Joint Estimation of 3D Hand Position and Gestures from Monocular Video for Mobile Interaction

Jie Song[1], **Fabrizio Pece**[1], **Gábor Sörös**[1], **Marion Koelle**[1,2], **Otmar Hilliges**[1]
[1]ETH Zurich, [2]University of Passau
{jsong|fabrizio.pece|gabor.soros|otmar.hilliges}@inf.ethz.ch, marion.koelle@uni-passau.de

ABSTRACT
We present a machine learning technique to recognize gestures and estimate metric depth of hands for 3D interaction, relying only on monocular RGB video input. We aim to enable spatial interaction with small, body-worn devices where rich 3D input is desired but the usage of conventional depth sensors is prohibitive due to their power consumption and size. We propose a hybrid classification-regression approach to learn and predict a mapping of RGB colors to absolute, metric depth in real time. We also classify distinct hand gestures, allowing for a variety of 3D interactions. We demonstrate our technique with three mobile interaction scenarios and evaluate the method quantitatively and qualitatively.

Author Keywords
mobile interaction; gesture recognition; machine learning
ACM Classification Keywords
H.5.2 User Interfaces: Input devices and strategies

INTRODUCTION
We are currently witnessing how ultra-mobile devices such as smartwatches and head-worn displays (HWDs) are rapidly becoming commoditized. However, how the user will interact with these ever richer forms of virtual information is becoming a pressing issue. Current commercial offerings, such as Google Glass, Epson Moverio, or other AR research prototypes commonly leverage smartphone technologies such as touchscreens or acceleration sensors for user input. This limits interaction mostly to manipulations of *2D* content – rather than *3D* information embedded into the real world.

Natural user interface (NUI) research has enabled rich, contactless 3D input using hands, fingers and the full body. NUI is now synonymous with depth sensing technologies such as Kinect or the LeapMotion controller. Most available depth sensing technologies are based on either a stereo setup or a combination of a single camera and some form of active illumination such as structured light or time-of-flight. These dependencies set hard lower bounds on the size, weight and the power consumption of such devices. Hence, most depth sensing technologies are currently prohibitive in wearables.

CHI 2015, April 18 - 23, 2015, Seoul, Republic of Korea
Copyright ©2015 ACM 978-1-4503-3145-6/15/04...$15.00
http://dx.doi.org/10.1145/2702123.2702601

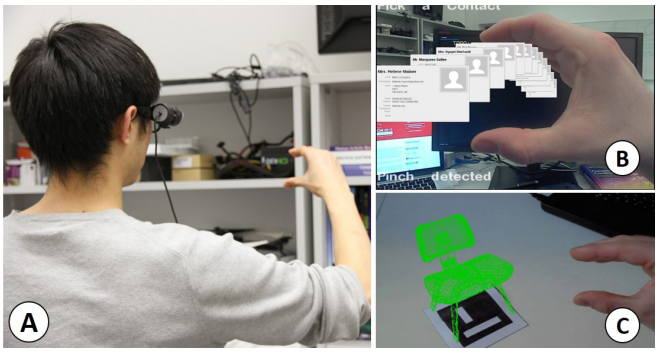

Figure 1. Joint classification of gesture and regression of metric depth from a single head-worn camera for spatial interaction. (A) User wearing a monocular RGB camera attached to AR glasses. Depth-aware gestures allow for rich interactions. (B) Invoking a contacts app by gesture and selecting a contact card by depth (C) Simultaneous control over discrete and continuous parameters in 3D apps.

At the same time almost all wearable computing devices and HWDs are equipped with RGB cameras for image capture. We leverage these and propose a novel machine-learning based algorithm that *jointly* recognizes gestures and estimates the 3D position of hands. In contrast to previous work [2], where instrumentation of the camera is required to obtain depth information, our technique uses *only* the built-in RGB camera, and hence is suitable for off-the-shelf, unmodified mobile devices, including compute-limited HWDs.

Our technique successfully copes with variation in gesture execution, and varying lighting conditions. To achieve this, we take a data-driven machine learning approach similar to those presented in [2, 13] but extend this framework to a hybrid classification-regression scheme, capable of learning a mapping from 2D images to 3D hand positions plus gestures.

