In-air Gestures Around Unmodified Mobile Devices

Jie Song[1], **Gábor Sörös**[1], **Fabrizio Pece**[1], **Sean Fanello**[2], **Shahram Izadi**[2], **Cem Keskin**[2], **Otmar Hilliges**[1]

[1]Department of Computer Science ETH Zurich Switzerland

[2]Microsoft Research Cambridge UK

{jsong|gabor.soros|fabrizio.pece|otmar.hilliges}@inf.ethz.ch {a-seanfa|shahrami|cemke}@microsoft.com

Figure 1: Touch input is expressive but can occlude large parts of the screen (A). We propose a machine learning based algorithm for gesture recognition expanding the interaction space *around* the mobile device (B), adding in-air gestures and hand-part tracking (D) to commodity off-the-shelf mobile devices, relying only on the device's camera (and no hardware modifications). We demonstrate a number of compelling interactive scenarios including bi-manual input to mapping and gaming applications (C+D). The algorithm runs in real time and can even be used on ultra-mobile devices such as smartwatches (E).

ABSTRACT

We present a novel machine learning based algorithm extending the interaction space around mobile devices. The technique uses *only* the RGB camera now commonplace on off-the-shelf mobile devices. Our algorithm robustly recognizes a wide range of in-air gestures, supporting user variation, and varying lighting conditions. We demonstrate that our algorithm runs in real-time on unmodified mobile devices, including resource-constrained smartphones and smartwatches. Our goal is not to replace the touchscreen as primary input device, but rather to augment and enrich the existing interaction vocabulary using gestures. While touch input works well for many scenarios, we demonstrate numerous interaction tasks such as mode switches, application and task management, menu selection and certain types of navigation, where such input can be either complemented or better served by in-air gestures. This removes screen real-estate issues on small touchscreens, and allows input to be expanded to the 3D space around the device. We present results for recognition accuracy (93% test and 98% train), impact of memory footprint and other model parameters. Finally, we report results from preliminary user evaluations, discuss advantages and limitations and conclude with directions for future work.

Author Keywords

HCI; mobile interaction; mobile gestures; gesture recognition; mobile computing; random forests

UIST 2014, October 5–8, 2014, Honolulu, HI, USA.

http://dx.doi.org/10.1145/2642918.2647373

ACM Classification Keywords

H.5.2 User Interfaces: Input devices and strategies, Interaction styles; I.3.6 Methodology and Techniques: Interaction techniques; I.4.8 Scene Analysis: Object recognition

INTRODUCTION

Today's mobile devices have realized the vision of ubiquitous access to information. Data is now literally at our fingertips no matter where we are. It is clear that these devices have already changed the way how we consume and produce information. However, the question of how best to interact with mobile content is far from solved. While direct touch interaction is clearly intuitive and popular, it also comes with drawbacks. In particular, as mobile devices continue to be miniaturized, touchscreen real-estate becomes increasingly limited, leading to smaller on-screen targets and fingers causing occlusions of displayed content. This can be an issue during prolonged interaction, for example, while reading on a mobile device or when attempting to perform complex manipulations that require many on-screen controls.

Not surprisingly, many HCI researchers have attempted to extend the interaction space around the device. For example using infrared (IR) proximity sensors [3, 23] or handheld magnetic tags [20]. Others have attempted to address the input scarcity issue by leveraging the human body [11] or surfaces in the environment [10, 34] as interactive platforms. However, all of these require hardware modifications of the device and/or user instrumentation. This can be a major barrier to adoption, and limits seamless unencumbered user interaction.

Our work builds upon and extends this body of research, in so doing, we leverage the fact that almost all mobile devices contain RGB cameras for video and image capture. We propose a novel machine learning based algorithm to extend the interaction space around mobile devices by detecting rich gestures performed behind or in front of the screen. The technique

uses *only* the built-in RGB camera, and recognizes a wide range of gestures robustly, copes with user variation, and varying lighting conditions. Furthermore, the algorithm runs in real time entirely on off-the-shelf, unmodified mobile devices, including compute-limited smartphones and smartwatches.

Our goal is not to replace the touchscreen as primary input device but to complement it and to enrich the interaction vocabulary. While touch input works well in many scenarios, there are numerous interaction tasks such as mode switches, application and task management, menu selection and certain types of navigation, where touch input can be cumbersome and the size of mobile devices, and hence that of on-screen controls, becomes problematic. While multi-touch gestures are a feasible way to increase the expressiveness of touch interaction, they cover even more screen real-estate and worsen the occlusion issue.

We argue that such tasks and interactions can be complemented well by sporadic, low-effort gesturing behind or in front of the device. For example, while sitting comfortably on a couch a quick flick of the wrist behind the phone may be used to advance the page in an e-book. Similarly, while using touch to pan a map, symbolic gestures could be used to adjust the map-viewing mode, to adjust the zoom level or to enable additional data views such as traffic information. Furthermore, we present a number of compelling interaction scenarios that would be difficult to achieve with a touchscreen only. For example, the non-dominant hand can be used to invoke and control a fish-eye lens while the touch screen may be used to select targets in the magnified region.

Contributions and overview of the paper

Our proposed gesture recognition algorithm is based on random forests (RF) [2], which have proved to be powerful for prediction of static hand gestures, hand pose [19] and body pose [33], but using PCs and depth sensing cameras. However, these classifiers trade discriminative power and run-time performance against memory consumption. Given a complex enough problem and large enough training set, memory requirement will grow exponentially with tree depth. This is of course a strong limitation for application on resource constrained mobile platforms.

To our knowledge, we present the first real-time implementation of RFs for mobile devices for a pixel labelling task. Furthermore, the algorithm does not rely on highly discriminative depth features but works using only 2D images. We describe a method to robustly and efficiently classify hand states (i.e., gestures) and salient features (i.e., fingertips) using only binary segmentation masks. We also describe techniques to make the classifier robust to in-plane rotation (i.e., wrist articulation or rotating the device itself) and to variations in depth of the hand. We introduce different techniques, specifically designed for mobile devices, to compactify the forests and to reduce memory consumption drastically, while maintaining classification accuracy. These techniques are necessary to reduce the memory footprint of the classifier for mobile phones but generalize to any computing device.

The remainder of the paper is structured as follows: we discuss the relevant literature in the areas of augmented mobile devices, touch and in-air gesture recognition. We then introduce a number of compelling interaction techniques and application scenarios. This is followed by an in-depth discussion of the gesture recognition method and several techniques to reduce the memory footprint of the RF classifier. The algorithm is simple enough to be replicated quickly. In this regard, we provide pseudo-code detailing the classification pipeline alongside training data in order to enable the community to replicate and extend our algorithm. We present our initial results in terms of recognition accuracy, impact of tree depth and other parameters. Finally, we report results from preliminary user evaluations, discuss advantages and limitations and conclude with directions for future work.

RELATED WORK

Advances in processing, sensing and display technologies have enabled rich, powerful computational platforms to fit into our hands and pockets . The current generation of mobile computing relies on touch for interaction but this limits the expressiveness of input. Here we review the literature that expands the input space from the touchscreen to the areas immediately above, behind and around it.

