A Unified Framework for 3D Hand Tracking

Rudra P K Poudel, Jose A S Fonseca, Jian J Zhang, Hammadi Nait-Charif

Bournemouth University, Pool BH12 5BB, UK,
{rpoudel, jfonseca, jzhang, hncharif} @bournemouth.ac.uk

Abstract. Discriminative techniques are good for hand part detection, however they fail due to sensor noise and high inter-finger occlusion. Additionally, these techniques do not incorporate any kinematic or temporal constraints. Even though model-based descriptive (for example *Markov Random Field*) or generative (for example *Hidden Markov Model*) techniques utilize kinematic and temporal constraints well, they are computationally expensive and hardly recover from tracking failure. This paper presents a unified framework for 3D hand tracking, utilizing the best of both methodologies. Hand joints are detected using a *regression forest*, which uses an efficient voting technique for joint location prediction. The voting distributions are multimodal in nature; hence, rather than using the highest scoring mode of the voting distribution for each joint separately, we fit the five high scoring modes of each joint on a tree-structure *Markovian model* along with kinematic prior and temporal information. Experimentally, we observed that relying on discriminative technique (i.e. joints detection) produces better results. We therefore efficiently incorporate this observation in our framework by conditioning 50% low scoring joints modes with remaining high scoring joints mode. This strategy reduces the computational cost and produces good results for 3D hand tracking on RGB-D data.

1 Introduction

The hand is often considered as one of the most natural and intuitive interaction modalities for human-human interaction [1]. It is also the most natural interaction interface with the physical world because it is used to manipulate objects such as grasping, pushing and twisting [2]. In human-computer interaction (HCI), proper 3D hand tracking is the first step in developing a more intuitive HCI system which can be used in applications such as virtual object manipulation and gaming. In recent years, built-in cameras in most of consumer electronic devices and the low price of depth sensors have opened new venues for hand gesture recognition research and applications. However, 3D hand gesture recognition, which is directly dependent on the accuracy of the hand tracking, remains a challenging problem due to the hand's deformation, appearance similarity, high inter-finger occlusion and complex articulated motion.

Hand tracking techniques can be divided into two major categories: *appearance-based* and *model-based* techniques [3]. Appearance-based techniques [4,5] extract features from image then classify the hand postures into meaningful gestures;

hence the quality of the hand tracking depends mainly on the robustness of the features. The model-based techniques [6,7,8,9] first sample the 3D model of the hand and evaluate it against the observed data. Model-based techniques extract the hand configuration more accurately than appearance-based techniques; however model-based techniques are computationally expensive. Moreover, based on how individual hand parts are used to estimate the hand pose, hand tracking techniques can be divided into two categories, *joint evidence techniques* and *disjoint evidence techniques* [9]. Joint evidence techniques [7,9] efficiently handle occlusions but they are computationally very expensive because of the larger search space as hand having 27 degrees-of-freedom. Disjoint evidence techniques [6,8,5] are computationally efficient because they reduce the search space but need additional mechanisms to handle the occlusion and collision. The unified framework presented in this paper falls under appearance-based and disjoint evidence techniques. However, our technique does not require additional occlusion or collision handling mechanisms unlike other disjoint evidence techniques [6,5]. Our proposed framework consist of three modules: i) hand region segmentation: segments the hand region using skin and depth cues; ii) hand pose estimation: uses a regression forest to estimate the positions of the hand joints ; iii) hand tracking: uses the pose estimation, kinematic prior and temporal information to track the hand in 3D.

Inspired by the work of Girshick et el. [10] which used a *regression forest* to efficiently predict occluded human body joints, our joint estimation module uses a discriminative *random forest* [11] to classify the hand-parts and learn joints offsets at leaf nodes. Since the voted joint offsets are multimodal in nature a *mean-shift* [12] voting aggregation technique is used. Unlike Girshick et el. [10] which selected human body joint proposal independently, we optimize joint proposals with kinematic prior and temporal constraints globally with a *Markov random Field* (MRF) [13]. We added temporal information on the same semantic level and modelled as MRF (ref. fig. 1).

