Hand pose estimation from a single RGB-D image

Alina Kuznetsova and Bodo Rosenhahn *

Institute for Information Processing (TNT),
Leibniz University Hanover, Germany
{kuznetso,rosenhahn}@tnt.uni-hannover.de

Abstract. Hand pose estimation is an important task in areas such as human computer interaction (HCI), sign language recognition and robotics. Due to the high variability in hand appearance and many degrees of freedom (DoFs) of the hand, hand pose estimation and tracking is very challenging, and different sources of data and methods are used to solve this problem. In the paper, we propose a method for model-based full DoF hand pose estimation from a single RGB-D image. The main advantage of the proposed method is that no prior manual initialization is required and only very general assumptions about the hand pose are made. Therefore, this method can be used for hand pose estimation from a single RGB-D image, as an initialization step for subsequent tracking, or for tracking recovery.

1 Introduction

Precise hand pose estimation and tracking is an important task in areas such as HCI, sign language recognition and robotics. Different data sources and methods are used to solve this task. One of the possible settings is to use recently introduced consumer range cameras that produce both color(RGB) and depth(D) data streams. Although the use of such devices makes the hand pose estimation task much more feasible, it is still unsolved, both because of general problems of the hand pose estimation and drawbacks of consumer RGB-D cameras.

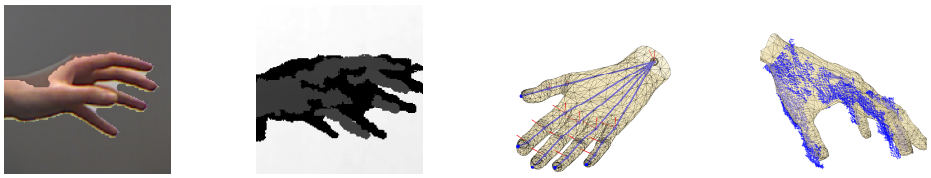
In general, hand pose estimation is very challenging due to the many degrees of freedom (DoFs) of the hand as an articulated object, which leads to great variability in hand appearance and self-occlusions.

The main drawbacks of the available RGB-D cameras are low resolution and missing range data.

There are three main groups of methods used to estimate the hand pose: template-based methods, model-based methods and machine learning methods, as well as different combinations.

* This work has been partially funded by the ERC within the starting grant Dynamic MinVIP

In our work, we concentrate on a model-based hand pose estimation method combined with feature detection. We use both color and depth images to overcome the problem of missing depth data. As a result, our method is able to correctly determine the hand pose from a single RGB-D image.



(a) (b) (c) (d)

Fig. 1: Initial data (1a — color image, overlaid with hand silhouette, 1b—depth data), hand model (1c), fitted hand and point cloud in 3D(1d).

2 Related works

All existing methods for hand pose estimation can be divided into three rough groups: template-based methods, model-based methods and machine learning methods [1]. There exist, of course, different combinations of these methods. Each group of methods has its own strengths and weaknesses.

Template-based methods. Template-based methods are usually applied in a single or multiple image setting [2]. To apply these methods, one firstly creates a database of possible hand poses and then uses this database to find a pose closest to the input image. The work for shape matching was recently extended to incorporate depth data in [3]. There are several limitations of this approach: firstly, this approach can only be used to distinguish between a limited number of poses and the discrimination power decreases with the increasing number of poses; and secondly, it requires the creation of a database that strongly depends on experimental setup.

Model-based methods. Recently, more works appeared on model-based approach for hand pose estimation and tracking. In these methods, a general model of a hand, parametrized by continuous parameters, is fitted to an image, based on the evaluation of similarity between the model and the image. In case of a single image, one often constraints the number of parameters to a subset of the 26 full DoF [4], or strong assumptions about the image are made [5, 6]. Several types of clues are used to evaluate the model, such as edges, optical flow [7], silhouettes, shading [8], color gloves [9], etc. The main disadvantages of model-based methods are that they are not real-time, not stable due to the high number of degrees of freedom and occlusions, and require prior initialization. It is also challenging to apply these methods in case of missing data.