A number of systems have used *cameras* to *extend* the interaction space beyond the display of mobile devices. LucidTouch uses a rear-mounted, protruding 2D camera to detect multi-touch input on the back of mobile devices [40]. Niikuura et al. [28] leverage IR LEDs and a 2D camera to recognize a single fingertip for in-air typing. Samsung has shipped several basic in-air swipe and hover gestures with their Galaxy S4* using a single IR proximity sensor. Others have proposed using body-worn cameras and diffuse IR illumination [9, 36] or RGB cameras [26, 37], demonstrating simple 2D pinch gestures [9] or detecting fingers using markers [26]. [36, 37] classify a wider set of discrete hand postures e.g. for sign language but require a desktop PC for processing. Jones et al. find, in a study of around the device interaction, that in-air gestures can perform as well as touch and define comfort zones for interaction [17].

Researchers have also explored handheld [16, 27] and shoulder-worn [10] depth cameras, focusing on sensing touch interaction with planar [10] or more complex physical surfaces [16, 27]. Others have attempted to leverage the human body directly as input source either sensing muscle or tendon activity to recognize a small set of discrete hand gestures [31, 32], leveraging acoustics to coarsely localize touch on the body [11], or using wristbands of IR proximity sensors for detecting coarse gestures performed on the forearm [29]. Wrist-worn camera based sensors have also been demonstrated to reconstruct full 3D hand pose [21].

The approaches above require new (or augmented) mobile hardware, body-worn cameras, user instrumentation or external tracking, and oftentimes off-board processing. These issues pose serious barriers for user adoption. We propose a real-time system that runs on unmodified mobile devices and recognizes a rich set of gestures alongside accurate fingertip detection for pointing and stroke-based interaction.

The challenges of camera based systems have led researchers to experiment with *non-camera based sensing* around mobile devices. For example, using simple IR proximity sensors fac-

*`http://www.samsung.com/global/microsite/galaxys4/`

ing outwards [3] or upwards [23, 24]. *Magnetic field sensing* has been used to track rigid motion around a device by pairing permanent magnets and magnetometers either by wearing both [4] or by using the device's built-in IMU (inertial measurement unit) [1, 15, 20]. Just like optical sensing solutions, magnetic field sensing can suffer from coarse sensing fidelity and input is limited to tracking a restricted number of discrete points and often with limited degrees-of-freedom (DoF). More recently, a transparent *electric field sensing* antenna has been demonstrated that enables sensing of 3D hand and fingertip locations [7]. While more compact than external camera augmentations, these approaches still require device or user augmentation. Our technique allows for richer gestures that go beyond tracking of discrete points and does not require any augmentation of the device nor the user.

A particularly important aspect of our work is that we aim to complement touch input with gestures. In this sense we share commonality with prior work combining pen+touch input [14], motion+touch [13], in-air and on-body gestures [25], or input from multiple devices [5]. We explore new interaction techniques and application scenarios that combine touch input on and gestural interaction around a mobile device.

Our work clearly relates to the vast body of literature on computer vision methods for hand gesture recognition. Early work focused on hand tracking from 2D cameras. However, recognizing complex hand gestures is clearly challenging from such 2D input [6]. The recent advancements in processing power and the emergence of consumer grade depth cameras, however, have enabled a number of high fidelity gestural interactive systems in HCI [10, 12, 21] and fine-grained 3D hand-pose estimation in real-time [19, 30, 35]. The current state-of-the art can be broken down into methods relying on model-fitting and temporal tracking [30, 35], and those leveraging per-pixel hand part classification [19, 38].

Our algorithm is designed for the recognition of rich and varied gestures and detection of salient hand parts (i.e., fingertips) rather than full hand pose estimation. It is important to note though, that our algorithm works without relying on highly discriminative depth data and rich computational resources of a high-end desktop machine. To our knowledge we present the first implementation of a RF-based gesture recognition algorithm that i) runs in real-time on off-the-shelf mobile devices ii) relies only on the built-in RGB camera(s) and iii) makes no strong assumptions about the user's environment and iv) is reasonably robust to rotation and depth variation.

IN-AIR GESTURES ON THE MOVE

Our system opens up the input space around a mobile device. In doing so, our aim is to couple regular touchscreen input with gestures performed around the device. Rather than replace one modality with the other, we believe that it is this combination that is powerful.

Before detailing the technical gesture recognition approach, we illustrate a number of compelling interaction scenarios, enabled by in-air gestures around mobile devices, in particular bi-manual, multi-modal input. We highlight scenarios where the user tries to perform complex operations that require mode-switching or simultaneous input on different screen locations. Such interactions are typically mapped to multi-touch gestures, which can cause occlusion issues and decrease pointing performance, especially on very small screens.

Figure 2: Usage scenarios: (A) Panning and zooming a document. (B) Bimanual map browsing with a magnifier lens (C) Using gestures to control mode-switches and tool parametrization in a drawing app. (D) Shooting in a scrolling shooter game.

Fig. 2 summarizes a selection of interactive scenarios enabled by our approach. In its most basic form, our algorithm can enable *division of labor* bimanual input [8]. For example, a user sitting on a couch can use touch input to continuously navigate a map or document while the non-dominant hand can be used to occasionally adjust the zoom level of the document, without occluding the screen, or interrupting the touch-based navigation (see Fig. 2 (A)).

The two hands can also work in a more tightly integrated fashion, where an in-air pinch gesture invokes a magnifier lens to magnify a particular area of interest on the document (e.g., an underground map). The magnifier lens can then be positioned by moving the hand behind the device. This illustrates the usefulness of jointly recognizing gestures and fingertips. The magnification factor can be adjusted by opening or closing the pinch gesture. Touch input can then be used to highlight text in a document, place a positional marker on a map or to otherwise interact with the magnified content (see Fig. 2 (B)). Combining touch and gestures makes interactions that simultaneously modify multiple parameters more integrated than what would be possible with touch input alone.

Similarly, gestures around the device can be used to streamline interaction with more complex user-interfaces that require explicit mode changes and allow the user to parametrize tools and functionalities. We have implemented a simple drawing application to illustrate this (Fig. 2 (C)). Here the user can draw using touch input, while a pinch gesture will invoke a brush-selection menu directly at the current touch location. The menu selection can then be performed using touch, without requiring the user's finger to travel to a menu. Analogously, the open pinch (a C-shaped hand) can invoke an in-place color picker. A final, compelling interaction scenario is mobile gaming. We interface our gesture recognition engine with an open-source 2D scroller game[†]. Touch input is used to control the helicopter position, whereas gestures are used to shoot weapons (see Fig. 2 (D)).

[†]`https://code.google.com/p/orange-grass`

Interacting with other mobile devices

One of the main contributions of our work is supporting a classifier that is efficient enough to run in real time on very resource-constrained platforms. The fact that cameras are now ubiquitous on many mobile platforms, makes our system applicable to a wide range of mobile computing scenarios, including mobile phones, tablets but also smartwatches.

Figure 3: Our algorithm runs on a variety of mobile form factors and enables different interaction scenarios. (A) A tablet media player in the kitchen can be controlled using simple, effortless gestures without touching the screen with wet hands. (B) In-air pointing gesture in front of the smartwatch triggers a photo sharing application.