Keskin et al. [5] and Hamer et al. [8] are relevant for our approach. While Keskin et al.[5] used a *classification forest* to classify hand-parts, an *artificial neural network* for occlusion handling and *translation vector* to push joints from the finger surface to their inside positions; our technique does not require any extra occlusion handling mechanism and directly predicts the hand joints. Moreover, Keskin et al.[5] track the hand by detection (pose estimation), while we also incorporate temporal motion and hand-part length prior. On the other hand Hamer et al. [8] used a model based approach, while we use an appearance based approach. In our MRF model, joints represent MRF nodes, while hand-parts represents MRF nodes in Hamer et al. [8], hence our technique is more flexible for different hand sizes. Additionally, half of the nodes in our MRF model are fixed as explained in section 3.3.

The focus of this paper is on single hand tracking using a Kinect sensor [14]. Our contributions are as follows: i) a unified framework for 3D hand tracking which efficiently combines discriminative and descriptive techniques; ii) a regression forest based technique for hand pose estimation which performs better

than classification forest based techniques; iii) a simple way of generating better features from a larger feature space (ref. *feature pool*, section 4). The paper is organised as follows: section 2 describes how artificial training data is generated; section 3 presents the proposed 3D hand-tracking framework. Experiments and results are presented and discussed in section 4. Finally, section 5 concludes the paper and presents future works.

2 Artificial Data Generation

The aim of this paper is to build a markerless 3D hand tracking system using a RGB-D sensor. The system is to be trained to detect 3D hand poses in an RGB-D image stream. To generate enough hand pose training data, various CG hand were used, and the trained system is expected to generalize and work equally well on the real data. To simulate the Kinect noise, Gaussian noise was added to the CG generated data. The system is trained to detect 15 joints of the hand (palm's one, thumb's two and 12 joints of the remaining four fingers) and 5 finger tips, which is depicted in figure 1. Similar to the work of [15,5] the classification forest is trained on 21 regions of the hand; 15 regions are centred around each hand joints and 5 regions for finger tips and one extra region to cover up middle part of the palm as shown in figure 1.

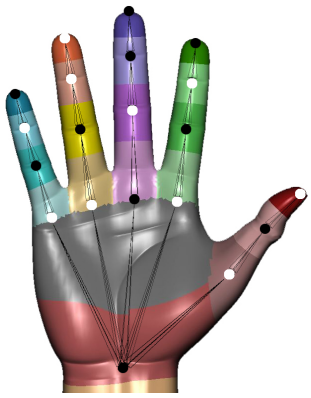


Fig. 1. Marked hand regions and MRF model, where white nodes/joints are conditioned on black nodes/joints/fixed-nodes.

3 3D Hand Tracking

The proposed 3D hand tracking framework has three sub modules: *hand region segmentation*, *hand pose estimation* and *hand tracking*. The input to the proposed framework is a stream of RGB-D images. The hand region segmentation module takes both RGB and depth images as input, while hand pose estimation module only takes segmented depth image as an input. The final hand tracking module takes five high scoring modes of each joint. All three modules are described in detail below.

3.1 Hand Region Segmentation

Both skin color and depth cues are used for the segmentation of the hand.
Skin cue: A histogram based Bayesian classifier [16] is used for skin color detection. Densities of skin and non-skin color *histograms* are learned from the 14 thousands images of *compaq dataset* [16] which contains images from all ethnic groups with uncontrolled illumination and background conditions. Training

using such a huge dataset makes the proposed technique equally applicable to unconstrained backgrounds, ethnic origins and lighting conditions.
Depth cue: At the initialization step we assume that the hand is the closest object in the scene to the Kinect sensor and roughly at the centre of the image. Then for a depth frame D at a time t, we assume that the hand will be inside a cuboid of $width = 10\ pixels$ and $depth = 5\ cm$ centred around the hand depth image at time $t-1$. The use of a cuboid mask instead of a spherical mask makes the query of image pixels easier. Also, the hand is more likely to move either up/down or left/right faster than in diagonal directions.
Hand region segmentation: given the skin and the depth cues described above, we extract the largest region which is later provided as an input for the hand pose estimation module (ref. section 3.2).

