Towards the automatic definition of the objective function for model-based 3D hand tracking

Konstantinos Paliouras and Antonis A. Argyros

Institute of Computer Science - FORTH
and
Computer Science Department - University of Crete
Heraklion, Crete, Greece

Abstract. Recently, model-based approaches have produced very promising results to the problems of 3D hand tracking. The current state of the art method recovers the 3D position, orientation and 20 DOF articulation of a human hand from markerless visual observations obtained by an RGB-D sensor. Hand pose estimation is formulated as an optimization problem, seeking for the hand model parameters that minimize an objective function that quantifies the discrepancy between the appearance of hand hypotheses and the actual hand observation. The design of such a function is a complicated process that requires a lot of prior experience with the problem. In this paper we automate the definition of the objective function in such optimization problems. First, a set of relevant, candidate image features is computed. Then, given synthetic data sets with ground truth information, regression analysis is used to combine these features in an objective function that seeks to maximize optimization performance. Extensive experiments study the performance of the proposed approach based on various dataset generation strategies and feature selection techniques.

1 Introduction

The automatic capture and analysis of human motion has several applications and high scientific interest. Long standing unresolved human-computer interaction problems could be solved by directly using the human body and, in particular the human hands for interacting with computers.

Several solutions have been proposed for the problem of 3D hand tracking [6, 11]. These solutions can be divided into two main categories, the *appearance-based* and the *model-based* approaches. The appearance-based approaches try to solve the problem by defining a map between the feature space and the solution space. This map is usually constructed with offline training of a prediction model, which can be either a regression or classifier model. The regression models are used to predict the exact configuration of the hand in the solution space, while the classifier models usually try to predict the posture of the observed hand. In contrast, model-based approaches operate directly on the solution space. Usually this involves making multiple hypotheses in the solution space that are evaluated in feature space by comparing the appearance of the observed hand and the estimated appearance of the hypotheses. This is formulated as an optimization problem whose objective function evaluates hypotheses (candidate hand

configurations that are rendered through graphics techniques by taking into account its kinematic model) against the actual hand observations. The objective function has a determining role in the quality of the obtained solutions as well as to the convergence properties of the optimization process. Its formulation requires prior-experience on the problem and a lot of fine tuning that is performed on a trial-and-error basis.

In this paper, our aim is to automate the process of objective function definition with a methodology that does not demand deep prior-knowledge of the problem. More specifically, a set of features that are relevant to the problem are supposed to be given. Then, regression analysis is used to define an objective function that given the available features, approximates as better as possible the true distance between different hand poses. To do so, synthetic datasets are used which are built by sampling the multi-dimensional solution space with two different strategies.

2 Related work

A lot of published work exists for recovering the full 3D configuration of articulated objects, especially the human body or parts of it like hands, head etc. Moeslund et al. [11] have made an extensive survey on vision-based human body capture and analysis. Hand tracking and body tracking problems share many similarities, like hierarchical tree structure, problem dimensionality and complexity, occlusions and anatomic constraints. Erol et al. [6] present a variety of methods for hand pose estimation or tracking. Depending on the output of these methods they are divided in partial and full pose estimation methods. Further categorization is between appearance-based and model-based methods. Typically appearance-based methods [14, 15, 17] solve the problem by modelling the mapping between the feature space and the pose space either analytically, or through machine learning techniques that are trained on specific datasets to perform either classification or regression. Appearance-based methods perform fast on prediction and are suitable for gesture recognition. A common problem though, for these methods, is that they usually demand large training dataset which, thus, is typically relevant to a specific application and/or scenario. To compensate for potential bias of the training dataset, Shotton et al. [15] generated a large synthetic dataset, permitting good generalization.

On the other hand, model-based methods [5, 8, 10, 12] search directly for the solution in the configuration space. Each hypothesis is rendered in feature space. An error/objective function evaluates visual discrepancies, which usually demands high computational resources. On the positive side, model-based methods do not need offline training, making them easier to employ as they are not biased to specific training datasets.

