Real-time Hand Tracking - A modular approach
DL 2018 Project

Luca Cavalli
luca3.cavalli@mail.polimi.it

Gianpaolo Di Pietro
gianpaolo.dipietro@mail.polimi.it

Michele Bertoni
michele2.bertoni@mail.polimi.it

Matteo Biasielli
matteo.biasielli@mail.polimi.it

Mattia Di Fatta
mattia.difatta@mail.polimi.it

Politecnico di Milano

Abstract

We propose a real-time hand tracking system. Its final objective is to track a 3D model of a hand, decomposed in 21 joints, running on common hardware. Real-time hand tracking is not a novel task: state of the art models achieve very good accuracy, but these approaches can run in real-time only on very high-end hardware. We break up the problem into steps to achieve adaptive trade-off between resource consumption and accuracy. We created a new dataset, to deal with new aspects, i.e., visibility of a joint and side of the hand. Up to now, with this approach we built a hand localization system showing reasonable accuracy, while being computationally cheap.

1. Introduction

Real-time hand tracking has the potential to become a new disruptive pervasive human-machine interface that takes advantage of the incredible dexterity and complexity of hand movements, but to be adopted on a large scale it needs to lower its computational requirements: current approaches need to employ very expensive hardware (Intel Xeon CPU + NVIDIA GeForge or Titan GPU).
Our objective is to decrease resource consumption while keeping reasonable accuracy, experimenting different approaches with the freedom given by a full-stack development. We built light models, compromising accuracy with resource requirements, we organized them in pipelines and we used lightweight interpolation to adaptively change the quality of service/performance compromise on different hardware conditions. As we deal with new features for 3D model inference, we built our own dataset, with all the needed labels.
Currently only the hand localization task has shown acceptable results, while we are still working on the joints identification model.

2. Related work

The problem of tracking hand position from images is very active for its possibilities in developing new human-machine interfaces. The most common setting is the use of single RGB-D images or multiple RGB images to infer key-point locations to track the movement of one hand. Also, some attention is put to correctly track hands over large occlusions, like in *Mueller et al.* [6].
Very recently, a new work [5] achieved very good precision with single RGB images, opening a new gamma of applications in video analysis, still claiming real-time performances in a desktop environment.
Common methods for tracking involve the use of pixel classification and clustering algorithms to fit the hand images with a gaussian mixture model of the hand volume [7, 10, 8], while more recently CNNs are adopted to directly infer 2D and 3D joint locations [5, 6].
Data collection and labeling recently switched from manual [9, 8, 6] or automated [3] labeling of real images to the use of generative models for realistic synthetic hand images [5, 6] used in combination with real images. All works achieving real-time that declare the hardware used for latency measurements actually use very high-end hardware (usually an Intel processor of the Xeon family coupled with the latest NVIDIA GPU) [6, 8].
In our project we want to answer the need of lowering computational requirements more than the last real-time approaches do, to run in real-time on different hardware plat-