Machine learning methods. Machine learning methods have been rarely applied to hand tracking due to the high variability in hand appearance and difficulty to distinguish different hand parts and find stable features [10]. However, due to the appearance of the RGB-D sensors, machine learning methods can be

used for hand pose estimation more efficiently [11]. It is still difficult to apply them though, due to the shortcomings of the data delivered by depth sensors.

In our work, we concentrate on model-based approach for hand pose estimation and propose to overcome several disadvantages mentioned above with a method that:

- determines the hand pose from a single image;
- does not require any initialization or model fitting;
- partially overcomes the problem of missing data;
- can be easily extended to the tracking setting.

This paper has the following structure. In Section 3, we present the data and explain the main challenges of hand pose estimation. In Section 4, we explain the main steps of the algorithm. In Section 5, we evaluate our approach both on real and synthetic data. We conclude our work in Section 6 with a discussion of the advantages and the disadvantages of our approach and the possible extensions.

3 Experimental setting and data

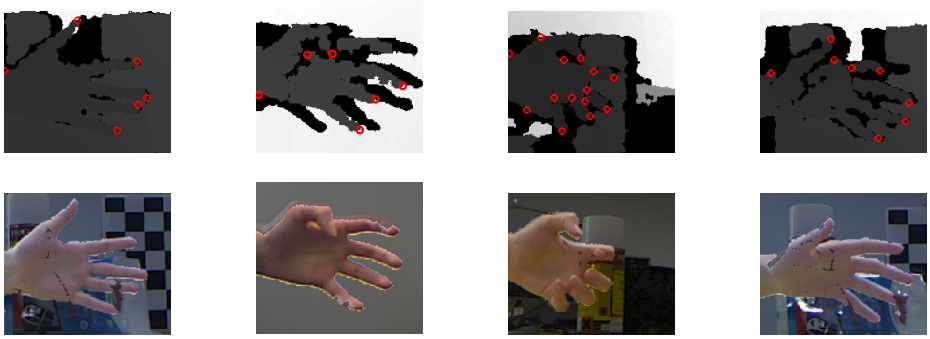


Fig. 2: Several examples of the RGB-D data; black areas of the depth images denote missing depth data; color images are overlaid with the extracted silhouette; initial finger tips detections are shown as red points.

For our work, we use the Microsoft Kinect sensor. This sensor delivers depth and color data at VGA resolution. Depth data is available at a distance of 0.6m-4m.

The data itself is hard to work with due to noise and missing depth data (see Fig. 2). Missing depth data occurs because of the two reasons: self occlusions prevent Kinect to acquire the correct depth values for some parts of the hand and the fingers themselves are hard to capture, because they are thin objects.

Because of the problems mentioned above, only a single depth frame alone can not be used for hand pose estimation. Therefore, we use RGB data to create hand silhouettes using background subtraction, so that we are able to partially recover from missing depth data. On the other hand, as mentioned in Section 2,

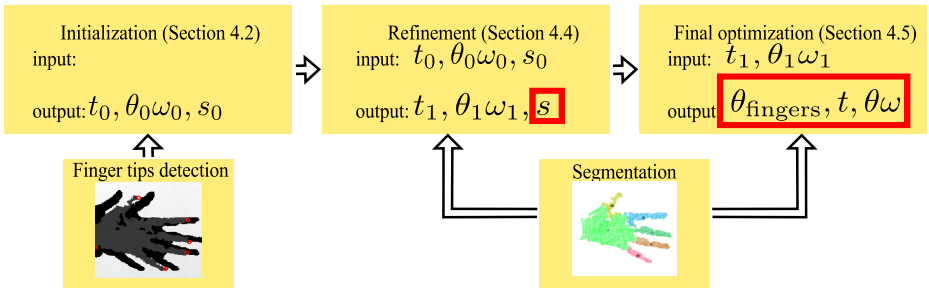


Fig. 3: The work-flow of the algorithm; the final output parameters are marked in red.

single color images are usually not enough for hand pose estimation, and many ambiguities can be resolved by using depth data.

