Real-Time Hand Model Estimation from Depth Images for Wearable Augmented Reality Glasses

Bill Zhou[*] Alex Yu[†] Joseph Menke[‡] Allen Y. Yang [§]

University of California, Berkeley

Abstract

This work presents a hand model estimation method designed specifically with augmented reality (AR) glasses and 3D AR interface in mind. The proposed work is capable of estimating the 3D positions of all ten finger from a single depth image. By leveraging a low-dimensional hand model and exploiting hand geometries from an ego-centric view, we build a lightweight algorithm that is accurate, environment agnostic, and runs in real time on mobile hardware. One major consideration in our design for AR is that the user‘s hand is likely to interact with planar surfaces since they serve as ideal touchscreens. As a result, our method will not fail to detect the hand even when the hand is in physical contact with a surface such as a table, wall, or even another palm. Our experiment shows using the CVAR database that the accuracy with clear background at 98% and with cluttered background at around 85%.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Mixed/augmented reality; Computing methodologies—Artificial intelligence—Computer vision—Computer vision problems

1 Introduction

Hand model estimation via computer vision is important in human-computer interaction and has been extensively studied [2, 11, 16]. However, it still remains a challenging problem, especially in the context of augmented reality, due to several reasons:

- **High Degrees of Freedom:** The human hand is an articulated deformable object with high degrees of freedom and complex finger movements.
- **Low-Latency:** Using hand models as an input modality in human-computer interface requires its estimation algorithms to be fast and low-latency, preferably to match those of keyboard and mouse input.
- **Light Weight:** Its particular usage in the emerging wearable augmented reality (AR) field often limits the available computing hardware to be only lightweight and low-power.
- **Robustness to Cluttered Background:** If the hand input is intended for mobile use, such as in vehicles or in factories, the solutions must work well under complex lighting conditions and cluttered background.
- **Robustness to Touch/Tactile Feedback:** In addition, we prefer a solution that is robust to the hand contacting physical surfaces since the most common utility of the human hand is touching physical surfaces in the world. Further, this provides tactile feedback to the human user compared to no feedback when performing gestures in mid air.

Despite recent advances in hand tracking, few solutions have been tailored to augmented reality applications and can adequately fulfill the above challenges. For example, much of the prior work has been focused on tracking the user‘s hand from a third person camera perspective with the assumption of little background interference [10, 14]. Some systems also use highly complex hand models to capture the hands motion which is very resource-intensive [8]. The work in [1] use a polygon model optimization but require a GPU to achieve real-time performance.

In this paper, we present a novel hand model estimation solution tailored specifically for head-mounted augmented reality glasses. First, in order for the solution to acquire accurate finger locations in 3D space under complex lighting conditions, we choose to use depth images from active depth cameras as our input (e.g., Intel RealSense and Microsoft Kinect).

Second, we employ a simpler hand model with 8 key points (five fingers, one palm center, and two wrist positions). We find that these 8 points are sufficient for most augmented reality applications. While many other hand tracking solutions use a full skeletal model, we have found that given the ego-centric vantage point, most of these skeletal points are not observable and hence are approximated with best guess generated based on probability. Most AR applications that we care about only need to know the 3D locations of the hand fingertips together with the palm, which are fully modeled and tracked in our solution.

In the proposed algorithm, we only track the 8 key points through a geometric parsing of the hand contour and a simple SVM model for false positive rejection. Despite its simplicity, our experiment shows that our hand tracking solution is very accurate and can be implemented in real time under the constraints of augmented reality use cases and hardware specifications.

Finally, our algorithm explicitly deals with modeling any dominant surfaces that the hands may contact with. The additional modeling of all possible contacting surfaces allows the application to ignore the surface structure and track hand gestures that interact with the physical surface. To this end, we have made our implementation of the algorithm available as an open source package online.

