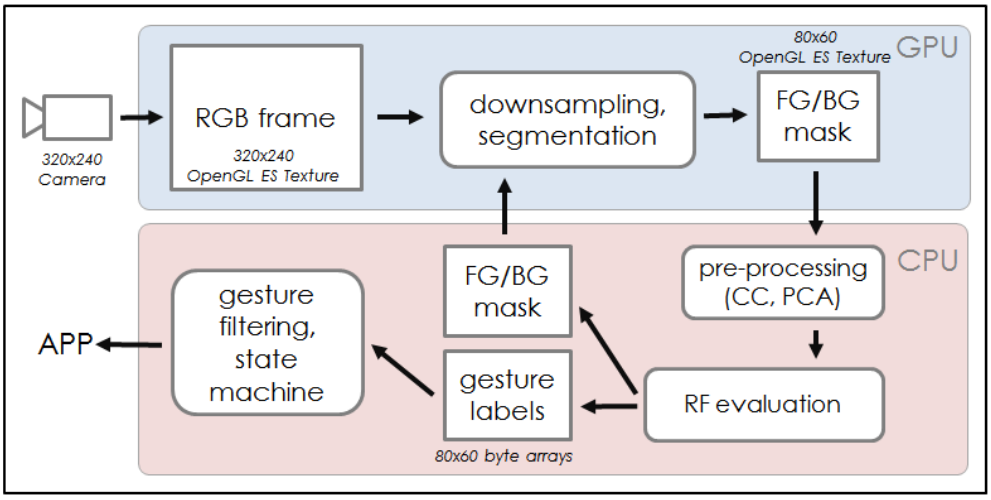


Figure 4.22: Overview of the recognition engine on an Android device. The recognized gestures can be injected into various applications built on top of our engine.

As most of the pipeline consists of pixel-wise computations, the algorithm is well suited for parallelization. The proposed tree storage as a matrix and tree traversal through array indices are also suitable for GPU implementation, where the trees can be stored as textures. We leave the GPU implementation for future work.

4.7.5 Device-specific modifications

The same architecture is portable across various Android devices, however, there are a few differences. Most importantly, the cameras of smartphones, smartwatches, and smartglasses have different field of view (matched to their respective applications) and therefore different training data is required for accurate classification. It also needs to be considered whether the user is allowed to use only left, right, or both hands. This depends not only on user preferences, but also on social conventions (watch is usually worn on the left hand, so gestures need to be performed with the right hand) and device properties (the Google Glass [289] has a camera only on the right side which implies interaction with the right hand). In general, as our hands are symmetric, it is sufficient to record training data with only one hand, and flip the images (or the feature vectors) if we want to recognize the other hand.

As the underlying hardware is different, adjustments may be necessary in the GPU texture handling. In particular, some GPU drivers support reading RGBA textures while some drivers support only BGR or other formats which needs to be handled carefully. The memory of smartwatches is especially limited hence shallower forests are very important. Furthermore, as the displays of the three devices are of very different size, the GUI elements need to be designed accordingly. For instance, as on smartwatches and smartglasses there is no space for visualizing both the camera input and the gesture labels separately, we visualize them overlaid on top of each other in a separate rendering pass.

4.8 Evaluation

We have conducted several experiments to evaluate the performance of our algorithms. The Android experiments were performed by the author, the Matlab experiments were performed by Jie Song. More results can be found in [203, 204]. We first evaluate an implementation of the 2D recognition approach with $6-12$ gestures. Afterwards, we evaluate an implementation of the 3D approach with only 3 gestures. The latter problem is significantly more complex in nature, hence only fewer gestures can be handled within our given resource constraints.

4.8.1 Gesture classification

In the first experiment, we test the performance of our classifier and we report accuracy figures from a single SCF, trained on the entire dataset. Looking first only at the SCF proves the discriminative power of the algorithm, and as we show later, this serves as lower bound and the multi-stage version performs equally well or even better.

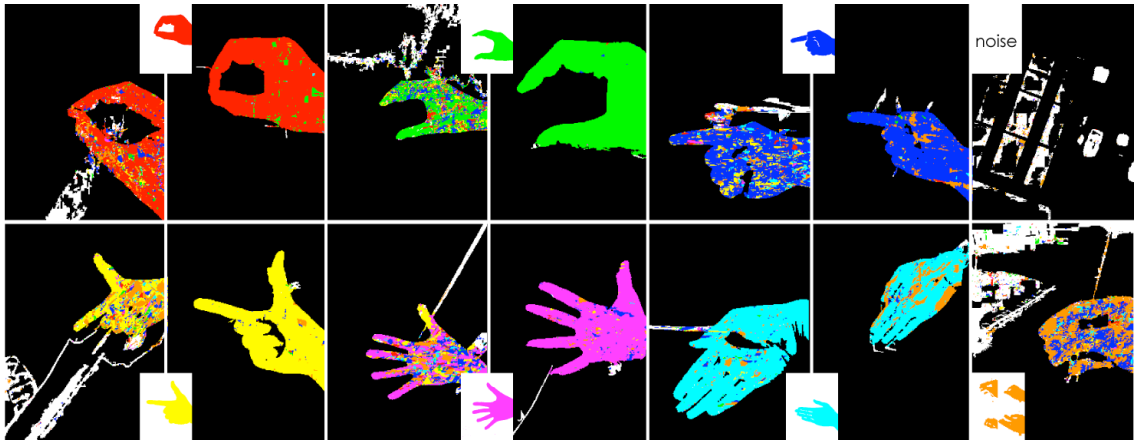


Figure 4.23: The current gesture set recognized by our classifier, two examples of each gesture class. In addition, an example of only `noise` and an example of `no_gesture` hand are shown on the right. The perfectly labeled ground truth gestures are shown in the white insets. Note the variation in scale and rotation as well as robustness to false positives from the segmentation step. Figure adapted from Fabrizio Pece [204].

Qualitative results

Figure 4.23 demonstrates qualitative results of the SCF classifying different hand gestures. Note the variations in hand orientation, distance to the camera, and different backgrounds, the classifier is robust to these variations. The robustness to segmentation errors is also apparent in the figures, the background `noise` class is labeled white by the classifier. The orange `no_gesture` class represents hand poses that do not belong to the six gestures but may happen between two gestures.

Confusion matrix

We summarize our results of the classification accuracy as a confusion matrix for the entire gesture set, using half training-half test (Table 4.1), and leave-one-out (Table 4.2) cross-validation methods. The confusion matrices are normalized over columns to show

which actual classes are responsible for a given classification output. The technique achieves a mean per-class per-frame accuracy of 98% and 93%, respectively. This means a robust per-frame gesture classifier even without temporal filtering. In our practical applications, we further improve the stability by adding a temporal filter over three frames, i.e., we inject a gesture event in the application when we see the the same gesture over three consecutive frames.

	green	red	blue	yellow	magenta	cyan	orange
pinch_open	0.98	0	0	0	0	0	0.01
pinch_close	0	1.00	0	0	0	0	0.01
pointing	0.02	0	0.99	0	0	0	0
gun	0	0	0	1.00	0	0	0
splayed_hand	0	0	0	0	0.99	0	0
flat_hand	0	0	0	0	0.01	0.99	0.07
no_gesture	0	0	0.01	0	0	0.01	0.91

Table 4.1: Confusion matrix (data from 20 users) using half training - half test cross-validation, average accuracy is 98%. Table adapted from Jie Song [204].

	green	red	blue	yellow	magenta	cyan	orange
pinch_open	0.88	0.03	0	0	0	0	0.02
pinch_close	0	0.93	0.05	0	0	0	0.01
pointing	0.02	0.01	0.90	0.04	0	0	0.01
gun	0	0	0.02	0.95	0	0	0
splayed_hand	0	0	0	0.01	0.99	0	0
flat_hand	0.05	0	0	0	0.01	0.99	0.11
no_gesture	0.05	0.03	0.03	0	0	0.01	0.85

Table 4.2: Confusion matrix (data from 20 users) using leave-one-out cross-validation, average accuracy 93%. Table adapted from Jie Song [204].

The effect of tree depth

The classification accuracy is increasing with the depth of the trees and the number of trees in the forest, however, we have to trade this for increasing memory requirement (and increasing training time). Table 4.3 and Table 4.4 summarize the classification performance at different tree depths. In our applications, we use 3 trees of depth 20 in the SCF which gives very good results. We report results with both leave-one-out and half training-half testing cross-validation.

	15	16	17	18	19	20
PPCR	0.79	0.82	0.85	0.87	0.89	**0.92**
PFCR	0.88	0.92	0.94	0.96	0.97	**0.98**

Table 4.3: Per-pixel classification rate (PPCR) and per-frame classification rate (PFCR) accuracy for half training - half-testing cross-validation of the shape classification forest (SCF) for different tree depths. The values are given for the SCF alone, averaged over all gestures. Figure adapted from Jie Song [204].

	15	16	17	18	19	20
PPCR	0.75	0.77	0.78	0.79	**0.80**	**0.80**
PFCR	0.86	0.89	0.92	0.92	0.92	**0.93**

Table 4.4: Per-pixel classification rate (PPCR) and per-frame classification rate (PFCR) accuracy for leave-one-out cross-validation of the shape classification forest (SCF) for different tree depths. The values are given for the SCF alone, averaged over all gestures. Figure adapted from Jie Song [204].

Staged and unstaged architecture

Figure 4.24 compares the performance of a single SCF and a combination of DCF+SCF when large hand depth variations occur in the gesture set. The SCF alone was not able to robustly cover this large variation within the maximum allowed tree depth (20). The staged classifier clearly outperforms the single SCF at all tree depths.

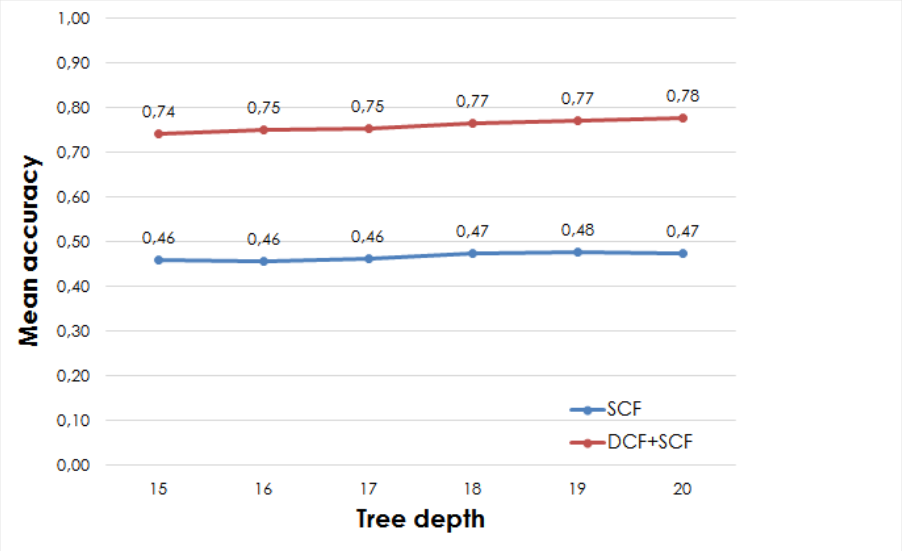


Figure 4.24: Comparison of classification accuracy (averaged over all classes) as a function of architecture (SCF alone vs. DCF and SCF together, different tree depths) when large hand depth variations occur in the test gesture set.

In our implementation, a single SCF of depth 20 with 3 trees consumes about 168 MB memory, assuming that we store each value as a 32-bit `integer` or `float`. In contrast, the combination of a DCF of depth 10 with 3 trees and an SCF of depth 15 with 3 trees consumes only about 5.3 MB memory, but the staged classifier performs $\sim 30\%$ better. Note that the memory footprint of a forest increases exponentially with tree depth and linearly with the number of trees. The memory requirements can be further reduced if we store the each split parameter and each probability value as `signed char` and `unsigned char`, respectively. Note that the jump vectors can be negative, but as we work with very small images, the range of $(-128, 127)$ is sufficient. For storing the probabilities, the range of $(0, 255)$ gives sufficient resolution as long as we do not have a large number of classes.

Extended gesture set

In the above experiments, we have considered a set of *six* gestures. To demonstrate the scalability and discriminative power of the approach, we have conducted further experiments with a larger gesture set. We selected *twelve* gestures from the American Sign Language alphabet (ASL) that have a unique contour. As the algorithm relies only on binary images, differences in appearance that are fully contained within the silhouette of a hand pose cannot be distinguished using the binary features. For example, the letters A, M, N, S, and T produce near identical outlines. Furthermore, we excluded signs with dynamics (i.e., motion) as we currently only recognize static gestures. Figure 4.25 shows the confusion matrix for the twelve gestures from the ASL alphabet. On this gesture set, we achieved similar performance to our original gestures. The overall average per-class accuracy is 83% in leave-one-out validation. This additional experiment further demonstrates the power of the classification algorithm and we believe that even more complex gesture sets could be handled.

Continuous gesture switching

While our algorithm handles only static gestures, many applications require continuous valued input. For example, imagine a gesture to control the volume of an audio player dynamically. One possibility to do using our approach is to exploit the per-pixel nature of the classifier. We have observed that the probabilities of symmetric gesture pairs (e.g., open and closed pinch) increase and decrease smoothly during the transition between the two gestures. Figure 4.26 plots the raw, unfiltered relative probability ratio between a pair of gestures (`pinch_open` and `pinch_close`). The ratio between these two values can then be mapped to continuous inputs.

O	0.78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C	0.03	0.90	0.04	0.03	0.04	0.07	0.18	0.00	0.00	0.00	0.00	0.01
D	0.00	0.02	0.89	0.14	0.00	0.01	0.00	0.00	0.01	0.07	0.00	0.06
L	0.00	0.00	0.00	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12
5	0.00	0.00	0.00	0.00	0.85	0.00	0.00	0.00	0.00	0.01	0.00	0.00
B	0.02	0.02	0.02	0.03	0.00	0.86	0.00	0.00	0.00	0.03	0.00	0.00
E	0.00	0.00	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00
Y	0.01	0.00	0.00	0.13	0.06	0.00	0.00	0.92	0.00	0.03	0.00	0.00
P	0.13	0.05	0.05	0.09	0.05	0.04	0.05	0.08	0.99	0.08	0.00	0.05
3	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.77	0.00	0.00
I	0.03	0.01	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	1.00	0.00
X	0.01	0.00	0.00	0.10	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.76

Figure 4.25: Confusion matrix for 12 gestures selected from American Sign Language.

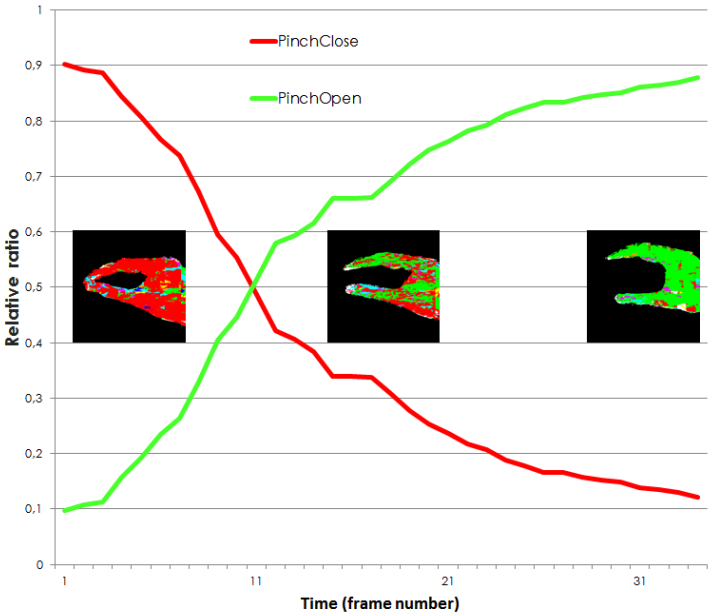


Figure 4.26: Continuous gesture input can be implemented by looking at the ratio of the first two modes in the per-pixel class probability distribution.

4.8.2 Depth estimation

In this section, we evaluate an implementation of the joint gesture-depth estimation approach with 3 gestures. Note that this is a different test setting and not a simple extension of the 2D experiments above. In particular, we use only 3 gestures due to the difficulty of the problem and the given resource constraints. We compare our technique against ground-truth (GT) data acquired from a Creative Senz3D [278] depth camera as baseline and against a naïve depth estimation technique based on raw hand size (foreground pixel count). This naïve baseline is calibrated offline and per user by moving the hand repeatedly to and away from the camera. We record *min* and *max* pixel counts and corresponding depth values. At runtime, we simply interpolate the depth between these two values. While not a very robust method, this actually works reasonably well – in particular with constant hand shape and linear motion along the z axis. Figure 4.27 visualizes the ground truth depth map, the average hand depth calculated from the depth map, and the estimates of our method and the naïve method.

