Conference and Exhibition of the European Association of Virtual and Augmented Reality (2014)
G. Zachmann, J. Perret, and A. Amditis (Editors)

New Techniques for Hand Pose Estimation Based on Kinect Depth Data

Simon Hummel, Victor Häfner, Polina Häfner, and Jivka Ovtcharova

Karlsruhe Institute for Technology, Germany

Abstract

In this paper we present new techniques for extension and improvement of approaches to markerless 3D hand pose estimation: A new algorithm for finger segmentation in point clouds of hands is presented which makes use of the narrow and flexible shape of the fingers. The kinematic of a 3D hand model gets aligned to the geodesic paths of the fingers which provides a very natural configuration of the model. Therefore a new technique for optimization of those geodesic paths is introduced. Another benefit of this system is, that it's only data resource is the depth stream of a Kinect. So no additional hardware, markers or training data is needed. A first implementation of our approach provides a proof for the new concepts and looks promising for further investigation.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

1. Introduction and Related Work

Using hands is one of the most important types for interaction with our environment. Therefore it is very natural - and increases immersion - to use hands for interaction in virtual environments as well. For this kind of interaction the hands of the user have to be captured and their configurations have to be evaluated - this process is called hand pose estimation.

In general the hand pose is defined by the global pose of the hand and the local configuration of the fingers. As a first step the approximate position of the hand has to be detected. A common approach to this is to assume that the hand is the closest object to the camera [LYTZ13, RYZ11]. A more general and correct detection and tracking of the hand is a difficult problem itself [TSA12, JWC14]. Approaches to optical hand pose estimation can be divided into two types: Systems using markers and markerless systems. This work focuses on the second type since this is more natural and therefore increases immersion. In those markerless approaches data is gathered directly (color, depth or both) by one or more sensors. They can be separated into appearance based and model based . In appearance based approaches, usually a function is trained using labelled samples. This function maps recorded features of an image to a certain hand pose. Romero et al. [RKK10] propose a system which matches the color image of each frame with a large database of labelled hand poses. This makes the system robust to occlusions and independent of model parameters. On the other side such systems need a big database which is expensive in creation and hand poses maybe ambiguous due to the non-linearity of the mapping [LYTZ13]. Model based approaches usually generate a 3D hand model which is fitted into the features extracted from data obtained by a camera. For example Oikonomidis et al. [OKA11] assume a joint parametric hand-object model using 3D geometric primitives and then minimize an objective function over this model using color data. Its main drawback is the use of multiple cameras which makes the system quite immobile. Ballan et al. [BTG12] make use of discriminatively learned salient points on the fingers. They estimate the hand pose by minimizing an objective function that also takes into account edges, optical flow and collisions. Their implementation works well for strong occlusions, but since it uses color based features it depends on illumination. On the contrary Liang et al. [LYTZ13] make use of depth data obtained from a Kinect camera. Their proposed framework basically consists of a hand parser, a fingertip detector and an inverse kinematic solver. The hand parser assigns a label

DOI: 10.2312/eurovr.20141355

- namely the corresponding finger/hand part - to each pixel in the depth image. This is done by comparing a 3D hand model - whose pose was configured to the estimated pose in the last frame - with the depth data of the current frame. Fingertip detection is done by finding points with maximum geodesic distance to the palm center. The IK solver can reconstruct the hand pose in the current frame.

This work follows the approach of markerless hand pose estimation using only depth data obtained by a Kinect. It is model based and also ties in with the idea of finding geodesic maxima in a point cloud. Among others new approaches to finger segmentation in point clouds and to local configuration of the hand model are presented.

2. Hand Pose Estimation

Our system uses depth data of a Microsoft Kinect Xbox 360 and rough hand positions of a given hand tracking system as input. An overview of the system can be found in Figure 1. Single processing steps are described in the following sections. Development and evaluation was performed on desk-

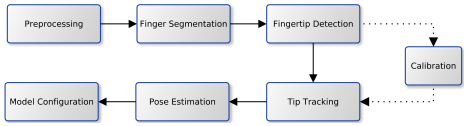


Figure 1: Overview of the hand pose estimation system.

top computer with a quad-core 2.8 GHz processor and 8 GB RAM. The Kinect is placed next to the screen and slightly tilted upwards, towards the user.

