Hand Shape and 3D Pose Estimation using Depth Data from a Single Cluttered Frame

Paul Doliotis[1,2], Vassilis Athitsos[1], Dimitrios Kosmopoulos[2], and Stavros Perantonis[2]

[1] Computer Science and Engineering Department
University of Texas at Arlington
Arlington,Texas
USA
[2] Institute of Informatics and Telecommunications
NCSR Demokritos
Athens, Greece

Abstract. This paper describes a method that, given an input image of a person signing a gesture in a cluttered scene, locates the gesturing arm, automatically detects and segments the hand and finally creates a ranked list of possible shape class, 3D pose orientation and full hand configuration parameters. The clutter-tolerant hand segmentation algorithm is based on depth data from a single image captured with a commercially available depth sensor, namely the $Kinect^{TM}$. Shape and 3D pose estimation is formulated as an image database retrieval method where given a segmented hand the best matches are extracted from a large database of synthetically generated hand images. Contrary to previous approaches this clutter-tolerant method is all-together: user-independent, automatically detects and segments the hand from a single image (no multi-view or motion cues employed) and provides estimation not only for the 3D pose orientation but also for the full hand articulation parameters. The performance of this approach is quantitatively and qualitatively evaluated on a dataset of real images of American Sign Language (ASL) handshapes.

1 Introduction

Gesture recognition has become an essential component for many natural user interface (NUI) systems. It provides humans the ability to interact with machines naturally without the use of any cumbersome mechanical devices. A gesture can be defined by any bodily motion, however hand gestures are more commonly used and can be found in a wide range of applications such as: sign language recognition, robot learning by demonstration and gaming environments, just to name a few. Recognizing hand gestures is a very challenging task and requires solving several sub-problems like automatic hand detection and segmentation, 3D hand pose estimation, hand shape classification and in some cases estimation of the full hand configuration parameters.

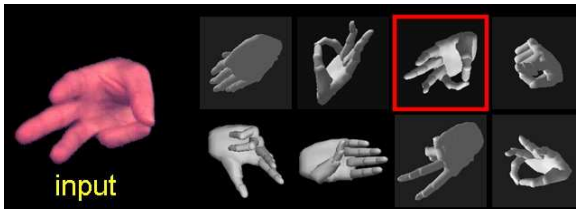

Fig. 1. System input and output. Given the input image, the system goes through the database of synthetic images in order to identify the ones that are the most similar to the input image. Eight examples of database images are shown here, and the most similar one is enclosed in a red square. The database currently used contains more than 100,000 images.

In this work we specifically address the problem of 3D hand pose and shape estimation. Towards developing an effective solution several challenges may arise and some of the main ones are listed bellow:

- High dimensionality of the problem
- Noisy hand segmentation due to cluttered backgrounds
- Increased pose variability and self-occlusions that frequently occur when a hand is in motion

Hand pose estimation is formulated here as an image database retrieval problem. The closest matches for an input hand image are retrieved from a large database of synthetic hand images. The ground truth labels of the retrieved matches are used as hand pose estimates from the input (Figure 1). This paper is motivated by the work presented in [1]. However, one limitation of that method was that it required manual segmentation in order to define a bounding box for the gesturing hand. We propose an automatic hand segmentation method that relies on depth data acquired from the Microsoft $Kinect^{TM}$ device [2]. Another contribution is that we achieve improved performance under clutter by using a similarity measure which is also based on the depth data. A main assumption we make is that the gesturing arm is the closest object to the camera and so it can easily be segmented from the rest of the body and other objects based on depth. To measure the effectiveness of this new method we have collected a dataset of American Sign Language(ASL) handshapes.

In the following Section 2 of this paper we briefly review previous work on 3D hand tracking and pose estimation. Afterwards, in Section 3 we describe in detail our automatic hand segmentation method, the result of which is given as input to our 3D hand pose estimation algorithm (Section 4). Finally we present experimental results (Section 5) and we conclude with a discussion on ideas for future-work (Section 6).