2 Related Works

Glove and Markers: Marker-based systems rely on reflective markers to be fixed to the hand directly or on gloves [12]. The 3D position of the markers is then estimated using a multi-camera setup which infers a full skeletal pose based on inverse kinematics. These methods require expensive and specialized hardware.

Multiple Views: Multiple cameras provide a means to overcome errors from various forms of occlusions. The work in [13] creates hand tracking from two cameras using a discriminative approach to quickly find the closest pose in a predefined database. [7] used 8 cameras in a studio setup to tracking both hands manipulating

[*] e-mail: billzhou@berkeley.edu

[†] e-mail: sxyu@berkeley.edu

[‡] e-mail: joemenke@berkeley.edu

[§] e-mail: yang@eecs.berkeley.edu

an object. However, these methods are not feasible within the hardware constraints of augmented reality. The head-mounted display must small and light. It is difficult to create two cameras that observe very different viewpoints on this type of form factor.

Single Depth Camera: The introduction of off-the-shelf depth sensors has led to an explosion of hand tracking methods that utilize depth information. [5] proposed a method for tracking hands directly in depth by physics simulation. [9] uses a multi-layered discriminative reinitialize strategy for per-frame pose estimation followed by a generative model fitting to recover the hand pose.

Random forests are widely used in full body tracking and a number of hand tracking methods adopt a similar strategy. [4] proposed a method of finger gestures from a monocular depth camera by training a decision forest. [3, 15] proposed supervised learning methods to regress over 3D hand poses. However, the problem with these methods that they require a large amount of training data and it is unclear how well they will perform for new users that may be out of distribution with the training data.

Commercial Products: In parallel, practitioners can adopt several hand tracking systems available in the market. Products such as Leap Motion and uSens Fingo leverages a similar approach in that they use a single active depth sensor. These products uses fully articulated skeletal hand and provide outstanding accuracy. However, they require the end user to purchase specialized hardware (instead of a generic depth camera). Further, the Leap Motion and uSens Fingo will both lose track if the hand physically contacts a surface.

Products such as the Oculus Touch and the HTC Vive Wand adopt the glove and markers approach. These systems have the lowest latency and highest accuracy. However, both rely on hand held devices and external sensors.

Finally, the Hololens and Hololens2 hand tracking functions are most similar to our solution, designed from the ground up for wearable AR devices. However, the functions will lose track of the hand if it contacts a surface. Furthermore, the implementations run on specially designed accelerated hardware, while our solution is an open-source software kit.

3 Method

Two major goals of our algorithm is to support interaction with virtual buttons augmented on physical surfaces and provide low latency on mobile hardware without sacrificing accuracy. To achieve this, we leverage modern depth sensors such as the Intel Realsense SR300. We construct a low-dimensional representation of the hand consisting of 8 key points rather than the traditional full skeletal model. We assume an ego-centric view which allows us to quickly eliminate unrealistic hand postures via simple rules and a lightweight SVM. Our method leverages simple geometric algorithms rather than kinematic model simulations or quadratic optimizations to reduce computation while maintaining high performance. An overview of the pipeline is shown in Figure 1.

The architecture of the algorithm is divided into three main stages: hand extraction, false positive rejection, and finger classification. During the hand extraction stage, we remove all background interference and extract all blobs that can be the hand. In the false positive rejection stage, we determine which candidate blobs can be the hand blob(s) using a lightweight SVM classifier trained on a small collection of manually engineered features. Finally, in the finger classification stage, we identify all 3D finger positions in the frame.

3.1 Background Subtraction

In order to track a hand that is in physical contact with a surface, we start by modeling all the planar surfaces in the scene.

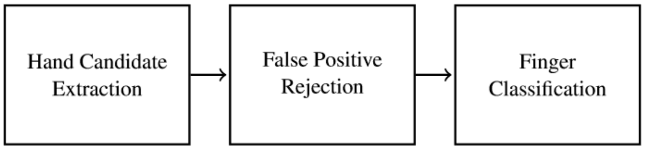

Figure 1: High level overview of our pipeline. We start with a raw depth image and classify fingers by their 3D positions.