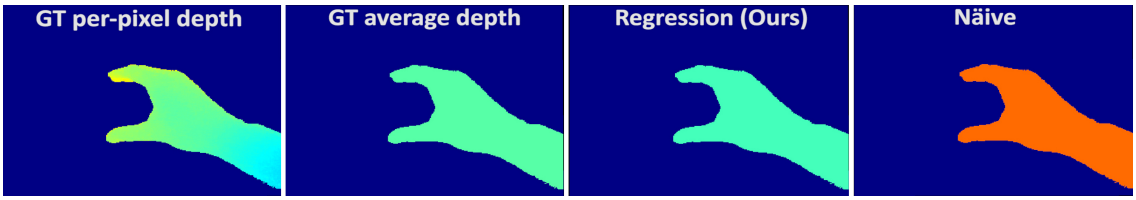


Figure 4.27: Visualization of the ground truth depth map, the average truth depth value, and estimation of our method and the naïve method.

Single gesture

Figure 4.28 plots the error of the two depth estimates as function of distance from the camera, compared to the ground truth depth value. Our technique compares favorably to the naïve method and also tracks the ground truth well. Qualitatively, our technique performs significantly better than the naïve approach.

Multiple gestures

The experiments so far were conducted with a single gesture. A more realistic scenario is evaluated in Figure 4.29 where we show depth estimates over $2,000$ frames and under gesture variation. In this sequence, the user shows three gestures at various depths, the plot is divided into three areas corresponding to the different gestures. Gray rectangles indicate the time between gestures. Note that the naïve method has big jumps in these

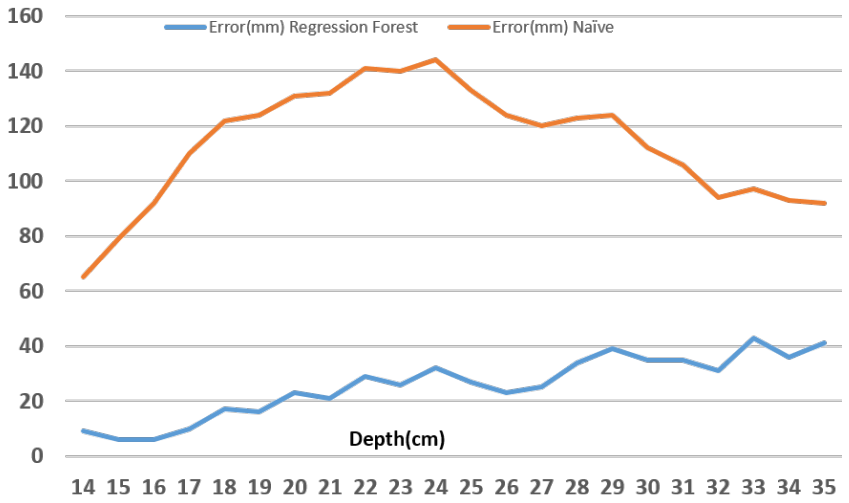


Figure 4.28: Depth estimation error as a function over depth. Our method (blue, average error 24 mm) tracks the hand with significantly less error than the naïve method (orange, average error 87 mm). The error of our method increases only slightly with distance (the maximum error is 4 cm). The error of the naïve method is higher, and increases non-linearly. Figure adapted from Jie Song [203].

areas. Our method tracks the ground truth closely, with small recurring spikes at the boundaries between the coarse depth levels. Recall that our three coarse depth levels are defined as $90-150$ mm, $150-240$ mm and $240-390$ mm, and the SCFs are switched at the level borders. We could detect and filter the spikes, or we could train the regression forests with training data that has more overlap across the level boundaries. In contrast, the naïve technique systematically over and undershoots the ground truth and exhibits a much larger average error (17.3 mm vs. 81.1 mm).

4.9 Applications

To demonstrate the accuracy and robustness of our algorithm, we implemented a set of compelling interaction techniques and application scenarios. In this section, we introduce a subset of these; a larger selection is presented in [203, 204] and accompanying videos (see Appendix). With these demonstrations, we would like to hint at the fact that our technique is by no means limited to barcode scanning but opens up a wide variety of possibilities. Our algorithm lends itself to a variety of novel interactions that extend the input space of wearable devices. We categorize our applications by typical device type, but we note here that any combination of these is possible, the only requirement is a color camera.

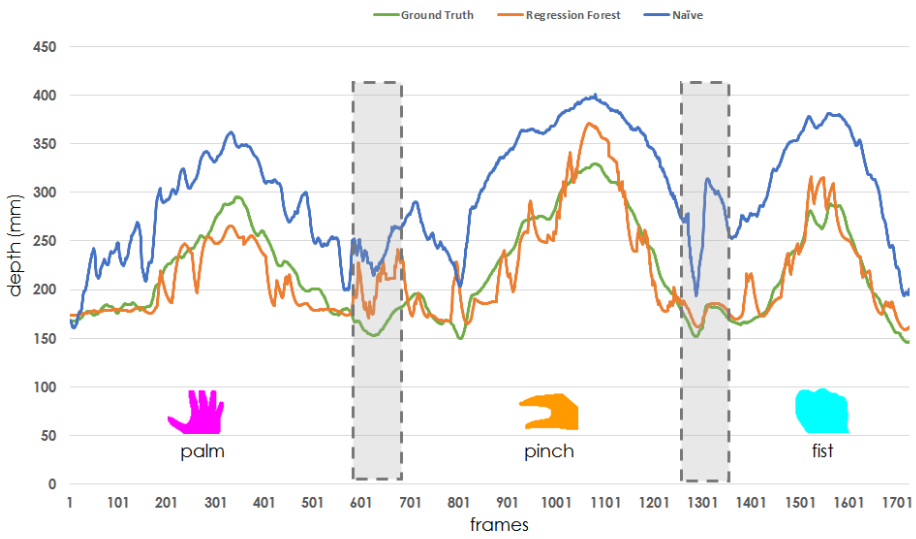


Figure 4.29: Depth estimation under gesture variation: the ground truth (green) is tracked closely by our method (orange), while the naïve method (blue) is significantly worse. Figure adapted from Jie Song [203].

4.9.1 Smartphone and tablet applications

In our smartphone demonstrations, our aim is to enable low-effort gestures that complement the primary touch screen input. There are numerous interaction primitives such as mode switches, menu selection, and certain types of navigation, where traditional input can be complemented or even replaced by in-air gestures. Furthermore, gesture input behind the display overcomes the issue of screen coverage on smaller touchscreens of wearables.

Figure 4.30 illustrates a map browsing application in which the user can control map style switching (1 and 2), zooming (3 and 4), and continuous panning (5 and 6) with in-air gestures behind the device. This allows a new type of bi-manual interaction in which the coarser tasks are performed by the non-dominant hand while the finer tasks are operated on the touch screen by the dominant hand as usual.

For showcasing the continuous gesture switching, we have also implemented a magnifier lens on the map that can be invoked by an in-air `pinch_open` gesture. The lens magnifies a particular area of interest on the map and can be positioned by moving the hand behind the device. The size of the lens (and hence magnification factor) can be controlled by opening and closing the pinch gesture. Cross-fading between the gesture probabilities can mapped to continuous interaction. The location of the lens can then be controlled for

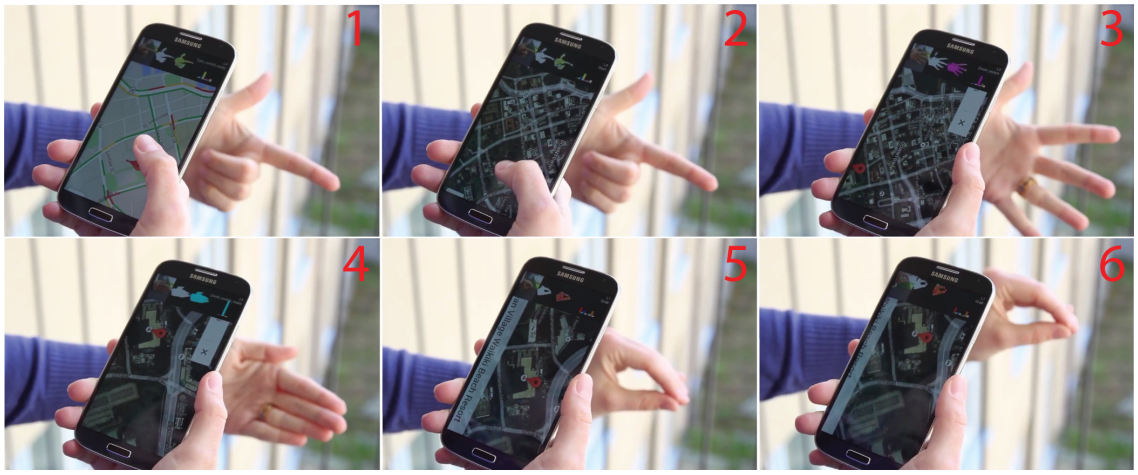


Figure 4.30: A map application with bi-manual user input. Gestures by the non-dominant hand behind the device successfully complement the touchscreen operated by the dominant hand. (1)-(2) style switching, (3)-(4) zooming, (5)-(6) panning.

instance via the fingertip locations (see Figure 4.31). This illustrates the usefulness of jointly recognizing gestures and fingertips.

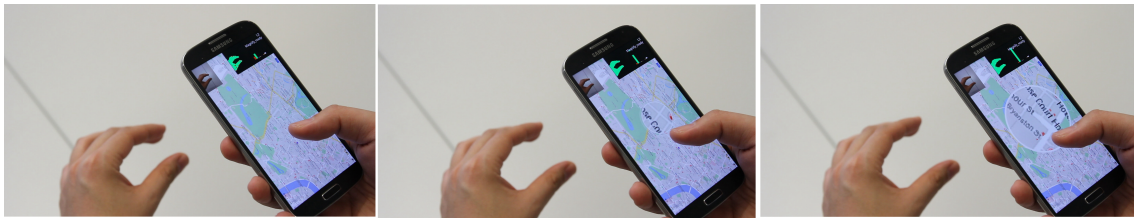


Figure 4.31: The size of magnifier on the map can be adjusted by the continuous transitions between the gestures `pinch_open` and `pinch_close`.

Next, we show a couple of scenarios for controlling smart appliances via our wearable computers (cf. Section 4.2 on universal user interfaces). In general, an appliance can be selected by scanning an attached tiny visual tag. Alternatively, an object can be identified via visual features (see Section 1.1.6 on visual recognition engines). Specifically, a display can also be recognized based on the displayed content as natural visual tag, like in previous works of the author [143, 211].

In Figure 4.32 (1), a tablet in the kitchen is controlled using simple, effortless gestures without touching the screen with wet hands. Here, we leverage the front-facing camera for gesture recognition. We implemented a video player and a recipe browser application for cooking, however, similar distant input is also often necessary in medical applications. Figure 4.32 (2) illustrates a medical visualization system [211] that can interfaced with our gesture recognition engine. In this concrete example, we apply the tablet's upward-facing

camera for recognition. Figure 4.32 (3) shows a network configuration application for smart homes [143]. A tablet recognizes a smart electricity meter based on visual features and visualizes its network connections. The user can accept or deny certain connections by simple gestures.

Figure 4.32: (1) in-air gestures controlling a cooking app. (2) gesture interface of a medical visualization screen. (3) an augmented reality network traffic visualizer.

4.9.2 Smartwatch applications

While multi-touch is an effective input method on larger-screen smartphones an tablets, devices with a very small touch screen like smartwatches especially benefit from in-air gestures that complement touch. Fortunately, the low computational requirements make our approach also applicable on smartwatches. Figure 4.33 illustrates how we can control a Web-enabled light bulb with simple gestures performed around our smartwatch.

Figure 4.33: Controlling a Web-enabled light bulb with simple gestures performed around the smartwatch. The subsequent `gun` and `pointing` gestures change the color of the light.

4.9.3 Smartglasses applications

We have also built proof-of-concept demonstrators on smartglasses. Our simpler gesture method can be used to select barcodes in a picking scenario. Figure 4.34 shows screenshots of our algorithm running on a Google Glass [289].

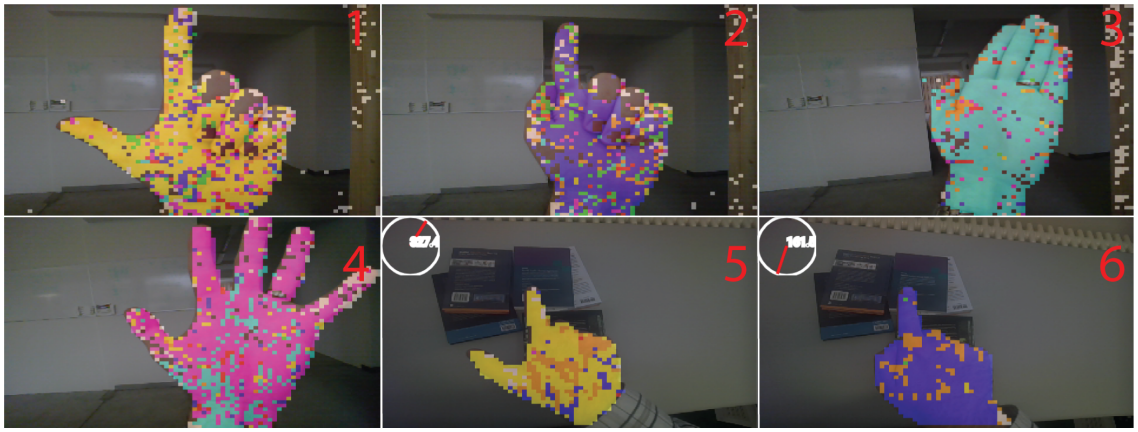


Figure 4.34: (1)-(4) screenshot of our gesture classifier on Google Glass [289] smartglasses. (5)-(6) Barcode selection in a picking scenario.

Finally, we show two applications where it is necessary to estimate the gesture and the hand position simultaneously, which allows the user to jointly control discrete and continuous variables. For example, gestures and depth may be used to invoke and browse linear list controls like selecting a contact card on smartglasses. Figure 4.35 shows a 3D phone book application. The contact list can be invoked by the pinch gesture, then a stack contact cards appear in 3D and the user can scroll through the cards by swiping closer or further. The fist gesture selects a contact and places the call. Figure 4.36 illustrates a

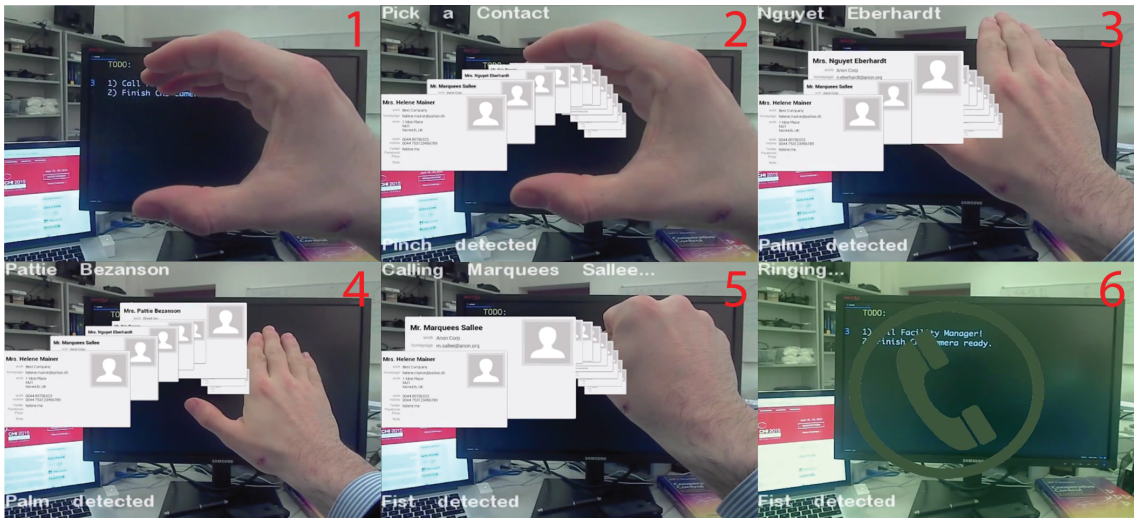


Figure 4.35: 3D gesture control in an augmented reality phone book. The user can open the phonebook by the pinch gesture (1)-(1), swipe through the contact cards by moving the hand further or closer (3)-(4), and pick a contact by grabbing it (5).

hierarchical 3D menu in which the menu and submenu items are represented as planets

and moons, respectively. The 3D position of the hand is used to cycle through menu items. Left and right swipes rotate the 3D menu, bringing the palm close to the camera expands a menu item, submenus can be browsed with left and right swipes, and the grab (fist) gesture selects an item. Our depth+gesture method is particularly suitable for continuous 3D manipulation of virtual objects in see-through AR scenarios, and for browsing and selecting 3D items.