In all optimization problems, there are two major components, the error/objective function and the optimization algorithm. Different options exist for both components but the performance is dependent on the correct combination of the two. As an example, de La Gorce et al. [5] used a quasi-newton optimization algorithm and a hand made error function that considers textures and shading of the model which proved to perform well on the problem. Oikonomidis et al. [12] used the *Particle Swarm Optimization* (PSO) algorithm and they also crafted a special objective function, taking into

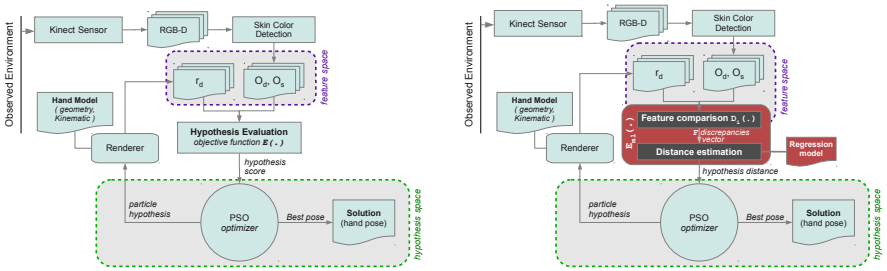

Fig. 1: Overview of baseline method pipeline (left) and the proposed modification (right) In the proposed work, the objective function is not handcrafter but rather estimated through an offline trained regression model.

account the skin color and depth information provided by a RGB-D sensor or from a multi-camera setup [13]. Recent works have tried to combine the advantages of both methods basically using machine learning. Xiong et al. [18] mentioned the effectiveness of 2^{nd} order descendent methods, as well as the difficulty in using them in computer vision, mainly because it is hard to analytically differentiate the objective function. They proposed instead, a supervised descent method that models the behaviour of an objective function in offline training. At prediction time, this method consults the learned model to estimate the descent direction, allowing the usage of any non linear objective function. Cao et al. [4] have presented an appearance-based method for tracking facial expression. In their work, they have used regression analysis for modelling the correlation between the feature space and the 3D positions of a face shape model.

3 The baseline method

For the purposes of studying and evaluating an alternative method for constructing the objective function of model-based hand-trackers, we used the work of Oikonomidis et al. [13]. In that work, the authors present a model-based approach for recovering and tracking the full configuration of a human hand with 27 DOFs. The problem is formulated as an optimization problem seeking the best solution in the configuration space of 27 dimensions. The methodology can be divided into 3 main components.

- *Observation*: Responsible for acquiring input from the sensor and pre-processing data. At the pre-process stage, it detects and isolates all the areas with skin color [2].
- *Hypothesis evaluation*: For each hypothesis made in configuration space its discrepancy to the observed hand pose must be evaluated. This component is responsible for quantifying this discrepancy by considering the visual discrepancies in feature space. To estimate the appearance of each hypothesis, rendering techniques are used to project the hypothesis in the feature space. To achieve this, the hand shape and its kinematic model are supplied to the rendering pipeline.
- *Optimization*: Performs optimization on solution space in order to find the hypothesis with the minimum distance to the observed hand. The PSO [9] optimization

algorithm was chosen for this task and the hypothesis evaluation component is used to score all candidates during the optimization process.

Figure 1 (left) illustrates a high-level flow diagram of the methodology followed in [13]. The objective function defined in [13] receives the configuration of the hypothesized hand h and the visual observation model of the tracked hand O and quantifies their discrepancy as:

$$D(O,h,C) = \frac{\sum min(|o_d - r_d|, d_M)}{\sum(o_s \vee r_m) + \varepsilon} + \lambda \left(1 - \frac{2\sum(o_s \wedge r_m)}{\sum(o_s \wedge r_m) + \sum(o_s \vee r_m)}\right) + \lambda_k \cdot kc(h). \tag{1}$$

The first term of Eq. (1) measures the difference between depth maps of an observed and a hypothesized hand pose. The second term of Eq. (1) performs a comparison of the segmented region of the hand with the corresponding map of the hypothesis. To perform the comparison, a mapping of the hypothesis h to feature space is applied by means of rendering. Finally, the third term adds a penalty to kinematically implausible hand configurations by penalizing adjacent finger interpenetration. For more details in that approach the reader is referred to [13].