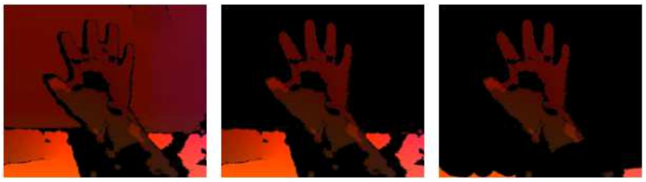

Figure 2: Pipeline to extract the hand cluster from background noise. We first removal all planar surfaces which the hand might contact with. Next, we remove all background noise and clutter.

Since we have access to an ordered point cloud, we choose to fit planes through region growing of similar normal vectors under cosine similarity on the depth image. We first compute the normal vector at each point by taking the cross product of two vectors each determined as an average of four neighbors. Next, we aggregate points with similar normal vectors to grow the plane. Finally, we fit a plane equation to all points with similar normal vectors using ordinary least squares. We find that region growing is a much faster and more reliable approach than RANSAC when an ordered point cloud is present yet the number of planes present is unknown.

Using the detected plane models, content creators can augment virtual buttons directly on top of these planar surfaces. When the user interacts with these buttons, the table top or wall surface will provide natural tactile feedback.

3.2 Hand Extraction

As shown in Figure 2. we first remove all the planes in the depth image. Otherwise, the hand cluster can merge with the planar surface if they are in contact. Next, we flood fill from the bottom of the frame to extract all possible hand blobs. Since we specially designed this solution for head-mounted devices, we assume an egocentric view which means the users wrist must intersect with a line near the bottom of the depth camera frame. This assumption allows the system to quickly discard all clusters (including other users hands) that are not near to the bottom edge. We iterate through voxels near the bottom of the frame and use a watershed floodfill to identify all clusters whose areas are between $0.01m^2$ and $0.02m^2$. Clusters that are not within this size range are discarded since they are unlikely to represent a hand.

From the cluster we compute the contour. The remaining computations are performed on the contour for reduced computation.

3.3 False Positive Rejection

In this stage of the pipeline, we use an SVM to rapidly classify the likelihood each detected contour is a hand. To be more precise, we use a *Support Vector Regressor* (SVR) that produces a confidence score for the classification, allowing for finer thresholding. Traditionally, false positive rejection is performed at the end of the pipeline when potential fingers are identified. However, computing finger positions is a relatively resource-intensive task. To maximize performance, we choose to perform false positive rejection early in the pipeline with features that are easily computed from the depth image to avoid performing finger detection on contours that are not

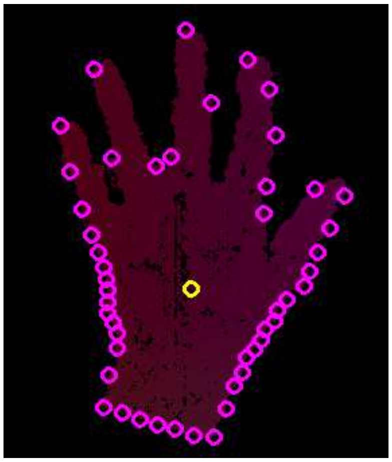

Figure 3: Visualization of the 48 SVM features for the open hand

the hand. The SVM utilizes 48 features (Figure 3) computed as follows:

1. **Enforce Rotation and Translation Invariance**: To make the algorithm rotation and translation-invariant, we transform the cluster's contour so that the *palm center* is at position zero and the *dominant direction* is up in the image space. We define the rough palm center to be the center of the largest inscribed circle within the hand's contour. This is determined by computing the Voronoi diagram of the contour, which is first downsampled to improve performance, and then iteratively checking each vertex inside the contour to find the one furthest from the contour.

   We simply define the dominant direction to be the vector from the palm center to the contour point farthest from the palm center.