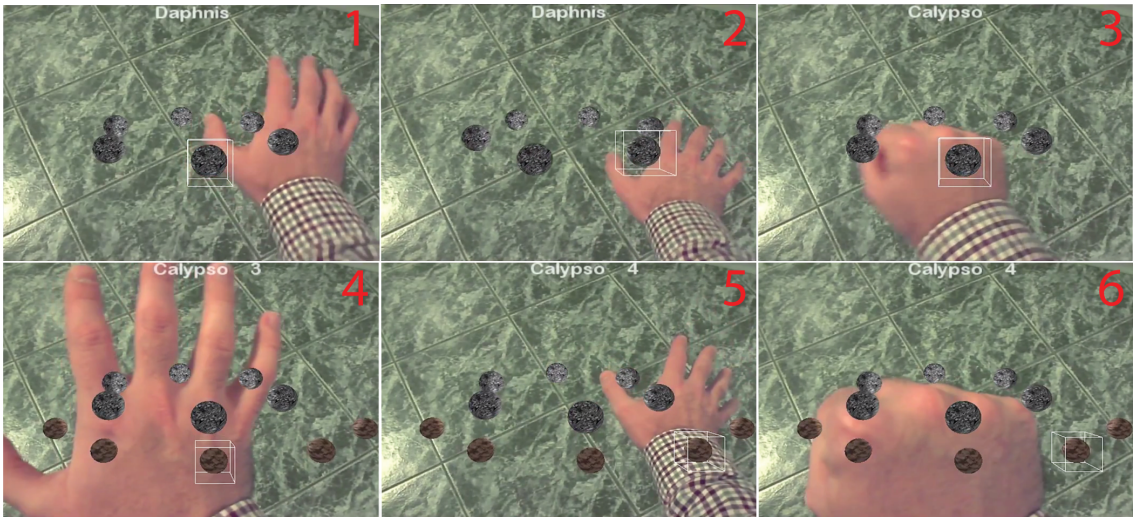


Figure 4.36: A hierarchical 3D menu system for augmented and virtual reality applications. The menu items can be browsed by swiping left or right (1),(2),(5), submenus can be opened by moving closer (3)-(4), and an item can be selected by grabbing it (6).

4.10 Discussion

The initial quantitative and qualitative results of our method are promising. The classification results performed by users in different lighting conditions and with changing backgrounds illustrates that our prototypes can be used to recognize a diverse set of gestures on off-the-shelf tablets, smartphones, but even smartwatches and smartglasses. To the best of our knowledge, this is the first real-time implementation of randomized decision forests on such resource-constrained devices, with remarkable robustness for gesture recognition. However, there are also a few issues to improve. In this section, we outline limitations and our ideas for future work.

4.10.1 Other gestures

Note that as our recognition engine is based on machine learning, new gestures can be simply added by recording new examples and repeating the training. Future work could explore the feasibility of using an even larger gesture set and more discriminative features that go beyond classifying binary images.

Our experiment with 12 gestures showed promising results (see Figure 4.25). However, our chosen binary features limit the type of gestures to those that produce visually distinguishable contours. In the future, we could investigate different types of features to overcome this limitation and to exploit the additional information that is present in the data (e.g., texture, edges). For example, a feature that captures texture information may be able to differentiate between a left hand facing down and a right hand facing up.

In our demonstrations, we used the same gestures in all applications, however, we could also train different decision forests for different applications. Different games require different gestures, for example. While the classification engine could be embedded in the wearable operation system, the app-specific (pre-trained) forests could be shipped as plug-ins for each application. It also needs to be further explored which gestures (in terms of duration, distance, etc.) are acceptable in public settings [5] in different cultures. A further interesting research question to explore is how we can systematically propose gestures so that the recognition rate will be maximized.

4.10.2 Dynamic gestures

Furthermore, the extension to dynamic gestures (e.g., by adding temporal features) should be explored. Note that we already allow a number of continuous interactions by fingertip tracking and continuous switch between open and closed pinch (see Figure 4.26), but our classifier is static by design. For future work, we want to explore the possibility to capture dynamic gestures such as flicks and swipes directly. We plan to built upon recent work demonstrating decision forests for recognizing motion gestures [28, 229].

4.10.3 Lighting conditions

As all vision-based techniques, our method is influenced by the ambient lighting. We demonstrated that the system can cope with a natural lighting both indoors and outdoors, and also artificial indoor lighting. While the method is sufficiently robust to the art of illumination, sudden changes such as entering or leaving a building or switching on strong

direct light sources may cause an problem. Other failure cases we experienced were under very low lighting conditions when the image was dark and suffered from color noise, or under red-toned ambient lighting, for instance at sunset time.

4.10.4 Segmentation

While our technique is robust to false positive segmentation failure, it is sensitive to false negatives as these erode the shape of the hand. False negatives can be counteracted by the help of the user in setting the the skin detection parameter with a slider. We can optionally feed back the per-pixel classification results (hand/no hand) to the segmentation step to automatically adjust the color thresholds, but we found that the improvement in segmentation is is not worth the extra computational effort.

Thanks to the wide variety of artificially generated background noise patterns, our framework does not need any prior knowledge of the scene or background, but only sufficient initial training data. This allows us to run in unconstrained environments and in a variety of scenes. Ideally, more sophisticated and reliable skin detection mechanisms would be desirable, perhaps also implementing the segmentation as a decision forest classifier at the very first stage.

4.10.5 Depth estimation

Our depth estimation method also has a few limitations. First, it is not a general purpose depth estimation technique, but is only trained to recover depth of a hand and only at close range in front of the camera.

Second, we only regress a single depth value of the hand. While a single depth value together with the x, y position of the hand's centroid is sufficient for many applications, it cannot compete with high-fidelity commercial depth cameras. The main advantage of our method remains that RGB cameras can already be found in practically all wearable devices, while the size and power consumption of depth cameras are still hindering their use in small devices.

While our training data consists of gestures performed by many people at our institute, we could not cover all variations in hand shape over the population. In particular, the depth estimation stage obviously overestimates the distance of children hands as they are smaller than the hands in the training data.

Our future work will include experimenting with estimating depth of the individual parts of the hand such as the fingertips and the palm. Given that the method operates on hand silhouettes only, it is unclear whether this can be extended to fine-grained depth estimates, but offline methods [125, 126] have shown promising results.

4.11 Conclusions

In this chapter, we explored new ways of interaction with tagged smart appliances via personal wearable computers. We generalized the role of a universal interaction device from the smartphone to other always-available wearable devices and combinations of them. Starting from the concept of user interface outsourcing from smart appliances to universal interaction devices, we introduced the concept of capability insourcing from our wearables to our appliances. We explored through a few examples what new functionalities the capability sharing allows. In particular, we highlighted that natural input techniques implemented on our wearable computers can be applied to control our smart appliances via the insourcing mechanism. We therefore reviewed the existing wearable natural input techniques and identified the need for gestural input without hardware modifications. Next, we proposed a novel machine learning based algorithm for robust gestural interactions around unmodified mobile and wearable devices, including tablets, smartphones, smartwatches, and smartglasses. To the best of our knowledge, this is the first random forest based gesture recognizer running in real time on off-the-shelf devices. As input, the algorithm requires only a regular color camera which is omnipresent in wearable devices. We introduced a multi-stage classification framework in order to achieve high accuracy at low memory footprint. Furthermore, we showed that by switching to a classification-regression pipeline, the framework is able to estimate continuous 3D hand position for 3D mobile interaction without a depth camera. We also proposed a number of compelling interaction scenarios built around our gesture recognition engine, allowing for in-air gestural interaction and smart appliance control.

Chapter 5

Conclusions and outlook

This chapter briefly summarizes the achievements and main contributions of this dissertation to the field of wearable barcode scanning. We also outline a few remaining open issues and suggest directions for future research.

5.1 Summary of contributions

Our everyday appliances are getting smarter and are able to sense, communicate, and act autonomously and cooperatively. Our cloud databases are storing almost unlimited information generated about smart objects but also about virtual counterparts of conventional objects. This enormous amount of information is becoming the fuel of a new economy, which brings with it the fundamental prerequisite of uniquely identifying and linking objects and information. Today, visual codes like barcodes and QR codes are the most prevalent linking elements, but the scanning of these codes usually requires expensive dedicated barcode scanners. In parallel, we are currently witnessing how wearable computers with extensive sensing and computing capabilities are transforming from early research prototypes to commercial products. These computers are not only serving as ubiquitous communication tools, but are also reshaping the way we consume information, and are also taking the role of universal interaction devices. To apply these devices as always-available wearable barcode scanners thus seems to be a promising idea.

Our overall goal is to bring easy information access to everyone by enabling the scanning of visual codes on commodity wearable computers. The algorithms and methods proposed in this dissertation compensate important drawbacks of smartphones, smartwatches and smartglasses when compared to traditional barcode scanners, and are pushing forward the

state of the art in terms of accuracy, speed, and robustness. We presented several novel methods, their fast and robust prototype implementations, and their evaluation in practical settings. Our main contributions on visual code localization, motion blur compensation, and gesture control are summarized below.

5.1.1 Visual code localization

We developed a fast and robust localization approach for both 1D and 2D visual codes in large digital images. Our method can simultaneously localize codes of various symbologies, even at different orientations, and over a fairly wide range of scales. In addition, the method is more robust to blur than previous approaches. Robust scanning of multiple codes, and potentially several symbologies at the same time, is important for many enterprise applications. We described the implementation, optimization, and evaluation of the algorithm on the CPU as well as on the embedded GPU of smartphones and smartglasses.

We tested the effectiveness of our implementation in qualitative and quantitative experiments on images degraded by synthetic and real blur. Our results with publicly available data sets indicate that the method provides robust orientation and scale invariance, and localizes both 1D and 2D barcodes in sharp and blurry images. With UPC/EAN codes, we achieve sharp detection rates similar to other approaches but with no limiting assumptions on code size, code orientation, or code position. Additionally, our algorithm also detects close to 17% of the blurry codes, which means in 17% of the cases, a blurry decoder can already start decoding the barcode even before the autofocus was triggered. This results in much faster scanning and better user experience. Furthermore, the algorithm can also detect various sharp and blurry 2D codes.

The resolution of current generation smartphone cameras is already high enough for decoding multiple distant codes that are cut out from a high-resolution image. Hence, applying our localization algorithm as a preprocessing step in existing (blurry) decoding algorithms, we can realize simultaneous scanning at a distance, which was not possible before with consumer devices. However, the true advantage of scanning multiple small codes is unleashed with head-mounted cameras and smartglasses. Scanning multiple products (codes) on a shelf may soon become possible without the need for holding the items close to the camera.

5.1.2 Motion blur compensation

We presented a fast and robust method for reading severely blurred QR codes in images casually captured on the go with mobile devices. Our new, practical blur removal algorithm is specifically tailored for decoding motion-blurred QR codes on CPU- and memory-constrained wearable computers. We improved on existing general-purpose deblurring methods by adapting each step to the specific properties of QR codes that ensures fast convergence to the correct solution. We also proposed and empirically evaluated a new initialization scheme that greatly improved convergence and the quality of the results in removing large motion blur.

We presented fast PC and Android implementations and showed in thorough experiments that our iterative restoration-recognition algorithm can quickly decode QR code images degraded by synthetic or real motion blur. From the comparison with the state of the art, we concluded that our restoration quality is on par with existing methods while the restoration speed is about a magnitude faster.

In addition, we presented a method to render piecewise uniform blur kernels from built-in gyroscope measurements. This fast approximation can be used as an initialization step for our blur removal algorithm so that less iterations are required in the multiscale restoration loop. Edge-based and sensor-based blur estimation approaches are well applicable in smartphone and smartglasses scenarios. We applied existing off-line and on-line methods for gyroscope and camera calibration; however, a robust on-line calibration method is still an open question.

Fast blur compensation means that no precise code alignment is required, but the user can simply swipe the camera in front of the code. For instance, we can imagine a tablet computer as a point of sale system for small businesses; the cashier can simply swipe the products over the front camera. For the post and the logistics industry, blurry QR scanning saves time because codes can be scanned without stopping the parcels. Our restoration-recognition algorithm brings cheap wearable QR scanning one step closer to enterprise-grade performance with a great potential for practical applicability.

5.1.3 Gesture control

We revisited the role of the smartphone as an always-available universal interaction device and generalized the notion to other wearable devices and also combinations of them. We described how user interface primitives of tagged smart appliances can be automatically "outsourced" to our wearables, and introduced how capabilities of the wearables can be

"insourced" to smart appliances. In particular, any advanced input method implemented on wearables can be also insourced to other appliances in a smart environment.

Therefore, we focused our attention on improving the wearable input from small buttons and touch screens to more natural ways like hand gestures. We proposed a fast and robust gesture recognition algorithm for unmodified mobile and wearable devices including tablets, smartphones, smartwatches, and smartglasses. While there is extensive prior work on gesture recognition, our multi-stage gesture classification framework is unique in a sense that *i)* it uses a single camera *ii)* and runs in real time *iii)* on resource-constrained devices *iv)* without external sensors.

In addition, we showed how to extend our framework to jointly classify discrete gestures and regress continuous 3D hand position for 3D input on smartglasses, still requiring only monocular camera input. We evaluated our methods quantitatively and qualitatively, and showcased various scenarios where in-air gestures successfully complement touchscreen input, and where in-air gestures control smart appliances.

Fast and robust gesture control is not only advantageous in barcode scanning, but is in general widely applicable in wearable augmented reality, a truly ubiquitous user interface that seamlessly blends the physical and the digital. By bringing vision-based natural hand gesture control even to devices that have no camera, we open the door for new smart functionalities.

5.1.4 Demonstrators

Along with the dissertation, we have designed and implemented several showcase scenarios and demonstrators of our contributions. All proposed algorithms have been implemented to be compatible with various platforms (e.g., Android, iOS, Windows Phone) and device families (e.g., smartphone, smartwatches, smartglasses, tablets) with their resource constraints in mind. Major parts of the research results of this thesis have been orally presented at conferences [145, 203, 204, 205, 206, 208, 209, 210, 212] and demonstrated running directly on smartphones and/or wearable devices.

Overall, we successfully addressed all challenges identified in Section 1.4, and presented fast and robust solutions on computationally restricted wearable devices for enabling the vision of wearable barcode scanning.

5.1.5 Contribution statement

The contributions presented in this thesis partially rely on work and ideas of other people as listed below:

Chapter 2: The visual code localization methods, their efficient implementations, and their evaluation are original work of the author.

Chapter 3: Most of the blur removal work has been done by the author, except a few implementation parts by Bachelor and Master students. Recording sensor data on the mobile phone was implemented by Carlo Beltrame. Rendering blur kernels from sensor data was implemented by Severin Münger. The Android camera control is partly the work of Luc Humair. Parts of the document deblurring code from Stephan Semmler was reused in our QR deblurring pipeline. The students have received paper co-authorship for their contributions [210, 212].

Chapter 4: The prototype of user interface beaming is joint work with Simon Mayer who implemented the semantic description of user interface primitives and the Web interface of the presented smart appliances. The concept of user interface insourcing roots in discussions with Friedemann Mattern. The gesture classification and the 3D interaction method is the result of joint work with the Advanced Interactive Technologies Group of ETH Zurich led by Otmar Hilliges. Most of the results have been published as peer reviewed conference papers [203, 204, 205]. All co-authors contributed to the design of the algorithms and to the papers. The training of the classification forest was performed by Jie Song and Sean Fanello. The training of the classification-regression forest was the work of Jie Song and Fabrizio Pece. The Matlab experiments for evaluation were performed by Jie Song, the Android experiments were performed by the author. The Web interface of the LIFX light bulb [303] is the work of Wilhelm Kleiminger. The real-time random forest implementation on smartphones, smartwatches, and smartglasses is sole work of the author. The applications around the algorithm, the 3D user interaction models, and their implementations are original work of the author.