4 Methodology

In this work, various simple features are computed to create a vector of scalar feature discrepancies. Then, by using *regression methods*, the correlation between the vector of feature discrepancies and the *true distance* is modeled and a learned function $E_{ml}(.)$ is defined to replace *baseline objective function* $E(h,O)$ (Eq. (1)). Figure 1 (right) shows how the new $E_{ml}(.)$ function is integrated in the *baseline method*. The proposed method consists of three-steps, (a) create a set of algorithms to calculate per feature discrepancy, quantified in a scalar variable (Sec. 4.1), (b) construct a dataset that will be used to train and evaluate the performance of various objective functions (Sec. 4.2) and, (c) train a machine-learned function using the dataset from the previous step. This function, will form the new objective function E_{ml} (Sec. 4.3).

4.1 Features and their comparison

We consider a number of features as well as functions $D_i(O,h)$ that measure the discrepancy of each of the feature between the observation O and a hypothesis h. In the following, o_d is the depth map of the observation, o_s is the skin map segmented using skin color detection, and r_d is the rendered depth map of hypothesis h. In all cases N is the total number of pixels of the feature maps.

Sum of depth differences $D_1(.)$ Depth discrepancy is very informative of the correlation between two poses. Unlike the baseline method, we define it without any data type of post-processing (e.g., clamping etc):

$$D_1(O,h) = \sum |o_d - r_d|. \tag{2}$$

Variance of depth distances $D_2(.)$ This provides another statistical perspective of the depth discrepancy and is defined as:

$$D_2(O,h) = \frac{1}{N}\sqrt{\sum (o_d - r_d)^2}. \tag{3}$$

Occupied area $D_3(.)$ This is defined as the difference of the areas (in pixels) covered by the segmented hand observation and hypothesis The area is calculated based on the corresponding depth maps as the number of non zero-valued pixels:

$$D_3(O,h) = \left| \sum_{pixel(o_d)\neq 0} 1 - \sum_{pixel(r_d)\neq 0} 1 \right|. \tag{4}$$

Accuracy of the skin map $D_4(.)$ Measures the compatibility between the observed skin color map and that of the hand hypothesis. This is quantified as the $F1$ measure between the two maps as:

$$D_4(O,h) = 2TP/(2TP + FP + FN), \tag{5}$$

where $TP = \sum o_d \wedge r_d$, $TN = \sum \neg o_d \wedge \neg r_d$ and $FN = \sum o_d \wedge \neg r_d$.

Depth map edges $D_5(.)$ Edges are computed with the Canny edge detector [3] resulting in the edge maps o_e and r_e for observation and hypothesis, respectively. Then the Euclidean distance map (ℓ_2 distance transformation) o_{ed} is generated over the edge map o_e, using [7]. A scalar value is the computed as:

$$D_5(O,h) = \sum o_{ed} \wedge r_e. \tag{6}$$

Hand contour $D_6(.)$ The method used in feature discrepancy $D_5(.)$ is also applied to compare contours rather than skin colored regions. Specifically, the o_s map and the binary mask r_m that corresponds to the occupied pixels of r_d are used to generate the edge maps of the observation and the hypothesis. Thus,

$$D_6(O,h) = \sum o_{sd} \wedge r_d, \tag{7}$$

where o_{sd} is the distance transform of o_s.

4.2 Dataset

Given two hand poses h_α and h_β we define their true distance $\Delta(h_\alpha, h_\beta)$ as

$$\Delta(h_\alpha, h_\beta) = \frac{1}{37}\sum_{i=1}^{37} ||p_i(h_\alpha) - p_i(h_\beta)||. \tag{8}$$

$\Delta(h_\alpha, h_\beta)$ is the averaged Euclidean distance between the centers of the 37 primitive shapes p_i comprising the hand model [13].

A dataset with examples of compared hand poses is needed for modeling the correlation between feature discrepancies and *true distance*. Every example in this dataset consists of the feature discrepancy vector F_i and the *true distance* Δ_i between an observed hand model and a hypothesis. To create a dataset, the search space of the optimization procedure must be sampled appropriately. The size of this space is huge to permit dense sampling. Therefore, two different sampling strategies are introduced. The first one uses a low-discrepancy sequence to select hand poses quasi-randomly. The second samples densely around the area that is mostly used during optimization.