We make general assumptions about the hand in the image — we assume that it is a left hand, rotated so that the palm is at least partially visible.

4 Hand pose estimation algorithm

Our algorithm consists of a number of steps (see Fig 3). We describe each step of the algorithm in more details in the following sections.

Hand model. We use a 26 DoF hand model, that consists of a 3D mesh of medium resolution of $N_m = 1317$ vertices and a skeleton connected to the model using the standard skinning technique [12]. The hand pose is defined by 20 joint angles $\boldsymbol{\theta}_{\text{fingers}}$ between the bones. The bone transformation is parametrized using exponential mapping [13], which is typically used for model-based human pose estimation [14]. Additionally, 6 parameters $\boldsymbol{t}, \theta\boldsymbol{\omega} \in \mathbb{R}^3$ encode position and rotation of the hand. Here $\theta\boldsymbol{\omega}$ is the rotation by the angle θ around the axes $\boldsymbol{\omega}$.

4.1 Initialization step

For initialization, we detect finger tips using geodesic extrema (see [15]). Geodesic extrema are computed by propagating distance from one point on the point cloud to the other points and then taking the maximum. Note, that the detected points are most likely not the positions of the actual finger tips (see Fig 2). We use these points to determine an approximate hand rotation and scaling. Since we do not know correspondences between the detected finger tips and those of the model fingers, we check all possible matches. For each match, we find the corresponding rotation and translation of the model, and evaluate the error between the model and the detections as the average Euclidean distance between corresponding points. We then select the match that delivers the smallest cost. An example of initialization result is given in Fig 5.

4.2 Point cloud segmentation using region growing

For pose estimation, we use an ICP-based optimization technique [16]. It is well known that such technique is prone to false matches. Therefore, we pre-segment our point cloud to differentiate between fingers and palm parts.

We use region growing based segmentation on the depth images: a pixel is included into a current region if the two distance measures, (1) and (2), are smaller then the corresponding thresholds. We define empirical thresholds at the 10mm for the Euclidean distance (1) and 0.8750 for the scalar product (2) between normals.

$$d_{eucl}(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{3} (x_i - y_i)^2 \right)^{\frac{1}{2}}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3 \ , \tag{1}$$

$$d_{norm}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{n_x}, \boldsymbol{n_y} \rangle \ . \tag{2}$$

Here $\boldsymbol{n_x}$ and $\boldsymbol{n_y}$ are the normals to the point cloud at the corresponding points $\boldsymbol{x}$ and $\boldsymbol{y}$. To compute the normals we use principal component analysis (PCA) as in [17].

After initial segmentation we merge small regions and divide large regions. The segmentation results are presented in Fig 4.

(a) (b) (c) (d)

Fig. 4: Segmentation results for the real data produced by Kinect (4a and 4b) and the artificial dataset [18] (4c and 4d).

After the segmentation, we determine the palm region as the largest region, and the rest of the points belong to the fingers. The point indices of the point cloud, corresponding to the palm, are denoted by I_{palm}, whereas the indices, corresponding to the i-th finger, are denoted by I^i_{finger}; the union of all non-palm points is denoted by I_{fingers}.

4.3 Hand position and size refinement

As mentioned in Subsection 4.1, the initial hand pose and scaling estimation can be inaccurate. Therefore, we firstly refine these parameters using an ICP technique. The parameter vector has seven dimensions: $(\boldsymbol{t}, \theta\boldsymbol{\omega}, s)$, where $\boldsymbol{t}$ and $\theta\boldsymbol{\omega}$ define hand position and rotation, and s defines scaling.