5.1.6 Other work not included in the thesis

During the PhD studies, I also had the chance to contribute to topics other than barcode scanning, in particular mobile augmented reality and interaction. Based on my master

thesis, together with Peter Rautek, Hartmut Seichter, and Eduard Gröller, we presented the concept of *augmented visualization* [211] where a handheld mobile device acts as a magic lens in front of a large visualization screen and allows for discovering viewer-dependent details overlaid on the common content. With Florian Daiber and Tomer Weller, we developed a *wearable biking assistant* [207] that provides instant performance feedback and context-aware notifications overlaid on a biker's view. During my internship at Qualcomm Research, I had the chance to work on the *Vuforia* augmented reality engine for a better tracking of textureless 3D objects. In collaboration with Simon Mayer, we developed a system for *visualizing network traffic* in smart home environments [142, 143]. Also, we co-invented *user interface beaming* [145], a method that allows users to conveniently interact with smart things in their environment by combining the capabilities of various personal wearable computers. Both works make use of a simple and fast *appliance recognition* technique that we presented in [144]. I also helped Frederik Hermans and Liam McNamara in developing a *new visual code* whose decodability is gracefully degrading with the scanning distance [80]. With Marian George and Mihai Bâce, we worked on *assistive technologies* in the context of shopping [65] and product assembly scenarios using smartphones and other wearable cameras. I have also supervised various student projects on wireless sensor networks, mobile optical character recognition, 3D reconstruction, and augmented reality.

5.2 Open problems and future work

With our original goals and our accomplished contributions in mind, we now mention some open challenges and provide directions for interesting future work.

5.2.1 Energy and durability

One of the key issues with wearable computer vision is the high computational time with large input images, and this is also coupled with high energy consumption and heat problems in wearable devices. We highlighted that most parts of our algorithms are suitable for parallel processing. Shifting the processing to the GPU that has more cores at lower frequency than the CPU means lower energy consumption. We also believe it would be beneficial to design hybrid cameras for wearables that can be operated in two modi: a lower quality but low-energy, always-on video capture mode, and the normal high-quality picture capture mode. We also expect that dedicated computer vision processors will become widespread in the close future.

Another important issue is durability for industrial applications. An industrial scanner should operate in a wide range of temperatures such as $-30\,^{\circ}C$ to $50\,^{\circ}C$, should be dust and water or at least drop resistant, and should survive drops to concrete from a height of 2 meters. We believe that these requirements can be fulfilled by designing insulated and rugged coatings for smart devices.

5.2.2 Multi image deblurring and super resolution

If we want to scan small and distant codes as well, the resolution of the image sensor may not be high enough, so the options for super-resolution from subsequent video frames are worth to be investigated. The subsequent frames in the camera feed are slightly shifted recordings of the same scene, so with sub-pixel image registration techniques, it might become possible to read codes smaller than what the actual camera resolution allows. However, as the motion estimation needs to be very precise for this, higher than $2\times$ resolution increase is infeasible in practice with current super resolution algorithms. As also mentioned in Section 3.5, having multiple images is also advantageous for deblurring, but new robust image alignment methods are necessary. Multi image restoration with joint alignment, deblurring, and super resolution at practical speed is a very interesting direction for future work. Recent approaches show promising results but are still rather slow [262]. Robust detection and decoding of tiny codes in the environment could find numerous applications in ubiquitous computing.

5.2.3 Machine learning for image restoration

Very recently, a couple of machine learning based methods have been proposed for denoising [56], deblurring [190, 218, 253], and also for super resolution [88, 191]. These methods apply either random forests or deep neural networks that can learn and infer the necessary restoration kernels by looking at degraded image patches. It is an interesting research question how such methods would perform on visual codes or text, or whether the special structure would allow certain speed optimizations in the training and inference.

In addition to visual codes, we could explore how our blur removal algorithm can be applied in other domains. We think it might work for example with text deblurring where the text consists of two-color structures. This is an important issue for an eye-worn camera in recording personal memories, whiteboards, documents, or to-do lists. Fast blur removal and super resolution would allow reading far away text through smartglasses.

5.2.4 Robust visual code designs

Assuming we might soon reach the limits in how much we can improve our cameras, we can also investigate in how we can improve the robustness of our visual codes against known degradations. As a new research area, we can design new visual codes that are tolerant to blur or noise at the price of possibly reduced decodable information. In [81], the authors presented a visual code that encodes the data in the Fourier domain instead of the spatial domain, thereby the code's decodability is gracefully degrading with the scanning distance. Once super resolution at practical speeds and robust code designs appear, they will enable *long-range barcode scanning* with numerous practical applications.

5.2.5 Automatic camera and motion sensor calibration

We have shown in Section 3.5 that camera properties and capture settings can highly influence the success of blur removal algorithms. In particular, signal non-linearities introduced during the capture process may cause blur removal algorithms to fail or not converge at all. We could overcome this in our experiments with high-end models that allow manual control and access to linear pixel measurements. However, most devices today – as also shown in our experiments with Google Glass – do not allow manual control and require calibration of the camera response function (CRF). This is a crucial limitation knowing that the CRF is different for each camera and even in different settings of the same camera. Because wearable cameras are equipped with numerous automatic image enhancement functionalities, the CRF actually needs to be estimated continuously. We are not aware of any method for automatic online calibration of the CRF.

Relying on inertial sensors for blur estimation is fast, however, the quality of the estimated kernels is largely affected by the quality of the sensor data and the calibration of the camera and the sensors. In particular, we noticed in our experiments that the delay between timestamps of camera frames and sensor measurements is slightly varying over time, therefore online calibration is necessary also here. In future work, we could build on the theory of visual-inertial odometry for online auto-calibration, keeping our resource constraints in mind.

5.2.6 Wearable hardware development

We believe that wearable computers will become even more powerful in the close future, mainly driven by mobile entertainment and computational photography. The high demand of such applications will push the industry forward in developing new standards, which is

evident when we take a look at the industry-wide Khronos standardization group. Khronos is developing new APIs for accelerated computer graphics (*OpenGL ES* [314]), accelerated computer vision (*OpenVX* [316]), better inter-operability and long-term convergence of graphics and computing (*Vulkan* [345]), better camera handling (*OpenKCam* [315]), better sensor fusion (*StreamInput* [337]), etc. On the hardware side, special-purpose computer vision chips and machine learning chips are in development that will greatly accelerate our restoration and recognition tasks.

We presented separate components for a new generation of barcode scanning; however, integrating all these components requires further design and engineering effort. All our algorithms have been carefully designed with resource constraints in mind, and with a high degree of data independency (mostly per-pixel operations). These aspects make our methods well suited for parallelization and applicable in the future, even when technology changes.

5.2.7 New applications

We believe that using wearable computers for scanning visual codes can make daily processes of people more efficient. We imagine, for example, a system connecting different wearable computers to realize a personal shopping assistant. Such an assistant would reduce the time required to find all products on the shopping list and could also automate the checkout process. In a more general sense, the combination of the wearables allows new, innovative applications given that the smartglasses share the user's viewpoint, the smartwatches provide an always-available touch screen, and the smartphone has advanced sensors and fast Internet connectivity. In the shopping scenario, for example, the camera of any device can be applied to scan the barcodes of articles present in the shop. Through scanning an object, a lot of information can be retrieved, such as in which category the products are to be found, pricing, exact names, and the location in the store. If a digital map of the store is available, in-store navigation using the camera or other sensors of the devices could further assist the user. These and previous examples in the text illustrate that wearable barcode scanning could create new use cases that are not possible with today's technology.

5.3 Closing remarks

We have shown in this dissertation that in the area of wearable barcode scanning and generally in the world of binary images, difficult computer vision problems like *object*

(a) Object segmentation (b) Image restoration (c) Shape classification

Figure 5.1: We have shown that in the world of binary images, these generally very difficult problems can have fast and robust solutions even on resource-constrained wearable devices.

detection, *image restoration*, and *shape classification* can have efficient solutions even on resource-constrained wearable devices. Our methods are pushing forward the state of the art in terms of accuracy, speed, and robustness, and thereby help to bring the comfort, the productivity, and the new business opportunities of wearable scanning also to the professional users who are accustomed to the performance of their laser scanners.

Realizing our goal of easy and ubiquitous information access can have a significant impact on business processes and mobility hardware in the enterprise domain, but it can also bring many benefits to consumers. For employees, the smartphones and smartglasses could be, as a first step, simply put into rugged shells and replace the expensive dedicated scanning hardware. For consumers, the improved performance means that barcode scanning gets even faster and easier facilitating price comparison, accessing product information, and building shopping lists. Tiny visual codes allow easy configuration of our smart environment and interaction with smart appliances, while robust visual codes allow communication over screen-camera links facilitating context-aware and location-based services.

Overall, robust wearable barcode scanning bears the potential to become one of the successful application domains of wearable computing, and it can assert the role of the smartphone and pave the road for smartglasses to become the essential tools for bridging the gap between the physical and digital world.

Bibliography

[1] Robert Adelmann. Mobile phone based interaction with everyday products – on the go. In *Proceedings of the International Conference on Next Generation Mobile Applications, Services and Technologies*, NGMAST '07, pages 63–69, 2007.

[2] Robert Adelmann. *An efficient bar code recognition engine for enabling mobile services*. PhD thesis, ETH Zurich, 2011.

[3] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 3D tracking via body radio reflections. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI '14, pages 317–329, Berkeley, CA, USA, 2014. USENIX Association.

[4] Amit Agrawal, Yi Xu, and Ramesh Raskar. Invertible motion blur in video. *ACM Transactions on Graphics*, 28(3):95:1–95:8, July 2009.

[5] David Ahlström, Khalad Hasan, and Pourang Irani. Are you comfortable doing that?: Acceptance studies of around-device gestures in and for public settings. In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '14, pages 193–202, New York, NY, USA, 2014. ACM.

[6] Jonas Alfthan. Robust detection of two-dimensional barcodes in blurry images. Master's thesis, KTH Stockholm, Sweden, 2008.

[7] Brian Amento, Will Hill, and Loren Terveen. The sound of one hand: A wrist-mounted bio-acoustic fingertip gesture interface. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 724–725, New York, NY, USA, 2002. ACM.

[8] Yali Amit and Donald Geman. Shape quantization and recognition with randomized trees. *Neural Computing*, 9(7):1545–1588, October 1997.

[9] Christoph Amma, Thomas Krings, Jonas Böer, and Tanja Schultz. Advancing muscle-computer interfaces with high-density electromyography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 929–938, New York, NY, USA, 2015. ACM.

[10] Shigeru Ando. Image field categorization and edge/corner detection from gradient covariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2):179–190, February 2000.

[11] Shigeru Ando and Hidekata Hontani. Automatic visual searching and reading of barcodes in 3-D scene. In *Proceedings of the IEEE International Vehicle Electronics Conference*, IVEC'01, pages 49–54, 2001.

[12] Daniel Ashbrook, Patrick Baudisch, and Sean White. Nenya: Subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2043–2046, New York, NY, USA, 2011. ACM.

[13] Hyeoungho Bae, Charless C. Fowlkes, and Pai H. Chou. Accurate motion deblurring using camera motion tracking and scene depth. In *Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision*, WACV '13, pages 148–153, Washington, DC, USA, 2013. IEEE Computer Society.

[14] Rafael Ballagas, Jan Borchers, Michael Rohs, and Jennifer G. Sheridan. The smart phone: A ubiquitous input device. *IEEE Pervasive Computing*, 5(1), January 2006.

[15] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 640–653, Berlin, Heidelberg, 2012. Springer-Verlag.

[16] Edward Barkan and Jerome Swartz. System design considerations in bar-code laser scanning. *Optical Engineering*, 23(4), 1984.

[17] Benedicte Bascle, Andrew Blake, and Andrew Zisserman. Motion deblurring and super-resolution from an image sequence. In *Proceedings of the 4th European Conference on Computer Vision - Volume II*, ECCV '96, pages 573–582, London, UK, 1996. Springer-Verlag.

[18] Eric P. Batterman and Donald G. Chandler. Omnidirectional wide range hand held bar code reader, January 3 1995. US Patent 5,378,883.

[19] Steven Bell, Alejandro Troccoli, and Kari Pulli. A non-linear filter for gyroscope-based video stabilization. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne

Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8692 of *Lecture Notes in Computer Science*, pages 294–308. Springer International Publishing, 2014.

[20] Moshe Ben-Ezra and Shree K. Nayar. Motion-based motion deblurring. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):689–698, June 2004.

[21] Xavier Benavides, Judith Amores, and Pattie Maes. Invisibilia: Revealing invisible data using augmented reality and Internet connected devices. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '15 Adjunct, pages 341–344, New York, NY, USA, 2015. ACM.

[22] Pierre J. Benckendorff, Pauline J. Sheldon, and Daniel R. Fesenmaier. *Tourism Information Technology, 2nd Edition*. CABI, 2014.

[23] Mark Billinghurst, Nigel Davies, Marc Langheinrich, and Albrecht Schmidt. Augmenting human memory - capture and recall in the era of lifelogging (Dagstuhl Seminar 14362). *Dagstuhl Reports*, 4(8):151–173, 2015.

[24] Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch. Touch Projector: Mobile interaction through video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296, New York, NY, USA, 2010. ACM.

[25] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, October 2001.

[26] Jian-Feng Cai, Hui Ji, Chaoqiang Liu, and Zuowei Shen. Blind motion deblurring using multiple images. *Computational Physics*, 228(14):5057–5071, August 2009.

[27] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. CyclopsRing: Enabling whole-hand and context-aware interactions through a fisheye ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pages 549–556, New York, NY, USA, 2015. ACM.

[28] Liwei Chan, Chi-Hao Hsieh, Yi-Ling Chen, Shuo Yang, Da-Yuan Huang, Rong-Hao Liang, and Bing-Yu Chen. Cyclops: Wearable and single-piece full-body gesture input devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3001–3009, New York, NY, USA, 2015. ACM.

[29] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. uTrack: 3D input using two magnetic sensors. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 237–244, New York, NY, USA, 2013. ACM.

[30] Ling Chen. A directed graphical model for linear barcode scanning from blurred images. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV '12, pages 524–535, Berlin, Heidelberg, 2013. Springer-Verlag.

[31] Xiaogang Chen, Xiangjian He, Jie Yang, and Qiang Wu. An effective document image deblurring algorithm. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 369–376, June 2011.

[32] Hojin Cho, Sunghyun Cho, Young Su Moon, Junguk Cho, Shihwa Lee, and Seungyong Lee. Analysis of the practical coverage of uniform motions to approximate real camera shakes. In *Proceedings of the SPIE, Computational Imaging*, volume 8296, 2012.

[33] Hojin Cho, Jue Wang, and Seungyong Lee. Text image deblurring using text-specific properties. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV '12, pages 524–537, Berlin, Heidelberg, 2012. Springer-Verlag.

[34] Sunghyun Cho, Hojin Cho, Yu-Wing Tai, and Seungyong Lee. Registration based non-uniform motion deblurring. *Computer Graphics Forum*, 31(7pt2):2183–2192, September 2012.

[35] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. *ACM Transactions on Graphics*, 28(5):145:1–145:8, December 2009.

[36] Sunghyun Cho, Yasuyuki Matsushita, and Seungyong Lee. Removing non-uniform motion blur from images. In *Proceedings of the 11th IEEE International Conference on Computer Vision*, ICCV '07, pages 1–8, Oct 2007.

[37] Taeg Sang Cho, Anat Levin, Fredo Durand, and William T. Freeman. Motion blur removal with orthogonal parabolic exposures. In *Proceedings of the 2010 IEEE International Conference on Computational Photography*, ICCP '10, pages 1–8, March 2010.

[38] Taeg Sang Cho, Sylvain Paris, Berthold K. P. Horn, and William T. Freeman. Blur kernel estimation using the Radon transform. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 241–248, Washington, DC, USA, 2011. IEEE Computer Society.

[39] Rustum Choksi and Yves van Gennip. Deblurring of one dimensional bar codes via total variation energy minimization. *SIAM Journal of Imaging Science*, 3(4):735–764, October 2010.

[40] Hung-Kuo Chu, Chia-Sheng Chang, Ruen-Rone Lee, and Niloy J. Mitra. Halftone QR Codes. *ACM Transactions on Graphics*, 32(6):217:1–217:8, November 2013.