Fig. 3 (A), shows our system running on an off-the-shelf tablet. Here we leverage the front-facing camera for gesture recognition. The user controls a media application where a flat, closed hand pauses the playback of a video and the flat, splayed hand resumes playback. The user may also control other applications such as advancing a photo slideshow using swipe gestures, or controlling (semantic) zoom in a PDF reader. We argue that this type of low-effort, casual gestures can complement touch in certain scenarios for example when the tablet is rested on the user's lap or on a table.

Ultra-mobile devices such as smartwatches are clearly becoming important mobile platforms. However, their input expressiveness is severely limited by the small size of the touchscreens, making typical multi-touch gestures cumbersome or infeasible. Not surprisingly, multiple efforts have been undertaken to enable gestural interaction with such devices (e.g., [23, 31]). However, these typically require modification of the watch and suffer from limited input fidelity. We enable recognition of the same rich gestures as on tablets and phones (e.g., static gestures and fingertip detection) on a smartwatch[‡]. For example, to take pictures and navigate the photo stream without occluding the screen (see Fig. 3 (B)).

Data-driven gesture recognition

The application scenarios discussed here give a flavor for the types of interactions enabled by our system. While already enabling new scenarios, we do not claim that the gestures currently implemented constitute the best possible or even final gesture set. Quite contrary, the proposed algorithms are data driven and therefore it is a matter of capturing new training data, and retraining the classifier, to redefine the gesture set – not a matter of rewriting any code. We highlight two further aspects of this work:

First, due to power and size constraints, currently no readily procurable mobile devices exist with integrated depth sensors. However, state-of-the art gesture and hand-pose estimation algorithms rely on depth data. Robustly detecting hand gestures from a monocular, moving camera, in uncontrolled lighting conditions and with arbitrary backgrounds remains a challenging and difficult task - in particular on mobile devices. At the same time, our main design goal is to support unmodified mobile devices to lower the barrier of use.

Second, the initial gesture set is what we consider to be a useful but also a challenging gesture set. The gestures are different enough from each other that it would be difficult to conceive a single, non-probabilistic algorithm (for example based on geometric heuristics) that could detect them all. Moreover, there is sufficient similarity between the pairs of gestures (e.g., flat hand, splayed-hand) that distinguishing these from each other (heuristically) is an even more challenging task. Moreover, in many interaction scenarios we are not only interested in hand states (i.e., gestures) but also salient hand parts such as fingertips or the wrist, further complicating the detection problem.

These two aspects suggest that recognizing this gesture set or a similar set alongside hand parts would be challenging using heuristics-based approaches only. We therefore explore a novel recognition architecture that combines very simple and computationally efficient image processing techniques with powerful machine learning classification techniques for robust and extensible gesture recognition.

SOFTWARE PIPELINE

So far we have concentrated on describing our system at a high-level and discussing interactive possibilities it affords. We now provide an overview of the software pipeline before detailing the gesture classification algorithm. The pipeline consists of fairly easy and established image processing steps interwoven with a new, staged gesture and part classification pipeline. All components of the pipeline have been carefully chosen and designed with runtime and memory efficiency in mind, in order to achieve real-time performance even on ultra-mobile and resource restricted devices.

Segmentation and Pre-processing

Our gesture recognition algorithm relies on binary masks of the hand, segmented from the background. Therefore, the first step of our pipeline is foreground extraction via skin color detection. Despite much research this problem remains a challenging task under arbitrary lighting conditions. A good summary of various techniques is provided by [18]. We have experimented with many techniques, including HSV and GMM-based adaptive thresholding. However, due to the large variation in skin color and the influence of lighting and background, currently none of these methods provide high enough accuracy of segmentation under arbitrary conditions.

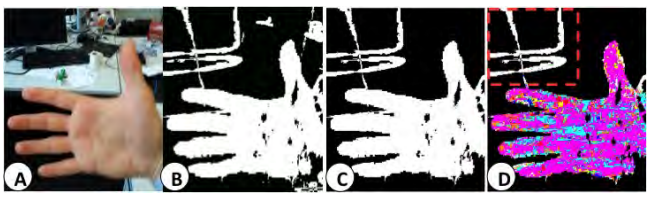


Figure 4: Segmentation by classification. A) Input frame. B) Foreground mask after color thresholding. C) Salt'n'pepper noise is cleaned up by connected component analysis; larger connected non-hand regions remain. D) Result after classification. Colored pixels indicate class label, white pixels indicate noise (red box).

[‡]`http://www.samsung.com/global/microsite/galaxynote3-gear/`.

Therefore we opted for a very simple thresholding technique that is i) easy to implement ii) computationally cheap and iii) provides a good compromise between true positives and false negatives. We segment the hand from the background by:

$$S_t(u) = \begin{cases} 1, if min(R_t(u) - G_t(u), R_t(u) - B_t(u)) > \tau_t \\ 0, otherwise \end{cases} \quad (1)$$

where R_t, G_t and B_t denote the RGB-components at frame t and image pixel u, τ_t is a threshold value and S_t the segmentation. This technique provides reasonable segmentation but suffers from *false positives*, that noise outside the hand region (see Fig. 4 (B)). Some of this noise can be suppressed using connected component analysis but regions of noise will remain if connected to the hand (see Fig. 4 (C)). However, the proposed hand state classification algorithm can be made robust to this particular type of noise as detailed below. At this stage it is important to attain a segmentation that does not suffer from too many false negatives, in particular we wish to persist the contour of the hand for accurate classification results. Please note that the technique only relies on a single parameter τ that can be tuned at runtime (via an on-screen slider), and hence it can be quickly adapted to work under many different illumination conditions.

Hand State Classification Method

Our gesture recognizer is based on random forests (RF) [2] which have been used successfully for a number of vision problems, including body pose estimation [33] and hand pose and state estimation [19, 38] using depth cameras, but also in HCI using different sensing modalities [7, 22, 39].

A RF is an ensemble of decision trees, each tree in the ensemble produces a noisy classification result. However, accurate classification results can be attained by averaging the results from multiple, non-biased classifiers together. Each tree consists of split nodes and leaf nodes; the split nodes themselves can be seen as very primitive classifiers. Each node will evaluate a simple and computationally inexpensive function and as a result forward the currently evaluated datum (in our case a pixel) to its left or right child until the datum reaches one of the leaf nodes. The split functions can be seen as simple questions where each answer adds a little more information, or in other words reduces the uncertainty about the correct class label for the particular pixel. The leaf nodes contain probability distributions describing the likelihood with which a pixel belongs to any of the classes in the label space.

Recent work based on RFs (e.g., [19, 33, 38]) leverages highly descriptive depth images and complementary depth-invariant features for classification. These approaches use continuous valued features (typically simple depth differences), and the split functions compare the feature response against a continuous valued threshold. Our approach cannot rely on depth. Furthermore, we cannot rely on color, as the variation in appearance of different hands under arbitrary lighting conditions is too large to cover all variations. Hence, our method relies only on shape (i.e., binary masks cf. Eq. (1)) to infer hand states and features. We use split criteria of the form:

$$C_j^L(w, v, \Gamma_j) = \{(S, u) | F_{w,v}(S, u) = \Gamma_j\} \quad (2)$$

$$C_j^R(w, v, \Gamma_j) = \{(S, u) | F_{w,v}(S, u) \neq \Gamma_j\} \quad (3)$$

where F_j is the feature response and Γ_j is a selector value stored at each split node j. Here C_j^L and C_j^R are the mutually exclusive sets of pixels assigned to the left and right children of the split node. For a given segmented image S we define the feature response at the pixel location u as

$$F_{w,v}(u) = [S(u + w), S(u + v)] \quad (4)$$

this is a 1×2 vector where w and v are random offsets and $S(u)$ is the binary pixel value at that location. The feature response is discrete and each element of it can take on only two values; 1 for foreground and 0 for background pixels. At evaluation time we simply compare this vector with $\Gamma_j \in \{[0, 0], [1, 0], [0, 1], [1, 1]\}$.