3.2 Hand Pose Estimation

Our technique uses an ensemble of decision trees [11] called a random forest to regress the hand joints. Following [17,10], we use classification nodes to split a tree and a hough voting technique at leaf nodes of the tree for joint proposals. Since using all votes from the training pixels is inhibitive due to limited memory and available processing power, reservoir sampling [18] has been used. Then mean-shift model finding technique [12] is applied on those joint proposals. The highest scoring mode of each joint is used for pose estimation, and for the hand tracking, five high scoring modes are used. The feature details, training and testing methodologies are described below.

Depth feature The quality of features has a significant influence on the quality of hand parts classification. However, because of the computational demand of random forests, simple features are used to achieve real-time computation. Hence, an efficient depth comparison feature from [15] is used, which requires only five arithmetic operations. For a pixel d of depth image D, the depth value at d is denoted by $D(d)$, and the depth difference feature $\theta = (u, v)$, the feature value F is defined as follows,

$$F_\theta(D, d) = D(d + \frac{u}{D(d)}) - D(d + \frac{v}{D(d)}) \tag{1}$$

where u and v are 2D pixel offset values. The maximum length of an adult hand is 23 centimetres [19], hence a threshold is applied to F_θ such as $-25cm \leq F_\theta \leq +25cm$. The division by depth value of a given pixel d makes sure the feature is invariant of depth. However, since the feature (ref. equation 1) is not rotationally invariant, all possible samples of the targeted system are provided.

Classification forest Each decision tree of the random forest is trained using depth difference feature described above to classify the hand regions (ref. figure 1). Each split node of a decision tree is trained with collection of features and thresholds τ. The aim is to split all training pixel examples to *left*, L, or *right*,

R, child nodes to reduce the uncertainty of the hand region classes C. We use Shannon entropy, S, to measure the uncertainty of hand region classes defined as following,

$$S = -\sum_{c \epsilon C} p(c) log(p(c)) \tag{2}$$

where, $p(c)$ is a normalized discrete probability of a hand region class, calculated using the histogram of all training examples at the given node. Hence, the information gain I of the split node is defined as

$$I = S - \sum_{i \epsilon \{L,R\}} \frac{|S^i|}{|S|} S^i \tag{3}$$

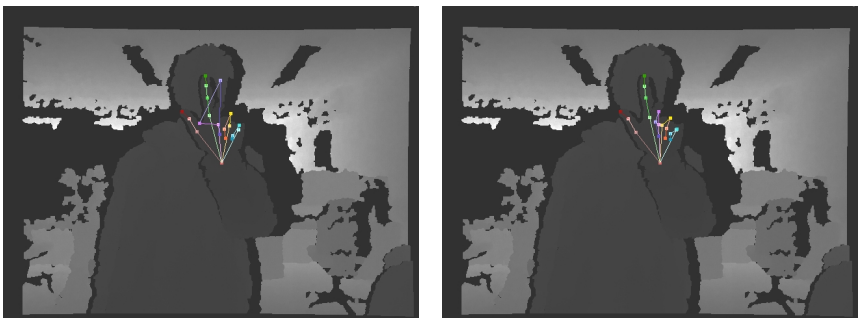
Finally, each split node chooses the best combination of feature and threshold which maximizes the information gain.

Regression of joints positions at leaf nodes Unlike a regular classification tree which stores the discrete probabilities of each class at the leaf node, we store 3D offsets for each joint (i.e regression of joint position). However, voting joint position from long distance is not reliable, hence the votes which do not satisfy a distance threshold criteria are discarded. The details of leaf node training and testing techniques are defined below.