We then optimize the functional:

$$E(\boldsymbol{t}, \theta\boldsymbol{\omega}, s) = E_{\text{palm}} + \alpha E_{\text{fingers}} + \beta E_{2D} \ . \tag{3}$$

Here E_{palm} is the error term responsible for the distance between the point cloud region, identified as palm, and the palm of the model.

$$E_{\text{palm}} = \frac{1}{|I_{\text{palm}}|} \sum_{i \in I_{palm}} (\boldsymbol{p}_i^{\text{pc}} - \boldsymbol{p}_{n(i)}^m)^2, \quad \boldsymbol{p}_i^{\text{pc}}, \boldsymbol{p}_{n(i)}^m \in \mathbb{R}^3 \ . \tag{4}$$

Here, for each point $\boldsymbol{p}_i^{\text{pc}}$ of the point cloud we search for the nearest palm point $\boldsymbol{p}_{n(i)}^m$ in the hand model. We pre-build a kd-tree [19] on the data points to accelerate point search.

The E_{fingers} term is responsible for the distance between non-palm points in the point cloud and non-palm points of the model:

$$E_{\text{fingers}} = \frac{1}{|I_{\text{fingers}}|} \sum_{i \in I_{\text{fingers}}} (\boldsymbol{p}_i^{\text{pc}} - \boldsymbol{p}_{n(i)}^m)^2, \quad \boldsymbol{p}_i^{\text{pc}}, \boldsymbol{p}_{n(i)}^m \in \mathbb{R}^3 \ . \tag{5}$$

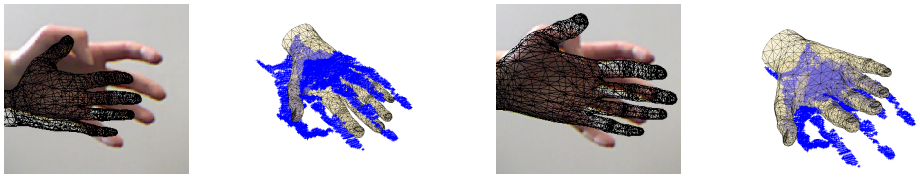
Here $\boldsymbol{p}_{n(i)}^m$ is the nearest non-palm point from the model. Finally, the E_{2D} term defines the distance between the 2D silhouette of the hand, and the projection of the model:

$$E_{2D} = \frac{1}{N_{\text{sill}}} \sum_i (\boldsymbol{p}_i^{\text{sill}} - \Pr(\boldsymbol{p}^m)_{n(i)})^2 + \frac{1}{N_m} \sum_j (\boldsymbol{p}_{n(j)}^{\text{sill}} - \Pr(\boldsymbol{p}^m)_j)^2, \tag{6}$$

$$\boldsymbol{p}_i^{\text{sill}}, \Pr(\boldsymbol{p}^m)_j \in \mathbb{R}^2 \ . \tag{7}$$

Here $\boldsymbol{p}_i^{\text{sill}}$ is a 2D point of the silhouette with N_{sill} points, $\Pr(\boldsymbol{p}^m)_{n(i)}$ is the nearest projected model point, where $\Pr(\boldsymbol{p})$ is the perspective projection operator with the default Kinect calibration parameters.

Since the whole functional can be represented as a sum of squares, we use the trust region approach for non-linear optimization (see, for example, [20]). Of course, the minimum found is not garanteed to be the global minimum, and therefore the fit is not perfect. We fix the weight from (3) $\alpha = 0.1$ and $\beta = 1.3$ for the real data. After this stage, we get a much better fit between the hand model and the point cloud (see Fig 5).



(a) (b) (c) (d)

Fig. 5: Hand fitting initialization (5a and 5b) and refined hand pose and size (5c and 5d). The final hand pose estimation follows in the last step.