An important aspect is that the order in which the split functions are concatenated (the tree structure) and that the final probability distributions are learned from annotated training data. In the case of RF this is done by randomly selecting multiple split candidates 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:

$$I_j = H(C_j) - \sum_{i \in L,R} \frac{|C_j^i|}{|C_j|} H(C_j^i) \quad (5)$$

where H is the Shannon entropy of C. Decrease in entropy, denoted by I_j, means increase in information, or in other words that the uncertainty about the decision boundary goes down. At training time we exhaustively enumerate all possible splits (w, v, Γ_j) and select the one maximizing I_j (for more details see [2, 33]). We provide pseudo-code for forest evaluation, for others to easily extend our results (cf. appendix).

The binary features employed in our method allow training of a complex data-set while keeping the computational time and memory footprint low. This, however, does not impact precision. Conceptually, the binary and shift values encode the hand shape information and hence the probability that a given pixel belongs to the hand and to which gesture. As the classifier is trained and operates only on such data, the amount of variance in the learning data that needs to be modelled is drastically reduced compared to training directly on color images. Ideally, we could use a larger number of offsets to increase the discriminative power of our classification technique. However, this would strongly affect the training time as the number of offset combinations increases exponentially.

Multi-stage Classification

Now that we have characterized how we leverage the RF framework for per-pixel classification of binary images, we next detail how this can be utilized to recognize a rich set of gestures and hand-parts, highlighting aspects that are specific to computationally restricted mobile devices.

Given the interactive scenarios outlined earlier, our classifier has to infer much from binary images alone. First, the gestures themselves are challenging, due to the similarity between pairs of gestures. Second, we are not only interested in hand states (i.e., whole image classification) but also more fine-grained part information, in particular the location of fingertips. Third, the classifier has to work with hands of different shapes and sizes. Fourth, due to the complexity of skin

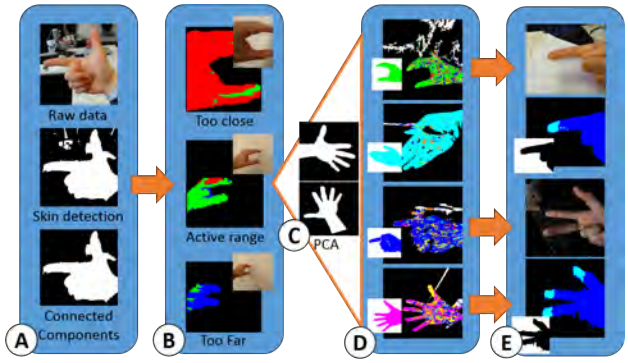


Figure 5: Multi-stage classification. A) Pre-processing segments hands from background. B) DCF coarsely estimates the depth of hand and in-range frames are forwarded. Effective range is between 15-50 cm. C) PCA is used to find in-plane rotation of hands. D) SCF classifies foreground pixel into 6 gesture classes + plus noise. E) PCF classifies location of fingertips for pointing-like gestures.

appearance the classifier has to operate on noisy segmentation data, in particular with regions of noise that are connected to and are indistinguishable from fingers (cf. Fig. 4 (C)). Fifth, natural gestures display a lot of variation in gesture execution (across users but also within), in particular, in terms of rotations of the hand relative to the camera and of course distance between the mobile device and the user's hand. Our goal was to produce a classification scheme that would be able to deal with all of these challenges to a high degree and in real-time.

One approach to achieve this goal is to amass a training database that covers all the expected variations and train a single, potentially very deep forest on this data. The classification step is very efficient so that even deep trees can be evaluated on modern hardware in only a few milliseconds. However, the size of the forest can quickly become an issue as the memory footprint of the trees grows exponentially with tree depth. A further issue is that the different objectives of the classifier are not necessarily aligned with each other. For example, we experimentally confirmed that gesture classification accuracy tremendously benefits from near perfect segmentation. Hence, jointly classifying noise and hand state might not be the optimal approach. A final concern is that of overfitting. With very deep trees the danger exists that a forest learns how to precisely separate the data in the training set but does not have the model capacity to generalize well on unseen test data.

A different approach is that of multi-stage or multi-layered forests [19]. This is a configuration where expert forests are trained for a particular task (e.g., finding hand parts after first determining the overall hand shape) and only those images corresponding to a particular class are forwarded to a second forest, trained only on examples from this class. Each of the forests then needs to model less variation and hence can be comparatively shallow. We propose an algorithm similar to [19]. Ours differs in that it combines forests trained on completely separate tasks and combines forests that modify the image before forwarding it to the next stage (e.g., removing noise). This can be thought of as a pipeline of independent but inter-related classifiers as illustrated in Fig. 5.

Stage 1: Coarse Depth Classification The segmented but noisy foreground mask $S(u)$ is classified into three levels of depth (see Fig. 6 (A)). The depth classification forest (DCF) serves a dual purpose. By coarsely estimating the depth of the hand and by rejecting frames where the hand is either too close or too far from the device, the DCF greatly reduces the variation in depth that the downstream pipeline has to deal with. Equally important, at this stage, the system can detect whether it will be able to correctly classify a gesture with sufficient probability, information that can be fed back to the user. Currently the system is trained to accept gestures performed in an interval of 15-50cm, corresponding to a comfortable arm pose (see Fig. 6 (A)).

Stage 2: Shape Classification On the foreground masks that passes the DCF we run principal component analysis (PCA). The angle of the hand is computed from the major axis (see Fig. 5, C). Instead of rotating the image, which would be costly on mobile hardware, we pass this angle to the next classifier and during evaluation, we rotate the features. This makes the features pseudo-rotation invariant and allows us to reduce the number of images necessary in the training data (and consequently to train shallower forests), while not sacrificing runtime performance. The shape classification forest (SCF) classifies the images into (currently) six gesture classes, one additional *no-gesture* class, and a per-pixel noise class (i.e. eight classes in total). The latter is necessary to deal with remaining false positives from the segmentation, and the former to robustly reject *non-gesture* motion in front of the camera. Fig. 6 illustrates typical classification results with colored pixels indicating the class label and white pixels being classified as noise.

Stage 3: Part Classification Images containing pointing like gestures (i.e., 'point', 'gun', 'splayed hand') are processed by a part classification forest (PCF, see Fig. 5, E), performing per-pixel hand part classification. Due to the lack of depth information we do not attempt to classify all hand parts (cf. [19]), but only select parts of interest for interaction. Concretely we detect the location of fingertips and the wrist which can then be used to implement more fine-grained, localized in-air gestures. The PCF benefits from the previous steps, as practically all noise is cleaned up and only images with visible fingertips will be forwarded to it (see Fig. 6 (C)). Hence, the forest is fairly accurate at a very shallow depth of 10 levels.