2 Related work

Some successful early works require specialized hardware or the use of cumbersome mechanical devices. In [3] Schneider and Stevens use a motion capture system while in [4] Wang and Popović employ visual markers with a color glove. Unfortunately such methods impede the user's natural interaction in the scene and they require a costly and complex experimental setup.
Nowadays research is more focused on purely vision-based methods that are non-invasive and are more suitable for Natural User Interface(NUI) systems. The most recent review on vision-based hand pose estimation methods has been published by Erol *et al.* [5]. They define a taxonomy where initially these approaches are divided in two main categories: "partial pose estimation" and "full DOF pose estimation". "Partial pose estimation" methods can be viewed as extensions of appearance-based systems. They usually take as input image features and map them a small discrete set of hand model kinematic parameters. A main disadvantage is that they require a large amount of training data and hence are not scalable. Appearance-based methods for hand pose recognition, like [6,7,8,9], can tolerate clutter, but they are limited to estimating 2D hand pose from a limited number of viewpoints. Our method can handle arbitrary viewpoints.

"Full DOF pose estimation" approaches are not limited to a small, discrete set of hand model configurations. They target all the kinematic parameters (i.e., joint angles, hand position or orientation) of the skeleton of the hand, leading to a full reconstruction of hand motion. These approaches can be further divided into two other categories: (1) "Model-based tracking" and (2) "Single frame pose estimation".

"Model-based methods" [10,11,12,13] typically match visual observations to instances of a predefined hand model. Formally this is expressed as an optimization problem where an objective function is required in order to measure similarity between actual visual observations and model hypotheses. The main drawback is increased computational complexity due to the high dimensionality of the model's parameter space. On the other hand they require less training and are easily scalable.

"Single frame pose estimation methods" try to solve the hand pose estimation problem without relying on temporal information. The lack of temporal information increases the difficulty of the problem. However successful approaches can tackle the negative effect of motion blur and can also be employed to initialize tracking-based systems. Athitsos *et al.* [1] have proposed such a single pose estimation method by creating a large database of synthetic hand poses using an articulated model and retrieve the best match from this database. However they require manual segmentation of the test data.

Most recently, due to the advent of commercially available depth sensors, there is an increased interest in methods relying on depth data [14,15,10,16]. Keskin *et al.* [14] train Random decision forests (RDF) on depth images and then use them to perform per pixel classification and assign each pixel a hand part. Then, they apply the mean shift algorithm to estimate the centers of hand parts

to form a hand skeleton. However they don't explicitly address the automatic hand segmentation problem.

According to the aforementioned taxonomy, this paper describes an "appearance-based method" aiming at 3D orientation estimation using features from a single frame. This work builds on top of the work described in [1], where hand pose is estimated from a single cluttered image. The key advantages of the method described here over [1] are that we integrate an automatic hand segmentation method and we use a similarity measure that is based on depth data.

3 Hand segmentation

As a first step we need to perform a rough segmentation by thresholding the depth data in order to obtain the gesturing arm. Given the assumption that the hand is the closest object to the camera we can automatically find the lower depth threshold. As an upper threshold we take an initial rough estimation, since at this point we are only interested at segmenting the arm. A more precise thresholding is needed however if we need to further segment the hand. To find the palm cutoff point we need to perform the following steps:

1. Compute the axis of elongation for the gesturing arm.
2. Create a sequence of widths.
3. Perform a gradient descent on the sequence of widths in order to identify the local (or global) minimum, at which the palm cutoff point is located.

3.1 Finding the axis of elongation

The result of the initial rough segmentation is a blob representing the gesturing arm. The boundary of that blob is essentially a set $\mathcal{S}$ of m points in 2 dimensional space: $\mathcal{S} = \{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^2$. Noisy smaller groups of pixels are usually part of $\mathcal{S}$. To remove them, we morphologically open the binary image by eliminating all connected components (objects) that have fewer than 20 pixels, considering an 8-connected neighborhood. The remaining boundary pixels will belong to a new set $\mathcal{S}' = \{x'_1, x'_2, \ldots, x'_k\} \in \mathbb{R}^2$. In order to define the elongation axis of the gesturing arm we will compute the *Minimum Enclosing Ellipsoid(MEE)* for the boundary pixels $x'_i \in \mathcal{S}'$. The major axis of the *MEE* coincides with the arm's axis of elongation (Figure 2).
An ellipsoid in center form can be given by the following equation:

$$\mathcal{E} = \left\{x' \in \mathbb{R}^2 | (x' - c)^T E (x' - c) \leq 1\right\} \tag{1}$$

where $c \in \mathbb{R}^2$ is the center of the ellipse $\mathcal{E}$ and E is a 2×2 positive definite symmetric matrix, $E \in \mathbb{S}^2_{++}$.
So finding the Minimum Enclosing Ellipsoid can be formulated as an optimization problem as follows:

$$\begin{aligned} \underset{E,c}{\text{minimize}} \quad & det(E^{-1}) \\ \text{subject to} \quad & (x_i' - c)^T E(x_i' - c) \leq 1,\ i = 1, \ldots, k \\ & E \succ 0 \end{aligned} \tag{2}$$