4.4 Final hand pose estimation

In this stage, we do the final hand pose estimation. On the previous step, we estimated hand pose and size, but the fingers parameters $\boldsymbol{\theta}_{\text{fingers}}$ are still unknown. The two steps are separated because otherwise dependencies between

the parameters and the error functional become more complex. Moreover, initialization described in Subsection 4.1 is not good enough to avoid false matches. We now use a variation of a non-rigid ICP algorithm to fit the pose of our model to the point cloud and the silhouette.

As mentioned above, every ICP algorithm is prone to false matching. It is especially hard problem for such a highly articulated object as the hand, since fingers tend to be matched incorrectly. Therefore, prior to optimization, we match different parts of the model to different parts of the point cloud. For matching each part of the point cloud to a part of the model, we define a cost for each match as Euclidean distance between the centers of two segments. Each model segment can only be matched to zero or one point cloud segment. The problem can be mathematically written as follows:

$$\mu = \text{argmin}_\mu \sum_i d_{\text{eucl}}(\boldsymbol{c}_i^{\text{pc}}, \boldsymbol{c}_{\mu(i)}^m), \quad \mu(i) \in \{1, \ldots, 5\} \ . \tag{8}$$

where $\boldsymbol{c}_i^{\text{pc}}$ is the center of the i-th region of the point cloud and $\boldsymbol{c}_{\mu(i)}^m$ is the center of the $\mu(i)$-th region of the model. Note, that palm region is already known, so it is excluded from the matching procedure. The solution μ is found using brute force search, since the number of possible combinations is low.

The full pose of the hand is parametrized by 26 parameters: $(\boldsymbol{\theta}_{\text{fingers}}, \boldsymbol{t}, \theta\boldsymbol{\omega})$, where $\boldsymbol{t}, \theta\boldsymbol{\omega}$ are defined in the previous section, and $\boldsymbol{\theta}_{\text{fingers}}$ is a 20-dimensional vector of joint angles defining fingers' position. We also add natural constraints on the $\boldsymbol{\theta}_{\text{fingers}}$ parameters.

To find the parameter values, we optimize the following functional:

$$E(\boldsymbol{\theta}_{\text{fingers}}, \boldsymbol{t}, \theta\boldsymbol{\omega}) = E_{\text{palm}} + \left(\sum_i E_{\text{fingers}}^i \right) + \zeta E_{2D} \ . \tag{9}$$

The first and the last term have exactly the same meaning as in the previous step. The additional term can be described as follows:

$$E_{\text{fingers}}^i = \frac{1}{|I_{\text{finger}}^i|} \sum_{j \in I_{\text{finger}}^i} (\boldsymbol{p}_j^{\text{pc}} - \boldsymbol{p}_{n(j,\mu(i))}^m)^2 \ . \tag{10}$$

Here $n(j, \mu(i))$ is the closest model point to the point j, coming from $\mu(i)$ region of the model and $|I_{\text{finger}}^i|$ is the number of indices in the set I_{finger}^i. In this way, smaller finger regions gain more weight and have more influence in the optimization.

The last 2D term is weighted with the weight $\zeta = 1$ for the real data, which determines the importance of the silhouette.

We use the same algorithm for constrained non-linear optimization, as in the previous section. The solution of the optimization problem delivers the full pose and size estimation of the hand.

5 Evaluation

We evaluate our results both on the real data and the synthetic data (consisting of depth data and the corresponding silhouette).

Experiments with the synthetic data. We evaluated our algorithm on the synthetic data used for evaluation in [18]. Since we are not doing tracking, it would be unfair to compare the results we achieved directly with the numbers provided in [18]. Note, that the model used to produce synthetic data in this case differs significantly from our model in shape and proportions.

Since for synthetic data joint positions are known, we measured the distance between the joint positions of the fitted model and the ground truth joint positions. We evaluate the error after each step of our method to show its impact.

In Fig 6, mean error for each hand part is represented. One can see, that after the hand position and scale refinement, fingers error becomes larger. The explanation for this is simple — the position refinement gives more weight to the palm, and therefore finger error can actually become greater. As expected, after the final fit the error is the smallest then on the two previous steps.