[41] Gabe Cohn, Sidhant Gupta, Tien-Jui Lee, Dan Morris, Joshua R. Smith, Matthew S. Reynolds, Desney S. Tan, and Shwetak N. Patel. An ultra-low-power human body motion sensor using static electric field sensing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 99–102, New York, NY, USA, 2012. ACM.

[42] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. Humantenna: Using the body as an antenna for real-time whole-body interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1901–1910, New York, NY, USA, 2012. ACM.

[43] Andrea Colaço, Ahmed Kirmani, Hye Soo Yang, Nan-Wei Gong, Chris Schmandt, and Vivek K. Goyal. Mime: Compact, low power 3D gesture sensing for interaction with head mounted displays. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 227–236, New York, NY, USA, 2013. ACM.

[44] Enrico Costanza and Jeffrey Huang. Designable visual markers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1879–1888, New York, NY, USA, 2009. ACM.

[45] Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Publishing Company, 2013.

[46] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[47] Mauricio Delbracio and Guilermo Sapiro. Removing camera shake via weighted fourier burst accumulation. *IEEE Transactions on Image Processing*, 24(11):3293–3307, Nov 2015.

[48] Laura Dipietro, Angelo M. Sabatini, and Paolo Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, July 2008.

[49] Markéta Dubská, Adam Herout, and Jiří Havel. Real-time precise detection of regular grids and matrix codes. *Journal of Real-Time Image Processing*, 11(1):1–8, February 2013.

[50] Wolfgang Engel, editor. *GPU Pro, Advanced Rendering Techniques*. A K Peters, 2010.

[51] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, October 2007.

[52] Selim Esedoglu. Blind deconvolution of bar code signals. *Journal of Inverse Problems*, 20(1):121–135, 2004.

[53] Gabriele Fanelli, Matthias Dantone, and Luc Van Gool. Real time 3D face alignment with random forests-based active appearance models. In *Proceedings of the 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–8, April 2013.

[54] Gabriele Fanelli, Jürgen Gall, and Luc Van Gool. Real time head pose estimation with random regression forests. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 617–624, June 2011.

[55] Sean Ryan Fanello, Cem Keskin, Shahram Izadi, Pushmeet Kohli, David Kim, David Sweeney, Antonio Criminisi, Jamie Shotton, Sing Bing Kang, and Tim Paek. Learning to be a depth camera for close-range human capture and interaction. *ACM Transactions on Graphics*, 33(4):86:1–86:11, July 2014.

[56] Sean Ryan Fanello, Cem Keskin, Pushmeet Kohli, Shahram Izadi, Jamie Shotton, Antonio Criminisi, Ugo Pattacini, and Tim Paek. Filter forests for learning data-dependent convolutional kernels. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1709–1716, Washington, DC, USA, 2014. IEEE Computer Society.

[57] Hany Farid and Eero P. Simoncelli. Differentiation of discrete multidimensional signals. *IEEE Transactions on Image Processing*, 13(4):496–508, April 2004.

[58] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics*, 25(3):787–794, July 2006.

[59] Per-Erik Forssen and Erik Ringaby. Rectifying rolling shutter video from hand-held devices. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 507–514, June 2010.

[60] Markus Funk, Alireza Sahami Shirazi, Sven Mayer, Lars Lischke, and Albrecht Schmidt. Pick from here!: An interactive mobile cart using in-situ projection for order picking. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 601–609, New York, NY, USA, 2015. ACM.

[61] Jürgen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, Nov 2011.

[62] Orazio Gallo and Roberto Manduchi. Reading challenging barcodes with cameras. In *Proceedings of the 2009 Workshop on Applications of Computer Vision*, WACV '09, pages 1–6, Dec 2009.

[63] Orazio Gallo and Roberto Manduchi. Reading 1D barcodes with mobile phones using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1834–1843, 2011.

[64] Maribeth Gandy, Thad Starner, Jake Auxier, and Daniel Ashbrook. The Gesture Pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *Proceedings of the 4th IEEE International Symposium on Wearable Computers*, ISWC '00, pages 87–, Washington, DC, USA, 2000. IEEE Computer Society.

[65] Marian George, Dejan Mircic, Gábor Sörös, Christian Flörkemeier, and Friedemann Mattern. Fine-grained product class recognition for assisted shopping. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, ICCVW '15. IEEE, December 2015.

[66] Amit Goldstein and Raanan Fattal. Blur-kernel estimation from spectral irregularities. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, pages 622–635, Berlin, Heidelberg, 2012. Springer-Verlag.

[67] Anhong Guo, Shashank Raghu, Xuwen Xie, Saad Ismail, Xiaohui Luo, Joseph Simoneau, Scott Gilliland, Hannes Baumann, Caleb Southern, and Thad Starner. A comparison of order picking assisted by head-up display (HUD), cart-mounted display (CMD), light, and paper pick list. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, ISWC '14, pages 71–78, New York, NY, USA, 2014. ACM.

[68] Ankit Gupta, Neel Joshi, C. Lawrence Zitnick, Michael Cohen, and Brian Curless. Single image deblurring using motion density functions. In *Proceedings of the*

11th European Conference on Computer Vision: Part I, ECCV '10, pages 171–184, Berlin, Heidelberg, 2010. Springer-Verlag.

[69] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. SoundWave: Using the Doppler effect to sense gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1911–1914, New York, NY, USA, 2012. ACM.

[70] Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. Imaginary interfaces: Spatial interaction with empty hands and without visual feedback. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 3–12, New York, NY, USA, 2010. ACM.

[71] Sean Gustafson, Christian Holz, and Patrick Baudisch. Imaginary phone: Learning imaginary interfaces by transferring spatial memory from a familiar device. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 283–292, New York, NY, USA, 2011. ACM.

[72] Dong Han, Jianfu Teng, Zhaoxuan Yang, Yanwei Pang, and Meng Wang. 2D barcode image binarization based on wavelet analysis and Otsu's method. In *Proceedings of the International Conference on Computer Application and System Modeling*, volume 5 of *ICCASM'10*, 2010.

[73] Gustav Hanning, Nicklas Forslow, Per-Erik Forssen, Erik Ringaby, David Tornqvist, and Jonas Callmer. Stabilizing cell phone video using inertial measurement sensors. In *Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops*, ICCVW '11, pages 1–8, Nov 2011.

[74] Tian Hao, Ruogu Zhou, and Guoliang Xing. Cobra: Color barcode streaming for smartphone systems. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 85–98, New York, NY, USA, 2012. ACM.

[75] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, 1988.

[76] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. OmniTouch: Wearable multitouch interaction everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 441–450, New York, NY, USA, 2011. ACM.

[77] Chris Harrison and Scott E. Hudson. Abracadabra: Wireless, high-precision, and unpowered finger input for very small mobile devices. In *Proceedings of the 22nd*

Annual ACM Symposium on User Interface Software and Technology, UIST '09, pages 121–124, New York, NY, USA, 2009. ACM.

[78] Chris Harrison and Scott E. Hudson. Minput: Enabling interaction on small mobile devices with high-precision, low-cost, multipoint optical tracking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1661–1664, New York, NY, USA, 2010. ACM.

[79] Chris Harrison, Desney Tan, and Dan Morris. Skinput: Appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 453–462, New York, NY, USA, 2010. ACM.

[80] Frederik Hermans, Liam McNamara, Gábor Sörös, Christian Rohner, Thiemo Voigt, and Edith Ngai. Focus: Robust visual codes for everyone. In *Proceeding of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, June 2016.

[81] Frederik Hermans, Liam McNamara, and Thiemo Voigt. Demo: Scalable visual codes for embedding digital data in the physical world. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 457–458, 2015.

[82] Valentin Heun, James Hobin, and Pattie Maes. Reality editor: Programming smarter objects. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 307–310, New York, NY, USA, 2013. ACM.

[83] Valentin Heun, Shunichi Kasahara, and Pattie Maes. Smarter objects: Using AR technology to program physical objects and their interactions. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 961–966, New York, NY, USA, 2013. ACM.

[84] Masakazu Higuchi and Takashi Komuro. AR typing interface for mobile devices. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 14:1–14:8, New York, NY, USA, 2013. ACM.

[85] Lars Erik Holmquist, Hans-Werner Gellersen, Gerd Kortuem, Albrecht Schmidt, Martin Strohbach, Stavros Antifakos, Florian Michahelles, Bernt Schiele, Michael Beigl, and Ramia Mazé. Building intelligent environments with Smart-Its. *IEEE Computer Graphics and Applications*, 24(1):56–64, January 2004.

[86] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-Werner Gellersen. Smart-Its Friends: A technique for users to

easily establish connections between smart artefacts. In *Proceedings of the 3rd International Conference on Ubiquitous Computing*, UbiComp '01, pages 116–122, London, UK, UK, 2001. Springer-Verlag.

[87] Wenjun Hu, Jingshu Mao, Zihui Huang, Yiqing Xue, Junfeng She, Kaigui Bian, and Guobin Shen. Strata: Layered coding for scalable visual communication. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 79–90, New York, NY, USA, 2014. ACM.

[88] Yan Huang, Wei Wang, and Liang Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In *Advances in Neural Information Processing Systems 28*, NIPS '15, pages 235–243. Curran Associates, Inc., 2015.

[89] Toshiyuki Inoue. Pen-type bar code reader, July 21 1987. US Patent 4,682,016.

[90] Atsushi Ito, Aswin C. Sankaranarayanan, Ashok Veeraraghavan, and Richard G. Baraniuk. BlurBurst: Removing blur due to camera shake using multiple images. *ACM Transaction on Graphics*, 2014.

[91] Yuta Itoh and Gudrun Klinker. Vision enhancement: Defocus correction via optical see-through head-mounted displays. In *Proceedings of the 6th Augmented Human International Conference*, AH '15, pages 1–8, New York, NY, USA, 2015. ACM.

[92] Yuta Itoh, Jason Orlosky, Kiyoshi Kiyokawa, and Gudrun Klinker. Laplacian vision: Augmenting motion prediction via optical see-through head-mounted displays. In *Proceedings of the 7th Augmented Human International Conference 2016*, AH '16, pages 16:1–16:8, New York, NY, USA, 2016. ACM.

[93] Chao Jia and Brian L. Evans. Online calibration and synchronization of cellphone camera and gyroscope. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*, GlobalSIP '13, pages 731–734, Dec 2013.

[94] Chao Jia and Brian L. Evans. Online camera-gyroscope autocalibration for cell phones. *IEEE Transactions on Image Processing*, 23(12):5070–5081, Dec 2014.

[95] Jiaya Jia. Single image motion deblurring using transparency. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '07, pages 1–8, June 2007.

[96] Neel Joshi, Sing Bing Kang, C. Lawrence Zitnick, and Richard Szeliski. Image deblurring using inertial measurement sensors. *ACM Transactions on Graphics*, 29(4):30:1–30:9, July 2010.

[97] Neel Joshi, Richard Szeliski, and David Kriegman. PSF estimation using sharp edge prediction. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '08, pages 1–8, June 2008.

[98] George E. Chadima Jr. and Vadim Laser. Instant portable bar-code reader, April 26 1983. CA Patent 1,145,470.

[99] Jinki Jung, Jinwoo Jeon, Hyeopwoo Lee, Kichan Kwon, Jamal Zemerly, and Hyun Seung Yang. Augmented keyboard: A virtual keyboard interface for smart glasses. In *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '14, pages 159–164, New York, NY, USA, 2014. ACM.

[100] Praveen Kakumanu, Sokratis Makrogiannis, and Nikolaos G. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, March 2007.

[101] Alexandre Karpenko, David Jacobs, Jongmin Baek, and Marc Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical Report CTSR 2011-03, Department of Computer Science, Stanford University, September 2011.

[102] Stephan Karpischek. *Mobile barcode scanning applications for consumers*. PhD thesis, ETH Zurich, 2012.

[103] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth extraction from video using non-parametric sampling. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, pages 775–788, Berlin, Heidelberg, 2012. Springer-Verlag.

[104] Hiroko Kato, Keng T. Tan, and Douglas Chai. *Barcodes for Mobile Devices*. Cambridge University Press, 2010.

[105] Melinda Katona and László G. Nyúl. Efficient 1D and 2D barcode detection using mathematical morphology. In *Proceedings of the International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, ISMM '13, pages 464–475, 2013.

[106] Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota. Bringing gesture recognition to all devices. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 303–316, Berkeley, CA, USA, 2014. USENIX Association.

[107] Cem Keskin, Furkan Kiraç, Yunus E. Kara, and Lale Akarun. Randomized decision forests for static and dynamic hand shape classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '12, pages 31–36, June 2012.

[108] Cem Keskin, Furkan Kiraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 852–863, Berlin, Heidelberg, 2012. Springer-Verlag.

[109] Hamed Ketabdar, Amin Haji-Abolhassani, and Mehran Roshandel. MagiThings: Gestural interaction with mobile devices based on using embedded compass magnetic field sensor. *International Journal on Mobile Human Computer Interaction*, 5(3):23–41, July 2013.

[110] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: Freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 167–176, New York, NY, USA, 2012. ACM.

[111] David Kim, Shahram Izadi, Jakub Dostal, Christoph Rhemann, Cem Keskin, Christopher Zach, Jamie Shotton, Timothy Large, Steven Bathiche, Matthias Niessner, D. Alex Butler, Sean Fanello, and Vivek Pradeep. RetroDepth: 3D silhouette sensing for high-precision input on and above physical surfaces. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1377–1386, New York, NY, USA, 2014. ACM.

[112] Jongbae Kim and Heesung Jun. Vision-based location positioning using augmented reality for indoor navigation. *IEEE Transactions on Consumer Electronics*, 54(3):954–962, August 2008.

[113] Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. The gesture watch: A wireless contact-free gesture based wrist interface. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, ISWC '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[114] Tim Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 191–199, New York, NY, USA, 2002. ACM.

[115] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirhana Spasojevic. People, places, things: Web presence for the real world. In *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications*, pages 19–28, 2000.

[116] Mathias Kölsch. *Vision Based Hand Gesture Interfaces for Wearable Computing and Virtual Environments*. PhD thesis, University of California, Santa Barbara, 2004.

[117] Carl H. Knowles. Hands-free body mounted laser scanner and method of use, July 22 1993. WO Patent App. PCT/US1993/000,461.

[118] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VII*, ECCV'12, pages 27–40, Berlin, Heidelberg, 2012. Springer-Verlag.

[119] Hao Wang Kongqiao Wang, Yanming Zou. 1D barcode reading on camera phones. *International Journal of Image and Graphics*, 7(3):529–550, 2007.

[120] Sven Kratz and Michael Rohs. HoverFlow: Expanding the design space of around-device interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 4:1–4:8, New York, NY, USA, 2009. ACM.

[121] Bernard Kress and Thad Starner. A review of head-mounted displays (HMD) technologies and applications for consumer electronics. *Proceedings of SPIE*, 8720, 2013.

[122] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-Laplacian priors. In *Advances in Neural Information Processing Systems 22*, NIPS '09, pages 1033–1041. Curran Associates, Inc., 2009.

[123] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *Proceedings of the 2011 IEEE Conference onComputer Vision and Pattern Recognition*, CVPR '11, pages 233–240, June 2011.

[124] Aliasgar Kutiyanawala, Xiaojun Qi, and Jiandonsi Tian. A simple and efficient approach to barcode localization. In *Proceedings of the International Conference on Information, Communications and Signal Processing, 2009*, ICICS'09, pages 1–5, 2009.

[125] L'ubor Ladický, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 89–96, Washington, DC, USA, 2014. IEEE Computer Society.

[126] L'ubor Ladický, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Proceedings of the 13th European Conference on Computer Vision - Volume Part V*, ECCV '14, pages 468–484, Cham, September 2014. Springer International Publishing.

[127] Tobias Langlotz and Oliver Bimber. Unsynchronized 4D barcodes: Coding and decoding time-multiplexed 2D colorcodes. In *Proceedings of the 3rd International Conference on Advances in Visual Computing - Volume Part I*, ISVC'07, pages 363–374, Berlin, Heidelberg, 2007. Springer-Verlag.

[128] Douglas Lanman and Gabriel Taubin. Build your own 3D scanner: 3D photography for beginners. In *ACM SIGGRAPH 2009 Courses*, SIGGRAPH '09, pages 8:1–8:94, New York, NY, USA, 2009. ACM.