2.1. Preprocessing

At first a rectangular cropping is performed around the hand in the depth image. This is done by using an axis aligned rectangle which is centered at the position given by a hand tracking system. Then distant points in the region of interest (ROI) not belonging to the hand are filtered out using a depth threshold. After that a binary image *Bin* from the ROI is created. The next step is applying a distance transform to *Bin* - the point with the largest distance to a non-hand-point is defined as palm center. Unprojecting this point in 3D space, the 3D palm center can be determined. Kinect data is noisy itself and can get even noisier in certain environments. Since erroneous data is treated as zero this can lead to small wholes in *Bin*. This again has a quite big influence on the result of the distance transform since a zero point in *Bin* reduces the distance for all surrounding points. Therefore the morphological closing operation is applied to *Bin* before the distance transform is conducted. So small holes which distort the result of the distance transform get filled. Depending on the current pose of the hand and the current camera perspective there are possibly still parts of the forearm in *Bin* (as seen in Figure 2a). To determine and remove these parts, the Principal Component Analysis of *Bin* is utilized: The first eigenvector is computed and its origin is set to the found palm center. Next this vector is inverted and the point of intersection **s** with the circle centered at the palm position with radius r is computed, where r is the result of the distance transform at the palm center. Then the tangent to the circle on point **s** is calculated and all points in the area behind the tangent can be set to zero. This process is illustrated in Figure 2b. The set of points which have been set to zero is marked as grey area. The position of the carpus is set to the intersection point. Unprojecting this point in 3D space, we determine the 3D carpus position (green point in Figure 2c).

(a) (b) (c)

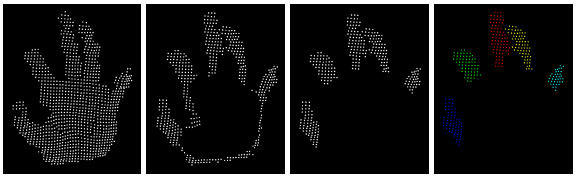
Figure 2: Steps of forearm cropping and carpus detection in a binary image of a hand.

2.2. Finger Segmentation

In the finger segmentation process the point cloud gets separated into points of the single fingers and points of the hand palm. The main differences between fingers and palm are that the fingers form a narrower region and that they are flexible. Therefore finger segmentation is basically done by a three-step-procedure:

1. Filter the hand cloud by the number of neighbours in a certain radius.
2. Filter the resulting cloud by the distance of each point to the palm center and the plane described by the palm.
3. Conduct euclidean clustering on the filtered cloud.

In the first step we iterate through the hand point cloud (see Figure 3a) and filter out each point whose number of neighbours in a fixed radius is lower than the average number of neighbours. This is already a rough approximation of the segmentation but still contains points of the palm (see Figure 3b). Now points whose distances to the palm center or to the plane in which the palm lays are over a threshold get extracted and defined as finger points (see Figure 3c). In the last step an euclidean clustering is applied to finger points to get the point clusters of the single fingers. An exemplary result of the segmentation process is illustrated in Figure 3. As one can see the result is not optimal but since only the distal parts of the fingers (containing the finger tips, cf. Section 2.3) are needed this approximation is precise enough for our purposes.



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

Figure 3: Steps of the finger segmentation process.

2.3. Fingertip Detection

Inspired by Liang et al. [LYT12] fingertip detection is performed by searching for geodesic maxima in the point cloud P of the hand. This idea was introduced in a more general context by Plagemann et al. [PGKT10]. We assume that the fingertip point (for each finger) is the point which has the highest geodesic distance from the palm center.