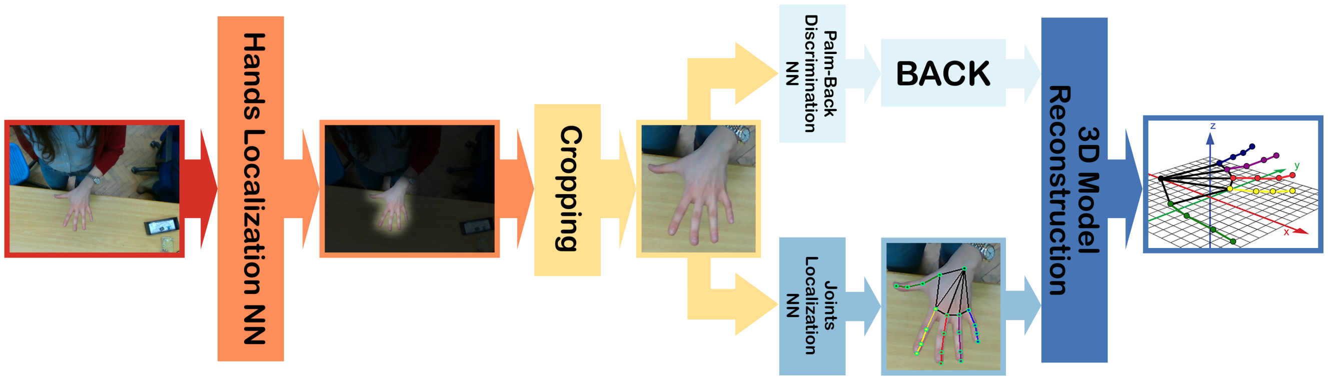

Figure 1. The end-system architectural scheme from the RGB image of a hand to its 3D model

forms under varying workload.

3. Proposed approach

To achieve good performances, we split the task into three main functions: hands localization (supervised), joints localization (supervised) and 3D reconstruction (unsupervised). Each step is considered as a function of time and is independently enclosed in a wrapper module: we can assign and control specific working frequencies for each one of them, decoupling their execution. Each wrapper needs an input source, then dynamically pipelines its assigned operation to achieve the target working frequency, and finally uses results to offer to the outside a lightweight interface from which to sample function values at any time instant.
The input of our system is an image stream, from which we first localize and crop hands. A small set of lightweight features is extracted from each located hand, to recognize it in the next frame, and a time-coherent ID is assigned.
If the side of the hand of each ID has not been firmly classified yet, we determine with an independent module whether subsequent frames are seen by palm or back, deducing if it's a left or right hand. The next module (i.e., joints localization) is able to find, for each hand crop, the 2D location and visibility of the joints. With relocation information of the crop, we can then reconstruct the position of the joints into the original 2D image space and we can geometrically build a 3D model in camera space of the hand skeleton.
The overall system architecture and dependencies are visible in Figure 1.

3.1. The Function Wrapper Module

A core factor to achieve cheaper real-time is the utilization of resources. We consider models as functions that take a non negligible time to be computed, that have to be computed with a controllable working frequency. The modern trend on hardware growth is towards thread level parallelism, thus we must enclose our models into some abstraction layer that is able to dynamically increase throughput and hide latency through adaptive depth pipelining and output interpolation. In our system the model is enclosed into a scheduler that performs a forward inference with a given frequency. To maintain the frequency, the scheduler keeps a running average of the forward inference time and decides the depth of a multithreaded pipeline needed to keep that pace. If the requested frequency is too high and the processor starts to execute the threads in time sharing, the average execution time grows and triggers an adaptive reduction of the model execution frequency to bring the pipeline back to real parallelism. The outputs of each run are labeled with their reference time and collected for interpolation. The output interface of this abstraction layer is the interpolation module that collects the most recent N samples and uses them to answer requests of the values of the function at specific time instants, using linear interpolation with polynomial features of order M. Using low order polynomial features makes the interpolation procedure very cheap and an accurate predictor when the latency is not too high and the function is not too irregular over time. Through pipelining we increase throughput up to the parallel capabilities of the host machine and through interpolation we implement the abstraction of a continuously available and cheap function that can be queried for any time instant at any moment thus hiding latency at the cost of accuracy.

3.2. Data collection

Most of the available datasets of RGB-D labeled images of hand joints offers frames captured in few different settings and from few repeated subjects, using a variable number of joints and resolution. Although there is a good variety of them, we decided to build our own dataset to label and consider different unconventional aspects of frames like the direct visibility or occlusion of a particular joint, being also able to reconstruct long sequences of connected frames with a well known framerate. The visibility information for example would enable fast refinement of the final 3D model with depth information, while guarantees on framerate would allow building more stable time-recurrent

models. This does not exclude the use of known datasets in some stages of the work: indeed we used egohands [1] to augment our hand localization dataset and SynthHands [6] to test it.
To build our dataset we shoot short videos of one hand from ego, third person view and subject perspective, with either open, clean or cluttered backgrounds with the Intel RealSense ZR300 RGB-D camera. We collected hands of different shapes and colors, moving at different speeds. A minority of videos was captured also while manipulating small objects causing additional hand occlusions. Some videos were captured using a clean blue background to augment them with different backgrounds. We also registered outdoor videos in different light conditions and with no hand at all to be used just as animated backgrounds for blue-screened hands. Our dataset currently holds 35 videos for a total of 7019 manually labelled frames overall and with linear interpolation we can reach a total number of 10055 usable frames.