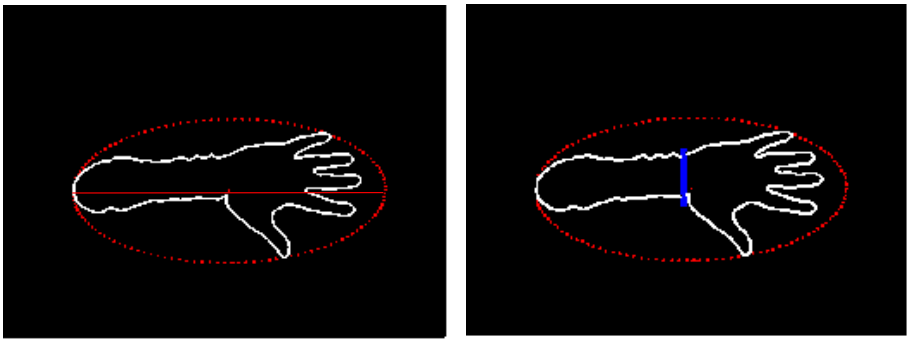

Fig. 2. To the left is an example of a *Minimum Enclosing Ellipsoid(MEE)* along with the major axis. To the right we visually demonstrate the desired palm cutoff location(blue line)

An implementation of a solver based on the Khachiyan Algorithm [17] can be found at the web [18]. The major axis for the arm boundary pixels will coincide with the major axis of the Minimum Enclosing Ellipsoid.

3.2 Creating the sequence of widths

After the major (or elongation) axis is obtained we can easily create a sequence of widths. In the discrete domain, the elongation axis is comprised of a set of pixels $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$. For each p_i we compute the maximum distance of arm pixels belonging to the line that goes through p_i and it's direction is perpendicular to the direction of the elongation axis. The main idea is that at the palm cutoff point the sequence will reach a global or local minimum. Since the contour of the segmented arm is very noisy our method could be prone to other local minima. To alleviate this effect we apply a smoothing on our 2D contour.

Smoothing 2D contours is achieved by using *Local Regression Lines*. Because of the linear nature of fitting it might be possible to loose important information in special cases like corners (or fingertips). To tackle this issue we opt to fit locally the line by employing Weighted Orthogonal Least Squares. The weights are generated from a Gaussian distribution. To be able to calculate the local regression lines we must define an order for the all pixels x_i' that $\in \mathcal{S}' = \{x_1', x_2', \ldots, x_m'\}$.

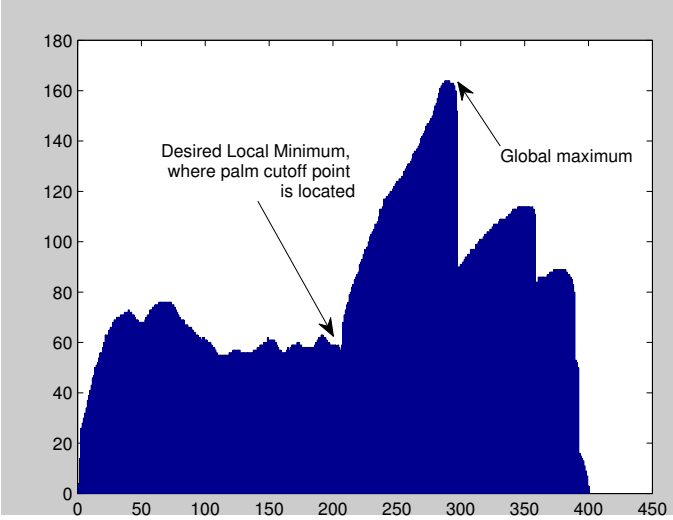

Fig. 3. This a plot of the sequence of widths. The desired local minimum is highlighted indicating the position for the palm cutoff point.