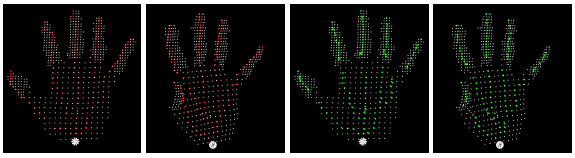
Therefore a Graph $G = \{V, E\}$ is build where V consist of all points in P. Then the k nearest neighbours $\mathbf{v}_0..\mathbf{v}_7$ of each $\mathbf{v} \in V$ are determined and an Edge $e = (\mathbf{v}_i, \mathbf{v})$ is added to E for every neighbour if the euclidean distance $d(v_i, v)$ is under a threshold τ. Furthermore a weight $\beta = d(\mathbf{v}_i, \mathbf{v})$ is assigned to every added edge. Since the resulting graph is not necessarily connected an union-find algorithm is applied to the graph for finding all connected components $\{K_0..K_k\}$. Then the connected component K_c containing the carpus position $\mathbf{c}$ is identified by iterating through the graph and comparing point indices of the components. Until now we basically followed the algorithm of Liang et al. [LYT12] but in the following we can make use of the previously found finger clusters $C_0..C_n$. We add the shortest possible edge from K_c to every finger cluster C_i. These edges can be found by finding the nearest neighbour of every point $p \in K_c$ in each cluster C_i. We now have a set of n potential connecting edges for every cluster, where n is the size of K_c. Next the shortest edge in each set is selected and it is added to E and weighted with its length. As a result every cluster is reachable from $\mathbf{c}$ in the graph. The next step is to locate the geodesic maximum (and thus the fingertip) from $\mathbf{c}$ in each finger cluster C_i. Therefore the geodesic distance from the palm center $\mathbf{c}$ to every vertex $v \in C_i \cup C_p$ is computed using Dijkstra's shortest path algorithm. The vertex with the highest geodesic distance is selected as a fingertip candidate $\mathbf{t}_i$. Since the accuracy of the path is only important for the fingers (and not for the palm), downsampling can be performed to C_p before performing Dijkstra's shortest path algorithm to improve performance. Finally we reject those fingertips whose distance to $\mathbf{c}$ is under a certain threshold, since we assume they arise from erroneously found clusters in Section 2.2.

As shown in Figure 5a, the paths corresponding to the found finger tips constitute a rough approximation of the real skeleton of the hand. Therefore we will use it for the estimation of our local hand configuration (see Section 2.6). To improve this approximation we apply a "centering" to each path: We create a k-d tree for the corresponding finger cluster and perform a range search with radius 1 cm for each point in the path. Thus we get the n neighbouring points N from each path point $\mathbf{p}_i$ in the cluster. $\mathbf{p}_i$ gets now replaced by the centroid of N. In Figure 4 this process is illustrated with a simplified 2D example: The yellow points are the points of the cluster, red points are points of the path. The big circles constitute the radii of the range search. In Figure 5 one can see

(a) (b)

Figure 4: 2D example path before (a) and after (b) centering.

the results of tip detection and path optimization on two real frames.



(a) Raw paths. (b) Paths after centering.

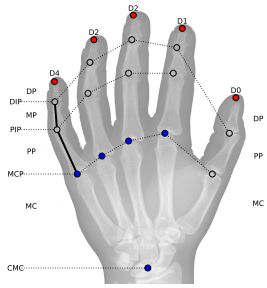
Figure 5: Geodesic paths and their optimizations.

2.4. The Hand Model

The design of the hand model is similar to real hand anatomy: Fingers D1 to D4 consist of the four joints CMC, MCP, PIP and DIP (see Figure 6). The thumb consists of CMC, MCP and DIP. The distance between the joints is defined by the lengths of the corresponding phalanges. For simplification it is assumed that the hand is rigid in area of palm and wrist. Thus the model can be reduced to use a single CMC joint for all fingers, which is the base root joint for the whole model. The eight carpal bones are not modelled. The MCP joints are the root joints for the kinematic chain of the fingers. Since the MC of the thumb is also flexible, the CMC is used as root joint for the thumbs kinematic chain. The fingertips are the end points of the kinematic chains. An illustration of the model can be found in Figure 6.

According to Buryanov and Kotiuk [BK10] there are typical relations of the lengths of the phalanges within each finger. Therefore it is possible to compute the length of each phalanx using these ratios and the finger lengths obtained in the calibration step.