RELATED WORK
Ultra-mobile interfaces have seen a lot of interest recently. In particular, researchers have tried to extend the input capabilities of small-screen devices such as smartwatches by adding touch sensitive wristbands [10] or by leveraging the skin for user input [3, 9]. Others have attempted to enable more free-form, in-air gestures using finger- [14] or wrist-worn [5] sensors. These approaches all require hardware modification, which allow for richer input but add bulk and power draw.

Researchers have also looked into expanding the input capabilities of emerging HWDs (e.g., [12]) but do not provide sensing solutions and rely on external infrastructure. More unconventional solutions to the mobile input problem include sensors placed on the tongue [11], behind ears [8], or on cheeks [12]. All these approaches require the user to wear

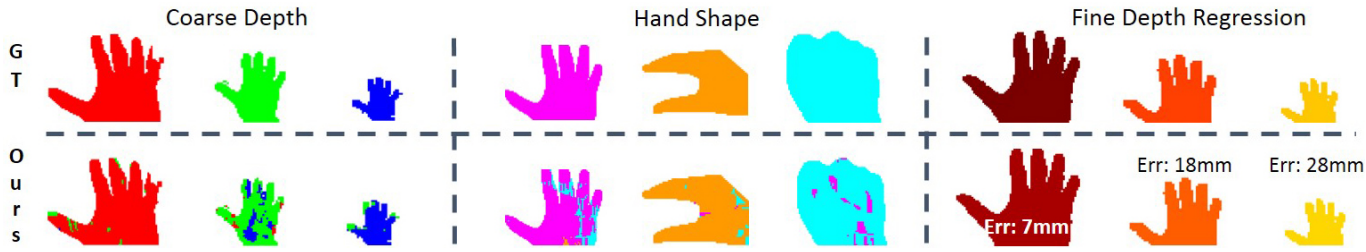

Figure 2. Classification-Regression pipeline overview. Top row: ground truth (GT) data used for training. Bottom row: posteriors outputted at each level by the forests. Left: Coarse depth classification. Middle: Hand-shape classification. Right: Fine-grained depth regression (average depth per hand shown at different depth). The inset shows the error in mm compared to the ground truth.

specialized sensor electronics and might not be feasible in real-world scenarios. Song et al. [13] recently introduced a data-driven gesture recognition approach that enables mid-air interaction on unmodified portable devices. However, the type of interactions are limited to 2D gestures.

Our approach leverages machine-learning techniques to jointly infer hand shape (gestures) and hand position in 3D from only monocular RGB imagery. Estimating depth from single images is ill-posed and a hard problem. Although several approaches exist that try to estimate surface normals or scene depth from single images. These methods rely on associating still images and ground-truth depth pairs, to then derive coarse scene depth [4, 7]. These approaches make use of complex and computationally costly algorithms, infeasible for interactive scenarios on mobile devices.

SPATIAL INTERACTION FOR WEARABLE COMPUTING

In this section we introduce our machinery for joint *classification* of discrete hand gestures and *regression* of scene depth of hands. Similarly to our work, Fanello et al. [2] learn a direct mapping between infrared intensity and depth measurements, to predict per-pixel depth values. Relying on infrared intensity fall-off, this requires mounting of IR illuminants and an IR pass filter, and as such, renders the camera unusable for other purposes while also increasing power draw.

Our method is similar to those proposed in [2, 13]. In [13] several randomized decision forests (RFs) are combined to enable the recognition of discrete 2D hand shapes or gestures. We extend [13] to jointly detect hand shapes and to regress the average hand depth. Together with the hand's centroid this gives full 3D hand position. This allows users of HWDs to interact with rich 3D graphics in a natural and direct fashion. The method relies only on 2D RGB camera imagery and works on completely unmodified mobile devices.

Cascaded Random Forests

The pipeline in [13] consists of simple image processing steps and three cascaded RF classifiers to detect i) coarse depth ii) hand shape and iii) hand parts. Our pipeline (see Fig. 2) is similar but differs in important aspects. We briefly compare our technique to the original algorithm:

Hand Segmentation

We use the same skin color segmentation method as [13] which is robust enough under normal lighting conditions. In this step the main importance is to keep the hand contour as complete as possible, whereas false positives (background noise) are handled later in the pipeline.

Coarse Depth Classification

After background segmentation we classify the image into three coarse layers of depth [2, 13]. For our purposes we are only interested in close range interaction (i.e., arms reach). By experimentation, we found that 90mm to 390mm measured from a head-worn camera to the center of the palm is a comfortable range for most users. We divide this global range into three smaller intervals: $90-150$mm, $150-240$mm and $240-390$mm (Fig. 2, left). Note how the interval ranges increase with distance. This is due to perspective effects causing the change in appearance of the users hand to decrease with distance from the camera.