2. **Find 48 Key Points**: We cast rays from the palm center in 48 directions on the 2D plane, starting with the "up" direction and with adjacent rays separated by 7.5 degrees. We select the nearest contour point to each ray as the corresponding keypoint.

3. **Create Features**: We compute the 3D Euclidean distance from the palm center to each key point computed by referring to the corresponding positions on the depth map. Please note that each pixel in the depth map has its corresponding 3D coordinates, when the camera is calibrated. The 48 distances obtained are organized into a feature vector.

We then can train an SVM with an RBF kernel on 18,880 real images manually labelled as either *hand* or *non-hand*. SVM is the preferred model for two reasons. First, the training data are of relatively low dimension and non-hand clusters are expected to be separable from hand clusters. Secondly, SVMs also have the advantage of running quickly and using very little computing resources.

3.4 Wrist Detection

To approximate the wrist positions, first we find the left-most and right-most points in the contour that intersect the bottom of the frame. If such points do not exist, we use the bottom-most (highest y-coordinate) point on the contour. We then move along the contour from each point in opposite directions until we first reach a predefined distance from the palm center. We call the nearest contour points the *wrist keypoints*.

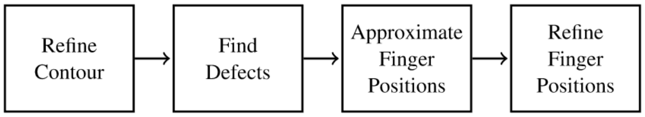

Figure 4: Our pipeline for classifying finger tips on the hand cluster is three step process. In the first step, we refine the contour with a median filter it to avoid noisy points on the edge of objects. In the second step, we approximate positions where finger tips might occur. The the third step, we use a series of hand engineered features to refine the previous approximates and prune false positives.

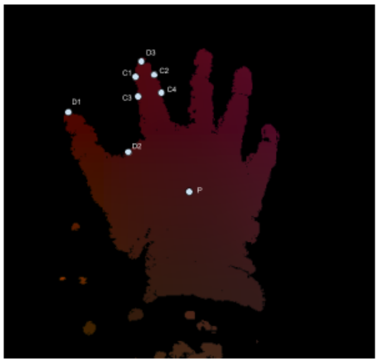

Figure 5: Keypoints used for finger classification. D1, D2, and D3 are computed from defects in the convex hull. P is the center of the palm. C1, C2, C3, and C4 are nearby points to the defect start and end.

3.5 Finger Detection

Given a contour, the role of the finger detector is to classify which points on the contour are fingertips (Figure 4). However, the Intel Realsense SR300, like most depth sensors, have higher noise around object edges. To compensate, for each contour point, we apply a 5 pixel median filter to obtain values from the less noisy interior points.

Next, we compute the convex hull and find the defects in the convex hull. Each defect is defined by three points, D1, D2, D3 (in Figure 7). For each defect, we add D1 and D3 to the list of potential fingers.

3.6 Heuristics

For each candidate finger point, we extract 8 key points (Figure 5). D1, D2, and D3 are the points computed from defects in the convex hull. P is the center of the palm. C1, C2, C3, and C4 are nearby points to the defect start and end. From these key point we generate 6 features which is used to determine whether a candidate point is a fingertip.

- **Finger Length:** Distance between D1 and D2.
- **Distance between defect to palm:** Distance between D2 and P.
- **Slope of Finger and nearest defect:** Slope of the line between D1 and D2.
- **Slope of finger and palm center:** Slope of the line between D1 and P.
- **Local curvature:** the curvature of the contour between C1, D3, and C2.
- **Non-local curvature:** the curvature of the contour between C3, D3, and C4, to contrast with *local curvature*.