According to the joint angles stated by Aristidou and Lasenby [AL10] we assume DIP of all fingers to have 1 DoF, PIP of D1 to D4 have 1 DoF, MCP of D1 to D4 have 2 DoF, MCP of D0 has 1 DoF, CMC of D1 to D4 have 0 DoF and MCM of D0 has 2 DoF. So there are 20 DoF in the finger joints. In addition to the 6 DoF of the global pose of the



Bones: Distal Phalanx (DP), Medial Phalanx (MP), Proximal Phalanx (PP), Metacarpal (MC)
Joints: Distal Interphaleangeal Joint (DIP), Proximal Interphaleangeal Joint (PIP), Metacarpal Phaleangeal Joint (MCP), Carpo Metacarpal Joint (CMC)

Figure 6: Hand model with joints and bones.

model (3 translational and 3 orientational). Thus the model has 26 DoF overall.

2.5. Model Calibration

Before proceeding with global and local pose estimation, the system has to be calibrated at least once. In particular the calibration of the system comprises an initial detection of the MCP positions, measurement of finger lengths and initial computation of the hand pose. The calibration process assumes that the whole hand is visible, fingers are fully stretched and there is at least a small distance between all fingers. Currently the calibration process is started manually by the user.

MCP Detection: First the convex hull of the contours of our binary hand image (red lines in Figure 7) is computed. Next all convexity defects of the contour, i.e. all parts of the contour which do not have a convex shape are located. In each defect area the point with the largest distance to the convex hull is taken. These points are marked blue in Figure 7. Now the points get successively connected, except the longest distance crossing the palm. The centers of the connecting line segments are stored as MCP positions. To get the MCP position for the thumb an additional line, connecting the defect point between thumb and index with its next point of the convex hull, is added. The resulting MCP positions are marked pink in Figure 7. As one can see the convexity

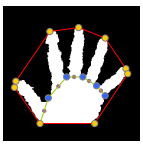


Figure 7: Detection of MCP joints.

defect between thumb and index differs from the other ones. Thus the MCP position of the index finger is corrected by shifting it in the direction of the corresponding corner of the convex hull. Unprojecting the MCP positions in the image into 3D space, 3D MCP positions can be determined.

Finger Lengths: Since we assume that fingers are fully stretched, finger lengths $l_0..l_4$ can be determined by computing the distances between the previously detected MCP positions $m_0..m_4$ and corresponding finger tips $t_0..t_4$ as detected in Section 2.3.

Initial Hand Pose: The global hand pose P is determined by its position $\mathbf{p}$ and its orientation. The position is set to the palm center as computed in Section 2.1. The orientation is defined as a direction vector $\vec{d}$ and an up vector $\vec{u}$. We assume that $\vec{u}$ is the normal of the hand, defined as a vector perpendicular to the surface of the hand palm. Based on the hand palm position it has to point to the back of the hand. $\vec{d}$ is defined as the normalized direction from carpus $\mathbf{c}$ (as determined in Section 2.1) to $\mathbf{p}$.

2.6. Model Configuration

Model Initializing: After the calibration phase the hand model gets initialized. For this purpose the base root of the model is set to the position of the initial hand pose. The root joints of finger D1 to D4 are set to the MCP positions detected during calibration. The remaining joints are not yet set in the initializing phase.

Global Pose Estimation: As stated in Section 2.5, the global pose P of the hand consists of position $\mathbf{p}$, direction $\vec{d}$ and up vector $\vec{u}$. The initial pose is determined in the calibration phase. Since the hand palm is assumed to be rigid, the global pose at frame$_t$ is estimated based on the global pose at frame$_{t-1}$ and the transformation (translation and rotation) describing the movement of the hand palm from frame$_{t-1}$ to frame$_t$. This transformation is estimated using the Iterative Closest Points (ICP) algorithm.

