Efficient Hand Pose Estimation from a Single Depth Image

Chi Xu
Bioinformatics Institute, A*STAR, Singapore
xuchi@bii.a-star.edu.sg

Li Cheng
Bioinformatics Institute, A*STAR, Singapore
School of Computing, NUS, Singapore
chengli@bii.a-star.edu.sg

Abstract

We tackle the practical problem of hand pose estimation from a single noisy depth image. A dedicated three-step pipeline is proposed: Initial estimation step provides an initial estimation of the hand in-plane orientation and 3D location; Candidate generation step produces a set of 3D pose candidate from the Hough voting space with the help of the rotational invariant depth features; Verification step delivers the final 3D hand pose as the solution to an optimization problem. We analyze the depth noises, and suggest tips to minimize their negative impacts on the overall performance. Our approach is able to work with Kinect-type noisy depth images, and reliably produces pose estimations of general motions efficiently (12 frames per second). Extensive experiments are conducted to qualitatively and quantitatively evaluate the performance with respect to the state-of-the-art methods that have access to additional RGB images. Our approach is shown to deliver on par or even better results.

1. Introduction

3D hand pose estimation has a wide range of applications including avatar animation and graphics [22, 24], robotic design [13], human-computer interaction, and Ergonomics. Despite of extensive research efforts as reviewed in *e.g*. [17, 10, 9, 13] and references therein, it remains a challenging problem, which is mainly due to the complex and dexterous nature of hand articulations. Recent advances in commodity-level RGB-D cameras such as Kinect [1] and Xtion have greatly simplified the problem, which enable a number of new developments to exploit this new source of information [16, 25, 15, 4]. The Kinect-type depth images, on the other hand, come with noticeable depth noises that significantly degrade the image quality, as illustrated in Figure 1. In particular, regions are sometimes missing, and there are ghost shadows around object boundaries – pixels with undefined depth values.

In this paper, we focus on efficient hand pose estimation from a single noisy depth image. Clearly this can not be accomplished by a trivial re-implementation of existing methods *e.g*. from human pose estimation, as hands are much smaller, flexible and there are many more self-occlusions. Worse, we often only have access to noisy observations with large portion of missing values. This leads us to propose a dedicated three-step pipeline: Initial estimation step provides an estimation of the hand's in-plane orientation and 3D location using a Hough forest model; Candidate generation step produces a limited set of plausible 3D poses from the voting space of a different Hough forest, using depth features that are now invariant to in-plane rotations; Verification step delivers the final 3D hand pose as the solution to an optimization problem. Experiments are conducted using the depth channel of a Kinect sensor, with some exemplar estimation results presented in Figure 1. Our system demonstrates its comparable or even better performance with respect to the state-of-the-arts, which usually make use of *additional* RGB information. Our system works at a speed of 12 frames per second (FPS) on an average desktop without multicore or GPU speedup. The main contributions are:

- Our approach estimates 3D hand poses from a single depth image, which is applicable to general motions (i.e. activity-independent). To our knowledge, this is the only such system to work with Kinect-type noisy depth images and reliably produces pose estimations of general motions [1].
- We analyze the depth noises, show that they are inherited from the geometric layout of the on-board sensors, and offer tips to minimize their negative impacts on the overall performance.
- Different from the often-used hand kinematic model in *e.g*. [18] where the palm is assumed flat, our model is able to simulate palm arching (Figure 3) – which is novel as *e.g*. discussed in sec. 2 of [20].

[1] More results are at http://web.bii.a-star.edu.sg/˜xuchi/handengine.htm.