Note, that our estimation accuracy is close to the one reported in [20] (in 74% the estimated pose deviation 40mm or less from the ground truth), even though we estimate the pose from a single frame instead of tracking and the model we use differs from the one used for creating the articial data signicantly.

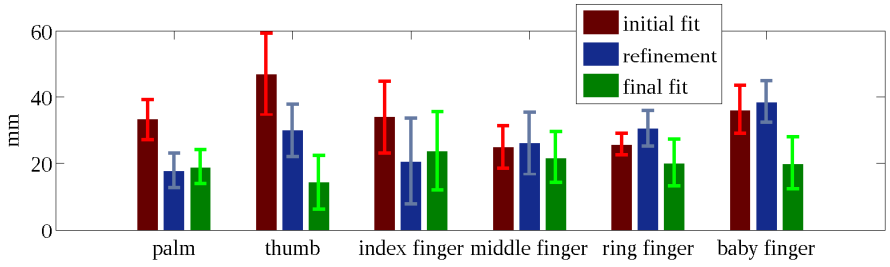


Fig. 6: Mean error of the six hand parts: palm, thumb, and four fingers; error bars show standard deviation; $\alpha = 0.1, \beta = 1.3, \zeta = 2$; the palm error is measured as the mean error of metacarpophalangeal joints and palm origin; the finger error is measured as the mean error of all joints of the finger.

In general, we could observe that the changes of the parameters $\alpha \in [0.1, 2]$ and $\zeta \in [0.1, 2]$ in the corresponding intervals do not affect the average results significantly. However, changes in β have significant influence on the results. For example, in case of $\beta = 5$ for the artificial data, there is almost no difference in error between the initial and the final fit. We attribute it to the fact, that the β parameter is crucial for determining the correct hand scaling.

Experiments with the real data. For these experiments, we used the data recorded by the Kinect sensor. In Fig 7, the results of the full fitting procedure are shown. In general, the pose of the hand is fitted correctly. Such estimation is enough for gesture recognition and also for tracking initialization. But one can see that the match is not perfect. We attribute it to several factors. Firstly,

there is a mismatch between hand form and proportions, and the real hand. Secondly, the scaling parameter is critical for the following pose estimations, so small mistakes in the scaling parameter directly lead to errors during fitting.

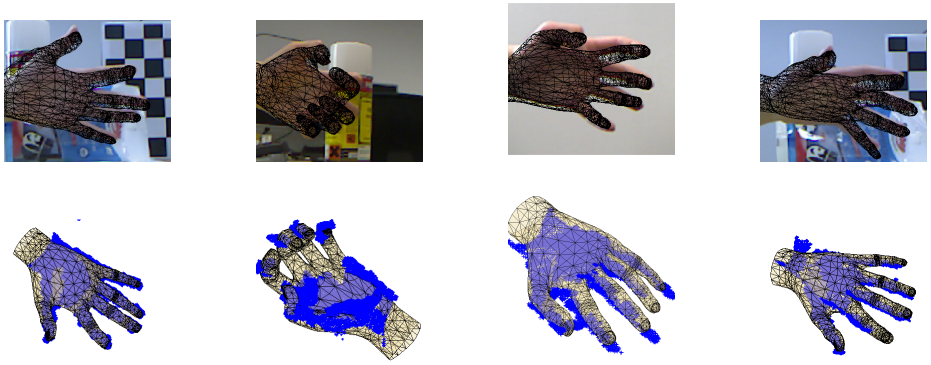


Fig. 7: Several model fitting results on the images from the database; the upper row shows color image with fitting results; the bottom row shows point cloud to model fit in 3D; the weights used are $\alpha = 0.1, \beta = 5, \zeta = 1$

Our algorithm performs good in case of an open hand and a partially closed hand (see Fig 7). However, we observed that in case of a fully closed hand our algorithm fails to estimate hand pose correctly, unless provided with initial pose, that is close to the pose on the image. We believe, that this problem can be potentially solved by learning an approximate pose from an image prior to optimization.