Training: The split nodes of a decision tree are learned using the classification technique described above. The voting offsets for each joint in each leaf node is learned separately. The ideal scenario is to use all voting offsets of training pixels for offset learning; however, it is practically difficult if not impossible. That is why *reservoir sampling* [18] with size 400 is used for offset vote collection. In the leaf node l the voting offset Δ for the joint j is defined by $\Delta_{lj} = P_j(x, y, z) - P_d(x, y, z)$, where P_j is the ground truth point in the 3D space for joint j and P_d is the unprojected point of a given depth pixel in 3D space. Then, the voting offsets are clustered using mean-shift algorithm. Similarly to [10] two voting offsets from the largest clusters are used and the weight w_{lj} is formed using the number of elements in the cluster. The training bandwidth b_t and the voting threshold λ_t are learned using grid search and are the same for all joints.

Joint Inference: In the testing phase, absolute joint proposal points are collected by compensating learned voting offsets form all depth pixel being tested. The weight w_{lj} of the proposal points are reweighted using depth value of the pixel as there are fewer pixels for objects further from sensor. Then the mean-shift mode finding algorithm is applied using the highest $N = 500$ weighted joint proposal points which satisfy the test time threshold criteria λ_j for each joint j. The bandwidth b_j and threshold λ_j for each joint is learned using grid search.

Learning parameters The training time voting threshold and mean-shift bandwidth, and test time per-joint voting threshold and mean-shift bandwidth



(a) Using discriminative technique (b) Propose Technique

Fig. 2. The example demonstrate the benefit of combining discriminative and descriptive model (MRF).

parameters are optimized independently. Normally these parameters are optimized together, but such a optimization is computationally expensive. Although this can be seen as a problem, the experiments show that our technique produce state-of-the-art results for hand pose estimation. The grid search is done with cross validation of 2,500 randomly selected hand poses. We found that training mean-shift bandwidth $b_t = 0.05$ cm and the voting threshold $\lambda_t = 15$ cm produces good result. Test time mean-shift bandwidth varies between 0.33 cm to 1.85 cm. Surprisingly, test-time voting thresholds varied significantly, from as low as 1.99 cm to 8.75 cm high. The used values of test-time bandwidths and thresholds are as follows (in cm; order: palm joint, thump metacarpals to little finger tip),

Bandwidths	0.68, 0.33, 0.45, 0.87, 1.85, 0.33, 0.73, 0.35, 0.75, 0.33, 0.92, 0.8, 0.92, 0.33, 0.43, 0.46, 1.2, 0.34, 0.33, 1.06, 0.68
Thresholds	6.31, 8, 3.78, 2.0, 2.04, 8.75, 4.77, 3.8, 2.96, 7.31, 3.08, 3.18, 2.84, 7, 3.6, 2.8, 1.99, 5.85, 3.68, 2.43, 2.49

3.3 Hand Tracking

3D hand pose estimation (ref. section 3.2) would be an ideal solution for hand tracking, which could easily overcome the problems of tracking failure and initialization. Unfortunately, due to the depth sensor noise and high inter-finger occlusion pose estimation fails. To improve the hand pose estimation hand-parts kinematics and temporal motion constraints are incorporated. In the initialization step, our technique expects the hand approximately in the center of the frame. When hand joints scores are more than a threshold value in pose estimation (ref. section 3.2), we start initializing hand parts length for the next 30 frames. This gives us the hand-parts length prior. Then, we apply MRF module, which incorporates joints proposals of pose estimation module (ref. section 3.2), hand-parts prior and temporal constraints as described below,

Temporal Coherence We add two additional joint proposals j_{t-1} (last position) and $j_{t-1} + v_j$ (projected position) with $s_{t-1} * R_l$ and $s_{t-1} * R_p$ scores respectively for 50% lower scoring joints. Where, v_j is the velocity of the joint j and $R_l = 0.4$ and $R_p = 0.5$ are the weights of last position and projected position scores respectively. Experimentally we found that assigning higher weight to projected position than to the previous position produced better results. Also, increasing last and projected positions weights provide some stability against noise but performs poorly under high occlusion and large motion. Some parts of the depth image are corrupted by noise, hence optimizing the hand-pose estimation only using joints proposals from pose estimation does not produce good results and addition of the temporal coherence feature improves our technique.