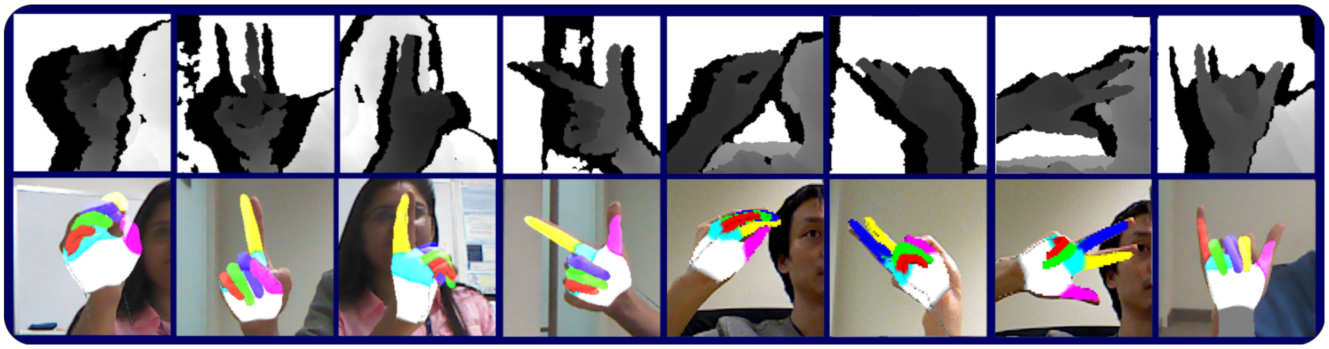


Figure 1. Exemplar 3D hand estimation results of the proposed approach for hands from various gestures, orientations, as well as from different people. The first row is the input depth images from Kinect, while the second row presents our corresponding results overlayed on the RGB images. The depth images are very noisy and of low-resolution, nevertheless our approach produces satisfactory results. Note the RGB images are not used by our system, and are shown here only for visualization purpose. The color tag is used to clearly visualize the extent of each finger as well as the upper and lower parts of the palm.

- A dedicated three-step pipeline is devised to provide a systematic solution. In particular, the depth features of the second step are reformulated to become invariant to in-plane rotations.

Related work In their work, Shotton *et al*. [19] focus on the problem of body part labeling, and produce impressive results. In particular, their method utilizes large-scale synthetic data during the training stage, as well as adopts the random forest paradigm of [5]. This framework is further followed by [11] for directly regressing the 3D locations of 16 body joints for body pose estimation. It is similarly used by Keskin *et al*. [15] to address the problem of hand gesture recognition, *i.e*. to recognize a limited set of predefined hand gestures. In particular its application focuses on letter recognition from the American sign language.

The problem of estimating 3D hand pose from a single RGB image has also been studied by *e.g*. [18, 21, 9] for a single RGB image and by *e.g*. [8] for multiple cameras. These approaches generally do not offer fast processing speed as the hypothesis generation and verification in this context is computationally rather demanding. More recently, a tracking-based 3D hand pose estimation method has been proposed in [16] by making use of both RGB and depth channels of Kinect. It however requires an explicit initialization to learn a hand skin model which may be cumbersome in some circumstances. By exploiting GPU parallel computing, their method is able to parse images at a frame rate of 15 frames per second (FPS).

The commercially available 3Gear system [2] is able to robustly estimate hand poses by accessing to the RGB-D channels of *two* Kinects. It however supports only 6 predefined gestures. The performance is excellent with these gestures at various orientations, but becomes unbearable when engaging with an unseen gesture. Leapmotion [3] is the most recent commercial system designed for close-range (within about 50cm in depth) hand pose estimation. As a closed system, its inner working mechanism remains unclear. Our observation is that it is not well tolerant to even moderate level of self-occlusions of finger tips, such that a finger will not be detected if it touches other fingers or the palm. In contrast, our system works beyond half a meter in depth, and work well when some of the finger-tips are occluded as it does not rely on detecting finger tips.

For the related problem of optical motion capture, Kinect has been utilized together with existing marker-based system [25], or multiple RGB cameras [4] for markerless optical motion capture.

2. Our Approach

Our approach starts with a preprocessing step to decrease the discrepancies between synthetic and real depth images. It then follows with a three-step pipeline: Step one provides initial estimation of in-plane orientation and 3D location using a Hough forest; Step two makes usage of a different Hough forest model to produce a set of 3D hand pose candidates, with the help of modified depth features that are invariant to in-plane rotations; Step three delivers the final pose estimation by solving an optimization problem.