Such an order can be defined with many techniques such as boundary tracing and chain codes. An implementation of this smoothing technique can be found at the web [19].

3.3 Gradient descent

After the original contour is smoothed the sequence of widths is further filtered with a 1D horizontal mask of ones and of size 5. The next step is to perform a gradient descent in order to identify the local minimum at which the palm cutoff point will lie. As a starting point we choose the global maximum (i.e. the highest width) which will always reside in the hand area. Then we move towards the end of the arm until we reach our local minimum. In Figure 3 we can see a plot of the sequence of widths along with the desired local minimum where the palm cutoff point is located.

4 Framework for Hand Pose Estimation

We model the hand as an articulated object, consisting of 16 links: the palm and 15 links corresponding to finger parts. Each finger has three links (Figure 4). There are 15 joints, that have a total of 20 degrees of freedom (DOFs). For the 20-dimensional vector of joint angles we use synonymously the terms “hand shape” and “hand configuration.”

The appearance of a hand shape also depends on the camera parameters. For simplicity, we consider only the camera viewing direction (two DOFs), and image

plane orientation. We use the terms "camera parameters," "viewing parameters" and "3D orientation" synonymously to denote the three-dimensional vector describing viewing direction and camera orientation. Given a hand configuration vector $C_h = (c_1, ..., c_{20})$ and a viewing parameter vector $V_h = (v_1, v_2, v_3)$, we define the hand pose vector P_h to be the 23-dimensional concatenation of C_h and V_h: $P_h = (c_1, ..., c_{20}, v_1, v_2, v_3)$.

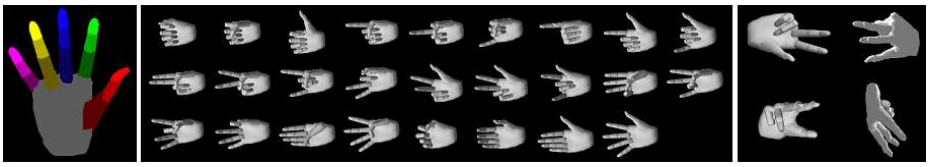

Fig. 4. Synthetic images of hands. Left: the articulated hand model. The palm and 15 finger links are shown in different colors. Middle: the 20 basic shapes used to generate model images in our database. Right: four 3D orientations of the same hand shape.

Using these definitions, our framework for hand pose estimation can be summarized as follows:

1. Preprocessing step: create a database containing a uniform sampling of all possible views of the hand shapes that we want to recognize. Label each view with the hand pose parameters that generated it.
2. Given an input image, retrieve the database views that are the most similar. Use the parameters of the most similar views as estimates for the image. The most similar views (Figure 1) are retrieved according to a similarity measure (e.g. Euclidean distance, chamfer distance)

4.1 Database

Our database contains right-hand images of 20 hand shape prototypes (Figure 4). Each prototype is rendered from 86 different viewpoints (Figure 4), sampled approximately uniformly from the surface of the viewing sphere. The rendering is done using a hand model and a computer graphics software ([20]). To accommodate rotation-variant similarity measures (like the chamfer distance), 48 images are generated from each viewpoint, corresponding to 48 uniformly sampled rotations of the image plane. Overall, the database includes 4128 views of each hand shape prototype and 82,560 images overall. We refer to those images using the terms "database images," "model images," or "synthetic images."

4.2 Similarity measures

Chamfer distance The chamfer distance [21] is a well-known method to measure the distance between two edge images. Edge images are represented as sets

of points, corresponding to edge pixel locations. The X-to-Y directed chamfer distance $c(X, Y)$ is defined as

$$c(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\| \ , \tag{3}$$

where $\|a - b\|$ denotes the Euclidean distance between two pixel locations a and b. The undirected chamfer distance $C(X, Y)$ is

$$C(X, Y) = c(X, Y) + c(Y, X) \ . \tag{4}$$

We will use the abbreviations DCD to stand for "directed chamfer distance" and UCD for "undirected chamfer distance."