Sampling with low-discrepancy sequence: The advantage of using low-discrepancy sequences is that they offer more uniform sampling of a multidimensional space, even for a few samples. Several such algorithms have been developed. In this paper we use the Sobol sequence [16]. The sampling procedure is performed in the following steps: (a) Create a set P of n quasi-random hand pose configurations (b) Generate the feature maps p_d and p_s (depth, skin map) for every hand pose in P and (c) for all possible combinations of P set, generate an example in the dataset that consists of the the true distance Δ_k and the F_k feature discrepancies vector calculated by the $D_i(.)$ functions. The bounds of each dimension are selected based on the usable frustum of the Kinect sensor and the possible movements of hand joints based on anatomical studies [1]. In particular, for the 27-DOF parameters of the hand pose the boundaries of the global position are selected so that the hand is always inside the perspective frustum. The boundaries of each finger are the same as the boundaries of the PSO optimization module in baseline method. In the special case of global orientation, another quasi-random algorithm is used to create random quaternions.

Sampling biased to optimization: The previous method provides a good strategy for uniformly sampling the search space. However, in practice, the optimization module of the baseline method considers solutions close to the actual one. This happens because the particles of PSO are distributed around the previous solution of the previous frame. Therefore, a second sampling strategy is proposed that samples the search space where the *baseline method* usually searches. To do so, we use the logs of previous hand tracked poses and the particles that PSO considered.

4.3 Regression Model

$E_{ml}(.)$ (Eq. (9)) is constructed by modelling the *training dataset* that was created using either of the methods described in Sec. 4.2. That is, given the outcomes of $D_i(.)$ functions (Sec. 4.1) we need to come up with a function $E_{ml}(.)$ that approximates as closely as possible the true distance $\Delta(.)$

$$E_{ml} = f(D_1(.), D_2(.), ..., D_n(.)) \tag{9}$$

Different regression analysis algorithms have been developed to find the correlation between parameters on a dataset, depending on the nature of their relation. In our problem formulation, we have employed and experimented with four different models, (a) linear model using mean squared error, (b) polynomial model of 2^{nd} degree, (c) polynomial model of 3^{rd} degree and (d) random forests with 6 sub-trees.

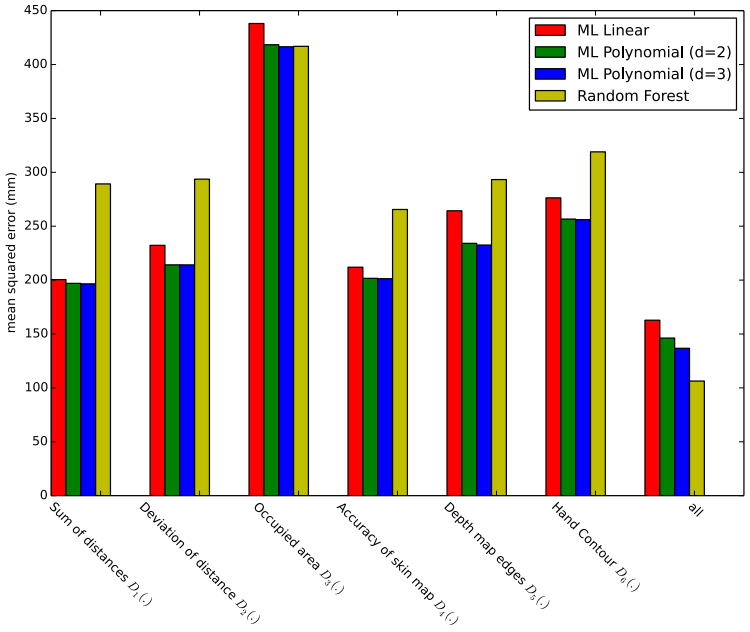

Fig. 2: Performance of models when trained on dataset that contained only one feature, compared to performance when trained on all features simultaneously.