This layer serves two purposes. First, it removes most of the noise coming from the simple segmentation method. Second, it constrains the variation in terms of hand appearance that the steps further down the pipeline have to deal with, as prior work has shown that classifying the scene into canonical depth regions helps in subsequent, more fine-grained depth estimation [6]. This means that overall we can use shallower trees, resulting in reduced memory footprint (as was the main goal in [13]). Analogously, this approach also allows us to increase the predictive power of the RF ensemble while keeping the memory footprint constant, for example to solve a more difficult task, as is our intention with this work.

Given an input image I with associated pixels x and a segmentation S, the forest at the top layer infers a probability distribution $p(l|x,S)$ over the three coarsely quantized depth ranges and an additional noise class, where $l = \{close, middle, far, noise\}$ are the labels. This per-pixel distribution (Fig. 2, left) is forwarded to the next layer.

Shape Classification

At the next level we evaluate each pixel x in I again to compute a gesture probability distribution $p(c|x,S)$, where c is the label for different hand shapes or gestures, in our case $c = \{splayed\_hand, pinch, closed\_hand\}$ for the three gestures. The predicted posteriors are combined:

$$p(c|x,S) = \sum_{l=1}^{L} w_l p_l(c|x,S) \qquad (1)$$

Here the output posterior p is the weighted sum over the estimates p_l of the forests trained on the three gesture classes. The weights w_l are the posterior probabilities estimated by

the first layer. Note that this means we need to run all shape classification forests simultaneously. In practice this is not a problem as the trees at depth 15 are relatively shallow. Finally we pool, and average, the probabilities across the image to attain a single value $p(c|S)$ (Fig. 2, middle).

Depth Regression

At the final level we switch from classification to regression forests. Here the goal is to map from an input pixel x in I to an absolute depth value (Fig. 2, right). Note that in contrast to the previous level here we only run one forest per gesture (they are trained only on examples of one hand shape). The continuous value $y(x|S)$ is attained as

$$y(x|c,S) = \sum_{l=1}^{L} w_l y_l(x|c,S) \quad (2)$$

Where l is the coarse depth level and weights w_l are again the posteriors from the first layer.

Prediction and Features

For prediction we follow [13] as closely as possible, with the predictions for $p(l|x,S)$, $p(c|x,S)$ and $y(x|c,S)$ are all done in a similar fashion. We pass individual pixels down several decision trees, forming an ensemble or forest. At each split node we evaluate a split function, passing the pixel to its left or right child, until it reaches a leaf node. The classification forests at the first two levels simply store learned, discrete probability distributions in their leaf nodes and output the mode. For regression, the distributions are multi-modal, and outputting the mean can lead to poor performance. Hence, we store a small set of distributions and perform a median filter over a small patch of pixels around x, resulting in the final depth prediction $y_l(x)$ (Fig. 2, right). We use the same binary split criteria and visual feature responses as in [13].

Training

Both the tree structure and the final probability distributions are learned from annotated training data. In the case of RFs, this is done by randomly selecting multiple split candidates and offsets and choosing the one that splits the data best. The quality metric for the split thereby is typically defined by the information gain I_j at node j. For the classification levels we use the Shannon-Entropy $E(S)$ of the empirical *discrete distribution* of coarse depth values and hand shapes, respectively, to compute I_j (cf. [2, 13]). For the regression forest, $E(S)$ is the differential entropy of the empirical *continuous density* $p(y|S)$, modeled as a 1D Gaussian. Thus $E(S)$ reduces to $E(S) = log(\sigma_s)$, where σ is the the Gaussian variance.

Our training data (cf. Fig. 2) consists of ground-truth images attained from a Creative Senz3D depth camera. We use simple depth thresholding and connected component analysis to attain clean segmentation of the hand. We also keep the segmented depth map for training of the regression forest. We record training data for the different hand shapes separately so that labeling can be automated. In addition we add artificial noise to the binary segmentation in the fashion of [13].

For coarse depth classification we train a single tree of depth 8 using 8K training images. This tree achieves an average classification accuracy of 90.1% in half-test-half-train cross-validation. For each gesture in the second layer we use 4K images for training, totaling 12K training data. Using this data we train three trees of depth 15 achieving an average accuracy of 95.3%. For regression, we train one forest per coarse depth interval and per gesture, each consisting of 6 trees ($3 \times 3 \times 6 = 54$ trees in total). Each tree is trained to depth 16 using 4K training images. As we only execute one of the regression forests at a time, the algorithm still runs in real time.