Training Data

The RF relies on good training data for high classification accuracy. To train a classifier, robust to the large amount of variation in hand shapes, sizes and distances to the camera

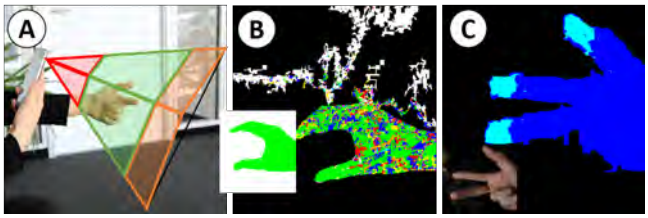


Figure 6: Multistage classification: (A) First hand is classified into three different coarse depth bins. (B) Next hand shape is classified into 6 classes. (C) Finally, fingertip parts are detected by PCF.

and gesture execution, we need a large and balanced training data set.

We asked 20 subjects from our institution to perform the seven gestures under natural variation and recorded short sequences of each gesture. This included one 'no-gesture' where participants were instructed to just casually move their hands and fingers. This was performed per participant and at different distances away from the phone (see Fig. 7 inset). We recorded $\sim 50K$ images covering enough variation in rotation, depth and appearance for training the SCF.

We collected additional $8K$ images to train the DCF, covering a larger depth range, selected $35K$ images from the original data set to train the SCF, and collected additional $5K$ images to train the PCF. The raw training data was processed using the runtime pre-processing pipeline. However, we have to attain clean segmentation masks and label false positives as such. This was achieved by first acquiring the gesture sequences in front of a uniform (typically black) background with carefully tuned threshold τ (see Fig. 7 (A+B)). We then capture empty scenes of arbitrary backgrounds, which are segmented with the same τ, producing some realistic noise signal (Fig. 7 (C+D)). The hand segmentation is then superimposed over the noise and the resulting image is cleaned up using connected component analysis. Finally, to attain a labeled image we subtract the clean hand from the noisy image to separate the hand region from connected false positives (Fig. 7 (E-G)).

This semi-automatic method is reasonably fast in producing training data that contains both clean hand segmentations as well as realistic, labeled noise.

SYSTEM EVALUATION

So far we have presented what we believe is the first real-time implementation of RFs for pixel-labelling tasks on mobile devices. The algorithm does not rely on highly discriminative depth features but works using only 2D images. We detail experiments to characterize the performance of our recognizer. Please note that we report accuracy figures from a single SCF, trained on the entire dataset. As evident from Fig. 10 this serves as lower bound, and as we show later, the pipelined version performs as well or better. The SCF is the best testbed to access the pure discriminative power of the algorithm.

Fig. 8 demonstrates qualitative results of our system classifying different hand gestures. Note the variation we see in the distance to the camera, orientation, of the hand, and the different backgrounds and lighting conditions, as well as the robustness of the segmentation step (labeled white).

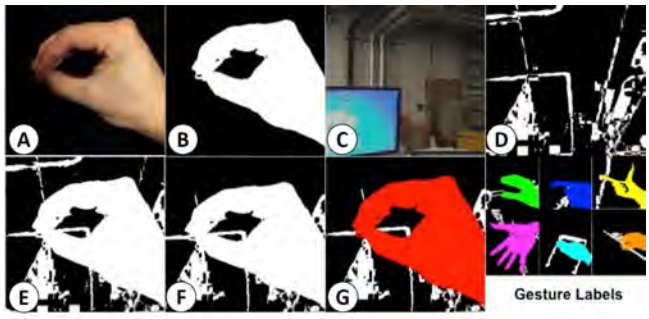


Figure 7: Training data generation and labels used for training.

Test Data and Forest Parameters

The forest size (number of trees) and the depth of the trees are the two main parameters affecting classifier performance. Increasing the number of trees can enhance accuracy in certain cases, but always increases computational cost and memory footprint linearly. To ensure that recognition is fast enough for real-time interaction at 30 Hz, we use three trees.

For test data, we instructed participants to gesture roughly at a range of $15 - 50$ cm, which is a comfortable range of operation, and well represented in the training data.

Gesture Recognition Accuracy

We report results using both leave–one–subject–out cross–validation and by using half of the gesture samples in the set for training and the remaining half for validation. Table 1 summarizes overall classification accuracy for a single SCF, averaged over all recognized gestures.

	15	16	17	18	19	20
Tr PPCR	0.79	0.82	0.85	0.87	0.89	0.92
Tr SCR	0.88	0.92	0.94	0.96	0.97	0.98
Ts PPCR	0.75	0.77	0.78	0.79	0.80	0.80
Ts SCR	0.86	0.89	0.92	0.92	0.92	0.93

Table 1: Per-frame accuracy for half–training / half–test (Tr) and leave–one–subject–out (Ts) cross validation of SCF. PPCR stands for per-pixel classification rate and SCR for state classification rate. Columns report accuracy for different depths.

Users have their own preferences for natural gestures, and the way gestures are performed may vary considerably. Hence, the accuracy of the leave–one–subject–out technique directly depends on the inter-personal variation in the dataset. As the number of subjects increases, so does the variation covered in the training set, and the model's capacity to generalize to previously unseen subjects. This accuracy (avg. 93%) can therefore be viewed as a lower bound and a good estimate of (unfiltered) real-world performance.

The half training–half test scheme is provided to give a better estimate of the power of the proposed method, independent of the quality and make-up of the training data. Overall, recognition performance is very good even with shallow trees (cf. Table 1). Fig. 9 summarizes classification accuracy as a confusion matrix for the entire gesture set, using both validation methods. Our technique achieves generally near perfect classification accuracies with a mean per-class, per-frame accuracy (mean of the confusion matrix's diagonal) of 98% and 93% respectively. In practice this translates to a very robust gesture recognizer with very little temporal filtering necessary (currently we use a 3 frame box filter), as evident by the video.

Increasing tree depth improves classification accuracy but also increases training time and memory footprint. On our data no overfitting effect was observed, indicating that we could increase mean accuracy even further by training deeper trees. However, beyond 20 levels we hit the memory limits of mobile devices. This issue of memory footprint for mobile uses reaffirms our need for multi-staged forests, which we evaluate in the next section.

Multi-staged Forests: Accuracy vs. Memory Footprint

In this section we quantify why splitting forests into separate specialized classifiers improves memory consumption, whilst

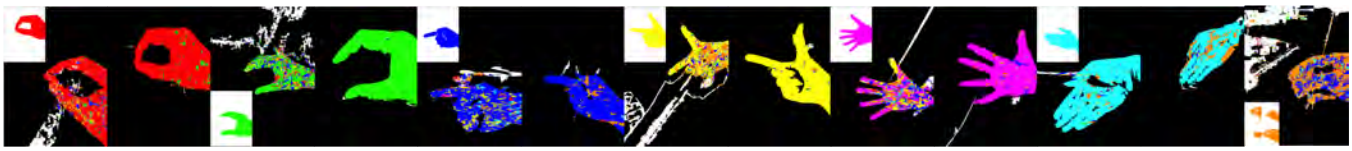


Figure 8: The current gesture set recognized by our classifier. Training data samples insets. Note the variation in scale and rotation as well as robustness to false positives from the segmentation step. the right-most image represents the background noise class.