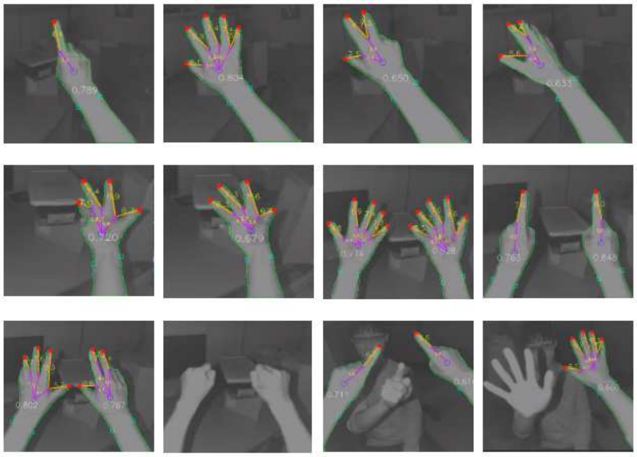

Figure 6: Finger classification under various hand poses, background interference, and false positive interference.

Each of the feature is thresholded against values that are tunable by the end user. The default values are picked based on the average observed measurements from the CVAR dataset [6].

3.7 One Finger Special Case

One of the most challenging poses is one finger since there is likely no defect in the convex hull and the overall shape does not look like a hand. We construct a pipeline to specifically handle this case since one finger gesture is very common in AR. To check for one finger poses, we first record the farthest point (p1) on the convex hull and points 0.05 meters around it. We then compute the local and non-local curvature around that point. We threshold the curvature to determine if it is sharp enough assuming the average finger width is between 1.5cm - 2.1cm. If the curvature is sharp enough, a candidate region is added and classified as above.

3.8 Limitations

Our algorithm being lightweight and real time has certain limitations. First, our hand model only contains finger tips and the palm. Thus, we do not directly capture the positions of other hand joints and cannot support hand interaction that requires precise estimation for those joints. Second, if the finger tips are occluded, we do not have a hidden model to estimate their locations. The hidden finger tips are being track with their last known position. Finally, our algorithm will lose track if the hand interacts with complex non-planar shapes in the real world.

4 Experiments and Results

4.1 Dataset

We evaluate the results on the CVAR hand dataset. The dataset contains 2800 unique hand poses captured from an egocentric perspective. CVAR dataset has folders named P1, P3, P4, P5, P6, and P7. P3 folder includes two hands. However only the 21 joints information for the right hand is given. In the modified dataset, we provide the algorithm with only the depth image for the right hand by masking the left hand. Further, we only evaluate on fingertip locations directly visible to the camera. CVAR dataset images are of the resolution 320*240 pixels. The depth images for the CVAR dataset collected from a Creative Senz3D camera.

4.2 Accuracy

To evaluate our algorithm, we compute the 3D Euclidean distance between the ground truth finger tip positions and the predicted position. A prediction is considered correct if the distance between the predicted position and the ground truth is less than 1.5 cm. In Table 1, we list the accuracy of our system evaluated on the various portions of the CVAR dataset.

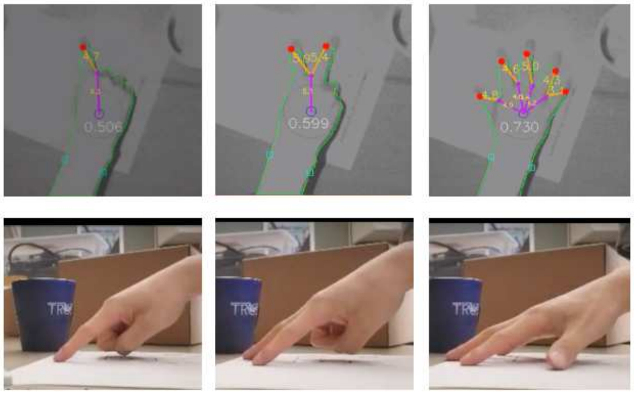

Figure 7: Finger detection when hand is in contact with surface. The value in the center of the hand is the SVM confidence.