Depth matching Test data have been captured via the $Kinect^{TM}$ device which offers two synchronized streams, one RGB and one depth stream. Frame resolution is 640×480. A depth image is a gray-scale image, where each pixel is assigned an intensity value according to how far or close it is located from the camera. We have also managed to create depth-maps for our synthetically generated database images using a 3D modeling and animation software [20]. Some examples of our depth-maps are depicted in Figure 5. Both, model and test depth images are normalized in order to achieve translation invariance for the z-axis. All depth values are in the range from 0 to 1. The depth similarity measure between two images is defined as the total sum of their pixel-wise Euclidean distances. Through the rest of this paper we will refer to the depth matching similarity measure as $depthSM$.

Weighted Depth matching and Chamfer distance Another similarity measure is defined by combining the first two similarity measures. In the following equation the $WeightedSM$ is a weighted sum of $depthSM$ and UCD:

$$WeightedSM = l_1 \times depthSM + l_2 \times UCD \tag{5}$$

In our experiments $l_1 = l_2 = 0.5$.

5 Experiments

We have tested our system on a challenging dataset of 94 right hand images of American Sign Language(ASL) handshapes. The images are captured with the $Kinect^{TM}$ device in a highly cluttered scene where the hand is not cleanly pre-segmented. We will refer to this dataset as *KinectHandshapes*. These 94 test images are provided in both formats, RGB and depth. We have manually established pseudo-ground truth for each test image, by labeling it with the corresponding shape prototype and using the rendering software to find the viewing parameters under which the shape prototype looked the most similar to

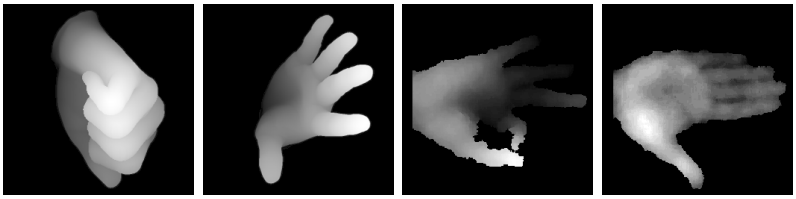

Fig. 5. The two depth-maps at the left side are "database depth-maps" and have been rendered with a 3D modeling software. The two depth-maps at the right side are "test depth-maps" and they have been captured by the $Kinect^{TM}$ device.

the test image. This way of estimating viewpoint parameters is not very exact; we found that manual estimates by different people varied by 10-30 degrees. Model views cannot be aligned perfectly because the anthropometric parameters (like finger lengths and widths) of hands in test images do not match those of the model, and because the hand shapes in the real images are not exact replications of the 26 shape prototypes.

We consider a database view V to be a *correct match* for a test image I if the shape prototype with which we label I is the one used in generating V, and the manually estimated viewing parameters of I are within 30 degrees of those of V ([22]). On average, there are 30.4 correct matches for each test image in the database. Our measure of retrieval accuracy for a given test image I is the rank of the *highest-ranking correct match* that was retrieved for I. 1 is the highest (best) possible rank.

Table 1. Results for KinectHandshapes dataset. For every method used, we show the percentage of test images for which the highest ranking correct match was within each range. DCD stands for "image-to-model directed chamfer distance". UCD depth contours, is undirected chamfer distance between hand contours from depth images and full edges from model images. In UCD color edges, skin color segmentation has been employed to extract the full edges

Method used	1	1-4	1-16	1-64	1-256
UCD depth contours	14.74	20.0	32.63	40.0	62.11
UCD color edges	9.57	14.89	21.28	32.98	55.32
depthSM	34.04	44.68	61.70	76.60	87.23
weightedSM	34.04	44.68	61.70	76.60	87.23

From the experiments on the *KinectHandshapes* dataset it is evident that using depth information from the depth-maps enhances the discrimination power of our method, even in highly uncontrolled and cluttered environments. It out-

performs even the case "UCD color edges" where full edges of the hand are available due to color information.

We have also evaluated quantitatively our automatic hand segmentation method. To this end we compared for all of our test images the distance between the automatically generated *palm cut-off points* and the manually specified ones. For the 80.85% of our images the distance (measured in pixels) is negligible. The distance is less than 15 pixels for the 85.11% and less than 25 for the 92.55% of the images (Table 2).