[129] Mathieu Le Goc, Stuart Taylor, Shahram Izadi, and Cem Keskin. A low-cost transparent electric field sensor for 3D interaction on mobile devices. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3167–3170, New York, NY, USA, 2014. ACM.

[130] Seungyon Claire Lee, Bohao Li, and Thad Starner. AirTouch: Synchronizing in-air hand gesture and on-body tactile feedback to augment mobile gesture interaction. In *Proceedings of the 15th Annual International Symposium on Wearable Computers*, ISWC '11, pages 3–10, June 2011.

[131] Taehee Lee and Tobias Hollerer. Handy AR: Markerless inspection of augmented reality objects using fingertip tracking. In *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, ISWC '07, pages 1–8, Washington, DC, USA, 2007. IEEE Computer Society.

[132] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3), July 2007.

[133] Anat Levin, Yair Weiss, Frédo Durand, and William T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '09, pages 1964–1971, June 2009.

[134] Cheng Li and Kris M. Kitani. Pixel-level hand detection in ego-centric videos. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3570–3577, June 2013.

[135] Tianxing Li, Chuankai An, Xinran Xiao, Andrew T. Campbell, and Xia Zhou. Real-time screen-camera communication behind any scene. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 197–211, New York, NY, USA, 2015. ACM.

[136] Yunpeng Li, Sing Bing Kang, Neel Joshi, Steve M. Seitz, and Daniel P. Huttenlocher. Generating sharp panoramas from motion-blurred videos. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '10, pages 2424–2431, June 2010.

[137] Ningzhong Liu, Xingming Zheng, Han Sun, and Xiaoyang Tan. Two-dimensional bar code out-of-focus deblurring via the increment constrained least squares filter. *Pattern Recognition Letters*, 34(2):124–130, 2013.

[138] Renting Liu and Jiaya Jia. Reducing boundary artifacts in image deconvolution. In *Proceedings of the 2008 IEEE International Conference on Image Processing*, ICIP 2008, pages 505–508, Oct 2008.

[139] Yifei Lou, Ernie Esser, Hongkai Zhao, and Jack Xin. Partially blind deblurring of barcode from out-of-focus blur. *SIAM Journal on Imaging Sciences*, 7(2):740–760, 2014.

[140] Ziyang Ma and Enhua Wu. Real-time and robust hand tracking with a single depth camera. *The Visual Computer*, 30(10):1133–1144, 2013.

[141] Friedemann Mattern. From smart devices to smart everyday objects. In *Proceedings of the Smart Objects Conference*, SOC '03, pages 15–16, 2003.

[142] Simon Mayer, Christian Beckel, Bram Scheidegger, Claude Barthels, and Gábor Sörös. Demo: Uncovering device whispers in smart homes. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, December 2012.

[143] Simon Mayer, Yassin N. Hassan, and Gábor Sörös. A magic lens for revealing device interactions in smart environments. In *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications*, SA '14, pages 9:1–9:6, New York, NY, USA, 2014. ACM.

[144] Simon Mayer, Markus Schalch, Marian George, and Gábor Sörös. Device recognition for intuitive interaction with the Web of Things. In *Proceedings of the 2013*

ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, UbiComp '13 Adjunct, pages 239–242, New York, NY, USA, 2013. ACM.

[145] Simon Mayer and Gábor Sörös. User interface beaming - seamless interaction with smart things using personal wearable computers. In *Proceedings of the 11th International Conference on Wearable and Implantable Body Sensor Networks, Workshop on Glass & Eyewear Computers*, BSN '14, pages 46–49, Jun 2014.

[146] Simon Mayer, Andreas Tschofen, Anind K. Dey, and Friedemann Mattern. User interfaces for smart things – a generative approach with semantic interaction descriptions. *ACM Transactions on Computer-Human Interaction*, 21(2):12:1–12:25, February 2014.

[147] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Proceedings of the 2001 IEEE International Conference on Computer Vision*, volume 1 of *ICCV*, pages 525–531 vol.1, 2001.

[148] Pranav Mistry and Pattie Maes. SixthSense: A wearable gestural interface. In *ACM SIGGRAPH Asia 2009 Sketches*, SA '09, pages 11:1–11:1, New York, NY, USA, 2009. ACM.

[149] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. Bokode: Imperceptible visual tags for camera based interaction from a distance. *ACM Transactions on Graphics*, 28(3):98:1–98:8, July 2009.

[150] Amin Motahari and Malek Adjouadi. Barcode modulation method for data transmission in mobile devices. *IEEE Transactions on Multimedia*, 17(1):118–127, Jan 2015.

[151] Adiyan Mujibiya and Jun Rekimoto. Mirage: Body motion and activity recognition using off-body static electric field sensing. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 65:1–65:2, New York, NY, USA, 2013. ACM.

[152] Hubert Nguyuen, editor. *GPU Gems 3*. Addison Wesley Professional, 2008.

[153] Takehiro Niikura, Yuki Hirobe, Alvaro Cassinelli, Yoshihiro Watanabe, Takashi Komuro, and Masatoshi Ishikawa. In-air typing interface for mobile devices with vibration feedback. In *ACM SIGGRAPH 2010 Emerging Technologies*, SIGGRAPH '10, pages 15:1–15:1, New York, NY, USA, 2010. ACM.

[154] Masa Ogata and Michita Imai. SkinWatch: Skin gesture interaction for smart watch. In *Proceedings of the 6th Augmented Human International Conference*, AH '15, pages 21–24, New York, NY, USA, 2015. ACM.

[155] Masa Ogata, Yuta Sugiura, Yasutoshi Makino, Masahiko Inami, and Michita Imai. SenSkin: Adapting skin as a soft interface. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 539–544, New York, NY, USA, 2013. ACM.

[156] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Tracking the articulated motion of two strongly interacting hands. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '12, pages 1862–1869, June 2012.

[157] Tatsuya Onoda and Kazuhiro Miwa. Clear two-dimensional code, article having clear two-dimensional code attached thereto, method for printing two-dimensional code and method for displaying two-dimensional code, March 5 2009. US Patent App. 11/887,747.

[158] Santiago Ortega-Avila, Bogdana Rakova, Sajid Sadi, and Pranav Mistry. Non-invasive optical detection of hand gestures. In *Proceedings of the 6th Augmented Human International Conference*, AH '15, pages 179–180, New York, NY, USA, 2015. ACM.

[159] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, Jan 1979.

[160] Jinshan Pan, Zhe Hu, Zhixun Su, and Ming-Hsuan Yang. Deblurring text images via L0-regularized intensity and gradient prior. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2901–2908, June 2014.

[161] Jinshan Pan, Risheng Liu, Zhixun Su, and Xianfeng Gu. Kernel estimation from salient structure for robust motion deblurring. *Signal Processing: Image Communication*, 28(9):1156–1170, October 2013.

[162] Junseong Park, Min Kim, Soonkeun Chang, and Kwae Hi Lee. Estimation of motion blur parameters using cepstrum analysis. In *Proceedings of the 2011 IEEE 15th International Symposium on Consumer Electronics*, ISCE '11, pages 406–409, June 2011.

[163] Sung Hee Park, Andrew Adams, and Eino-Ville Talvala. The FCam API for programmable cameras. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 707–710, New York, NY, USA, 2011. ACM.

[164] Sung Hee Park and Marc Levoy. Gyro-based multi-image deconvolution for removing handshake blur. In *Proceedings of the 2014 IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR '14, pages 3366–3373, Washington, DC, USA, 2014. IEEE Computer Society.

[165] Samuel David Perli, Nabeel Ahmed, and Dina Katabi. PixNet: Interference-free wireless links using LCD-camera pairs. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 137–148, New York, NY, USA, 2010. ACM.

[166] Simon T. Perrault, Eric Lecolinet, James Eagan, and Yves Guiard. Watchit: Simple gestures and eyes-free interaction for wristwatches and bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1451–1460, New York, NY, USA, 2013. ACM.

[167] Daniele Perrone and Paolo Favaro. Total variation blind deconvolution: The devil is in the details. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2909–2916, Washington, DC, USA, 2014. IEEE Computer Society.

[168] Henning Pohl and Michael Rohs. Around-device devices: My coffee mug is a volume dial. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '14, pages 81–90, New York, NY, USA, 2014. ACM.

[169] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*, MobiCom '13, pages 27–38, New York, NY, USA, 2013. ACM.

[170] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition HANDS Workshop*, CVPRW '14. IEEE, June 2014.

[171] A. N. Rajagopalan and Rama Chellappa. *Motion Deblurring: Algorithms and Systems*. Cambridge University Press, 2014.

[172] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: Motion deblurring using fluttered shutter. *ACM Transactions on Graphics*, 25(3):795–804, July 2006.

[173] Alex Rav-Acha and Shmuel Peleg. Two motion-blurred images are better than one. *Pattern Recognition Letters*, 26(3):311–317, February 2005.

[174] Rupert Reif, Willibald A. Günthner, Björn Schwerdtfeger, and Gudrun Klinker. Pick-by-vision comes on age: Evaluation of an augmented reality supported picking system in a real storage environment. In *Proceedings of the 6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '09, pages 23–31, New York, NY, USA, 2009. ACM.

[175] Felix von Reischach. *Product reviews via mobile phone – Prototyping, interaction, market*. PhD thesis, ETH Zurich, 2010.

[176] Jun Rekimoto. GestureWrist and GesturePad: Unobtrusive wearable interaction devices. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers*, ISWC '01, Washington, DC, USA, 2001. IEEE Computer Society.

[177] Jun Rekimoto and Yuji Ayatsuka. CyberCode: Designing augmented reality environments with visual tags. In *Proceedings of Designing Augmented Reality Environments*, DARE '00, pages 1–10, New York, NY, USA, 2000. ACM.

[178] William Hadley Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, Jan 1972.

[179] Erik Ringaby and Per-Erik Forssen. A virtual tripod for hand-held video stacking on smartphones. In *Proceedings of the 2014 IEEE International Conference on Computational Photography*, ICCP '14, pages 1–9, May 2014.

[180] Christof Roduner. The mobile phone as universal interaction device — are there limits? In *Proceedings of the 2006 Workshop on Mobile Interaction with the Real World*, MobileHCI '06, 2006.

[181] Christof Roduner. *Mobile devices for interacting with tagged objects*. PhD thesis, ETH Zurich, 2010.

[182] Michael Rohs. *Linking physical and virtual worlds with visual markers and handheld devices*. PhD thesis, ETH Zurich, 2005.

[183] Michael Rohs. Visual code widgets for marker-based interaction. In *Proceedings of the International Conference on Distributed Computing Systems Workshops*, volume 5 of *ICDCSW '05*, pages 506–513, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[184] Wilson M. Routt and Donald L. West. Coaxial optical scanner, August 24 1982. US Patent 4,346,292.

[185] Enrico Rukzio, Gregor Broll, Karin Leichtenstern, and Albrecht Schmidt. Mobile interaction with the real world: An evaluation and comparison of physical mobile

interaction techniques. In Bernt Schiele, Anind K. Dey, Hans Gellersen, Boris de Ruyter, Manfred Tscheligi, Reiner Wichert, Emile Aarts, and Alejandro Buchmann, editors, *Ambient Intelligence*, volume 4794 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2007.

[186] Medford D. Sanner. Ambient illumination bar code reader, October 17 1989. US Patent 4,874,933.

[187] T. Scott Saponas, Chris Harrison, and Hrvoje Benko. PocketTouch: Through-fabric capacitive touch input. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 303–308, New York, NY, USA, 2011. ACM.

[188] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 167–176, New York, NY, USA, 2009. ACM.

[189] Philipp M. Scholl, Matthias Wille, and Kristof Van Laerhoven. Wearables in the wet lab: A laboratory system for capturing and guiding experiments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 589–599, New York, NY, USA, 2015. ACM.

[190] Christian J. Schuler, Michael Hirsch, Stefan Harmeling, and Bernhard Schölkopf. Learning to deblur. In *Advances in Neural Information Processing Systems, Deep Learning and Representation Learning Workshop*, NIPS '14, 2014.

[191] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 3791–3799, June 2015.

[192] Byung-Kuk Seo, Junyeoung Choi, Jae-Hyek Han, Hanhoon Park, and Jong-Il Park. One-handed interaction with augmented virtual objects on mobile devices. In *Proceedings of the 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '08, pages 8:1–8:6, New York, NY, USA, 2008. ACM.

[193] Hossein Shafagh and Anwar Hithnawi. Poster: Come closer: Proximity-based authentication for the internet of things. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 421–424, New York, NY, USA, 2014. ACM.

[194] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Transactions on Graphics*, 27(3):73:1–73:10, August 2008.

[195] Qi Shan, Zhaorong Li, Jiaya Jia, and Chi-Keung Tang. Fast image/video upsampling. *ACM Transactions on Graphics*, 27(5):153:1–153:7, December 2008.

[196] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3633–3642, New York, NY, USA, 2015. ACM.

[197] Eli Shechtman, Yaron Caspi, and Michal Irani. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–545, April 2005.

[198] Shuyu Shi, Lin Chen, Wenjun Hu, and Marco Gruteser. Reading between lines: High-rate, non-intrusive visual codes within regular videos via ImplicitCode. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 157–168. ACM, 2015.

[199] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *ACM Communications*, 56(1):116–124, January 2013.

[200] Ondrej Sindelar and Filip Sroubek. Image deblurring in smartphone devices using built-in inertial measurement sensors. *Journal of Electronic Imaging*, 22(1), 2013.

[201] Ondrej Sindelar, Filip Sroubek, and Peyman Milanfar. A smartphone application for removing handshake blur and compensating rolling shutter. In *Proceedings of the 2014 IEEE International Conference on Image Processing*, ICIP '14, pages 2160–2162, Oct 2014.

[202] Ondrej Sindelar, Filip Sroubek, and Peyman Milanfar. Space-variant image deblurring on smartphones using inertial sensors. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 191–192, June 2014.

[203] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, and Otmar Hilliges. Joint estimation of 3D hand position and gestures from monocular video for mobile interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors*

in Computing Systems, CHI '15, pages 3657–3660, New York, NY, USA, 2015. ACM.

[204] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. In-air gestures around unmodified mobile devices. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 319–329, New York, NY, USA, 2014. ACM.

[205] Jie Song, Gábor Sörös, Fabrizio Pece, and Otmar Hilliges. Extended abstract: Real-time hand gesture recognition on unmodified wearable devices. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition HANDS Workshop*, CVPR '15. IEEE, June 2015.

[206] Gábor Sörös. GPU-accelerated joint 1D and 2D barcode localization on smartphones. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '14, pages 5095–5099, May 2014.

[207] Gábor Sörös, Florian Daiber, and Tomer Weller. Cyclo: A personal bike coach through the Glass. In *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*, SA '13, pages 99:1–99:4, New York, NY, USA, 2013. ACM.

[208] Gábor Sörös and Christian Flörkemeier. Towards next generation barcode scanning. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, pages 47:1–47:2, New York, NY, USA, 2012. ACM.

[209] Gábor Sörös and Christian Flörkemeier. Blur-resistant joint 1D and 2D barcode localization for smartphones. In *Proceedings of the 12th ACM International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 11:1–11:8, New York, NY, USA, 2013. ACM.

[210] Gábor Sörös, Severin Münger, Carlo Beltrame, and Luc Humair. Multiframe visual-inertial blur estimation and removal for unmodified smartphones. In *Proceedings of the 23rd International Conference on Computer Graphics, Visualization and Computer Vision*, WSCG '15, June 2015.

[211] Gábor Sörös, Hartmut Seichter, Peter Rautek, and Eduard Gröller. Augmented visualization with natural feature tracking. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, MUM '11, pages 4–12, New York, NY, USA, 2011. ACM.

[212] Gábor Sörös, Stephan Semmler, Luc Humair, and Otmar Hilliges. Fast blur removal for wearable QR code scanners. In *Proceedings of the 2015 ACM International*

Symposium on Wearable Computers, ISWC '15, pages 117–124, New York, NY, USA, 2015. ACM.

[213] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 3213–3221, June 2015.

[214] Thad Starner. Project Glass: An extension of the self. *IEEE Pervasive Computing*, 12(2):14–16, April 2013.

[215] Thad Starner, Steve Mann, Bradley Rhodes, Jeffrey Levine, Jennifer Healey, Dana Kirsch, Rosalind W. Picard, and Alex Pentland. Augmented reality through wearable computing. *Presence, Special Issue on Augmented Reality*, 7(4), 1997.

[216] Thad Starner, Alex Pentland, and Joshua Weaver. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, December 1998.