One may wonder why having the three steps instead of a single step, as what has been adopted for human body estimation [19, 11]. One chief motivation is the fact that a hand can easily roll sideways (*i.e*. in-plane rotation), a somehow rare case for everyday human body motions. As depth features are usually not rotational invariant, we instead consider separate steps – thus our first two steps in the pipeline. Furthermore, as a compound consequence of small hand size and dense self-occlusions, mode-seeking in the Hough voting space might not necessarily give the optimal 3D hand pose. The third verification step is thus useful to identify the best fit from a small pool of top-notch candidates.

Before proceeding further, let us review some key existing techniques.

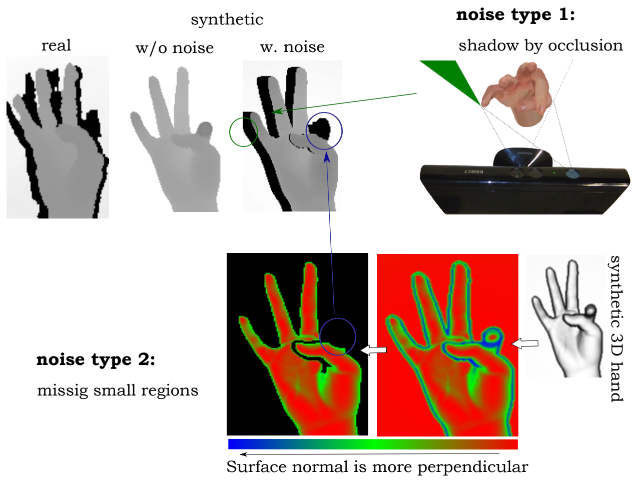


Figure 2. Depth noises. Top left panel presents three depth images: A real, a synthetic without noise, and a final synthetic with noise. Note the pixel in a depth image is darker as it is closer to the viewing camera. The pixels in pure black are those with unknown depth values. Two main sources of noises: (1) Shadow by occlusion in green. (2) Missing small regions in blue. See text for detail.

Depth features We adopt the same depth features as mentioned in [19]. That is, at a given pixel location x of an image I, denotes its depth value as a mapping $d_I(x)$, and construct a feature $f_I(x)$ by considering two 2D offsets positions u, v from x:

$$f_I(x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right). \qquad (1)$$

This feature is depth invariant. However, it is not rotational invariant.

Regression model of Hough forest The regression model of Hough forest is developed in [11] to estimate 3D locations of body joints. It is used in the first two steps of our pipeline to create two different regression models.

2.1. Preprocessing: Analyzing the Depth Noises

Our approach relies on *synthetic* hand images for training the model, which is then applied to *real* depth images for pose estimation. The assumption is synthetic and real images are *similar*. However, there are noticeable noises presented in typical Kinect-type depth images. As illustrated in Figure 2(a), they are in fact quite different. The issue is further complicated by the fact that the Hough forest model tends to pick up features along depth boundaries – which often coincide with the ares having large noises (first row of Figure 1). Empirically we have observed that simply ignoring their existence leads to much downgraded performance.

The noises, especially those with unknown values, turn out to be rather different from the typical image noises such as Gaussian noise or salt-and-pepper noise. We have identified two major noise sources. The first type of noises is the shadow around boundaries, as illustrated Figure 2 top-right, which is due to the occlusion from the geometric layout of the sensors: As shown in the green-colored area, the occlusion is introduced by the displacement of the IR light source and the CMOS imaging sensor. The second noise source contributes to the disappearance of small regions. As depicted in Figure 2 bottom, rendered from certain view of the synthetic 3D hand, the surface normal of every pixel is calculated, and its perpendicular tendency w.r.t. the viewing direction is measured. Those tend to be perpendicular are less visible, therefore their depth values become unknown (in black). As a result, often there are small regions surrounded by these black unknown pixels, which are subsequently missing from the final depth map, due to limited sample rate of the depth camera. By simulating according to these noise sources, as in Figure 2 top-left panel, our noise-added synthetic image is more similar to the real one, where the index finger area shrinks to a small red region due to perpendicular surface normals, then is entirely missing from the final synthetic depth image, as presented in Figure 2 bottom row. During our experiments, the synthetic images are added with these two source of depth noises.