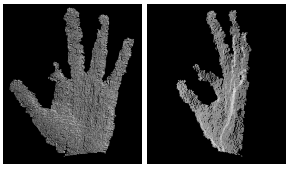
Now the model has to be fitted to the global pose and the fingertips resp. their geodesic paths, found in Section 2.3. Fitting the model to the global pose means that the base root and the root joints of the model are transformed by the transformation found for global pose estimation. Fitting the model to the geodesic paths of the fingertips is done by performing a kinematic alignment of the fingers of the model to these paths.

Kinematic Alignment: As suggested above the model has the following kinematic chains:

- MCP - PIP - DIP - Fingertip (fingers D1 to D4)
- CMC - MCP - DIP - Fingertip (finger D0)

The main idea of kinematic alignment is to align each kinematic chain ($\mathbf{p}_0..\mathbf{p}_k$) to the corresponding geodesic path found in Section 2.3. The alignment goes joint by joint, beginning at the first non root joint. For each of these joints a

point on the path has to be found whose distance to the previous joint is approximately equal to the length of the phalanx we are currently aligning. This point can be found by iterating through the path, beginning at the tip point, and comparing the distance of each point with the phalanxs lengths. Actually this distance does possibly not correspond exactly to the length of the phalanx, but this inaccuracy is minimal enough (usually < 1 mm) to ignore it. Since the Kinect uses structured light for retrieving depth information, a certain surface area is needed to project the IR pattern on. If this area is too small, depth data cannot be resolved. This can be the case for very small objects or plane surfaces parallel to the IR light beams. During our experiments we especially noted missing depth data when a finger is pointed directly to the device. In Figure 8 a sample frame where some depth data of a finger is missing can be found. As a consequence,



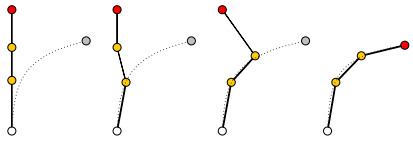
(a) Front view. (b) Side view.

Figure 8: Example for missing depth data of small structures.

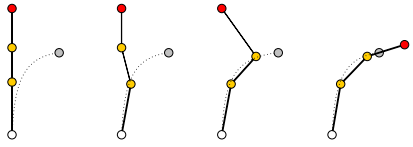
it is possible that the found tip is not the real fingertip but lies anywhere on the finger and therefore the corresponding geodesic path is shorter than the actual finger. Thus we have to distinguish between the following cases:

1. Path length $\geq$ length of kinematic chain.
2. Path length < length of kinematic chain.
   a. One remaining nodes.
   b. Multiple remaining nodes.