3.3. Data labeling

For each captured frame we need to reconstruct all the information content about the hand. The main description of a hand is a sequence of 21 joint locations, one for the wrist position and four for each finger; for each joint we need to know its 2D location on the image and its depth, hence the need for the occlusion information in order to be able to distinguish a correct depth value on the D image from a wrong one. Additional useful information is the fact that the hand is right or left and seen by palm or back; right-left distinction is useful for geometrical 3D model reconstruction and is very easy to label as it is a property of the entire video and can be diffused to each frame, while palm-back can be geometrically inferred from the right-left distinction and the joint 2D locations and has proved to be an easier class to be inferred by a CNN.
As right/left class is labeled by video, the per-frame needed labels are its image joint locations and occlusion. For this reason we built a small website (https://mdifatta.github.io/) to ask for help in labeling. On this website users can receive a sample of our choice, zoom on it and select each one of the 21 joint locations distinguishing between visible and not visible. When performing manual labeling it is important that the users have best possible visual feedback of the data they are providing to us to reduce the intrinsic noise that will be generated by different people, therefore a skeletal model of the hand appears gradually while each joint is inserted with different colors for each finger to give an immediate idea of how the described hand would appear. At last they can submit the labelled frame and eventually get one more. Moreover, the presence of a ranking listing the top contributors by nickname proved to motivate some users to help us more.
Manual labeling is a very long process that can provide labels only for a limited number of frames. In order to efficiently use the effort of our contributors, we propose the next frame by minimizing the maximum distance between two consecutive labeled frames. All unlabeled frames can be temporarily labelled by linearly interpolating the two nearest labels. This is possible because of the continuity of the space of our labels, and can be extended also to the occlusion flags modeling the uncertainty of the occlusion itself. The nearest the two extreme frames, the most accurate the interpolation will be. For this reason by minimizing the maximum distance between two consecutive labelled frames we are minimizing the maximum interpolation error. This way we can get acceptable results even by labeling one frame every four, and good results labeling only half of the frames. One more peculiar problem of manual labeling is the intrinsic noise over the produced labels, and the presence of some evident outliers in the labels due to human errors. To address this problem we built a custom video player that allows us to play our videos at different speeds visualizing also all the direct and interpolated labels provided by users. Finding evident outliers is a fast operation playing the video in natural or even slower speed because each video is not longer than a few seconds; once found, the player enables us to either discard some frame labels to be requested to the next users or to move the label points and perform the interpolation with the new points. Sometimes it is also useful to slightly correct the interpolation result itself on fast movements and keep the correction as new labelled data. When a video appears to have high quality data it can be fixed and never proposed to users anymore to give more space to others.
Labeled data is then converted into an intermediate format to enable merging it with different datasets and to optimize training speed. Moreover by standardizing data format we can merge more than one dataset into a single standard, uniform and larger source of data.

3.4. Heatmap Loss

Most of the models we propose are based on heatmaps to locate either the hand or its joints. As location heatmaps of small objects are mainly black, a common problem of all these models is the local minimum of a fully black output, which proved to be very attractive. This is clear when using the classical loss for heatmaps, the RSS, as all models always ended up converging from random output to black output. To balance the contributions of black and white we elaborated a custom loss L_h

$$L_h = E_H[dm(p_e - p_{gt})w(p_{gt}, H_{gt})] \tag{1}$$

where E_H is the expectation over the whole heatmap, p_e is the estimated label of a pixel, p_{gt} is the ground truth corresponding pixel label and H_{gt} is the whole ground truth heatmap. The dm function is a mapper of the estimation error difference, while the w function assigns an importance weight to each pixel based on the whole heatmap. The classic RSS loss can be obtained by this with the choices:

$$w(p_{gt}, H_{gt}) = 1 \tag{2}$$

$$dm(\epsilon) = \epsilon^2$$

To have explicit control over the different penalization of false positives and false negatives we propose an asymmetric function dm depending on the real hyperparameter δ:

$$dm(\epsilon) = \frac{\delta}{2}\epsilon^3 + (1 - \frac{\delta}{2})\epsilon^2 \tag{3}$$