2.2. Step 1: Initial Estimation

This step is to estimate the in-plane orientation and location of the hand. The parameters to be estimated are (x_1, x_2, x_3, θ), in which x_1, x_2, x_3 defines the 3D position of the hand base (*i.e.* the bottom of a hand), and θ is the in-plane orientation of the hand. A regression model of Hough Forest [11] is used to predict these parameters, as follows: Each image pixel is parsed by one of the T_1 trees, leading to a path from the root to certain leaf node that stores a collection of votes. Here each vote corresponds to a six-dimensional vector. The voting space is then formed by plainly mixing all the votes from every tree and every pixel, and the standard mean-shift method [7] is used to find the mode of the point cloud – the output of our initial estimation.

In case there are h hands in a image, parameters of the hands are obtained by identifying the top h sets of parameters, after applying non-maximal suppression over their locations.

2.3. Step 2: Candidate Generation

Given the in-plane orientation θ, the hand can be reversely rotated to its canonical pose in Figure 3 (c), as rotating each of the hand-related pixels x in-plane by $-\theta$ along the hand base. The depth feature can thus be computed from the processed image using Equation 1. Unfortunately a whole image rotation is computationally rather costly. Having this in mind, we instead redefine our depth features as follows:

$$f_I(x) = d_I\Big(x + \frac{u'}{d_I(x)}\Big) - d_I\Big(x + \frac{v'}{d_I(x)}\Big), \qquad (2)$$

where $u' = \mathrm{Rot}(u; x, -\theta)$, $v' = \mathrm{Rot}(v; x, -\theta)$. Here $\mathrm{Rot}(u; x, \theta)$ defines a mapping from the x-based relative 2D pixel location u to its new 2D location, as in-plane rotating u along the pixel x by θ. Note his new depth feature does *not* implement exactly the image rotating scenario described above. Nevertheless it is an efficient approximation, and as a result, our depth features becomes invariant to in-plane rotations.

A second regression model of Hough forest is then constructed to produce a pool of 3D hand pose candidates: Each image pixel is parsed by one of the T_2 trees, leading down the tree path to certain leaf node that stores a collection of 27-dimensional votes. The voting space is similarly formed as in step 1, and the standard mean-shift method [7] is used to find k local modes of the point cloud as the output pool of 3D hand candidates. We note that during training stage of this step, the out-of-plane orientations (*i.e.* a hand's palmar and dorsal sides as well as every orientation in-between) are each considered as a unique training example. Besides, we only need to consider the canonical hand poses with small in-plane perturbations, as the in-plane rotation issue has been explicitly addressed by step 1. In what follows, we discuss our 3D hand model.

Kinematic chain vs. 3D location of joints Instead of estimating the 3D location of body joints from a depth image [11], our model predicts the parameters of a predefined hand kinematic chain, which is then used to build the 3D hand. There are two main motivations for doing so. First, a kinematic chain model is better at dealing with self-occlusion, an scenario often seen in hand pose estimation. Comparing to 3D location of joints, kinematic chain is a global representation and is more tolerant to small local perturbations. Second, for human pose estimation, once the body location is known (*i.e.* the torso is fixed), the limbs and the head can be roughly considered as independent subchains: *e.g.* a change from left hand will not affect the other parts significantly. In contrast, motions of the five fingers and the palm are tightly correlated.