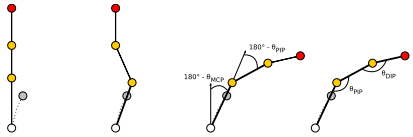
A 2D illustration of these cases can be found in Figure 9. Case 1 is the regular one and we can proceed as mentioned above. Formally we can compute the new position of each (non root) node $\mathbf{p}_i = \mathbf{p}_{i-1} + l \cdot \vec{d}_i$, where l is the length of the phalanx modelled by $\mathbf{p}_{i-1}$ and $\mathbf{p}_i$ and $\vec{d}_i$ is the directional vector of this phalanx: $\vec{d}_i = \frac{\mathbf{q}_k - \mathbf{p}_{i-1}}{|\mathbf{q}_k - \mathbf{p}_{i-1}|}$ where $\mathbf{q}_k$ is the point on the path whose distance to the previous joint is equal to the length of the phalanx. In case 2 this only works as long as the path length is greater than the partial kinematic chain $(\mathbf{p}_0..\mathbf{p}_i)$. If that is not the case the next link of the kinematic chain is aligned to the end of the path: $\vec{d}_i = \frac{\mathbf{q}_0 - \mathbf{p}_{i-1}}{|\mathbf{q}_0 - \mathbf{p}_{i-1}|}$ If there are now still nodes of the kinematic chain left (case 2b) we can not proceed as before, since there is no more information about the real path. So the finger configuration has to be estimated without a measurement of the distal part of the finger. Typically the remaining nodes correspond to the PIP and DIP joints. According to Lin et al. [LWH00] the bending of these joints is not independent: If the DIP joint is bent by an angle θ_{DIP}, the PIP joint must also be bent by an angle θ_{PIP}. The relation between the two angles can be approximated by $\theta_{DIP} = \frac{2}{3} \cdot \theta_{PIP}$. This relation does not hold for θ_{MCP} and θ_{PIP} in all types of gestures. Due to the lack of a better approximation we assume only natural grasping gestures for which this ratio should approximately be the same. $\vec{d}_i = \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{|\mathbf{p}_i - \mathbf{p}_{i-1}|} \cdot R$ where R is a rotation matrix which rotates a vector around axis $\vec{a}$ with angle θ. $\vec{a}$ is the vector perpendicular to the plane described by the direction of node $\mathbf{p}_{i-1}$ and the direction from $\mathbf{p}_{i-1}$ to the current value of $\mathbf{p}_i$: $\vec{a} = \vec{d}_{i-1} \times (\mathbf{p}_i - \mathbf{p}_{i-1})$ According to the angular relations mentioned above θ is $\frac{2}{3}$ of the angle between the two vectors describing this plane: $\theta = \frac{2}{3} \cdot \frac{\vec{d}_{i-1} \cdot (\mathbf{p}_i - \mathbf{p}_{i-1})}{|\vec{d}_{i-1}| \cdot |(\mathbf{p}_i - \mathbf{p}_{i-1})|}$ In Figure 9 the kinematic alignment of a simplified 2D example is illustrated for each case: The white point denotes the root joint of the kinematic chain. Yellow points are regular joints and the red point denotes the tip. The dotted line is the geodesic path. The alignment processes joint by joint from left to right. Figure 9a illustrates case 1, Figure 9b illustrates case 2a and Figure 9c illustrates case 2b. In Figure 10a



(a) Path length $\geq$ length of kinematic chain.



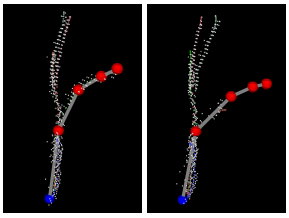
(b) Path length < length of kinematic chain with one remaining joint.



(c) Path length < length of kinematic chain with multiple remaining joints.

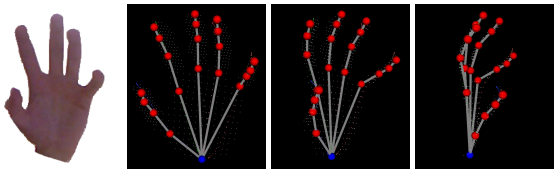
Figure 9: Different cases of the kinematic alignment.

a real example for regular alignment of the model of a finger to the corresponding geodesic path can be seen. Figure 10b shows an example of using estimated angles as described above. In Figure 11 a fully fitted model of a real frame can be found. To get a more realistic result (w.r.t human hand anatomy), the angular ranges of the finger joints, as found by Aristidou and Lasenby [AL10], can be incorporated. As proposed by Aristidou and Lasenby [AL11] this can be done by mapping the new joint position $\mathbf{t}$ to the region of possible positions. This region is defined by the angular constraints



(a) (b)

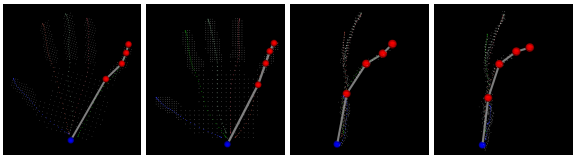
Figure 10: Aligning bent index using only the geodesic path (a) and using estimated angles (b).



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

Figure 11: Color image and views of the fitted hand model.

of each finger joint in left/right and up/down direction and is generally shaped like an irregular cone. The new constrained joint position **t'** can be found by mapping **t** to this cone. In Figure 12 a real world example of aligning pinky - with and without using angular constraints - can be found.



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

Figure 12: Aligning bent pinky to geodesic path with (b, d) and without (a, c) using constraints in front and side view.