Table 2. Results for our Hand segmentation method

range (measured in pixels)	0	0-15	0-25	0-45	0-68
percentage of test images	80.85	85.11	92.55	96.81	100

6 Discussion and future work

We have presented a new method for hand pose estimation from a single depth image. Our method combines a novel hand segmentation method and a similarity measure ($depthSM$) based on depth information from depth images.The similarity measure is used to retrieve the best matches from an image database, thus providing estimates for the 3D hand pose and hand configuration. Depth information increases the discrimination power of our method, according to the experiments conducted. Retrieval accuracy is still too low for the system to be used as a stand-alone module for 3D hand pose estimation. However estimating hand pose from a single image can be useful in automatically initializing hand trackers. Our system currently doesn't achieve real-time performance. In order to do so and since our method is inherently parallel, we are planning to take advantage of the GPU's processing power. Additional future work will be to define more sophisticated similarity measures further exploiting depth information.

Acknowledgments. This work was partially supported by NSF grants IIS-0812601, IIS-1055062, CNS-1059235, ECCS-1128296, CNS-0923494, and CNS-1035913

References

1. Athitsos, V., Sclaroff, S.: Estimating 3d hand pose from a cluttered image. In: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. Volume 2. (2003) II–432–9 vol.2
2. Microsoft Corp. Redmond WA.: Kinect Xbox 360. (`http://www.xbox.com/kinect`)

3. Schneider, M., Stevens, C.: Development and testing of a new magnetic-tracking device for image guidance. Proceedings of SPIE **7035** (2007) 65090I–65090I–11
4. Wang, R.Y., Popović, J.: Real-time hand-tracking with a color glove. ACM Trans. Graph. **28** (2009) 63:1–63:8
5. Erol, A., Bebis, G., Nicolescu, M., Boyle, R.D., Twombly, X.: Vision-based hand pose estimation: A review. Computer Vision and Image Understanding **108** (2007) 52–73
6. Moghaddam, B., Pentland, A.: Probabilistic visual learning for object detection. Technical Report 326, MIT (1995)
7. Triesch, J., von der Malsburg, C.: Robotic gesture recognition. In: Gesture Workshop. (1997) 233–244
8. Freeman, W., Roth, M.: Computer vision for computer games. In: Automatic Face and Gesture Recognition. (1996) 100–105
9. Wu, Y., Huang, T.: View-independent recognition of hand postures. Volume 2. (2000) 88–94
10. Oikonomidis, I., Kyriazis, N., Argyros, A.: Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In: Computer Vision (ICCV), 2011 IEEE International Conference on. (2011) 2088–2095
11. 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
12. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Markerless and Efficient 26-DOF Hand Pose Recovery. In: Asian Conference on Computer Vision, Queenstown, New Zealand (2010)
13. Rehg, J., Kanade, T.: Model-based tracking of self-occluding articulated objects. Computer Vision, IEEE International Conference on **0** (1995) 612
14. Keskin, C., Kiraç, F., Kara, Y.E., Akarun, L.: Real time hand pose estimation using depth sensors. In: ICCV Workshops. (2011) 1228–1234
15. Pugeault, N., Bowden, R.: Spelling it out: Real-time asl fingerspelling recognition. In: ICCV Workshops. (2011) 1114–1119
16. Mo, Z., Neumann, U.: Real-time hand pose recognition using low-resolution depth images. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Volume 2 CVPR06 **2** (2006) 1499–1505
17. Khachiyan, L.G.: Rounding of polytopes in the real number model of computation. Math. Oper. Res. **21** (1996) 307–320
18. Minimum Volume Enclosing Ellipsoid: Matlab Central. (`http://www.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid`)
19. Smoothing 2D Contours Using Local Regression Lines: Matlab Central. (`http://www.mathworks.com/matlabcentral/fileexchange/30793-smoothing-2d-contours-using-local-regression-lines`)
20. Smith Micro, Aliso Viejo, CA: Poser 8. (`http://poser.smithmicro.com/poser.html`)
21. Barrow, H.G., Tenenbaum, J.M., Bolles, R.C., Wolf, H.C.: Parametric correspondence and chamfer matching: two new techniques for image matching. In: Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2. IJCAI'77, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1977) 659–663
22. Athitsos, V., Sclaroff, S.: An appearance-based framework for 3D hand shape classification and camera viewpoint estimation. In: Automatic Face and Gesture Recognition. (2002)