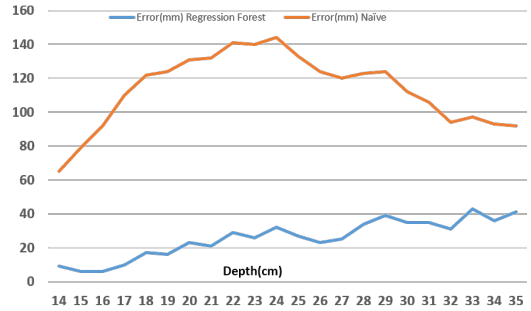

Figure 3. Error vs. ground-truth as a function over depth. Blue is our method (avg. avg. 24mm) and orange is the Naïve method (avg. 87mm).

SYSTEM EVALUATION

We have conducted a number of experiments to asses the accuracy of the algorithm. We compare our output to two separate data sources. First, we compare against the ground-truth (GT) data acquired from a Creative Senz3D camera as a baseline. We also compare to a naïve depth estimation technique based on raw hand size. This naïve baseline is calibrated offline and per user by moving the hand repeatedly to and away from the camera. We record min and max pixel counts and corresponding depth values. At runtime we simply interpolate the depth between these two values. While not a very robust method this actually works reasonably well in particular with constant hand shape and linear motion along z.

Fig. 3 plots the error of the two depth estimates as function of distance from the camera, compared to the GT depth value. Our technique compares favorably to the naïve method and also tracks the GT data well. The accuracy of our approach can also be assessed qualitatively. Again, our technique performs significantly better than the naïve approach (cf. Fig. 4).

The experiments so far were conducted using a single gesture. A more realistic scenario is evaluated in Fig. 5. Here we show depth estimates data over $2K$ frames and under gesture variation. The plot is divided into three areas corresponding to the different gestures. Our method tracks the GT closely, with small recurring spikes. These can be traced back to the boundaries between the coarse depth levels. One could detect and filter these, alternatively one could train the regres-

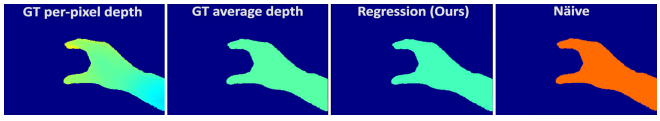

Figure 4. Qualitative Results compared to GT.

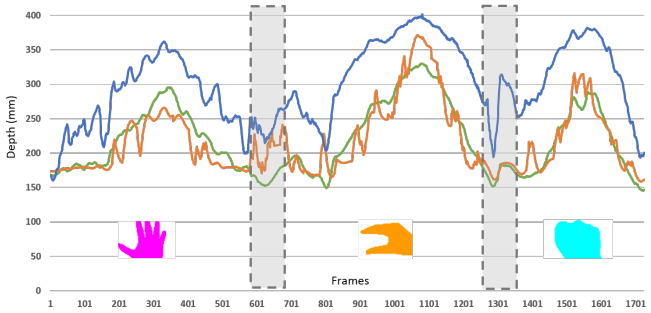

Figure 5. Depth estimation under gesture variation. GT (green) is tracked closely by Ours (orange). Naïve (blue) is significantly worse.

sion forests with training data that has more overlap across the depth boundaries. In contrast, the naïve technique systematically over and undershoots the GT and exhibits a much larger avg error (17.3mm vs. 81.1mm). A t-test reveals that this difference is statistically significant ($p = 0.001$).

APPLICATION SCENARIOS

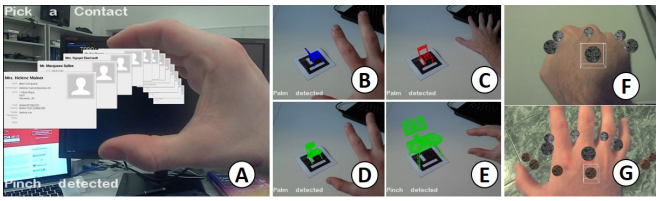

Figure 6. Application scenarios. See text and video figure.