6 Conclusion and future work

We presented an approach for model-based hand tracking, based on the classical non-rigid ICP algorithm for combined RGB-D data. We evaluated our approach both on the synthetic and the real-world data and showed, that it is capable of producing plausible hand pose estimations. Our approach can be used both for one-shot hand pose estimation and extended for tracking. Our approach allows to partially overcome the problem of missing data.

In future, we will focus on improving the speed and also on providing a more stable initialization. As we use local optimization methods, our approach is prone to get stuck in a local minima, so we would like to extend it to usage of global optimization methods. Another possible direction could be to couple it with machine learning or template-based matching, since they are proven to be helpful for hand pose recognition.

References

1. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. Comput. Vis. Image Underst. **108** (2007) 52–73

2. Stenger, B.: Template-based hand pose recognition using multiple cues. In: In Asian Conference on Computer Vision. (2006) 551–560
3. Doliotis, P., Athitsos, V., Kosmopoulos, D.I., Perantonis, S.J.: Hand shape and 3d pose estimation using depth data from a single cluttered frame. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Fowlkes, C., Wang, S., Choi, M.H., Mantler, S., Schulze, J.P., Acevedo, D., Mueller, K., Papka, M.E., eds.: International Symposium on Visual Computing (ISVC), Springer, Springer (2012)
4. Stenger, B., Mendona, P.R.S., Cipolla, R.: Model-based 3d tracking of an articulated hand. In: CVPR (2). (2001) 310–315
5. Bray, M., Koller-Meier, E., Mueller, P., Gool, L.V., Schraudolph, N.N.: 3d hand tracking by rapid stochastic gradient descent using a skinning model. In: 1st European conference on visual media production (CVMP). (2004) 59–68
6. de La Gorce, M., Fleet, D.J., Paragios, N.: Model-based 3d hand pose estimation from monocular video. IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011) 1793–1805
7. Ballan, L., Taneja, A., Gall, J., Gool, L.V., Pollefeys, M.: Motion capture of hands in action using discriminative salient points. In: European Conference on Computer Vision (ECCV), Firenze (2012)
8. de La Gorce, M., Paragios, N., Fleet, D.J.: Model-based hand tracking with texture, shading and self-occlusions. In: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA, IEEE Computer Society (2008)
9. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. ACM Trans. Graph. **28** (2009) 63:1–63:8
10. Stenger, B., Thayananthan, A., Torr, P., Cipolla, R.: Hand pose estimation using hierarchical detection. In: in Intl. Workshop on Human-Computer Interaction, Springer (2004) 102–112
11. Keskin, C., Kiraç, F., Kara, Y.E., Akarun, L.: Real time hand pose estimation using depth sensors. In: ICCV Workshops. (2011) 1228–1234
12. Jacka, D., Merry, B., Reid, A.: A comparison of linear skinning techniques for character animation. In: In Afrigraph, ACM (2007) 177–186
13. Murray, R.M., Sastry, S.S., Zexiang, L.: A Mathematical Introduction to Robotic Manipulation. 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1994)
14. Pons-Moll, G., Rosenhahn, B.: Model-Based Pose Estimation. Springer (2011)
15. Plagemann, C., Ganapathi, V., Koller, D., Thrun, S.: Real-time identification and localization of body parts from depth images. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on. (2010) 3108–3113
16. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14** (1992) 239–256
17. Rusu, R.B.: Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany (2009)
18. Iason Oikonomidis, N.K., Argyros, A.: Efficient model-based 3d tracking of hand articulations using kinect. In: Proceedings of the British Machine Vision Conference, BMVA Press (2011)
19. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18** (1975) 509–517
20. Coleman, T.F., Li, Y.: An interior trust region approach for nonlinear minimization subject to bounds. (1996)