# Gesture Recognition: Hand Pose Estimation

Ubiquitous computing seminar FS2014
Student report

*Adrian Spurr*
ETH Zurich
spurra@student.ethz.ch

**ABSTRACT**
In this report, different vision-based approaches to solve the problem of hand pose estimation are reviewed. The first three methods presented utilize random forest, whereas the last tackles the problem of pose estimation of two strongly interacting hands. Lastly, an outlook at the future of the field is given.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

**General terms:** Algorithms, Human Factors.

**Keywords:** Ubiquitous computing, gesture recognition, hand pose estimation, human computer interaction.

## 1. INTRODUCTION

In today's environment, computers have become ubiquitous. They are so seamlessly integrated in the lives of people, that we barely notice their presence and influence. Efficient interaction with these devices is vital for a productive society. To avoid having to spend too much time on learning on how to properly interact with them, it is vital that the interfaces are as natural to human action as possible. This is where hand pose estimation comes into play. Due to being a natural part of human interaction, as well as being the most dexterous body part human beings possess, there has been a great emphasis on enabling the computers to recognize and understand hand gestures. This ability can be used for a wide area of application, such as virtual/augmented reality, gaming, robot control/learning, and even medical applications. One of the first attempts to solve this problem was the data glove. Even though being highly accurate and having the possibility to provide haptic feedback, its disadvantages are too many for wide and practical use. These include, amongst others, the long calibration time, invasiveness and hindrance of natural movements being cabled, therefore unsuitable for mobile use. Vision-based systems are believed to be a promising alternative, not possessing any of these disadvantages. However, these systems bring with them an array of challenging problems, which need to be solved to achieve widespread utilization. These include self-occlusion, viewpoint change, noisy data, accuracy and performance. The field of hand pose estimation has made a rapid progress in recent years, mainly due to the availability of cheap enabling hardware, such as RGB-D cameras, or high powered GPUs. In this report, approaches on how these problems can be solved with the help of state-of-the-art algorithms are provided. Each approach will be summarized in detail and its extent of accuracy presented.

## 2. OVERVIEW

The general approach is similar for each algorithm assisted by a RGB-D camera, which provide the computer with either colour or depth images, or both. Normally the depth image is preferable, as it is easier to segment. However, inclusion of the colour image into the estimation process helps to perform more accurate prediction. The images are then preprocessed to assist in the estimation step. The main prediction is performed and the result is an output, in the form of the hand skeleton. This computer-usable form is a succinct and convenient way to represent the current configuration of the hand.

### Difficulties

Pose estimation, specifically hand pose estimation comes with challenging difficulties, which need to be considered to have an accurate and robust classifier.

*Noisy Data and segmentation.* Noisy data, which include low resolution images from low resolution cameras or if the person is far away. Noise also includes anything else in the image that is not of interest for pose estimation, in other words, the background. This can be seen as well as a segmentation problem. One would like to have the hand segmented without any additional noise. There are several approaches to solving this, such as a simple threshold on the depth, displaying only close up body parts, which ideally would be the hand. Or alternatively, one can segment based on the skin colour, with help of the colour image.

*Self-Occlusion and Viewpoint change.* Frequently, fingers will be covering other fingers and not all hand parts will be visible from the depth image. One has to therefore infer the position of occluded hand parts from what is visible. The change of viewpoint introduces additional occlusion, which have to be accounted for. Accurate approaches are able to deal with these situations.

*High Degree of Freedom per Hand* Each hand has 27 degrees of freedom, which account for about 280 trillion hand poses. These either have to be learned, or should be able to generalize to do so.

*Performance.* Low computational cost is critical. Any practical application of hand pose estimation requires real time performance. This is an issue for many approaches, which generally have high computational cost due to sophisticated algorithms. However, with today's widespread availability of high powered GPU's and CPU's, these are becoming increasingly more applicable for real time application.

### Approaches

In general, there are two main approaches used in today's algorithms and both have their advantages and disadvantages.