PinchOpen	0.98	0	0	0	0	0	0.01
PinchClose	0	1	0	0	0	0	0.01
Pointing	0.02	0	0.99	0	0	0	0
Gun	0	0	0	1	0	0	0
SplayedHand	0	0	0	0	0.99	0	0
FlatHand	0	0	0	0	0.01	0.99	0.07
No-Gesture	0	0	0.01	0	0	0.01	0.91

PinchOpen	0.88	0.03	0	0	0	0	0.02
PinchClose	0	0.93	0.05	0	0	0	0.01
Pointing	0.02	0.01	0.9	0.04	0	0	0.01
Gun	0	0	0.02	0.95	0	0	0
SplayedHand	0	0	0	0.01	0.99	0	0
FlatHand	0.05	0	0	0	0.01	0.99	0.11
No-Gesture	0.05	0.03	0.03	0	0	0.01	0.85

Figure 9: Confusion matrix; data from 20 users. Left: half–test / half–train cross-validation; avg. accuracy 98% Right: leave–one–subject–out; avg. per-frame accuracy 93%.

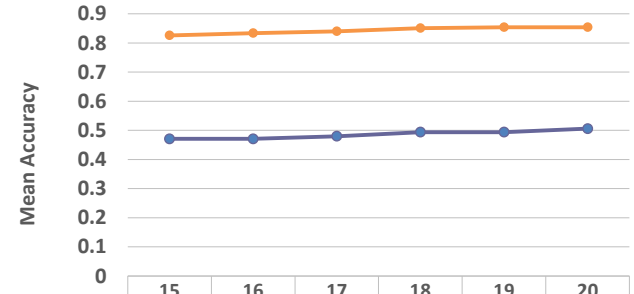
greatly improving accuracy. For this experiment, we assessed both the performance of the pipelined classifier (DCF+SCF) and the standalone forest (SCF). We chose a far more challenging test dataset, where six users performed an extreme set of gestures at different ranges from $2-90$cm, which is far beyond the expected range of interaction. This was intentional as we wanted to stress-test both classifiers.

Fig. 10 shows the comparison of the two classifiers. The drop in performance of SCF in this experiment is explained by the extreme hand poses and distances that we captured in this test set. SCF was not able to cope with this large variation, and could not exceed an accuracy much greater than 50%. Our DCF+SCF multi-stage classifier was able to continuously outperform the SCF producing an average accuracy of $\sim 85\%$. What is perhaps most compelling about this approach is the memory footprint of DCF+SCFs compared to SCFs. The combination of a DCF+SCF with depths 10 and 15 consumes only 4.5MB, versus 110MB for the SCF at depth 20 but performs $\sim 30\%$ better. At level 18 the DCF+SCF still saves 81MB and outperforms the single SCF by 35%. All memory figures are for a single tree and have to be multiplied by the number of trees in the forest.

MOBILE PLATFORM IMPLEMENTATION

The details of our implementation including pseudo-code for the forest evaluation can be found in the appendix. Here, we briefly summarize the most important aspects.

Our recognition pipeline is written in native C++ and handles the camera access, performs background segmentation, random forest evaluation, gesture filtering and visualization of the different stages. At runtime we read images from the camera directly into an OpenGL ES texture and perform skin detection and bilinear downsampling on the GPU. Next, we download the segmentation mask to the CPU and perform connected component analysis (CC) on the binary image, and we compensate for transformations via principal component analysis (PCA). The resulting binary mask is then classified by the DCF, SCF and, if applicable, by the PCF. Each forest updates a label image as well as the input mask. After classification we upload the gesture labels and the cleaned segmentation mask back to the GPU for display. The per-pixel label probabilities



	15	16	17	18	19	20
SCF	0.471	0.471	0.48	0.494	0.494	0.506
DCF+SCF	0.826	0.834	0.84	0.851	0.854	0.854

Figure 10: Comparison of classification accuracy as function of tree depth when extreme gesture depth variations occur. DCF+SCF (orange) outperforms single SCF (blue) in terms of accuracy averaged over all classes.

are pooled across the image $c^* = \underset{c}{\text{argmax}} \ \frac{1}{|u|} \sum_0^{|u|} p(c|S,u)$, where c^* is the final output mode or class label. Finally, a low-pass filter stabilizes the class labels temporally and a Kalman filter stabilizes the part labels spatially. The filtered gestures are injected as events to the Java-based Android apps. Camera capture, GPU computations, CPU computations, and user interaction run in four different synchronized threads.

PRELIMINARY USER EVALUATION

So far we have concentrated on evaluating the gesture recognition accuracy and the impact of *pipelining the classifier*, which we deem one of our main contributions. However, the main goal is to enable new forms of interaction and to enhance user experience on mobile devices. Hence, we gathered quantitative and qualitative data from a small pilot user study.

Bi-manual interaction One of the most compelling interaction scenarios for our technique is the use of touch and gestures in concert, freeing up display real-estate but allowing for complex bimanual interactions. To assess this hypothesis we ran a baseline experiment comparing multi-touch input with the combination of touch plus gestures in a comparative experiment. Participants had to repeatedly switch mode in a finger drawing application either using traditional menus in the touch condition or gestures in the bi-manual condition.

Experimental design We asked 9 subjects (6 male, 3 female) from our institution to trace a given figure using the *touch+gesture* interface or *touch only*. In both cases users had to repeatedly switch brush size and brush color. In the *touch only* condition both functionalities are controlled via a standard Android radial menu (see Fig. 11, left). In the *touch+gesture* condition the color chooser was invoked using the "C" gesture and the brush-size was toggled using the closed pinch gesture. Presentation order of the conditions

is counterbalanced. Each user is asked to trace a total of 10 drawings requiring a total of 6 mode changes each.

Our results, summarized in Fig. 11 right, indicate that the *touch+gesture* condition is faster on average (mean=80.66s, SD=17.62) than the pure *touch* condition (65.22s, SD=15.06). A student's t-test shows that this difference is statistically significant($p < 0.007$). This is inline with our observation that most users instantaneously picked up on how to effectively leverage the gestures in this task. Only one user in our sample was faster with touch (104 versus 111 sec) and this user reported issues with remembering which gesture was mapped to which function.

Qualitative Feedback

While the quantitative results are promising, not all aspects of our system lend themselves to a formal evaluation. To attain a more holistic impression on the perceived user experience we conducted informal sessions with 8 participants (5 male, 3 female) again recruited from our lab. We gave users the opportunity to experiment with the applications we have built and discussed in the section "In-Air Gestures On The Move". This was followed with informal interviews.

Overall the feedback was very positive and users were impressed by the robustness of the gesture recognition engine. Scenarios that tightly couple touch input and gestures, for example the gaming scenario, were particularly well received. Furthermore, most users commented on the usefulness of the 'back of device' zoom and 'page flip' functionality in the Reader app. However, the participants were split in their opinion about the zoom functionality on the map application; with those not liking it saying *'I'm simply too well trained on touch and pinch–to–zoom on mobile phones'*. Finally, the magnifier functionality was generally received positively, with many users highlighting that this could be very useful, remarking that current mobile OS do not allow for simultaneous positioning of the magnifier and interaction with magnified content. The lack of dwell-time for lens invocation was also appreciated.