This ensures that false positives ($\epsilon = 1$) weight 1 while false negatives ($\epsilon = -1$) weight $1 - \delta$.
Moreover we want to weight each pixel proportionally to the square of its distance from the average of the heatmap $w(p_{gt}, H_{gt}) \propto (p_{gt} - E[H_{gt}])^2$. To have more control over weights we introduce the real hyperparameter wp, white-priority, to move the importance of the positive labels against the negative ones: we bias the average $\mu = E[H_{gt}]$ with the formula:

$$\mu_{biased} = \sigma(-log(\frac{1}{\mu} - 1) - wp) \tag{4}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Notice, for $wp = 0$ we have $\mu_{biased} = \mu$. We then use μ_{biased} instead of μ to apply the bias on the loss. This weighting function as an undesired side effect would give overall more importance also to heatmaps with more variance, thus weights must be normalized:

$$w(p_{gt}, H_{gt}) = \frac{(p_{gt} - \mu_{biased})^2}{E[(p_{gt} - \mu_{biased})^2]} = \tag{5}$$

$$= \frac{(p_{gt} - \mu_{biased})^2}{Var(H_{gt}) + (\mu - \mu_{biased})^2}$$

3.5. Hand localization models

In this section we present some models that we tested for the task of hand localization.

Eta-net Given the similarity of the hand localization task with the image segmentation task, we took inspiration from segmentation models like the U-net and tried a similar approach. We convolved the RGB input down to a 16 times smaller dimension and then deconvolved it up to the desired heatmap dimension (1/4 of the input size, thus not being a perfect U-shaped network).

Mobilenet transfer Mobilenets are models intended for real-time performances on embedded devices [4]. We used the first 24 layers of a mobilenet pretrained on the Imagenet dataset, and appended 7 convolutions to output the hand presence heatmaps.

3.6. Joint localization models

In this section we present some models that we tested for the task of joint localization.

Gradient Injection net The Gradient Injection model is based on the idea of parallel learning paths and early loss application. Between each pair of pooling layers we put multiple parallel convolution paths with different lengths: this improves the complexity and power of the model but makes propagating gradient more difficult. To help the diffusion of the gradient we added a second early output applying the L_h loss near the first layers to directly enforce feature significance in the first layers.

Vector field joint inference The vector field model tries to simplify the task of joint localization by splitting it into two parts: finger area/direction distinction and final joint localization. The idea is inspired by a work on body tracking [2] where body parts are localized independently with a similar technique. The model takes as input cropped RGB images of hands and produces as output per-pixel vector fields of the direction and presence of each finger and the joint positions afterwards.

Joint regressor A completely different approach from heatmap segmentation is coordinate regression, sometimes coupled with heatmaps themselves [5, 6]. We built a simple CNN regressor using cropped RGB images of one hand to output the coordinate vector of its joints.

3.7. Miscellaneous models

In this section we present other models we trained to perform the task of right/left classification needed for 3D modeling.

Plain CNN palm/back classifier The palm/back classifier is the simplest model we built. It takes a cropped hand RGB image as input and classifies the it as viewed by palm or back of the hand. This model is a classical sequential CNN classifier that takes as input a plain RGB cropped hand and outputs a binary prediction interpreted as the palm/back class.