*Discriminative Approach* This approach maps the input directly to a hand skeleton configuration, with the help of the knowledge it gained through a training phase. This is both an advantage and a disadvantage, as often the quality of the prediction depends on the quality and quantity of the training data and not only on the actual approach. Access to good training data is laborious. For supervised methods, one has to either utilize synthetic data as proposed by [1] or manually label all the realistic data. It is hard to compare discriminative approaches with each other, as they are trained and tested on different data. Nevertheless, the advantage of the discriminative approach is that it is reasonably robust, both to noisy data and rapid change of hand movement.

*Model-Based Approach* In this approach, the algorithm keeps track of the hand pose internally, called hypothesis, and estimates the current hand pose with the help of the hypothesis and the observation delivered by the RGB-D camera. It therefore naturally exploits temporal information and requires no training. This is a huge advantage over the discriminative approach, as the accuracy of it is only dependent on the implementation of the algorithm and not on training data. However, due to using temporal information it is also very sensitive towards rapid changes in pose. In other words, the approach tends to be especially prone to errors during rapid hand movements.

### Random Forests

Random Forests were first proposed by [5]. They consist of decision trees that performs regression or classification. The final result of the Forest is the average of all the trees contained within. It belongs to a class of machine learning algorithm called ensemble method. These methods are particularly robust to noise due to the averaging step done to perform the final decision. They can be implemented efficiently on GPUs and are therefore fast. To classify an input $x$, one starts at the root of the tree and assigns it to the either left child or the right child, based on the feature $\theta(x)$ and a threshold $\tau$. Once a leaf node is reached, the containing learned distribution is used as the output of that tree. To perform the final classification, the distributions of each tree are then averaged over all the trees, and a final classification is performed, according to equation 1:

$$P(c, x) = \frac{1}{T} \sum_{t=1}^{T} P_t(c, x) \tag{1}$$

To build such a tree, a training phase is required. One can use either synthetic or realistic training data, provided that they are labelled. Each tree is trained on a separate random subset of the training data. On each split node the training set is separated according to formula 2 and 3.

$$Q_1(\theta, \tau) = \{x | \theta(x) < \tau\} \tag{2}$$

$$Q_2(\theta, \tau) = \{x | \theta(x) \geq \tau\} \tag{3}$$

The feature $\theta$ and the threshold $\tau$ chosen for the split is the pair which maximizes the information gain, computed by a quality function. They are selected out of a random subset sampled from the full set of features and thresholds. This node splitting is recursively applied to each child node, until a maximum tree depth is reached or until the child nodes are pure enough, with respect to the quality function. The distribution of the leaf node is computed according to the labelled training data reaching the node. Random Forests are utilized in many discriminative approaches. They can be used to classify elements according to body parts, creating a body label distribution out of which a 3D joint proposal can be done. This is applied in [1, 2]. Alternatively they can be used to perform regression, such as in [3], to directly infer the position of the joints.

### Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a method to optimize an objective function defined over a parameter space. To make use of the algorithm, one needs to define a parameter space and an objective function $f$ to optimize. To initialize, $n$ particles are randomly sampled across the parameter space, each assigned a velocity, uniformly distributed over a defined region. These particles iterate over the parameter space according to their velocity. Once a maximum number of iterations is reached, the algorithm is terminated and the best position found is output. At each iteration $k$, each particle $i$ updates their position $x_{k,i}$ according to 4:

$$x_{k+1,i} = x_{k,i} + v_{k,i} \tag{4}$$

At the same time, the velocity $v_{k,i}$ gets updated according to 5:

$$v_{k+1,i} = v_{k,i} + r_1(P_{k,i} - x_{k,i}) + r_2(G_k - x_{k,i}) \tag{5}$$

Where $P_{k,i}$ is the best position found by particle $i$ and $G_k$ is the best position found over all particles, both until generation $k$. In terms of equations:

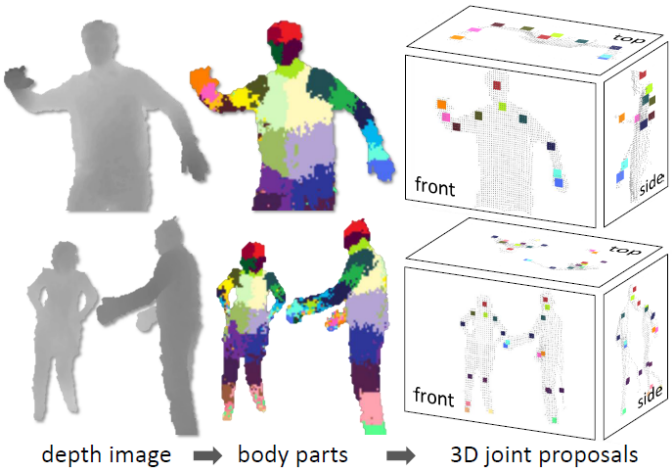


Figure 1: First the depth image is obtained. After passing through the random forest, a body label distribution is returned, with which the 3D joint proposals can be formed with help of mean shift algorithm.

$$P_{k+1,i} = \begin{cases} P_{k,i} & \text{if } f(P_{k,i}) \geq f(x_{k+1,i}) \\ x_{k+1,i} & \text{else} \end{cases}$$
$$G_{k+1} = P_{k+1,i},\ where\ i = \arg\max_i f(P_{k+1,i})$$

The advantage of PSO over other optimization methods is that it makes very little assumption over the objective function used. However, neither convergence nor finding the optimal position is guaranteed. One can assume that having enough particles and allowing the algorithm to iterate enough times, a sufficiently close position can be found. PSO is used in the last method presented in this report, a model-based approach using a objective function to penalize the discrepancy between the hypothesis and the observation.

## 3. METHODS

The first three methods described here all utilize random forests for the first part of the estimation step. The exception is the last method, which attempts to estimate the hand pose of two strongly interacting hand with the help of PSO.

### Real-Time Human Pose Recognition from Depth Image

[1] provides a solid basis for many, at that time, future papers to come. They were the first to propose the use of synthetic training data, that makes methods which have a training phase a lot easier. The idea is to use Random Forests to classify the depth image into body parts, resulting in a body part label distribution. These are then used to form the final joint proposals.

*Synthetic Training Data.* The idea is as simple as it is powerful. To have an accurate classifier, one needs enough variation in the training data to generalize well to all possible poses and to do so, one can vary real data. These are typically variations in body height, weight or clothing and hair. The advantage of this approach is, that one does not have to manually label the entire training set, as it suffices to have one labelled element upon which the variation bases on. A general overview can be seen in Figure 1.

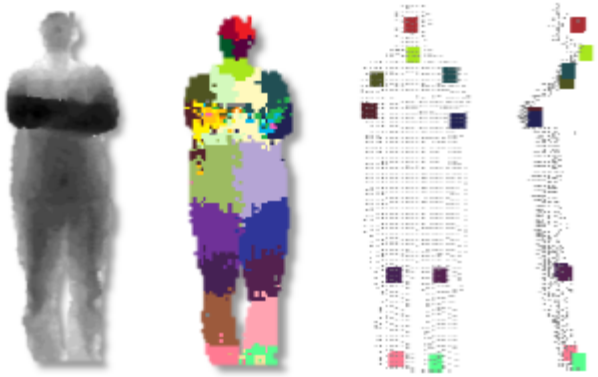

Figure 2: An example of a failed prediction of [1] due to occlusion.