5 Experimental Evaluation

The experimental evaluation of the proposed approach was performed based on both synthetic and real data. All experiments ran on a computer equipped with a quad-core intel i7 930 CPU, 16 GBs RAM and the Nvidia GTX 580 GPU with 1581 GFlops processing power and 1.5 GBs memory.

5.1 Evaluation on synthetic data

We evaluated the performance of tracking on a synthetic data set that was created using recorded real-life hand motion. Having such a dataset, we evaluate the performance of a tracker by measuring the error between the ground truth and the solution proposed by the tracker using the $\Delta(.)$ function in Eq. (8).

Regression models performance: We first evaluate the performance of each feature $D_i(.)$ depending on the employed regression model. For this test we trained the four proposed models (Section 4.3) using only one feature discrepancy function $D_i(.)$ and using simultaneously all six $D_i(.)$ functions. In all cases the models were trained on the same training dataset and were evaluated on the same ground truth dataset. The PSO was configured to 64 particles and 25 generations. Figure 2 illustrates the results of

this test. Note that the combination of all features produced better results than using any single feature. Using only one feature resulted in almost the same accuracy for all $D_i(.)$ functions except for $D_3(.)$ which showed reduced performance. It is interesting that the *random forest* algorithm had different performance compared to other models when using one $D_i(.)$ function or all six combined. Specifically, it was significantly outperformed by all other closed form algorithms when using only one feature. Still, it had better performance than all other algorithms when using all $D_i(.)$ functions together.

Tracking Performance: We evaluated different configurations for the regression model, the training dataset and the PSO algorithm. More specifically with respect to the regression model we considered (a) a linear model (b) 2^{nd} degree polynomial and (c) random forests. For the training dataset we employed one with 4096 poses using quasi-random sampling and a second one generated on previous tracking logs. PSO ran with 5, 10, 15, 20, 25 generations and 8, 32, 64 particles. Due to the stochastic nature of PSO, for each configuration, the test was run 20 times and the mean error was considered. The error was measured in *mm* using the $\Delta(.)$ function Eq. 8.

Figure 3 (left) shows tracking accuracy for the training dataset generated by *quasi-random* sampling. The baseline method has 10.7*mm* minimum error which is the best for all cases. However, a simple linear method with no prior knowledge of the problem complexity is only 8*mm* worse than the *baseline method*. The polynomial method has an error of 28.0*mm*. The explanation for this is that the polynomial function was over-fitted on the dataset which had more samples at long distances as explained in Section 4.2. This led to poor generalization at small distances which are extensively searched by the PSO. Finally, the *Random Forests* algorithm had the worst performance with 100.4*mm* minimum error and 1500*mm* maximum error. *Random Forests* make no assumption of the modelled space which makes them a bad predictor for areas where no training samples exist.

Figure 3 (right) shows the same test but using the training dataset generated from a previous tracking log. The major difference with Figure 3 is that all regression models have improved performance on low PSO budget. For higher PSO budget configurations, optimization performance varies in relation to the regression model used. More specifically, both the linear and the polynomial regression models perform worse than th previous datasets but now polynomial performs better that the liner model. In contrast, *Random Forests* have improved performance. This is expected as this dataset includes examples of poses that are not that far apart, so *Random Forests* managed to learn the behaviour of the objective function in that area of the pose space.

The results of this test show that the proposed method is sustainable and can, to some degree, replace the manual procedure of designing an objective function. At the same time, it has also been shown that the generation of a training dataset plays a key role on performance and should not be overlooked.

6 Discussion

In this work, we acknowledge the importance of the structure of the objective function on the optimization problem and the difficulties one has to face in order to construct