[217] Shuochen Su and Wolfgang Heidrich. Rolling shutter motion deblurring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1537, June 2015.

[218] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 769–777, June 2015.

[219] Libin Sun, Sunghyun Cho, Jue Wang, and James Hays. Edge-based blur kernel estimation using patch priors. In *Proceedings of the 2013 IEEE International Conference on Computational Photography*, pages 1–8, April 2013.

[220] Lars Sverker, Ture Lindbo, Paul Clarke, and Matthew D. Engle. Head-mounted code scanner, December 11 2014. WO Patent App. PCT/IB2014/061,897.

[221] Jerome Swartz, Barkan Edward, and Shelley A. Harrison. Portable laser scanning system and scanning methods, June 7 1983. US Patent 4,387,297.

[222] István Szentandrási, Adam Herout, and Markéta Dubská. Fast detection and recognition of QR codes in high-resolution images. In *Proceedings of the 28th Spring Conference on Computer Graphics*, SCCG '12, pages 1–8. Comenius University in Bratislava, 2012.

[223] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-ICP for real-time hand tracking. In *Proceedings of the Eurographics Symposium on Geometry Processing*, SGP '15, pages 101–114, Aire-la-Ville, Switzerland, Switzerland, 2015. Eurographics Association.

[224] Yu-Wing Tai, Xiaogang Chen, Sunyeong Kim, Seon Joo Kim, Feng Li, Jie Yang, Jingyi Yu, Yasuyuki Matsushita, and Michael S. Brown. Nonlinear camera response functions and image deblurring: Theoretical analysis and practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2498–2512, Oct 2013.

[225] Yu-Wing Tai, Hao Du, Michael S. Brown, and Stephen Lin. Correction of spatially varying image and video motion blur using a hybrid camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1012–1028, June 2010.

[226] Yu-Wing Tai, Ping Tan, and Michael S. Brown. Richardson-Lucy deblurring for scenes under a projective motion path. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1603–1618, Aug 2011.

[227] Hiroyuki Takeda and Peyman Milanfar. Removing motion blur with space-time processing. *IEEE Transactions on Image Processing*, 20(10):2990–3000, Oct 2011.

[228] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 3224–3231, Washington, DC, USA, 2013. IEEE Computer Society.

[229] Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. Type-hover-swipe in 96 bytes: A motion sensing mechanical keyboard. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1695–1704, New York, NY, USA, 2014. ACM.

[230] Ender Tekin and James M. Coughlan. BLaDE: Barcode localization and decoding engine. Technical Report 2012-RERC.01, The Smith-Kettlewell Eye Research Institute, December 2012.

[231] Koji Tsukada and Michiaki Yasumura. Ubi-finger: a simple gesture input device for mobile and ubiquitous environment. *Journal of Asian Information, Science and Life(AISL)*, 2(2):111–120, 2004.

[232] Yves van Gennip, Prashant Athavale, Jérôme Gilles, and Rustum Choksi. A regularization approach to blind deblurring and denoising of QR barcodes. *arXiv:1410.6333*, 2014.

[233] Mélodie Vidal, Andreas Bulling, and Hans Gellersen. Pursuits: Spontaneous eye-based interaction for dynamic interfaces. *GetMobile: Mobile Computing and Communications*, 18(4):8–10, October 2014.

[234] Steffen Wachenfeld, Sebastian Terlunen, and Xiaoyi Jiang. Robust 1D barcode recognition on camera phones and mobile product information display. In Xiaoyi Jiang, Matthew Y. Ma, and Chang Wen Chen, editors, *Mobile Multimedia Processing*, pages 53–69. Springer-Verlag, Berlin, Heidelberg, 2010.

[235] Daniel Wagner. *Handheld Augmented Reality*. PhD thesis, Graz University of Technology, 2007.

[236] Daniel Wagner and Dieter Schmalstieg. Making augmented reality practical on mobile phones, part 1. *IEEE Computer Graphics and Applications*, 29(3):12–15, May 2009.

[237] Daniel Wagner and Dieter Schmalstieg. Making augmented reality practical on mobile phones, part 2. *IEEE Computer Graphics and Applications*, 29(4):6–9, July 2009.

[238] Anran Wang, Chunyi Peng, Ouyang Zhang, Guobin Shen, and Bing Zeng. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, HotNets-XIII, pages 23:1–23:7, New York, NY, USA, 2014. ACM.

[239] Meng Wang, Li-Na Li, and Zhao-Xuan Yang. Gabor filtering-based scale and rotation invariance feature for 2D barcode region detection. In *Proceedings of the International Conference on Computer Application and System Modeling*, volume 5 of *ICCASM'10*, pages 34–37, 2010.

[240] Qian Wang, Man Zhou, Kui Ren, Tao Lei, Jikun Li, and Zhibo Wang. RainBar: Robust application-driven visual communication using color barcodes. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems*, ICDCS '15, June 2015.

[241] Sen Wang, Tingbo Hou, John Border, Hong Qin, and Rodney Miller. High-quality image deblurring with panchromatic pixels. *ACM Transactions on Graphics*, 31(5):120:1–120:11, September 2012.

[242] Kimberly A. Weaver, Hannes Baumann, Thad Starner, Hendrick Iben, and Michael Lawo. An empirical task analysis of warehouse order picking using head-mounted displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1695–1704, New York, NY, USA, 2010. ACM.

[243] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing and Communications Review*, 3(3):3–11, July 1999.

[244] Motorala Inc. Whitepaper. The next-generation warehouse – long range scanning and emergence of 2D bar codes. Technical report, Motorola Inc., 2011.

[245] Oliver Whyte. *Removing camera shake blur and unwanted occluders from photographs*. PhD thesis, Ecole Normale Superieure de Cachan, 2012.

[246] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. *International Journal of Computer Vision*, 98(2):168–186, June 2012.

[247] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. MIT paperback series. Technology Press of the Massachusetts Institute of Technology, 1964.

[248] Grace Woo, Andy Lippman, and Ramesh Raskar. VRCodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *Proceedings of the 2012 IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '12, pages 59–64, Washington, DC, USA, 2012. IEEE Computer Society.

[249] Norman J. Woodland and Silver Bernard. Classifying apparatus and method, October 7 1952. US Patent 2,612,994.

[250] Xiaolong Wu, Malcolm Haynes, Yixin Zhang, Ziyi Jiang, Zhengyang Shen, Anhong Guo, Thad Starner, and Scott Gilliland. Comparing order picking assisted by head-up display versus pick-by-light with explicit pick confirmation. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ISWC '15, pages 133–136, New York, NY, USA, 2015. ACM.

[251] Ying Xiong, Kate Saenko, Trevor Darrell, and Todd Zickler. From pixels to physics: Probabilistic color de-rendering. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '12, pages 358–365, June 2012.

[252] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 157–170, Berlin, Heidelberg, 2010. Springer-Verlag.

[253] Li Xu, Jimmy SJ. Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems*, NIPS '14, pages 1790–1798. Curran Associates, Inc., 2014.

[254] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L0 sparse representation for natural image deblurring. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1107–1114, Washington, DC, USA, 2013. IEEE Computer Society.

[255] Wei Xu and Scott McCloskey. 2D barcode localization and motion deblurring using a flutter shutter camera. In *Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision*, WACV '11, pages 159–165, Washington, DC, USA, 2011. IEEE Computer Society.

[256] Wei Xu and Scott McCloskey. 2D barcode localization and motion deblurring using a flutter shutter camera. In *Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision*, WACV '11, pages 159–165, Jan 2011.

[257] Saeed Yahyanejad and Jacob Ström. Removing motion blur from barcode images. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '10, pages 41–46, 2010.

[258] Xing-Dong Yang, Khalad Hasan, Neil Bruce, and Pourang Irani. Surround-see: Enabling peripheral vision on smartphones during active use. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 291–300, New York, NY, USA, 2013. ACM.

[259] Yitzhak Yitzhaky, I. Mor, A. Lantzman, and Natan S. Kopeika. A direct method for restoration of motion blurred images. *Journal of the Optical Society of America*, 15(6), June 1998.

[260] Lu Yuan, Jian Sun, Long Quan, and Heung-Yeung Shum. Image deblurring with blurred/noisy image pairs. *ACM Transactions on Graphics*, 26(3), July 2007.

[261] Ben Zhang, Yu-Hsiang Chen, Claire Tuna, Achal Dave, Yang Li, Edward Lee, and Björn Hartmann. HOBS: Head orientation-based selection in physical spaces. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction*, SUI '14, pages 17–25, New York, NY, USA, 2014. ACM.

[262] Haichao Zhang and Lawrence Carin. Multi-shot imaging: Joint alignment, deblurring, and resolution-enhancement. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2925–2932, Washington, DC, USA, 2014. IEEE Computer Society.

[263] Haichao Zhang, David Wipf, and Yanning Zhang. Multi-image blind deblurring using a coupled adaptive sparse prior. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1051–1058, June 2013.

[264] Haichao Zhang and Jianchao Yang. Intra-frame deblurring by leveraging inter-frame camera motion. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '15, pages 4036–4044, June 2015.

[265] Qi Zhang, Li Xu, and Jiaya Jia. 100+ times faster weighted median filter (WMF). In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 2830–2837, June 2014.

[266] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, August 1999.

Image sources and web references

All links were accessed on 1st April 2016 and were correct at the time of publication.

[267] Abby optical character recognition SDK. http://www.abbyy.com/mobile-ocr/.

[268] Adobe Photoshop CC Shake Reduction. https://helpx.adobe.com/photoshop/how-to/camera-shake-reduction-photoshop.html.

[269] Aipoly. http://aipoly.com/.

[270] Amazon Firefly. https://developer.amazon.com/public/solutions/devices/fire-phone/docs/understanding-firefly.

[271] Android microphone icon. http://www.lowcountryren.com/nestor/Documentation/ionicons-1.4.1/png/512/android-microphone.png.

[272] Barcode Generator. http://www.barcode-generator.org/.

[273] Barcode history. http://www.barcoding.com/information/barcode_history.shtml.

[274] Barcode on packaging lane. http://cfnewsads.thomasnet.com/images/cmsimage/image/barcode-reading.jpg.

[275] Brückner Callisto head-mounted display. http://www.brueckner.com/en/brueckner-servtec/services/remote-services/remote-service-tools/.

[276] CamFind. http://camfindapp.com/.

[277] Codecheck.info. http://www.codecheck.info.

[278] Creative Senz3D depth camera. http://en.europe.creative.com/p/archived-products/creative-senz3d.

[279] CrontoSign. http://www.cronto.com/.

[280] CUDA – Compute Unified Device Architecture. http://www.nvidia.com/object/cuda_home_new.html.

[281] Dacuda document scanning SDK. http://www.dacuda.com.

[282] Epson Moverio BT-200 smartglasses. http://www.epson.com/cgi-bin/Store/jsp/Landing/moverio-augmented-reality-smart-glasses.do.

[283] ETH Zurich In-Air Gestures project. http://ait.inf.ethz.ch/projects/2014/InAirGesture/.

[284] Fashwell. http://www.fashwell.com/.

[285] FFTW. http://www.fftw.org/.

[286] Fitbit sports accessories. https://www.fitbit.com.

[287] Gibraltar targets tourists with Wikipedia QR codes. http://www.bbc.com/news/technology-19544299.

[288] Google Glass developer page. https://developers.google.com/glass/.

[289] Google Glass (on Wikipedia). https://en.wikipedia.org/wiki/Google_Glass.

[290] Google Glass (patent). https://www.google.com/patents/USD659739.

[291] Google Goggles. https://en.wikipedia.org/wiki/Google_Goggles.

[292] Google Soli. https://www.google.com/atap/project-soli/.

[293] GS1. http://www.gs1.org/.

[294] Honeywell 8650 scanner ring. https://www.honeywellaidc.com/CatalogDocuments/8650BT_DS_EN.pdf.

[295] Honeywell N5600 series miniature area imaging engines specifications. https://www.honeywellaidc.com/CatalogDocuments/N5600_DS_RevD_1211_EN.pdf.

[296] Human readable quick response code. http://hrqr.org/.

[297] i.d. mate Quest wearable barcode scanner. http://www.envisionamerica.com/products/idmate/.

[298] Intermec CN70 handheld wireless computer. http://www.intermec.com/products/cmptrcn70a/index.aspx.

[299] Intermec SG20 handheld wireless scanner. http://www.intermec.com/public-files/brochures/en/Retail_brochure_web.pdf.

[300] iTiZZiMO smart business platform. http://www.itizzimo.com/.

[301] kooaba. http://www.kooaba.com/.

[302] Koubachi plant sensors. http://www.koubachi.com/.

[303] LIFX Web-enabled light bulb. http://www.lifx.com/.

[304] Logitech Z320 speaker system. https://secure.logitech.com/assets/15189/15189.png.

[305] Lytro lightfield camera. https://www.lytro.com/.

[306] Microsoft Bing Vision. https://en.wikipedia.org/wiki/Bing_Vision.

[307] Microsoft HoloLens head-mounted computer. https://www.microsoft.com/microsoft-hololens/.

[308] Monmouthpedia. https://en.wikipedia.org/wiki/Monmouth.

[309] Moodstocks. https://moodstocks.com/.

[310] Museum of the Risorgimento (Rome). http://www.nfctech.eu/media/images/mus-ris-vitt-em-qr.jpg.

[311] Nest learning thermostat. https://nest.com/.

[312] OpenCL – Open Computing Language. https://www.khronos.org/opencl/.

[313] OpenCV - Open Computer Vision. http://opencv.org/.

[314] OpenGL ES – Open Graphics Library for Embedded Systems. https://www.khronos.org/opengles/.

[315] OpenKCam – Advanced camera and sensor control. https://www.khronos.org/openkcam.

[316] OpenVX – Vision processing. https://www.khronos.org/openvx/.

[317] Osterhout R-7 smartglasses. http://www.osterhoutgroup.com/products-r7-glasses.

[318] Peel Smart Remote. https://www.peel.com/.

[319] Picavi Pick-by-Vision. http://www.picavi.com/.

[320] Price of RFID tags. https://www.rfidjournal.com/faq/show?85.

[321] Pristine Eyesight eye swapping service. https://pristine.io/.

[322] Pupil Labs eye tracker camera. https://pupil-labs.com/.

[323] QR Artist. http://qrartist.net/.

[324] QR in the supermarket. http://cdn.trendhunterstatic.com/phpthumbnails/130/130337/130337_1_468.jpeg.

[325] QR on train ticket. http://media0.faz.net/ppmedia/aktuell/wirtschaft/1246532942/1.3491292/article_multimedia_overview/fahrkartenkontrolle-im-zug.jpg.

[326] Recon Jet smartglasses. http://www.reconinstruments.com/products/jet/.

[327] RedLaser. http://www.redlaser.com.

[328] RenderScript – Parallel computing API by Google. http://developer.android.com/guide/topics/renderscript/index.html.

[329] Samsung Galaxy Gear smartwatch. http://www.samsung.com/uk/consumer-images/product/galaxy-gear/2014/SM-V7000ZKABTU/SM-V7000ZKABTU-79431-0.jpg.

[330] Samsung Galaxy S4 smartphone (on Wikipedia). https://upload.wikimedia.org/wikipedia/commons/thumb/8/86/Samsung_Galaxy_S4.svg/225px-Samsung_Galaxy_S4.svg.png.

[331] Scandit. http://www.scandit.com.

[332] ScanTrust. https://www.scantrust.com.

[333] Slyce. http://slyce.it/.

[334] SMark. http://smark.eu/.

[335] Smartphone-controlled Nespresso coffee machine (on wired.com). http://www.wired.com/2016/03/nespresso-prodigio/.

[336] Sonos Trueplay smartphone-based loudspeaker calibration (on wired.com). http://www.wired.com/2015/09/sonos-trueplay-play5/.

[337] StreamInput – Unified sensors API. https://www.khronos.org/streaminput/.

[338] Symbol DS6878-SR general-purpose cordless 2D imager. https://www.zebra.com/gb/en/products/scanners/general-purpose-scanners/handheld/ds6878-sr.html.

[339] The World Park. http://www.prnewswire.com/news-releases/the-world-park-reinvents-the-new-york-central-park-experience-92197279.html.

[340] Ubimax Industrial Wearable Computing Solutions. http://www.ubimax.de/.

[341] Various smartglasses. http://www.droiders.com/wp-content/uploads/12-glasses-b.png.