*Training and Prediction.* To perform classification with random forest, the following feature is used.

$$f(I,x)_{(u,v)} = I(x + \frac{u}{I(x)}) - I(x + \frac{v}{I(x)}) \qquad (6)$$

Where $I(x)$ is the depth value at location of pixel $x$. The offsets $u$ and $v$ are normalized to provide 3D translation invariance. This function compares two depth values with respect to $x$. This aims to convey spatial context of $x$. These features are very efficient to compute, but are weak individually, yet when used in conjunction with random forest, they provide sufficient spatial information. When training the random forest, the trees randomly sample a set of $u$, $v$ and $\tau$, bounded within an interval, and select the best according to the equation 7

$$G(u,v,\tau) = H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(u,v,\tau)|}{|Q|} H(Q_s(u,v,\tau)) \qquad (7)$$

where $H(Q)$ denotes the Shannon entropy function and $Q$ are the sets or subsets respectfully. Once the random forest is trained one can acquire the body part distribution. The mean of each distribution is computed with help of the mean shift algorithm [10]. Depending on the body part, the 3D joint proposals are set at the mean of each distribution or averaged, that average being then the proposal. However, because the mean lies on the surface of the body, it is pushed within it by a learned offset $\lambda$.

*Accuracy.* Accuracy of the random forest for body part labelling depends on a variety of different parameters. First, increasing the number of training images results in a logarithmic increase of label accuracy. This is intuitively clear, as the random forest has more information on how to classify. Allowing the trees of the random forest to have a greater maximum depth leads to a positive increase in correct labelling. By having deeper trees, we allow more splits to be performed, thus the end nodes can be more pure with respect to their labels. However, when using a smaller test set, overfitting is observed after a certain depth. This effect is counteracted by training on a bigger test set. Lastly, by having a larger interval to choose the offsets $u$ and $v$ of function 6 from, we allow the classifier to use more spatial context

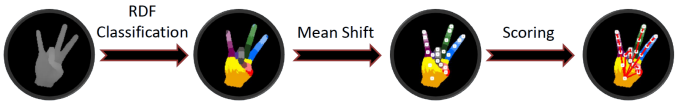


Figure 3: Keskins approach. Similarity can be seen to [1]

to perform its splitting decisions. The joint prediction accuracy using the predicted body parts suffers slightly when compared to the accuracy using the ground truth labels. Nevertheless, this occurs mainly for smaller body parts, such as the elbows and the arms. Overall, the body position is predicted fairly accurately. Miss-predictions occur mainly for heavily occluded body parts or for unseen poses, which differ by a lot from the training set, as shown in Figure .

**Multi-layered Random Forests**

Building up on [1] and proposed first by [2], this approach tackles the problem of having a big training set and applied for hand pose estimation. It divides the hand part label classification into two layers by first clustering the training data based on similarity, then training a random forest on and for each of these clusters. To perform prediction, the input is taken and fed into the first layer, which assigns it to the correct cluster. The assigned cluster performs the final hand part label classification. The first layer can be seen as a rough classification, the second a more fine tuned one. Once the hand part distribution is obtained, mean shift can be used to acquire the mean of each distribution, where the hand joint proposal get set, resulting in the hand skeleton. An overview is shown in Figure 3

*Clustering* Clustering is performed with the help of the spectral clustering algorithm. This separates the training data based on weighted difference, which can be fine tuned to reduce the workload on the second cluster, increasing the overall accuracy of the method. The procedure is based on distance of two skeletal configuration $v_i$,$v_j$ and the distance

$$D_{ij} = |W(v_i - v_j)| \tag{8}$$

Where $D_{ij}$ is the diagonal weight matrix. To perform the final clustering, k-means gets applied on $D$. These clusters are assigned their own label, after which a random forest is trained using these labels. The same features and quality function gets used as in [2]. This forms the first layer.

*Second Layer.* The second layer formed by training a random forest for each cluster, this time using the hand part labels, but using the same features and quality function. Each random forest is called expert.