3.8. Space 3D Model reconstruction

The objective of the model reconstruction is to find the transformations (wrist translation and rotations of mobile

joints of the hand) that minimize the discrepancy between the image-projected joint positions and the inferred 2D joint positions. We considered the inverse problem of finding the transformations that project the model points at the minimum distance from the rays projected from the origin and corresponding to each 2D image estimated joint position.
We experimented several methodologies for 3D reconstruction, the best performing is through geometrical interpolation. We need to start from the basic model of some hand at rest in standard coordinates to have a starting position with correct proportions, this model can be a standard fixed model or the result of some calibration procedure on the user's hand: the better the model, the better the reconstruction accuracy. Depending on the hand being classified as right or left, the base model may need to be mirrored. The first step is fitting the basic rigid triangle between wrist, base index and base pinky with fixed side lengths between the three corresponding rays. This problem can be solved numerically and has four solutions, reducible to two by central mirroring: these two solutions correspond to the hand facing up or down and can be both found by restarting the procedure several times.
Once the three base points have been found the corresponding base translation and rotation matrix can be easily computed to move all the other points of the model to position. Each finger rotation can be found by a constrained nonlinear optimization problem: fixed the center of the rotation, the direction around which the joint can rotate and the plane on which the main rotation is performed, we look for the rotations on the plane and perpendicular to the plane that minimize the distance of the relative point on the sphere from the projected line of the corresponding point estimation, keeping the angle into constrained intervals to allow only natural positions. This problem can be solved numerically for each mobile joint, starting from the solution to the unconstrained relaxed problem which can be easily solved analytically. Also when the plane-perpendicular allowed angles are small enough (as for the terminal tip finger joints) we can drop a dimension of the search space, allowing much faster execution.
The obtained model can finally be refined with depth values when the joint is labeled as visible: when the measured depth is near the estimated camera distance it is accepted as a more accurate estimation, otherwise it is considered an outlier and rejected.

4. Experiments

Hereafter we describe the experiments settings and results to validate our approach, both on the side of accuracy of the predictions and on the side of timing performances.
Unfortunately, we only experimented about the task of hand localization as no model for joint localization for now could give seemingly appreciable results.

4.1. Training Setup

To train our models we used a common training infrastructure that describes some important features of our validation environment. Training and validation sets are ensured to have no two frames of the same video belonging to different sets, so that video frame-dependencies do not influence validation. We use early stopping with its patience hyperparameter and checkpoints. Data samples are augmented online by shifting their HSV components by random values: performing random shifts at each iteration greatly helps training to generalize over different hand colors and light conditions. The probability to perform augmentation each time a sample is provided to the model for training is controlled by a hyperparameter. Moreover we always display sample images from both training and validation set to help the hyperparameter selection procedure described in section 4.2. For all models we used the Adam optimization algorithm.

4.2. Hyperparameter tuning

A collection of the most important hyperparameters of our model can be found in Table 3. An automatic hyperparameter tuning algorithm is far too expensive and takes far too long to evaluate the best configuration for a given model. Therefore we used a manual approach: we analyzed the evolution of the network output over time during training in order to diagnose the problems of the run and manually choose the needed change to hyperparameters. On each run we registered on a form the chosen hyperparameter values and the measured performances with both exact and fuzzy metrics to have a general overview of the explored points in the search space.
We must also consider that the hyperparameter-dependent loss we employed could not be used at all to compare the performances of different configurations, thus task-dependent metrics had to be designed each time.

4.3. Model validation

All the proposed models were built and trained with Adam optimizer using the manual hyperparameter selection procedure described in section 4.2. Here we show the best results for each proposed model. For the hand localization task both the eta-net and the mobilenet transfer performed well, but the latter performed better both on validation and inference speed (measured in Tables 2 and 1). Also the miscellaneous task of palm/back classification achieved acceptable results with a validation accuracy of 80% on our dataset, while direct right/left inference could not do better than 50%. No joint localization model showed acceptable results even on the training data, we think that this task needs more complex and specialized models than the ones tested.
A python numpy implementation of the 3D reconstruction

Processor	Lat	Freq	Freq w/o wrap
Intel Celeron CPU 877	960ms	1.5Hz	1.2Hz
Intel i7 7500U	360ms	4Hz	2.5Hz
Intel i7 7700HQ*	90ms	66Hz	9.2Hz

Table 1. Timings for live hand localization on different processors
* with GeForce GTX 1050 GPU