[342] Various smartphones. http://www.emarketingrobots.com/wp-content/uploads/2014/08/Smartphones.png.

[343] Various smartwatches. https://www.rudebaguette.com/assets/smart-watches.jpg.

[344] Volkswagen Touareg AR. http://eandt.theiet.org/magazine/2013/04/images/640_touareg-hybrid.jpg.

[345] Vulkan – Graphics and Compute API. https://www.khronos.org/vulkan/.

[346] Vuzix M100 smartglasses. https://www.vuzix.com/Products/m100-smart-glasses.

[347] Wahoo sports accessories. http://eu.wahoofitness.com/.

[348] Wikipedia: HSV color system. https://en.wikipedia.org/wiki/HSL_and_HSV.

[349] ZBar. http://www.github.com/ZBar.

[350] Zebra Technologies CS1504 keychain scanner. https://www.zebra.com/us/en/products/scanners/companion-scanners/cs1504.html.

[351] ZXing. http://code.google.com/p/zxing/.

Appendix A

The energy function of blind deconvolution

A.1 MAP formulation

The uniform blur model is formulated as

$$b = k * l + n$$

where the blurred image b is a result of convolving a sharp image l with a blur kernel k and adding Gaussian noise n. In blind deconvolution, we know only the blurred image b and we try to recover the latent sharp image l. This also requires estimating the blur kernel k. The problem of finding both l and k can be formulated as minimizing the following energy function:

$$\underset{l,k}{\operatorname{argmin}} ||b - k * l||_2^2 + \lambda \rho_l(l) + \gamma \rho_k(k)$$

To derive this energy function, we first express the noise on a single pixel. In the equations, the index i runs over all image pixels from 1 to N.

$$n_i = b_i - (k * l)_i$$

The probability distribution of a single pixel's additive noise is assumed to be Gaussian with zero mean and standard deviation σ:

$$p(n_i) \sim N(0, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2} n_i^2) \propto e^{-\frac{1}{2\sigma^2} n_i^2}$$

Assuming that the noise on each pixel is independent and identically distributed (i.i.d.), the probability of the image noise is the product of the pixel noise probabilities:

$$p(n) \propto \prod_{i=1}^{N} e^{-\frac{1}{2\sigma^2} n_i^2} = e^{-\frac{1}{2\sigma^2} \sum_{i=1}^{N} n_i^2}$$

Now let us look at the probabilities of l and k. As Bayes' rule says, the posterior probability is proportional to the product of the likelihood and the prior:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \propto p(y|x)p(x)$$

The term $p(y)$ is a normalization factor which does not play a role in the further minimization, as it will reduce to an additive constant after taking the logarithm. Introducing independent variables l and k hence results in

$$p(l,k|b) \propto p(b|l,k)p(l,k) = p(b|l,k)p(l)p(k)$$

The maximum a posteriori (MAP) estimates of the unknowns k and l are

$$\begin{aligned} \underset{l,k}{\operatorname{argmax}}\, p(l,k|b) &= \underset{l,k}{\operatorname{argmin}} \left[-\log\ p(l,k|b)\right] \\ &= \underset{l,k}{\operatorname{argmin}} \left[-\log\ p(b|l,k) - \log\ p(l) - \log\ p(k)\right] \end{aligned}$$

as maximizing the posterior probability is equivalent to minimizing its negative logarithm.

A.2 Likelihood term

The likelihood term follows from our blur model by expressing the noise term:

$$\begin{aligned} p(b|l,k) = p(n) &\propto e^{-\frac{1}{2\sigma^2} \sum_{i=1}^{N} n_i^2} \\ -\log\ p(b|l,k) &\propto C \sum_{i=1}^{N} [n_i]^2 = C \sum_{i=1}^{N} [b_i - (k*l)_i]^2 \propto ||b - k*l||_2^2 \end{aligned}$$

with C containing all the constant terms and i indexing the pixels in the image.

A.3 Image priors

While the pixel values can be very different across images, the (log-)distribution of the image derivatives follows a common pattern (see Figure A.1, left) in photographs. This property has been successfully exploited in the solutions of various image processing problems. This distribution is independent of the image scale and has a heavy tail which means while most gradients are around zero (flat image areas), some large gradients are also likely (edges). In particular, the distribution is *not* Gaussian. This is very unfortunate because a Gaussian prior would make the minimization problem very simple with a closed-form solution. Images restored with a Gaussian prior are often oversmoothed and/or contain ringing artifacts. Instead, the distribution is usually modeled by a Hyper-Laplacian function (see Figure A.1, right) with an exponent $\alpha < 1$, best values are $\alpha \in [0.5, 0.8]$.

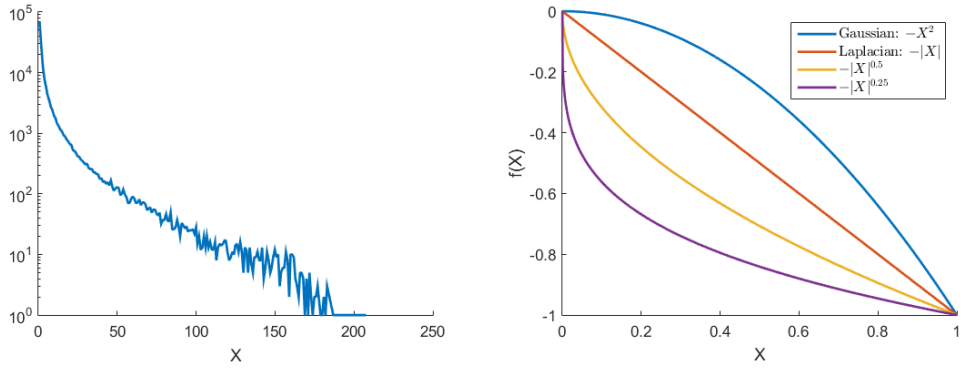


Figure A.1: Left: A natural image prior: typical log-gradient histogram of an image. The shape of this curve can be approximated by various parametric models. Right: Parametric models as natural image priors.

Let us denote the x- and y-derivatives of the image l at pixel i as $\partial_x l_i$ and $\partial_y l_i$, respectively. The prior $p(l)$ can then be formulated as

$$p(l) \propto e^{-\frac{1}{2\eta^2}\sum_i^N |\partial_x l_i|^\alpha + |\partial_y l_i|^\alpha}$$

In general form:

$$p(l) = \prod_{i=1}^{L} \Phi(\nabla l_i) = \prod_{i=1}^{L} e^{-\phi(\nabla l_i)}$$

and after taking the logarithm:

$$-\log\ p(l) = \sum_{i=1}^{L} \phi(\nabla l_i)$$

The two most commonly used curves are the **Gaussian prior**:

$$-\log\ p(l) = \sum_{i=1}^{L} |\nabla l_i|^2 = ||\nabla l||_2^2$$

and the **hyper-Laplacian prior**:

$$-\log\ p(l) = \sum_{i=1}^{L} |\nabla l_i|^{\alpha}$$

Other parametric curves are also used in the literature [58, 194].

In newest blind deconvolution algorithms [254], L_0 regularization terms were proposed as image prior. The L_0 norm is a quasi norm and stands for the number of non-zero elements. As blur smooths out sharp edges and distributes a single large gradient to multiple smaller gradients, a blurry image has much higher L_0 norm than its sharp counterpart. When applied on the gradients, the prior

$$-\log\ p(l) = ||\nabla l_i||_0$$

enforces the sharp image to have many flat regions and sharp edges inbetween. Furthermore, in document deblurring it is also beneficial to apply a similar L_0 prior also on the pixel intensities, because the number of zeros (black text pixels) is significantly different in a sharp document than in a gray blurred one [160]. This leads to the document deblurring prior

$$-\log\ p(l) = ||l_i||_0 + ||\nabla l_i||_0$$

However, the L_0 regularization term is hard to tackle computationally, therefore various approximations are used instead.

In our QR restoration algorithm, we apply the Laplacian prior ($\alpha = 1$) because it matches well the black and white code images and fast solution methods exists for minimizing the energy function:

$$\rho_l(l) = \sum_{i=1}^{L} |\nabla l_i|$$

A.4 Kernel priors

In the simplest case, the kernel prior $p(k)$ is assumed **uniform** so it is ignored. Earlier methods applied a **sum-of-exponentials** prior: For a single pixel k_i of the kernel k the distribution is a sum of D exponential distributions:

$$p(k_i) \sim \sum_{d=1}^{D} w_d \varepsilon_{\lambda_d}(k) = w_1 \lambda_1 e^{-\lambda_1 k} + w_2 \lambda_2 e^{-\lambda_2 k} + w_3 \lambda_3 e^{-\lambda_3 k} + \dots$$

For the whole kernel then:

$$p(k) \sim \prod_{k=1}^{K} \sum_{d=1}^{D} w_d \varepsilon_{\lambda_d}(k)$$

Its negative logarithm is:

$$-\log\ p(k) = -\sum_{k=1}^{K} \log \sum_{d=1}^{D} w_d \varepsilon_{\lambda_d}(k) =$$

$$= -\sum_{k=1}^{K} \log[w_1 \lambda_1 e^{-\lambda_1 k} + w_2 \lambda_2 e^{-\lambda_2 k} + w_3 \lambda_3 e^{-\lambda_3 k} + \dots]$$

Alternatively, a **Gaussian gradient prior** can be applied, this prior enforces connectedness.

$$p(k) = \prod_{k=1}^{K} e^{-c(\Delta k)^2}$$

$$-\log\ p(k) = C \sum_{i=1}^{K} (\Delta k)^2 = ||\Delta k||_2^2$$

In most methods and also in our method, a **Gaussian intensity prior** is applied on the kernel, this prior enforces small values and avoids Dirac kernels.

$$p(k) = \prod_{i=1}^{K} e^{-k_i^2} = e^{-\sum_{i=1}^{K} k_i^2}$$

$$-\log\ p(k) = \sum_{i=1}^{K} k_i^2 = ||k||_2^2$$

$$\rho_k(k) = ||k||_2^2$$

Also, $||k||_1 = \sum_{i=1}^{K} k_i = 1$ is always necessary so that the blurring does not change the overall image intensity.

Appendix B

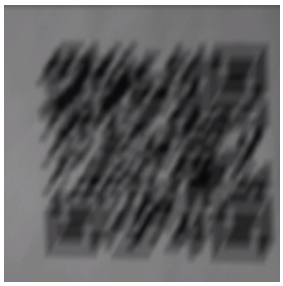
Video results

Large part of the research described in this thesis involves live camera stream processing hence the results can be better presented in videos. This appendix describes the videos attached to this document that can be downloaded from the online library of ETH Zurich: http://doi.org/10.5905/ethz-1007-50

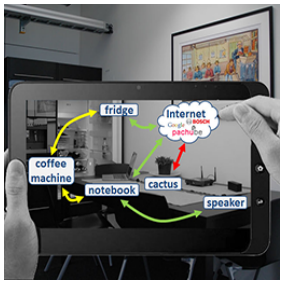
`code_localization.mp4`
This video presents our fast QR code localization method [206, 209] running entirely on the GPU of a smartphone. The method is able to detect multiple codes even at 1 m distance in full HD resolution images.

`blur_removal_examples.mp4`
Live blur removal from QR codes on Android devices. We show examples of some extreme blur cases and a synthetic example running on Google Glass smartglasses. Video adapted from [212].

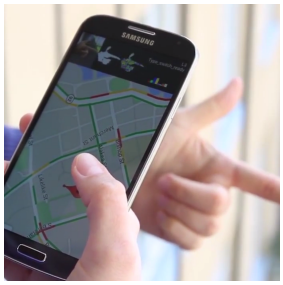
`blur_removal_initialization.mp4`
Further blur removal examples showing the superiority of our initialization scheme. The grid starting kernel is more effective in removing large blur.



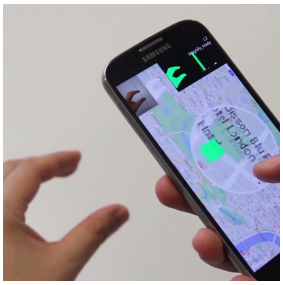
`blur_removal_extreme.mp4`
Inverting extreme blur where the structure of the QR code is highly distorted.



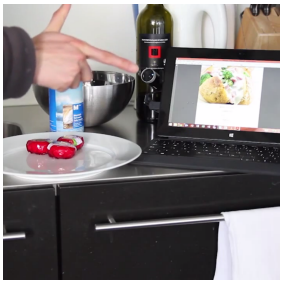
`augmented_reality_network_visualizer.mp4`
Live object recognition for visualizing network traffic in smart homes. The smart appliances in the camera's field of view are recognized by tiny visual tags or by their visual features, and network connections are overlaid on the camera preview. Video adapted from [143].



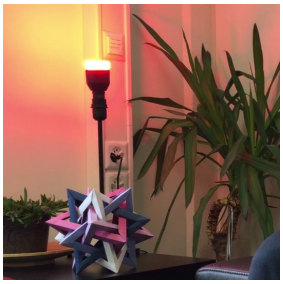
`gesture_control_map.mp4`
Bimanual interaction with a map application on a smartphone. The dominant hand can navigate on the touch screen while the non-dominant hand can perform gestures behind the device. Video adapted from [204].



`gesture_control_magnifier.mp4`
Continuous switching between gestures allows fine control of the size of a virtual magnifier lens overlaid on a map. Video adapted from [204].

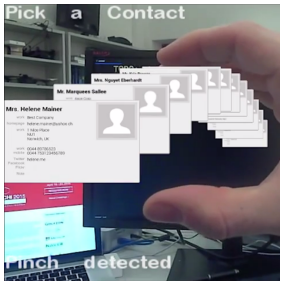


`gesture_control_tablet_watch.mp4`
Live gesture recognition running on a tablet and on a smartwatch. The user can control videos and switch between recipes in a cooking app. Our algorithm is simple enough to run even on a first-generation smartwatch. Video adapted from [204].

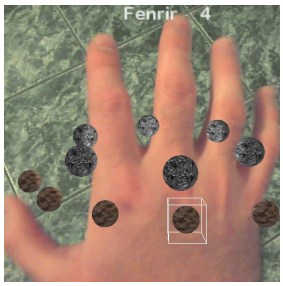


`gesture_control_lamp.mp4`
Controlling a Web-enabled light bulb with in-air gestures. The user can switch the lamp on and off and change the color of the light by performing simple gestures in front of a smartphone or a smartwatch camera.

`gesture_control_medical.mp4`
Controlling a medical visualization screen with in-air gestures. Touching the screen is sometimes not possible in medical scenarios therefore in-air gestures are particularly useful for interaction with the visualized content. We apply a tablet and a smartphone here but the camera could also be built into the screen.



`gesture_control_3d_phonebook.mp4`
Interaction with a 3D phonebook application on smartglasses. The user can open the contact cards with the pinch gesture, scroll through the cards by swiping the palm forward and backward, and place a call by grabbing a card. Video adapted from [203].



`gesture_control_3d_menu.mp4`
A 3D menu application on smartglasses. The user can scroll through menu elements by swiping the palm left or right, open sub-menus by swiping forward or backward, and select menu elements with the fist gesture. Video adapted from [203].

Short Curriculum Vitae

Gábor Sörös

Personal Data

Date of Birth	2 June 1986 in Veszprém, Hungary
Citizenship	Hungarian

Education

2011 – 2016	**Dr. Sc.** in Computer Science Department of Computer Science ETH Zurich, Switzerland
2009 – 2011	**M. Sc.** in Electrical Engineering *(with distinction)* Faculty of Electrical Engineering and Informatics Budapest University of Technology and Economics (BME), Hungary
2005 – 2009	**B. Sc.** in Electrical Engineering *(with distinction)* Faculty of Electrical Engineering and Informatics Budapest University of Technology and Economics (BME), Hungary
1994 – 2002	**Matura** *(with distinction)* Lovassy László Grammar School, Veszprém, Hungary

Professional Experience

2011 – 2016	Research and teaching assistant at the Institute for Pervasive Computing, ETH Zurich, Switzerland
2012 – 2015	External advisor, Scandit AG, Zurich, Switzerland
2014	R&D intern at Qualcomm Research Vienna, Austria
2010 – 2011	Undergraduate research assistant at the Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria
2008 – 2010	Undergraduate research assistant at the Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA-SZTAKI)
2009	Intern at the Institute of Information Processing Technology, Karlsruhe Institute of Technology (KIT), Germany
2007	Intern at the Department of Broadband Telecommunications and Electromagnetic Theory, BME, Hungary