*Prediction.* Predicting can be performed using one of two methods. Global Expert Network (GEN) assigns the input into the first layer, which outputs the votes for each cluster. This is averaged and the entire input is assigned to the top three clusters. The outputs get averaged and the final hand part label classification is performed. Local Expert Network (LEN) goes through the first layer, but assigns each pixel to the cluster it voted for. This results in LEN being able to generalize better than GEN, however GEN is more robust to noise due to the averaging step. Then, one can get the mean of each distribution utilizing mean shift algorithm, resulting in the 3D joint proposals, with which we can form the final hand skeleton.

*Accuracy.* The accuracy of the first layer is high. Moreover, it is very fast due to utilizing random forest, and can be trained using real images as well, as the clustering step is a form of unsupervised learning. An accuracy of 84 % on a testing set is reported. The second layer accuracy depends on similar parameters as in [1]. Additionally, two new factors are introduced: The number of clusters $K$ and the weight matrix $W$. By increasing $K$, it is ensured that only similar configurations fall into the same cluster. Therefore, it has a positive effect on the accuracy of both layers, reducing the complexity of them. On the other hand, $W$ can be determined in such a way that variations which are harder to detect are already clustered in the first layer, reducing the workload the second layer has. The accuracy difference of LEN and GEN are negligible, both having high accuracy of roughly 90%, with GEN being slightly better.

**Semi-Supervised Transductive Regression Forest**

The approach described in [3] aims to reduce the accuracy errors which occur when using synthetic training data during the training phase, such as in [1, 2], due to realistic-synthetic discrepancies. This is performed with help of transductive learning, which means one can learn from a target domain, in our case a labelled set of images, and apply knowledge transform onto a unlabelled image set. With this, the algorithm learns how to train with both labelled and unlabelled training data. The random forest used performs regression. The multi-layered aspect of [2] is combined into one layer, by utilizing different quality functions while growing the tree, and subsequently controlling what kind of classification is performed. The final output of the regression forest is the 3D joint location. One can utilize kinematic refinement on the output, to enforce bio-mechanical constraints of the hand, thus further increasing the accuracy of the prediction.

*Training.* Training is performed with both labelled realistic and synthetic training data, as well as with unlabelled realistic training data. The labelled elements consist of image patches, each label being a tuple $(a, p, v)$, with $a$ being the viewpoint, $p$ being the label of the closest joint and $v$ being a $16x3$ vote vector, containing the 3D location of all joints of that image. Additionally, realistic-synthetic associations $\psi$ are established by comparing their 3D joint location, contained in $v$:

$$\psi(r, s) = \begin{cases} 1 & if\ r\ matches\ s \\ 0 & else \end{cases}$$

Where $r \in R_l$, the labelled realistic training data and $s \in$ S, the synthetic training data.
When trees of the random forests are being grown, they randomly choose between two quality functions

$$Q_{apv} = \alpha Q_a + (1-\alpha)\beta Q_p + (1-\alpha)(1-\beta)Q_v \tag{9}$$

$$Q_{tss} = Q_t^{\omega} Q_u \tag{10}$$

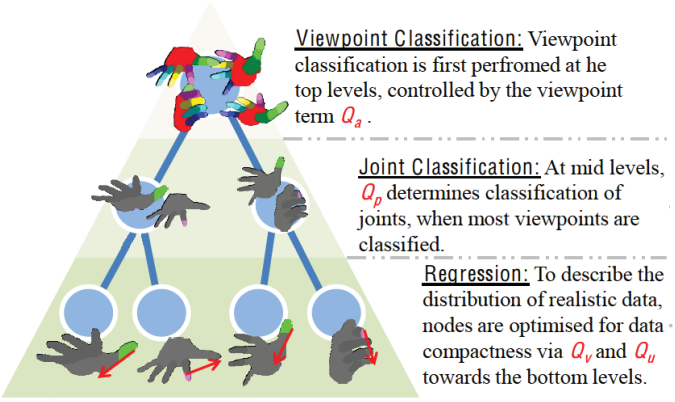


Figure 4: A tree in the regression forest of [3]. First, classification is performed based on the viewpoint label, then the joint label, and finally regression is done with the help of the vote vectors v