Table 1: Finger Classification Accuracy

Dataset	P1	P2	P3	P4	P5	P6	P7
Accuracy (%)	98.7	95.7	94.1	93.8	94.2	86.2	84.1

P1 contains clean hand data without any occlusion and we perform the best on that dataset. P2 - P5 contain various levels of background clutter. P6 and P7 contain samples where the hand interacts with non-planar objects that is not part of our initial consideration. However, our algorithm still can achieve reasonable accuracy on those datasets. Figures 6 and 7 show several realistic examples of our algorithm performance.

4.3 Performance

We evaluate our algorithm on an Intel i5-9600k and an Intel i3-8350k CPU with an Intel Realsense SR300 depth camera (Table 2). Currently, our implementation of the algorithm has not utilized a GPU. The performance is tested under three configurations and the FPS achieved by the algorithm are averaged. The speed results show that, even in the most cluttered of scenes, the algorithm is able to attain real-time performance above 30 FPS.

5 Conclusion

We have proposed a hand tracking algorithm to fit the specific needs of AR glasses. The core requirement is to design an algorithm that is accurate, environment agnostic, and can achieve real time performance on mobile hardware. We started with a low-dimensional 8 point model rather than the traditional full skeletal model. Through-

Table 2: Speed Performance

Configuration	Hand Only	Hand + Clutter	Hand + Clutter + Plane
i5-9600k (FPS)	45	42	36
i3-8350k (FPS)	35	32	31

out our pipeline, we heavily rely on hand engineered features and geometry rather than quadratic optimization or large learned models. We are able to achieve high accuracy and real time performance on a single ego-centric depth image.

Acknowledgments

This work was supported in part by ONR MURI Grant N00014-13-1-0341, ONR Grant N00014-19-1-2066, FHL Vive Center for Enhanced Reality Seed Grant, an Intel Corporation research grant, and a Ford Motor Company research grant.

References

[1] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–432, June 2003. doi: 10.1109/CVPR.2003.1211500

[2] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.*, 108(1-2):52–73, Oct. 2007. doi: 10.1016/j.cviu.2006.10.012

[3] L. Ge, Y. Cai, J. Weng, and J. Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] C. Keskin, F. Kra, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1228–1234, Nov 2011. doi: 10.1109/ICCVW.2011.6130391

[5] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface 2013*, GI '13, pp. 63–70. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2013.

[6] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit. Efficiently creating 3d training data for fine hand pose estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4957–4965, June 2016. doi: 10.1109/CVPR.2016.536

[7] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *2011 International Conference on Computer Vision*, pp. 2088–2095, Nov 2011. doi: 10.1109/ICCV.2011.6126483

[8] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Markerless and efficient 26-dof hand pose recovery. In *Proceedings of the 10th Asian Conference on Computer Vision - Volume Part III*, ACCV'10, pp. 744–757. Springer-Verlag, Berlin, Heidelberg, 2011.

[9] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 3633–3642. ACM, New York, NY, USA, 2015. doi: 10.1145/2702123.2702179

[10] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *2013 IEEE International Conference on Computer Vision*, pp. 2456–2463, Dec 2013. doi: 10.1109/ICCV.2013.305

[11] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1372–1384, Sep. 2006. doi: 10.1109/TPAMI.2006.189

[12] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, Jan 1994. doi: 10.1109/38.250916

[13] R. Wang, S. Paris, and J. Popović. 6d hands: Markerless hand-tracking for computer aided design. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pp. 549–558. ACM, New York, NY, USA, 2011. doi: 10.1145/2047196.2047269

[14] R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove. *ACM Trans. Graph.*, 28(3):63:1–63:8, July 2009. doi: 10.1145/1531326.1531369

[15] X. Wu, D. Finnegan, E. O'Neill, and Y.-L. Yang. Handmap: Robust hand pose estimation via intermediate dense guidance map supervision. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[16] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Yong Chang, K. Mu Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. Argyros, and T.-K. Kim. Depth-based 3d hand pose estimation: From current achievements to future goals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.