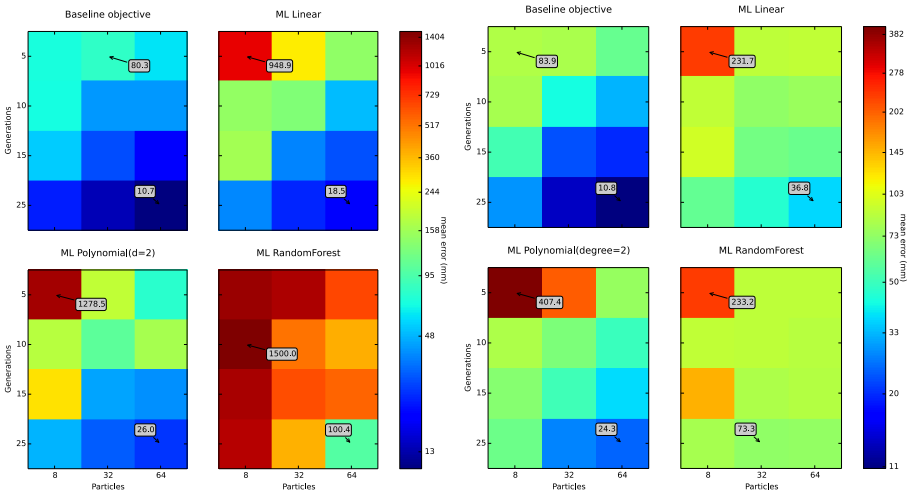

Fig. 3: Tracking performance tested on the ground truth "cooljason" dataset. Models were trained on dataset generated by 4096 quasi-random poses (left) and on the tracking logs using the objective function (right).

a function that performs well within the problem. The goal of this work was to find a systematic, automated method to construct such an objective function without a deep knowledge of the problem at hand. To apply and evaluate our method, different regression algorithms were tested on two different training dataset generation techniques. We evaluated the influence of several feature comparison functions $D_i(.)$ isolated against each tested regression model. The experiments showed that all $D_i(.)$ functions could approach the solution but the performance of their combination was by far the best. Another interesting result is that the Random Forests algorithm used for regression analysis was more effective in multi-dimensional space than any of the other methods. In the last step of evaluation, we measured the performance of hand-tracking by replacing the baseline objective function with the automatically estimated objective function $E_ml(.)$. For this experiment we tested multiple configurations of the PSO algorithm, regression models and training dataset generation approaches. It has been verified that none of the configuration profiles managed to outperform the baseline objective function but many profiles approached competitive results. This is quite important, especially considering the reduced demands of expertise on the problem of hand tracking. Finally, we should mention that this method is not constrained to the problem of hand-tracking but can be used in any optimization problem that depends on complex objective functions.

Acknowledgements

This work was partially supported by the EU project FP7-IP- 288533 Robohow. The contributions of Iason Oikonomidis and Nikolaos Kyriazis, members of FORTH/CVRL, are gratefully acknowledged.

References

1. Irene Albrecht, Jörg Haber, and Hans-Peter Seidel. Construction and animation of anatomically based human hand models. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 98–109. Eurographics Association, 2003.
2. Antonis A Argyros and Manolis IA Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Computer Vision-ECCV 2004*, pages 368–379. Springer, 2004.
3. John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.
4. Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 3d shape regression for real-time facial animation. *ACM Trans. Graph.*, 32(4):41, 2013.
5. Martin de La Gorce, Nikos Paragios, and David J Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference On*, pages 1–8. IEEE, 2008.
6. 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(12):52 – 73, 2007. Special Issue on Vision for Human-Computer Interaction.
7. Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.
8. Henning Hamer, Konrad Schindler, Esther Koller-Meier, and Luc Van Gool. Tracking a hand manipulating an object. In *Computer Vision, 2009 IEEE 12th International Conference On*, pages 1475–1482. IEEE, 2009.
9. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4. IEEE, November 1995.
10. N. Kyriazis, I. Oikonomidis, and A. Argyros. A gpu-powered computational framework for efficient 3d model-based vision. Technical Report TR420, ICS-FORTH, July 2011.
11. Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.
12. I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC 2011*. BMVA, 2011.
13. Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Markerless and efficient 26-dof hand pose recovery. In *Computer Vision–ACCV 2010*, pages 744–757. Springer, 2011.
14. Javier Romero, H Kjellstrom, and Danica Kragic. Monocular real-time 3d articulated hand pose estimation. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 87–92. IEEE, 2009.
15. 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. *Communications of the ACM*, 56(1):116–124, 2013.
16. Ilya M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.
17. Ying Wu and Thomas S Huang. View-independent recognition of hand postures. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 88–94. IEEE, 2000.
18. Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 532–539. IEEE, 2013.