Where equation 9 is the classification-regression term, which controls the classification and regression performed in the tree, and equation 10 is the transductive term, performing transductive learning. Equation 9 consists of three terms:
$Q_a$ is the viewpoint classification term, measuring the information gain with respect to the viewpoint label $a$.
$Q_p$ is the joint classification term. Similar to $Q_a$ it measures the information gain with respect to the joint label $p$.
$Q_v$ is the regression term. It measures the compactness of the vote vector $v$. This is defined as

$$Q_v = \left[1 + \frac{|L_{lc}|}{|L|}\lambda(J(L_{lc})) + \frac{|L_{lr}|}{|L|}\lambda(J(L_{lr}))\right]^{-1} \quad (11)$$

Where $\lambda(x) = trace(var(x))$.
$Q_t$ represents the transductive term, preserving the cross-domain association $\psi$:

$$Q_t = \frac{|\{r,s\} \subset L_{l,c}| + |\{r,s\} \subset L_{r,c}|}{|\{r,s\} \subset L|} \quad (12)$$

$\forall\{r,s\} \subset L$ where $\psi(r,s) = 1$
$Q_u$ is the unsupervised term, evaluating the appearance similarities of all realistic patches R within a node:

$$Q_u = \left[1 + \frac{|R_{lc}|}{|R|}\lambda(R_{lc}) + \frac{|R_{rc}|}{|R|}\lambda(R_{rc})\right]^{-1} \quad (13)$$

The parameters $\alpha$ and $\beta$ are dynamically set according the purity of the child nodes of the tree. With help of these and $\omega$, which can be set to weigh $Q_t$ against $Q_u$, one can choose based on what the tree classifies on each layer. The chosen mode for the approach is shown in Figure 4

*Prediction and Kinematic Joint Refinement.* When forming the prediction, the input is directly fed into the Regression Forest, returning the 3D joint location of the image directly. However, the results lack structural information to recover poorly detected joints due to occlusion. To avoid having an explicit hand model, a kinematic model is created with help of large hand pose data base $K$, which is specifically made for this cause. To create the models, $K$ gets split with respect to the viewpoint label $A$, i.e K = $\{K_1, ...K_{|A|}\}$. Then, for each viewpoint i in A, a N-Part Gaussian Mixture Model $G_i$ of the dataset $K_i$ gets fit, so that $G_i = \{\mu_i^1,...,\mu_i^n,...,\mu_i^N; \sigma_i^1,...,\sigma_i^n,...,\sigma_i^N\}$, where $\mu_i^n$ and $\sigma_i^n$ denote the mean and diagonal variance of the n-th Gaussian component in $G_i$ of view i. After obtaining the models, the Kinematic Joint Refinement step can be applied to compute the final joint location. To perform this step, the set of votes received by the j-th joint is fitted a 2-part GMM $G_j = \{\mu_j^1, \sigma_j^1, \rho_j^1, \mu_j^2, \sigma_j^2, \rho_j^2\}$, where $\mu$ is the mean, $\sigma$ the variance and $\rho$ the weight of the Gaussian components. If a joint prediction has high confidence, it forms a compact cluster of votes, which is shown by a high weighting and low variance of the GMM. On the contrary, a joint prediction containing distributed votes is a sign of low confidence. In mathematical terms, a strong detection is a joint having the Euclidean distance between $\mu_j^1$ and $\mu_j^2$ below a threshold $t_q$. These high confidence predictions remain as the final position and the joint refinement process is applied to the low-confidence votes. To perform this, a nearest-neighbour search (NN-search) is done, using only the high confidence joint location. The final joint position gets computed with respect to the weak joint location and the high confidence joint position found in the NN-search.

*Accuracy.* This approach has a high reported accuracy. In combination with kinematic joint refinement, it represents state-of-the-art accuracy. Testing was performed in many scenarios, including rapid changing and heavily occluded hand poses, each resulting in accurate prediction.