Joint Potential The joint/unary potential ϕ is defined as

$$\phi_i(u_i) = \frac{1}{1 + e^{-P(s)}} \tag{4}$$

where $P(s) = \frac{s}{\sigma_s}$, s is the score of the joint position hypothesis and $\sigma_s = 0.015cm$ is the score noise as well as normalization factor.

Kinematic Constraints The structural connection between hand joints are modelled as *kinematic constraints*, which is defined as

$$\psi_{i,j}(u_i, u_j) = e^{-(\frac{diff}{\sigma_{diff}})} \tag{5}$$

where $diff$ is the difference between hand-part length prior and the distance between connected joints (ref. MRF model fig. 1). $\sigma_{diff} = 10cm$ is the noise of a hand-part length.

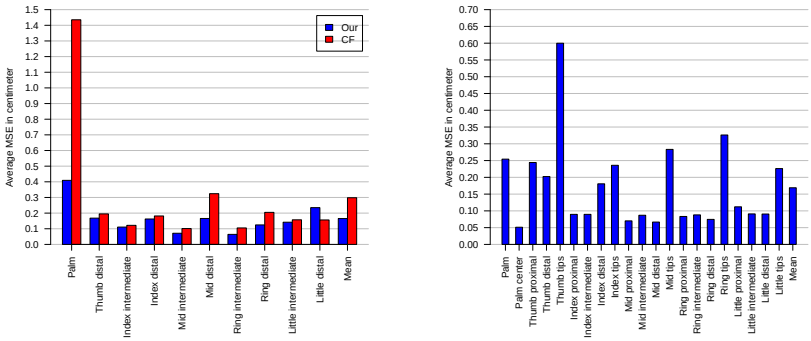
We used message passing algorithm, *belief propagation* [13,20], to maximize the likelihoods defined in equations 4 and 5 . Joint i with number of neighbours $N(i)$, sends a message to neighbour $j \varepsilon N(i)$ when it gets messages from all nodes except j. The message from i to j, $m_{i \to j}(u_j)$, for a joint proposal u_j is defined as

$$m_{i \to j}(u_j) = \sum \phi_i(u_i).\psi_{i,j}(u_i, u_j) \prod_{k \varepsilon N(i) \setminus j} m_{k \to i}(u_i) \tag{6}$$

Finally, the belief of a joint proposal is define as

$$b_i(u_i) = \phi_i(u_i) \prod_{j \varepsilon N(i)} m_{j \to i}(u_i) \tag{7}$$

We have used only one maximum scoring joint position for 50% higher scoring joints (fixed nodes) and the five higher scoring joint positions (modes of mean-shift) plus two additional positions as explained in the temporal coherence section for the remaining 50% nodes. This strategy allows us to give more weight to the discriminative technique and recover the best possible hand pose using kinematic constraints and temporal coherence. We do not use positional constraints for the joints as that would violate the tree-structure of the MRF model, also the processing time of *belief propagation* [20] would increase from 2.5 milliseconds (ms) to 6 ms per frame on a single core 3.33 GHz processor.