Our 3D hand model A widely-used hand kinematic chain model has been proposed by Rehg and Kanade [18], which has 21+6 degree-of-freedom (DoF). Unfortunately, in this kinematic model, the palm is assumed to be a rigid object, making it unable to mimic gestures with perceivable palm arching. This works well in scenarios with a extension (*i.e.* flat hand) type gestures. The limitation is however quite pronounced in the scope of activity-independent general motions, as *e.g.* when the fingers are in flexion motions

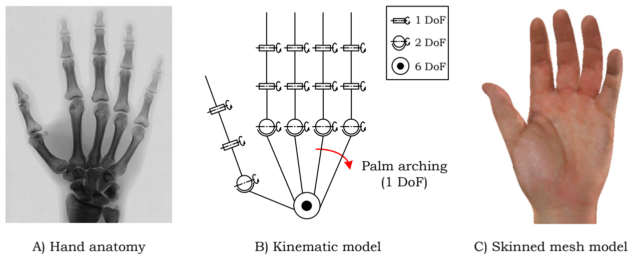


Figure 3. Our 3D hand contains 21+6 degrees of freedom, including the hand root position and orientation (6), and the relative angles of individual joints (21). From (A) to (C): The hand anatomy, the underlying skeleton kinematic model, and the skinned mesh model.

(*i.e.* curled inwards, column 1 & 5 of Figure 1) that are frequently seen in everyday activities of a hand.

Here we devise a new kinematic model. As illustrated as the red curved arrow in Figure 3 (b) connecting the last two metacarpus bones, our model explicitly considers the distal transverse arch [6], crucial for bending the palm. As *e.g.* discussed in Section 2 of [20], this has not been seen in existing literature. In our kinematic model, four DoF are attached to each finger from bottom up in Figure 3 (b): For each of the four fingers (*i.e.* index to pinky fingers), two DoF are used for the metacarpophalangeal (MCP) joint, and one each is for the proximal interphalangeal (PIP) joint and the distal interphalangeal (DIP) joint, respectively. The thumb, on the other hand, has a different structure from the other four: Two DoF are assigned to the trapeziometacarpal (TM) joint, and one each is for the MCP joint and the interphalangeal (IP) joint, respectively; Together 20 DoF are used for modeling the articulation of the five fingers, and 1 for palm bending. Similar to [18], the remaining 6 DoF are reserved for the global orientation and position of the hand. These model parameters collectively form a 27-dimensional vector denoted as Θ. It is the vote stored in the leaf node of step 2, and is also used to represent each of the 3D hand candidates.

2.4. Step 3: Verification by Minimization

Grenander and co-workers [12] have advocated the idea of "estimation by synthesis", which are used in many vision works. For example, it is adopted by [25] for motion capture and by [9] for estimating hand pose from one RGB image. It often involves the minimization of a discrepancy measure, ρ, between the observed and the synthetic depth image,

$$\Theta^* = \arg\min_{\Theta} \sum_x \rho\big(d_{I_{\text{syn}}}(x; \Theta) - d_I(x)\big), \qquad (3)$$

where I_{syn} denotes the synthetic image, and there are many options for ϕ, including the standard square loss function, and the robust M-estimators [14] such as the Huber loss function. For convenience we adopt the square loss function during our experiments.

This is a non-convex optimization problem in our context, as elements in the unknown vector of random variables Θ are tightly coupled. The usual way to address this problem is by iteratively descending along a direction related to current gradient, in searching for the fix-point solution in a continuous space. In practice it becomes unfortunately the time-consuming bottleneck [9], and especially so in our context as candidate poses are to be rendered on the fly. We instead consider a crude approximation alternative, by searching within a finite space of only k candidates handed over from step 2, and pick the one that minimizes Equation (3). This simple strategy has been empirically supported with satisfactory performance, meanwhile it greatly accelerates the processing speed.

3. Experiments

Datasets and Setup Based on an open source SDK Libhand [23], we have developed a new rendering engine to generate synthetic data. Our synthetic sets are randomly sampled from different orientations as well as a wide range of gestures: The kinematic model of over thirty American sign language letter and number gestures (after removing the dynamic guestures and and a few duplicate gestures) are adopted as bases. A number of random pairs are then selected, and from each pair, we smoothly interpolate over each of the joints to obtain a series of new gestures in-between. This way we construct our pool of synthetic hand gestures.