Metric/Threshold	0.2	0.4	0.6	0.8
Centroids Dist	6.17px	6.29px	6.26px	6.19px
Centroids Dist Std	3.59px	3.49px	3.42px	3.38px
Precision	0.83	0.85	0.83	0.75
Recall	0.42	0.34	0.30	0.27

Table 2. Accuracy metrics for the mobilenet transfer model on the SynthHands hand localization task, varying the positive threshold. Pixel values are taken for heatmaps of size 56x56.

geometric model takes 18ms to run on an Intel i7 7700HQ with no parallelization at all, providing good results and failing only on very convoluted positions. Its execution time can be improved but in the current configuration it does not represent a bottleneck.

4.4. Model evaluation

To ensure that the test set is not at all correlated with the training and validation sets, we used a third dataset, the SynthHands [6], to assess our measurements. On Table 2 we show the performances of the transfer mobilenet model referred to two metric classes:

- the average euclidean distance in pixels of the centroids of the two heatmaps (expected and predicted) and its standard deviation;
- precision and recall of the heatmaps. Precision is the number of true positives divided by the total positives. Recall is the true positives divided by the ground truth positive instances.

From the precision and recall analysis we can see that the model is very conservative in its predictions. These two metrics together allowed us to reliably determine what is the error we make when locating hands and helped us developing a fit cropping strategy that overcomes the error, which is now acceptable. This strategy consists in selecting the threshold that minimizes the distance and enlarging the cropped area to contrast the effect of low recall.

4.5. On-field testing

As a final system-wide validation step we created an application for hand localization through bounding boxes on the webcam image stream. The stream of images itself is very fluid as it is not blocked by the inference time of the

Hyperparam/Model	transfer mobilenet	eta-net
early stopping patience	10	10
learning rate	10^{-4}	$2 \cdot 10^{-5}$
wp	-2	-0.7
δ	1.5	15
dropout	0.5	0.15
augmentation prob.	0.4	0.25
L2 weight decay	10^{-5}	10^{-5}

Table 3. Selected hyperparameter values for the proposed hand localization models

hand localization network, while the bounding box placement is at times noisy but correct overall. We measured the full load operating frequencies and latencies of our system and reported results on Table 1 . Operating at lower frequencies than full load and masking the lower granularity with cheap interpolation lets the whole machine to operate other tasks more fluidly, which is our end-objective. Without interpolation the system clearly exposes the model working frequency to the outside, but output is more stable as errors don't influence it over time.
Interpolation with prediction to hide latency proved to partially mask the system working frequency, but it magnifies the occasional errors extending them over time. The result with interpolated prediction appears very unstable and inaccurate, while using interpolation without prediction gives slightly more stable results but doesn't hide latency.

5. Conclusion

We built a hand-localization system which regulates resource usage by leveraging on quality of service. Joint localization proved to be a much harder task to solve, which probably needs more specialized models. As future work we plan to build a joint localization model adding the camera depth information, available on our dataset, and to integrate it with the hand localization to put the full system to work and make more complete tests on performance control.

Acknowledgements

We owe great thanks to anybody who helped labeling our data: the main contributing nicknames are available at https://mdifatta.github.io/thanks.html . We also want to thank Alberto Bellini for his help on some components of the website. Special thanks go to Microsoft for offering free cloud computing power on Microsoft Azure to train our models. We thank Luca Morreale and Ewerton Lopes for their precious feedbacks, corrections and suggestions during the review of this manuscript.

References

[1] S. Bambach, S. Lee, D. J. Crandall, and C. Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 3

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 4

[3] S. Y. et al. Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. December 2017. https://arxiv.org/pdf/1704.02612.pdf. 1

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 04 2017. 4

[5] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 4

[6] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2017. 1, 3, 4, 6

[7] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[8] S. Sridhar, F. Mueller, M. Zollhoefer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 1

[9] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2013. 1

[10] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt. Real-time hand tracking using a sum of anisotropic gaussians model. In *Proceedings of the International Conference on 3D Vision (3DV)*, Dec. 2014. 1