3. Conclusion

This work presents a new approach to hand pose estimation for natural 3D interaction, in immersive environments. It builds on previously known approaches like finding geodesic maxima for fingertip detection and presents new techniques for finger segmentation in point clouds and for the kinematic alignment of an 3D hand model. A main benefit of this work is that its implementation only uses depth data of a Kinect as a single resource. So there is no need for additional devices, markers or expensive training material for machine learning algorithms and illumination is no issue.

Our implementation provides a first proof for the new concepts. Running on the hardware setup given in Section 2 it performs a frame rate of 15 to 20 frames per second. The main drawback currently is the low frame rate which leads to erroneous result on fast movements. Another disadvantage is the manual starting of the calibration process. For the future optimizations of algorithms and implementation and a quantitative evaluation of the results are planned. Another Future work is porting the algorithms to work on a GPU.

Acknowledgements

This work was developed in the scope of the project 3DConFu which is funded by the Central Innovation Program SME.

References

[AL10] ARISTIDOU A., LASENBY J.: Motion capture with constrained inverse kinematics for real-time hand tracking. *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, March (2010), 3–5. 3, 5

[AL11] ARISTIDOU A., LASENBY J.: FABRIK: A fast, iterative solver for the Inverse Kinematics problem. *Graphical Models 73*, 5 (Sept. 2011), 243–260. doi:10.1016/j.gmod.2011.05.003. 5

[BK10] BURYANOV A., KOTIUK V.: Proportions of Hand Segments. *Int. J. Morphol 28*, 3 (2010), 755–758. 3

[BTG12] BALLAN L., TANEJA A., GALL J.: Motion capture of hands in action using discriminative salient points. *European Conference on Computer Vision (ECCV)* (2012), 640– 653. 1

[JWC14] JOO S.-I., WEON S.-H., CHOI H.: Real-Time Depth-Based Hand Detection and Tracking. *The Scientific World Journal 2014* (2014), 1–17. doi:10.1155/2014/284827. 1

[LWH00] LIN J., WU Y., HUANG T. S.: Modeling the constraints of human hand motion. In *Workshop on Human Motion* (Los Alamitos, CA, 2000), IEEE Comput. Soc, pp. 121–126. doi:10.1109/HUMO.2000.897381. 5

[LYT12] LIANG H., YUAN J., THALMANN D.: 3D fingertip and palm tracking in depth image sequences. *Proceedings of the 20th ACM international conference on Multimedia (MM)* (2012), 785. doi:10.1145/2393347.2396312. 3

[LYTZ13] LIANG H., YUAN J., THALMANN D., ZHANG Z.: Model-based hand pose estimation via spatial-temporal hand parsing and 3D fingertip localization. *The Visual Computer 29*, 6-8 (May 2013), 837–848. doi:10.1007/s00371-013-0822-4. 1

[OKA11] OIKONOMIDIS I., KYRIAZIS N., ARGYROS A.: Full DOF tracking of a hand interacting with an object by modeling occlusions and physical constraints. *International Conference on Computer Vision (ICCV)* (Nov. 2011), 2088–2095. doi:10.1109/ICCV.2011.6126483. 1

[PGKT10] PLAGEMANN C., GANAPATHI V., KOLLER D., THRUN S.: Real-time identification and localization of body parts from depth images. In *IEEE International Conference on Robotics and Automation (ICRA)* (May 2010), IEEE, pp. 3108–3113. doi:10.1109/ROBOT.2010.5509559. 3

[RKK10] ROMERO J., KJELLSTRÖM H., KRAGIC D.: Hands in action: real-time 3D reconstruction of hands in interaction with objects. *IEEE International Conference on Robotics and Automation (ICRA)* (May 2010), 458–463. doi:10.1109/ROBOT.2010.5509753. 1

[RYZ11] REN Z., YUAN J., ZHANG Z.: Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. *Proceedings of the 19th ACM international conference on Multimedia (MM)* (2011), 1093. doi:10.1145/2072298.2071946. 1

[TSA12] TARA R., SANTOSA P., ADJI T.: Hand segmentation from depth image using anthropometric approach in natural interface development. *Int. J. Sci. Eng. Res 3*, 5 (2012), 1–5. 1