In our experiments, the number of trees for both Hough forests are the same as $T_1 = T_2 = 5$, the tree depth is fixed to 20, and the candidate pool size $k = 10$. To train the first Hough forest in the initial estimation step, 20k synthetic images with different gestures, orientations and scales were randomly generated for each tree. The range of in-plane rotation is [-90, 90] degrees. For the second forest in the candidate generation step, 100k synthetic images were randomly for each tree, thus totally 500k images for the entire forest. The in-plane rotation was set in a small range of [-10, 10] degrees to account for small perturbations, which enhances the system stability as illustrated in Figure 4. For both forests, 512 pixels are randomly sampled from each synthetic image, which roughly are evenly distributed over a hand image. The tree is grown in a way similar to [19]. At a decision node, the number of candidate tests is 1024, from which a best feature (a test & threshold pair) is selected.

Our 12 FPS performance is obtained from a desktop with Intel Core i7 CPU 960 3.20GHz and 24GB memory. We note that our code is not optimized, and with only 1 CPU being used during the experiments.

Effects of varying the internal parameters To examine the influence of the internal parameters, we have conducted a series of study. As depicted in the first row of Figure 5,

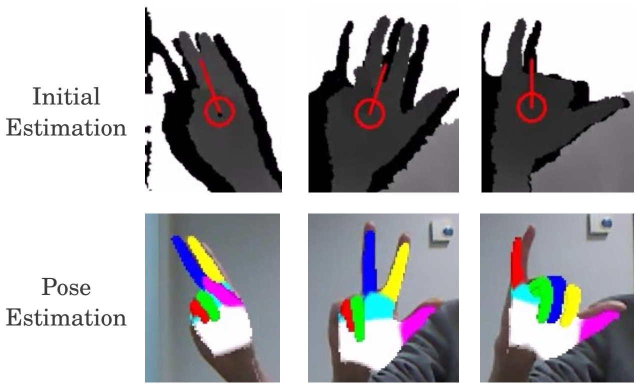
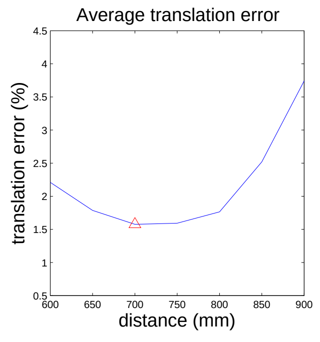
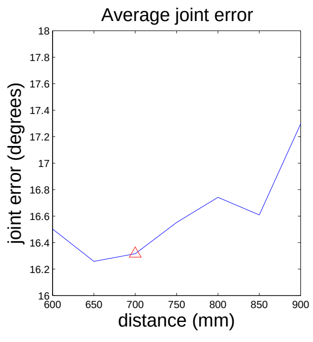


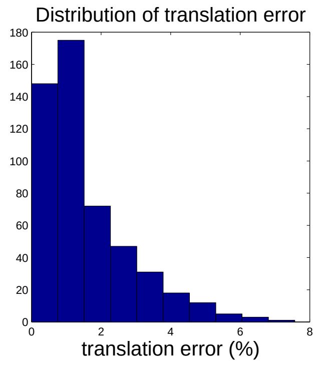
Figure 4. First row depicts exemplar results from initial estimation for in-plane rotation and 2D location of a hand. Second row presents the final estimation results. Sometimes, small in-plane rotation estimation error might occur such as the top-right subplot, this however will not affect the final result (bottom-right subplot). due to the introduction of local random perturbation to the in-plane rotation angle in the training set of the second forest, which significantly enhances the system stability.