Of course, these observations are only preliminary. Our gesture set was in part designed to evaluate the range of variation our recognizer can cope with. Nonetheless, the current gestures illustrate the utility of the bi-manual touch+gestures interaction style. Furthermore, only two participants thought that more gestures would have been useful, whereas the rest of the participants found the overall gesture set useful and easy to remember. One user commented on the compelling possibility to train different forests for different apps (*'for example, each game could come with it's own gestures'*). While the classification engine could be built into the OS, the specific trained forest could be shipped as middleware for each application. Finally, users commented that the absence of an explicit mode switch is very compelling. This can be seen as partial confirmation of our initial design goal of low-effort gestures, leveraging touch as primary input channel, complemented by smooth transitions to in-air gesturing then back.

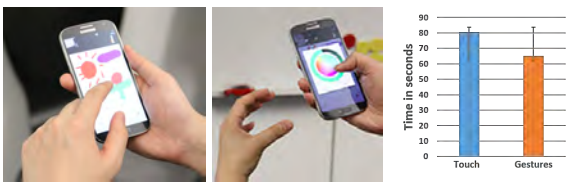


Figure 11: Tracing experiment. A) *Touch* condition B) *Touch+gestures* condition. C) Time to completion results in secs per drawing. Error bars signify standard deviation.

DISCUSSION AND FUTURE WORK

The initial quantitative and qualitative results of our system are promising. Classification results performed by a variety of users in different lighting conditions and with changing backgrounds illustrates that even our initial prototype can be used for a diverse set of gestures, on off–the–shelf mobile phones, tablets and even smartwatches. To our knowledge this is the first time that a real-time implementation of the RFs has been demonstrated on such devices, especially with this level of robustness for gesture recognition. One important aspect to note is that given our recognition engine is machine learning based, new gestures can be added simply by providing new examples, and retraining our system. There are also clearly many areas of improvement, which we discuss in this section, outlining limitations and future work.

More Gestures, Better Features To demonstrate the extendability of our system we have trained the recognizer on a gesture set compiled from the American Sign Language alphabet but taking only gestures that have a unique contour (see supplementary material). The resulting set contains 12 classes and on leave–one–subject out validation achieves an avg. accuracy of 83%. However this demonstrates that our approach is limited to recognizing and discriminating gestures that provide a unique shape or contour, and therefore can miss subtle differences in gestures. For future work we will explore the feasibility of using more discriminative features and going beyond classifying binary images. For example, a feature that captures texture information may be able to differentiate between a flat hand facing down and one facing up.

Dynamic Gestures While we already detect a number of continuous interactions such as tracking fingertips or computing the ratio between open and close pinch (cf. supplementary material), our recognizer is *static* in nature. For future work we want to explore means to extend this to be able to capture dynamic gestures such as flicks and swipes directly. We plan to built upon recent work demonstrating the usage of RFs for motion gestures [39].

Lighting conditions As with all vision systems, ambient light is an issue. We have demonstrated that the system can cope with a variety of natural lighting as can be found in many indoors and outdoors locations. However, sudden changes in lighting such as entering or leaving a building or switching on strong direct light sources are still an issue. In particular, our technique is sensitive to segmentation failure – primarily false negatives as these impact the shape of the hand. To some degree this is counteracted by using a skin detection scheme with a single parameter and keeping the user in the loop to tune this setting. However, ultimately more sophisticated and reliable detection mechanisms would be desirable. Inspired by the promising results we achieved on classifying false positives we will explore the possibility of pushing the entire segmentation into the random forest.

CONCLUSIONS

We have presented, for the first time, a random forest based gesture recognizer, running in real-time on off–the-shelf mobile devices, including tablets, mobile phones and smartwatches. The algorithm avoids any hardware modifications, requiring only images from a regular RGB camera which are now commonplace on such devices. This makes natural gestures on mobile devices accessible to a large user base. We have also proposed a number of compelling interaction scenarios, allowing users to seamlessly transition between touch and gestural interaction, and allowing for bi-manual, simultaneous touch+gesture interaction. To achieve maximum accuracy for minimal memory footprint we have introduced multi-stage classification forests. Our hope is to provide a compelling new platform for researchers and practitioners alike to create new gestural interfaces on mobile devices.

ACKNOWLEDGMENTS

We thank Jibin Ou for writing the mobile phone apps used in the user study and the logging software for the experiments. Furthermore, we thank Emily Whiting and Alec Jacobson for providing the voice-over for the video. Finally, we thank the study participants for their valuable time and feedback.

REFERENCES

1. Ashbrook, D., Baudisch, P., and White, S. Nenya : Subtle and Eyes-Free Mobile Input with a Magnetically-tracked Finger Ring. In *Proc. ACM SIGCHI* (2011), 2043–2046.
2. Breiman, L. Random Forests. *Mach. Learn. 45*, 1 (Oct. 2001), 5–32.
3. Butler, A., Izadi, S., and Hodges, S. SideSight: multi-touch interaction around small devices. In *Proc. ACM UIST*, UIST '08 (2008), 201–204.
4. Chen, K.-Y., Lyons, K., White, S., and Patel, S. uTrack: 3D input using two magnetic sensors. In *Proc. ACM UIST* (2013).
5. Chen, X. A., Grossman, T., Wigdor, D., and Fitzmaurice, G. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proc. ACM CHI'14* (2014).
6. Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.* (2007).
7. Goc, M. L., Taylor, S., Izadi, S., and Keskin, C. A Low-cost Transparent EF Sensor for 3D Interaction on Mobile Devices. In *Proc. ACM CHI* (2014).
8. Guiard, Y. Asymmetric Division of Labor in Human Skilled Bimanual Action. *J. Mot. Behav. 19*, 4 (1987).
9. Gustafson, S., Bierwirth, D., and Baudisch, P. Imaginary interfaces. In *Proc. ACM UIST* (2010), 3–12.
10. Harrison, C., Benko, H., and Wilson, A. D. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proc. ACM UIST*, ACM (2011), 441–450.
11. Harrison, C., Tan, D., and Morris, D. Skinput: Appropriating the Body As an Input Surface. In *Proc. ACM CHI* (2010), 453–462.
12. Hilliges, O., Kim, D., Izadi, S., and Weiss, M. HoloDesk: Direct 3D Interactions with a Situated See-Through Display. In *Proc. ACM CHI* (2012).
13. Hinckley, K., and Song, H. Sensor synaesthesia: touch in motion, and motion in touch. In *Proc. ACM CHI* (2011).
14. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. Pen + touch = new tools. In *Proc. ACM UIST* (2010).
15. Hwang, S., Ahn, M., and Wohn, K.-y. MagGetz: customizable passive tangible controllers on and around mobile devices. In *Proc. ACM UIST* (2013).
16. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proc. ACM UIST* (2011), 559–568.
17. Jones, B., Sodhi, R., Forsyth, D., Bailey, B., and Maciocci, G. Around device interaction for multiscale navigation. In *Proc. of MobileHCI* (2012).
18. Kakumanu, P., Makrogiannis, S., and Bourbakis, N. A survey of skin-color modeling and detection methods. *Pattern Recognit. 40*, 3 (2007), 1106–1122.
19. Keskin, C., Kirac, F., Kara, Y. E., Akarun, L., and Kiraç, F. Hand Pose Estimation and Hand Shape Classification Using Multi-layered Randomized Decision Forests. In *Proc. ECCV* (Berlin, Heidelberg, 2012).
20. Ketabdar, H., Roshandel, M., and Yüksel, K. A. Towards using embedded magnetic field sensor for around mobile device 3D interaction. In *Proc. MobileHCI* (2010).
21. 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).
22. Kim, D., Izadi, S., Dostal, J., Rhemann, C., Keskin, C., Zach, C., Shotton, J., Large, T., Bathiche, S., Nieß ner, M., Butler, A., Fanello, S., and Pradeep, V. RetroDepth: 3D Silhouette Sensing for High-Precision Input On and Above Physical Surfaces. In *Proc. ACM CHI* (2014).
23. Kim, J., He, J., Lyons, K., and Starner, T. The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface. In *IEEE ISWC* (2007), 1–8.
24. Kratz, S., and Rohs, M. HoverFlow: expanding the design space of around-device interaction. In *Proc. MobileHCI*, ACM (2009), 4.
25. Lee, S. C., Li, B., and Starner, T. AirTouch: Synchronizing In-air Hand Gesture and On-body Tactile Feedback. In *IEEE ISWC* (2011), 3–10.
26. Mistry, P., and Maes, P. SixthSense: a wearable gestural interface. In *ACM SIGGRAPH ASIA Sketches* (2009).
27. Molyneaux, D., Izadi, S., Kim, D., Hilliges, O., Hodges, S., Cao, X., Butler, A., and Gellersen, H. Interactive environment-aware handheld projectors for pervasive computing spaces. In *Pervasive Comput.* (2012).
28. Niikura, T., Hirobe, Y., Cassinelli, A., Watanabe, Y., Komuro, T., and Ishikawa, M. In-air typing interface for mobile devices with vibration feedback. In *ACM SIGGRAPH Emerg. Technol.* (2010).
29. Ogata, M., Sugiura, Y., Makino, Y., Inami, M., and Imai, M. SenSkin: adapting skin as a soft interface. In *Proc. ACM UIST* (New York, New York, USA, 2013).
30. Oikonomidis, I., Kyriazis, N., and Argyros, A. A. Tracking the Articulated Motion of Two Strongly Interacting Hands. In *IEEE CVPR* (Providence, 2012).
31. Rekimoto, J. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *IEEE ISWC* (2001).