### Estimating the Pose of Two Strongly Interacting Hands

This approach, proposed by [4] aims to solve the interesting problem of detecting the position and configuration of two strongly entangled hands. It does this with a model-based approach in mind, utilizing PSO described earlier. It is not sufficient to utilize a single hand pose estimator twice for each hand, as the additional occlusion resulting from the hand interaction must be accounted for. Its important to note that this approach does not run in real time.

*Tracking.* Tracking utilizes both depth and RGB image. From the RGB image, a skin map is obtained, with which the depth image is segmented, obtaining the hand part of the depth image. To use particle swarm optimization, a parameter space needs to be defined. Because the hand contains 27 degrees of freedom, a 54-dimensional parameter space, representing all the possible configuration of both hands, is used. The range of each dimension is linearly bounded, according to bio-mechanical constraints of the hand. Each point in the parameter space represents a configuration of both hands. These are appropriately artificially skinned, so that a skin map can be obtained for later use in the objective function. This function is essentially a penalty function to be minimized, defined as:

$$E(O,h,C) = P(h) + \lambda_k D(O,h,C) \quad (14)$$

Where $\lambda_k$ is a regularisation parameter, O is the observation delivered by the RGB-D Camera, h is the current hypothesis

held and C is calibration information of the camera, to synthesize more accurately the skin map from h.
$P(h)$ penalizes invalid articulation hypothesis. Because the linear constraints on the parameter space are not sufficient enough to avoid such configurations, $P(h)$ is needed.
$D(O, h, C)$ penalizes the discrepancies between the O and h. This is performed with help of the skin maps, as well as the depth images. Essentially, it measures the differences between both, summing and clamping them. Clamping is necessary to make the function robust to noise which could dominate and produce false high penalties. Additionally, through clamping, the function results in being smother. When utilizing PSO, function 5 gets adjusted to:

$$v_{k+1,i} = w(v_{k,i} + c_1 r_1 (P_{k,i} - x_{k,i}) + c_2 r_2 (G_k - x_{k,i})) \quad (15)$$

It introduces three new weight parameters $c_1$, $c_2$ and $w$. As proposed in [8], $w$ is set as

$$w = \frac{2}{|2 - \psi - \sqrt{\psi^2 - 4\psi}|} \quad (16)$$

where $\psi = c_1 + c_2$.

A new instance of PSO is performed by each observation delivered by the RGB-D camera. Each instance delivers $h_{min}$, so that:

$$h_{min} = argmin_h E(O, h, C) \quad (17)$$

To exploit temporal information, the hypothesis $h_{min}$ gets stored. When a new instance of PSO starts, the particles are sampled randomly in and around $h_{min}$, boosting performance and accuracy of method. However, this approach results in bad accuracy when the hands rapidly changes pose. Additionally, it needs to be initialized at the beginning. The nature of PSO allows for easy parallelization, increasing performance. Some example prediction can be seen in Figure 5.

*Accuracy.* Utilizing PSO, this approach has two main parameters to choose from: Number of particles and maximum number of iterations. Naturally, increasing either has a positive effect in accuracy. In the first case, we have more particles searching for the optimal position, therefore a higher probability of finding a sufficient close point in the parameter space. In the latter case, we allow the algorithm to search longer, again increasing the chances of finding the best or a sufficient close enough position. However, increasing both parameters has a detrimental effect on performance and a flat-off of accuracy can be observed after a certain threshold of the parameters is reached. It is important to note here that unlike the previous methods, this approach does not run in real-time.