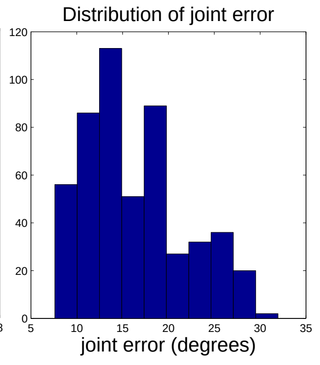
(a)



(b)



(c)



(d)

Figure 5. First two plots: Average translation error and average joint error. Average joint error is defined as the average error of the rotation of all hand joints. Distance is between hand and camera. The last two plots: The distribution of translation error and joint error, where the distances are both fixed to 700mm, as the red triangle locations in the first two plots.

the estimation errors and in particular the translation error of hand base (in %) & the average error of joints (in degrees) are evaluated as a function of distances. Due to the physical limitation of Kinect, the minimum depth is 0.5 meter, and the the hand area becomes too small beyond 1-1.2 meter in

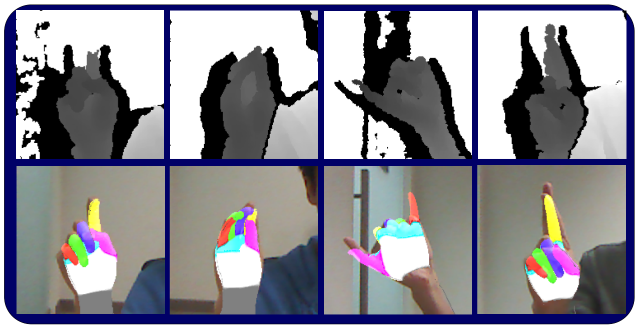


Figure 6. Exemplar results on *good* and *failure* cases of our approach. In particular, thet two columns on the right present incorrect estimation results.

depth. So we set our distance range to 0.6-0.9 meter. To produce the two plots, we take measure after every 50mm, and each time our system (which is trained at around 0.7m distance) is used to make predictions on 512 randomly sampled synthetic images. The error at a point is obtained by averaging over these predictions. The relative error metric for translation, %, is computed by dividing the absolution distance error by current distance, the horizontal value of the plot. In other words, the error metric is more sensitive when the hand is close, and is less sensitive when the hand is farther away from the camera. It is not a surprise to observe a nice test performance at around 700 mm distance, the same as what our system is trained with. The errors appear to become larger as the distance grows. However, the changes are both within relatively small ranges. To examine further, we also plot the empirical error distributions at 700 mm distance. The majority of the results seem to lead to small errors: Within 2% for location error, and within 20 degrees for angular error. Overall, our system performs rather robustly with respect to the change of depth.

Visual analysis of our results It is practically very important to consider the depth noises and in particular the missing of small regions in our synthesized training data. As with column 1 of Figure 6, the index finger tip is *missing* from the Kinect raw image, nevertheless our approach can still successfully estimate a proper 3D hand pose that is nicely aligned with the RGB image in hindsight. Moreover, the incorporating of palm arching parameter in our 3D hand model also turns to be very helpful in estimating the gestures with bending palm. Column 2 of Figure 6 provides such an example.

Our system may fail to estimate the right orientation for hand gestures with similar frontal and back sides in the depth map, as in the third column of this figure. Sometimes several fingers heavily overlap with each other, as the cross-finger case in the fourth column, they may be estimated as a single finger instead.

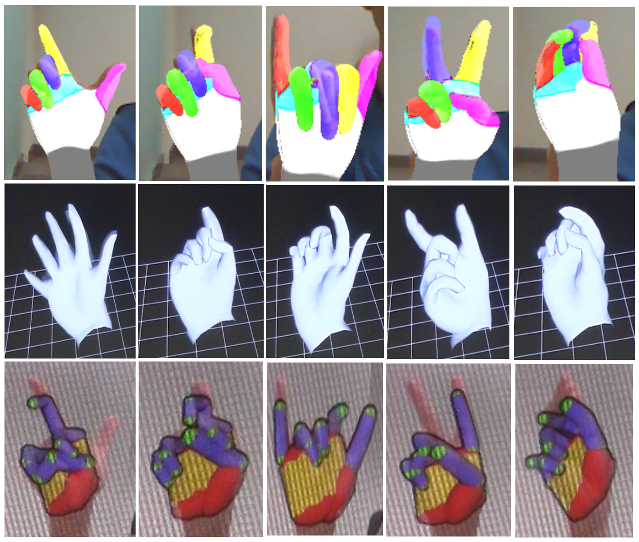


Figure 7. Comparing with the state-of-the-arts. First row: Our results. Second row: results of 3Gear [2]. Third row: results of Oikonomidis and Argyros [16].