32. Saponas, S., Tan, D., Morris, D., Balakrishnan, R., Turner, J., and Landay, J. Enabling always-available input with muscle-computer interfaces. In *Proc. ACM UIST* (2009).
33. Shotton, J., Fitzgibbon, A. W., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from single depth images. *Commun. ACM 56*, 1 (2013), 116–124.
34. Song, H., Guimbretiere, F., Grossman, T., and Fitzmaurice, G. MouseLight: Bimanual Interactions on Digital Paper Using a Pen and a Spatially-aware Mobile Projector. In *Proc. ACM CHI* (2010).
35. Sridhar, S., Oulasvirta, A., and Theobalt, C. Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data. In *IEEE ICCV* (2013).
36. Starner, T., Auxier, J., Ashbrook, D., and Gandy, M. The gesture pendant: a self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In *IEEE ISWC* (2000), 87–94.
37. Starner, T., Weaver, J., and Pentland, A. A wearable computer based American sign language recognizer. In *IEEE ISWC*, ISWC '97 (1997).
38. Tang, D., Yu, T., and Kim, T. Real-time Articulated Hand Pose Estimation using Semi-supervised Transductive Regression Forests. In *IEEE ICCV* (2013).
39. Taylor, S., Keskin, C., Hilliges, O., Izadi, S., and Helmes, J. Type-Hover-Swipe in 96 Bytes: A Motion Sensing Mechanical Keyboard. In *Proc. ACM CHI* (2014).
40. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. LucidTouch: A See-Through Mobile Device. In *Proc. ACM UIST*, ACM Press (2007).

APPENDIX

Extended gesture set

The main body of the paper demonstrates a gesture set consisting of only six gestures. In order to demonstrate the scalability and discriminative power of the approach, we have conducted further experiments with a larger gesture set. We selected 12 gestures from the American Sign Language alphabet (ASL) based on the uniqueness of their shape. As our algorithm relies only on binary images, differences in appearance that are fully contained within the silhouette of a hand pose are difficult to discriminate using the binary features. For example, the letters A, M, N, S, and T produce near identical outlines. Furthermore, we excluded signs with dynamics (i.e., motion) as we currently only recognize static gestures.

Fig. 12 shows the confusion matrix for the 12 gestures selected from the ASL alphabet. On this gesture set we achieved similar performance to our original gestures. The overall average per-class accuracy is of 83% for leave–one–out validation. This additional experiment further demonstrates the power of the classification algorithm and we believe that even more complex gesture sets could be handled. However, the feature we chose limits the type of gestures to those that produce visually discernible contours. In our future work we will investigate different types of features to overcome this limitation and to exploit the additional information that is present in the data (e.g., texture, edges).

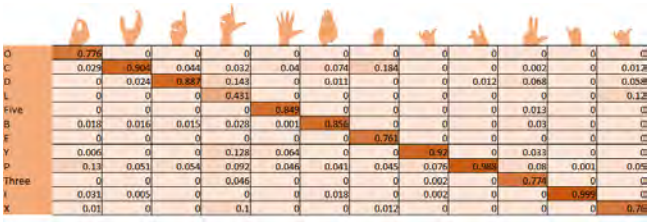


Figure 12: Confusion matrix for 12 gestures selected from American Sign Language.

Continuous gesture switching

Currently our algorithm can only handle static gestures. However, in many situations application programmers are interested in continuous valued input. For example, one might want to implement a gesture to control the volume of an audio player dynamically. One possibility to do so based on our approach is to exploit the per-pixel nature of the classifier. We have observed that the probabilities of symmetric gesture pairs (e.g., open and closed pinch) increase and decrease smoothly during the transition between the two gestures. Fig. 13 plots the raw, unfiltered relative probability ratio between the pair of gestures (*PinchOpen* and *PinchClose*). The ratio between these two values can be mapped to continuous inputs.

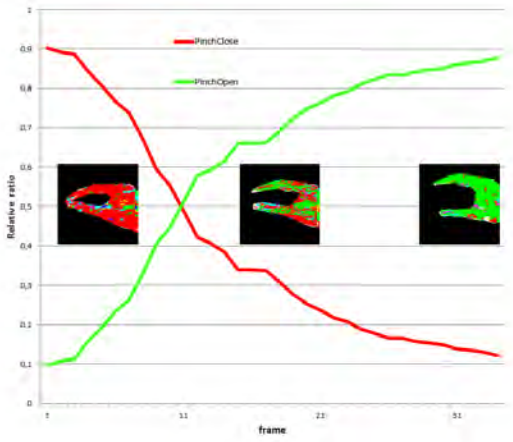


Figure 13: Continuous gesture input can be implemented by looking at the ratio of the first two modes in the per-pixel class probability distribution.

For instance, we have implemented a mapping application in which a fisheye lens can be invoked using the *PinchOpen* gesture. Subsequently, the size (and hence magnification factor) of the fisheye lens can be controlled by opening and closing the pinch gesture. The location of the lens can then be controlled via the fingertip locations (see Fig. 14).

Figure 14: The size of magnifier on the map can be adjusted continuously by the transitions between *PinchOpen* and *PinchClose* gesture pair.