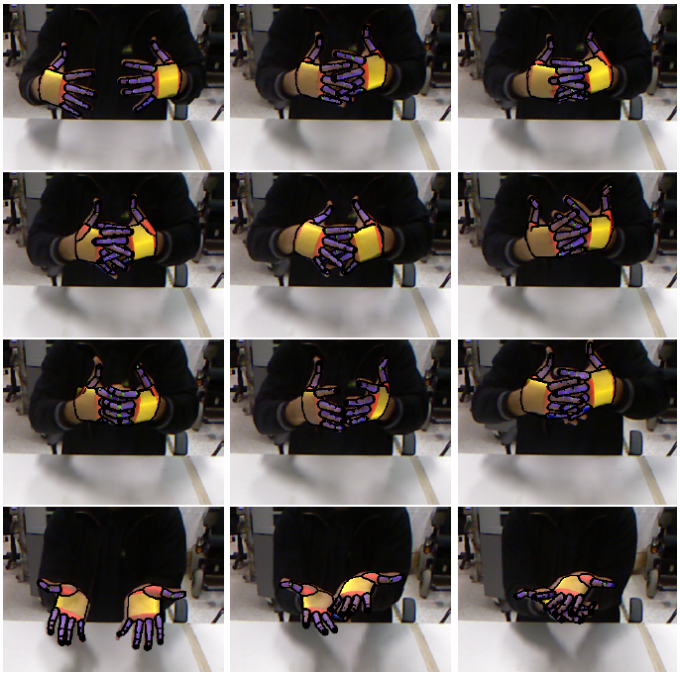

Figure 5: Some example results of [4].

## 4. FUTURE OUTLOOK

Research in the area of hand pose estimation has shifted its focus to recovering more information than just the hand pose. For example, there is a growing interest in recovering the hand and skin model, for use in areas like Telepresence or Virtual Reality. Conducting is required further research in areas of new features, as these heavily influence accuracy and robustness of estimation algorithms. Additionally semi-supervised approaches or unsupervised methods, not presented here, show a lot of potential in reducing the amount of synthetic training data needed, hence reducing the realistic-synthetic discrepancies, further increasing accuracy.

## 5. CONCLUSION

With all the different approaches presented here, tackling the problem of hand pose estimation. [1] was an essential building block for future methods, and was the first to utilize synthetic training data. This solves the problem of body pose estimation and is used in today's Kinect system. It uses random forest to assign a body label distribution over the depth image, and then places the 3D joint proposals based on the distribution. The next method, [2], tackles hand pose estimation. It introduced multi-layered random forests, a tool to divide the problem into sub-problems by clustering the training data and having a random forest trained on the clusters and for each cluster. This approach takes inspiration from [1], by assigning the depth images a hand label distribution and then, using the same approach, form the 3D joint proposals of the hand. [3] implements random regression forests, and combines the multi-layer aspect of [2] into one layer through varied use of different information gain function. The output of the regression forest directly form the 3D joint proposal. To further enhance the accuracy and robustness to occlusion, kinematic refinement is employed. This en-

forces somewhat bio-mechanical constraints which normally discriminative approaches lack. Lastly, [4], a model-based approach tries to solve the problem of pose estimation of two hands, which could be strongly interacting. It uses PSO to seek for a minimum of the objective function defined over the 54-dimensional parameter space. This objective function penalizes the discrepancies between the observation and the currently held model of the hand, the hypothesis, and invalid hand configurations, such as two fingers occupying the same physical space. However, this approach does not run in real time.

## REFERENCES


1. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman and A. Blake. Real-Time Human Pose Recognition in Parts from Single Depth Images. In: CVPR, 2011.

2. C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In ECCV, 2012.

3. D. Tang, T. Yu, T. Kim. Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests. In ICCV, 2013.

4. I. Oikonomidis, N. Kyriazis, A.A. Argyros. Tracking the Articulated Motion of Two Strongly Interacting Hands. In CVPR, 2012.

5. L. Breiman. Random Forests. 1999.

6. A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, X. Twombly. Vision-based hand pose estimation: A review. 2005

7. S. S. Rautaray, A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. 2012. 11

8. M. Clerc and J. Kennedy. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. Transactions on Evolutionary Computation, 6(1):58? 73, 2002

9. J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. PAMI, 2011.

10. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. IEEE Trans. PAMI, 24(5), 2002.