Comparisons with the state-of-the-arts Our results are visually compared with Oikonomidis and Argyros [16], 3Gear [2] on test images from Kinect. Some results are displayed in Figure 7. As expected, 3Gear [2] performs poorly on these hand poses, as the gestures are outside of the 6 predefined ones, at the expenses of employing two Kinects. [16] seems very difficult to pick up some gestures & orientations in tracking, as shown in this figure, as sell as in the supplementary videos. We hypothesize this is mainly due to the dense self-occlusions of fingers in these images.

4. Conclusion and Future Work

We have proposed a systematic approach to estimate 3D hand poses of general motions from a single depth image. Empirical simulations demonstrate that our system is able to deliver fast and accurate estimation results, and its performance is comparable with the best available systems that have access to additional RGB images. This is made possible by a carefully designed three-step pipeline, as well as a detailed analysis of the depth noises. On future work, we plan to exploit prior knowledge of hand motion constraints to work in a reduced parameter space, as well as to extend the current work to deal with scenarios where one hand is interacting with physical objects including other hands. We also plan to make available a GPU computing version for more efficient processing.

Acknowledgements

This research was partially supported by A*STAR JCO and IAF grants. We would like to thank Andrei Lucian Ionita for helpful discussions.

References

[1] Kinect. http://www.xbox.com/en-US/kinect/, 2011.

[2] 3gear. http://www.threegear.com/, 2012.

[3] Leapmotion. http://www.leapmotion.com/, 2013.

[4] L. Ballan, A. Taneja, J. Gall, L. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] E. Chao, K. An, W. Cooney, and R. Linscheid. *Biomechanics of the Hand: A Basic Research Study*. World Scientific, 1989.

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[8] T. de Campos and D. Murray. Regression-based hand pose estimation from multiple cameras. In *CVPR*, 2006.

[9] M. de La Gorce, D. Fleet, and N. Paragios. Model-based 3d hand pose estimation from monocular video. *IEEE Trans. Pattern Anal. Mach*, 33(9):1793–1805, 2011.

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

[11] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, 2011.

[12] U. Grenander. *Pattern Theory: From Representation to Inference*. Oxford University Press, 2007.

[13] A. Gustus, G. Stillfried, J. Visser, H. Jorntell, and P. van der Smagt. Human hand modelling: kinematics, dynamics, applications. *Biological Cybernetics*, 106(11-12):741–755, 2012.

[14] P. Huber. *Robust Statistics*. Wiley Press, 1981.

[15] C. Keskin, F. Kirac, Y. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, 2012.

[16] N. Oikonomidis and A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, 2011.

[17] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):677–695, 1997.

[18] J. Rehg and T. Knade. Visual tracking of high dof articulated structures: an application to human hand tracking. In *European Conference on Computer Vision*, pages 35–46, 1994.

[19] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[20] B. Siciliano and the DEXMART team. Kinematic modelling of the human hand. Technical report, University of Naples, 2009. http://www.dexmart.eu/index.php?id=13735.

[21] B. Stenger, P. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *CVPR*, 2001.

[22] S. Sueda, A. Kaufman, and D. Pai. Musculotendon simulation for hand animation. In *SIGGRAPH*, pages 83:1–83:8, 2008.

[23] M. Šarić. Libhand: A library for hand articulation, 2011. Version 0.9.

[24] R. Wang and J. Popović. Real-time hand-tracking with a color glove. In *SIGGRAPH*, pages 63:1–63:8, 2009.

[25] W. Zhao, J. Chai, and Y. Xu. Combining marker-based mocap and rgb-d camera for acquiring high-fidelity hand motion data. In *Eurographics Symposium on Computer Animation*, 2012.