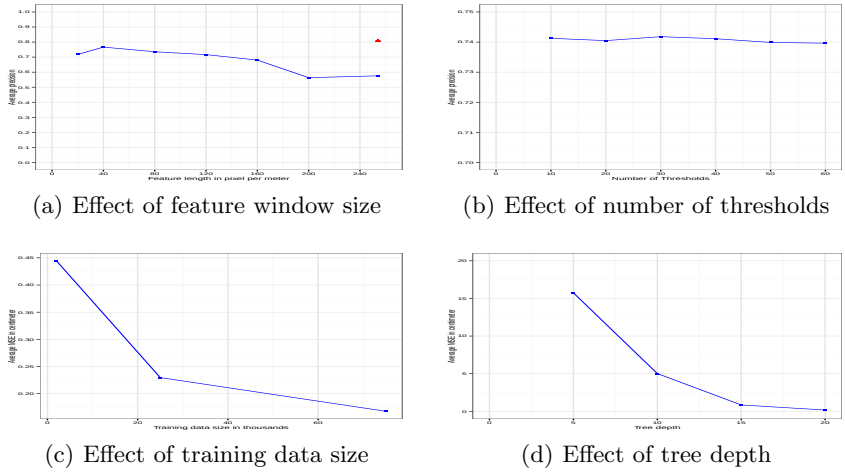


(a) MSEs of our vs CF techniques (b) MSEs of our technique

Fig. 3. Panel (a) compares the mean square error (MSE) between our regression forest and classification forest (CF) techniques for hand pose estimation. Panel (b) shows the mean square error of our technique for hand pose estimation using 150 thousands data for various hand poses.

4 Results and Discussion

The data for all experiments are artificially generated as mentioned in section 2. While this can be seen as a drawback but the experimental results show that it works equally well on the real data during the testing phase. In this section, first we compare regression forest and classification forest techniques [5] to justify the use of the regression forest in our proposed framework. For all experiments, we trained three trees with different poses- spreading, grasping, one finger, two fingers, three fingers, four fingers, pointing with index finger, shooting pose with thumb and index finger in wider directions (rotation angles in degree- along x-axis: -30 back to 85 front; along y-axis: -85 to 85; along z-axis: -85 to 85). **Comparisons with classification forest based technique:** we have compared the current state-of-the-art hand pose estimation technique [5] with our proposed pose estimation technique for straight hand spreading poses. Three classification trees are trained with tree depth 10 and three thousands data. As all hand parts are visible, there is no need for an occlusion handling module such as an artificial neural network used by authors in [5]. Transformation matrices are learned to push joints prediction from the surface to inside positions. Classification trees are common for both techniques, hence our technique inherits the same advantage and disadvantage created by the above mentioned experimental conditions. The *mean square errors* (MSEs) are presented in figure 3(a). We did not compare proximal joints because proximal joints are in the middle of the hand-part regions from back side of the hand but not the same case from front



(a) Effect of feature window size (b) Effect of number of thresholds

(c) Effect of training data size (d) Effect of tree depth

Fig. 4. Panel (a) hand-parts pixels classification precision plot against various window sizes. The triangle is precision of the *feature pool* technique (use of most frequently used features by split nodes of classification trees from all windows sizes). Panel (b) shows MSEs for range of thresholds. Panel (c) shows MSE for various training data size. Panel (d) shows MSEs of different tree depths.

and this condition is more favourable to our technique, which uses voting offset rather than finding the center of probability mass function in mean-shift [5]. The proposed hand-pose detection technique clearly outperforms classification forest based technique in estimating the positions of joints except distal joint of the little finger. We have also noticed that when the hand-region is large and the shape is not regular in all directions the MSE is higher.

Feature pool: We observed that there is a positive correlation between pixel classification accuracy and the regression of joint as our regression forest shares the same classification split nodes. Hence, for the feature selection we have measured pixels classification precision. First, we used 3200 uniform features with different window sizes, the results are presented in figure 4(a). We found that even though larger length features are useful, they are more sparse as the number of features is restricted to 3200. Hence, the performance decreased. Then, we selected the same number of the most frequently used features from all experiments and got better results. Such a pool of features is used for all other experiments.

Unlike other parameters, the number of thresholds has almost no or little effect on the accuracy (ref: figure 4(b)) of the hand pose estimation. We found that with higher numbers of training images, thresholds between 30-35 work better. In contrast, the value of the tree depth has significant effect in the accuracy. Due to the computational and memory limitations, we restricted the depth of trees

Fig. 5. Examples of hand pose estimation. The top row is Kinect depth images and bottom two rows are artificial data.

to 20 (ref: figure 4(d)). We noticed that with a tree depths lower than 10, the classification forest technique performs better than the regression forest based technique. The training dataset size is dependent upon the variation of hand poses as well. The proposed technique works reasonably well when the dataset contains more than 30 thousand training images (ref: figure 4(c)). Due to the limitation of computational resources we could not train our framework with more data. And we believe that the accuracy of our framework can be increased with more training data (ref. *effect of data size* fig. 4(c)).