We have built a variety of proof-of-concept demonstrators. These are illustrated in Fig. 6 and the accompanying video. The main advantage of our technique is that it can recover gesture and hand position simultaneously. This allows users to control discrete and continuous inputs jointly. For example, gestures and depth may be used to invoke and browse linear list controls. For instance, to find and select a contact card (Fig. 6, A). Similarly, in an AR furniture application a flat hand may be used to switch 3D models and the pinch gesture may be used to control size of the model (see Fig. 6, B–E). Finally, our approach could also be used to control a spatial, hierarchical 3D menu, where the x, y position of the hand is used to browse entries, z is used to select levels of the hierarchy, and gestures confirm the final selection (Fig. 6, F+G). A promising area for future work could be to combine our input with a trajectory based menu system akin to [1].

DISCUSSION AND CONCLUSION

We have presented a method to jointly estimate 3D hand positions and gestures for 3D input on HWDs. Additionally, we have demonstrated its feasibility quantitatively and qualitatively. However, our method also has clear limitations. First, it is not a general purpose depth estimation technique, but is only trained to recover depth for hands and only at close range. Furthermore, we currently only regress a single, average depth value per hand. This clearly affords less fidelity than what commercial depth cameras produce – albeit at the cost of size and power consumption. However, this depth estimate, when combined with the x, y position of the hand's centroid into a 3D position, is often what an application programmer is interested in. Alongside the possibility to detect gestures this already enables exciting interaction possibilities. For future work we are interested in experimenting with estimating depth for individual parts of the hand such as fingertips versus the palm [13]. Given that the method operates on hand shape only it is unclear whether this could be extended to per-pixel depth but offline methods [6, 7] suggest that there is room for further research and improvement.

REFERENCES

1. Bau, O., and Mackay, W. E. Octopocus: A dynamic guide for learning gesture-based command sets. In *Proc. ACM UIST* (2008), 37–46.
2. Fanello, S. R., Keskin, C., Izadi, S., Kohli, P., Kim, D., Sweeney, D., Criminisi, A., Shotton, J., Kang, S. B., and Paek, T. Learning to be a depth camera for close-range human capture and interaction. *ACM Trans. Graph. 33*, 4 (2014), 86:1–86:11.
3. Harrison, C., Tan, D., and Morris, D. Skinput: Appropriating the Body As an Input Surface. In *Proc. ACM CHI* (2010), 453–462.
4. Karsch, K., Liu, C., and Kang, S. B. Depth extraction from video using non-parametric sampling. In *Proc. of ECCV* (2012), 775–788.
5. Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., and Olivier, P. Digits: Freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proc. ACM UIST* (2012), 167–176.
6. Ladicky, L., Shi, J., and Pollefeys, M. Pulling things out of perspective. In *Proc. IEEE CVPR* (2013), 89–96.
7. Ladicky, L., Zeisl, B., and Pollefeys, M. Discriminatively trained dense surface normal estimation. In *Proc. of ECCV* (2014), 468–484.
8. Lissermann, R., Huber, J., Hadjakos, A., and Mühlhäuser, M. Earput: Augmenting behind-the-ear devices for ear-based interaction. In *ACM CHI Extended Abstracts* (2013), 1323–1328.
9. Ogata, M., Sugiura, Y., Makino, Y., Inami, M., and Imai, M. SenSkin: Adapting Skin As a Soft Interface. In *Proc. ACM UIST* (2013), 539–544.
10. Perrault, S. T., Lecolinet, E., Eagan, J., and Guiard, Y. Watchit: Simple Gestures and Eyes-free Interaction for Wristwatches and Bracelets. In *Proc. ACM CHI* (2013), 1451–1460.
11. Saponas, T. S., Kelly, D., Parviz, B. A., and Tan, D. S. Optically Sensing Tongue Gestures for Computer Input. In *Proc. ACM UIST* (2009), 177–180.
12. Serrano, M., Ens, B. M., and Irani, P. P. Exploring the use of hand-to-face input for interacting with head-worn displays. In *Proc. of ACM SIGCHI* (2014), 3181–3190.
13. Song, J., Soros, G., Pece, F., Fanello, S., Izadi, S., Keskin, C., and Hilliges, O. In-air Gestures Around Unmodified Mobile Devices. In *Proc. ACM UIST* (2014), 319–329.
14. Yang, X.-D., Grossman, T., Wigdor, D., and Fitzmaurice, G. Magic finger: Always-available input through finger instrumentation. In *Proc. ACM UIST* (2012), 147–156.