The MSE of our technique is plotted on figure 3(b). Finger tips are likely to be occluded in certain poses more than other hand-parts, hence MSEs of finger tips are higher. Figure 4 shows few examples of hand pose regression. These results clearly show how well the proposed technique was able to capture the 3D pose of the hand. Figure 2(b) shows the benefit of proposed technique over discriminative techniques. Proposed technique could not recover good hand pose when the noise continues for more than 4-6 frames and there are strong false positive joints proposals. Also, proposed technique fails on unseen hand poses in training time. We did not provide forward movement of single finger in training time, hence in demo it fails in those situations.

5 Conclusion

This paper presented a markerless 3D hand tracking framework, which efficiently combined discriminative and descriptive techniques. Giving more weight to discriminative technique by fixing high scoring joints/MRF-nodes taking full advantage of discriminative technique. Added temporal coherence enables to recover joints position from noises. Modelling hand joints as unary potential of the MRF model captures hand-parts length variation efficiently. We have demonstrated that the regression forest based technique outperforms the classification forest based technique for hand pose estimation. In our knowledge, the proposed technique is the first disjoint evidence technique which does not requires additional occlusion handling module for hand pose estimation. We have demonstrated that the feature pool technique is simple yet efficient way of generating features from larger feature spaces. In future, we aim to build a complete 3D hand tracking and gesture recognition system for better HCI experience.

References

1. Wang, X., Zhang, X., Dai, G.: Tracking of deformable human hand in real time as continuous input for gesture-based interaction. In: International Conference on Intelligent User Interfaces, ACM (2007) 235–242
2. Caridakis, G., Karpouzis, K., Drosopoulos, A., Kollias, S.: SOMM: self organizing markov map for gesture recognition. Pattern Recognition Letters **31** (2010) 52–59
3. 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
4. Wu, Y., Lin, J., Huang, T.: Analyzing and capturing articulated hand motion in image sequences. PAMI **27** (2005) 1910–1922
5. Keskin, C., Kirac, F., Kara, Y., Akarun, L.: Real time hand pose estimation using depth sensors. In: ICCV Workshops. (2011) 1228–1234
6. Sudderth, E., Mandel, M., Freeman, W., Willsky, A.: Distributed occlusion reasoning for tracking with nonparametric belief propagation. NIPS **17** (2004) 1369–1376
7. Martin, De La G., M., Paragios, N., Fleet, D.J.: Model-based hand tracking with texture, shading and self-occlusions. In: CVPR. (2008)
8. Hamer, H., Schindler, K., Koller-Meier, E., Gool, L.V.: Tracking a hand manipulating an object. In: ICCV. (2009) 1475–1482
9. Oikonomidis, I., Kyriazis, N., Argyros, A.A.: Efficient model-based 3d tracking of hand articulations using kinect. In: BMVC. (2011)
10. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV. (2011) 415–422
11. Breiman, L.: Random forests. Machine learning **45** (2001) 5–32
12. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. PAMI **24** (2002) 603–619
13. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory **51** (2005) 2282–2312
14. Kinect: Xbox kinect (2013) [Accessed 10 March 2013].

15. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR. (2011)
16. Jones, M.J., Rehg, J.M.: Statistical color models with application to skin detection. IJCV **46** (2002) 81–96
17. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: CVPR. (2009) 1022 –1029
18. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software **11** (1985) 37–57
19. Army, U.S.: Human engineering design data digest. US Army Missile Command, Redstone Arsenal, AL (1978)
20. Mooij, J.M.: libDAI: A free and open source C++ library for discrete approximate inference in graphical models. Journal of Machine Learning Research **11** (2010) 2169–2173