Human-Computer Interaction Using Robust Gesture Recognition

Matthias Endler Oleg Lobachev Michael Guthe

University Bayreuth, 95447 Bayreuth, Germany

ABSTRACT

We present a detector cascade for robust real-time tracking of hand movements on consumer-level hardware. We adapt existing detectors to our setting: Haar, CAMSHIFT, shape detector, skin detector. We use *all* these detectors at once. Our main contributions are: first, utilization of *bootstrapping*: Haar bootstraps itself, then its results are used to bootstrap the other filters; second, the usage of temporal filtering for more robust detection and to remove outliers; third, we adapted the detectors for more robust hand detection. The resulting system produces very robust results in real time. We evaluate both the robustness and the real-time capability.

Keywords

human-computer interaction, gesture recognition, computer vision, hand tracking

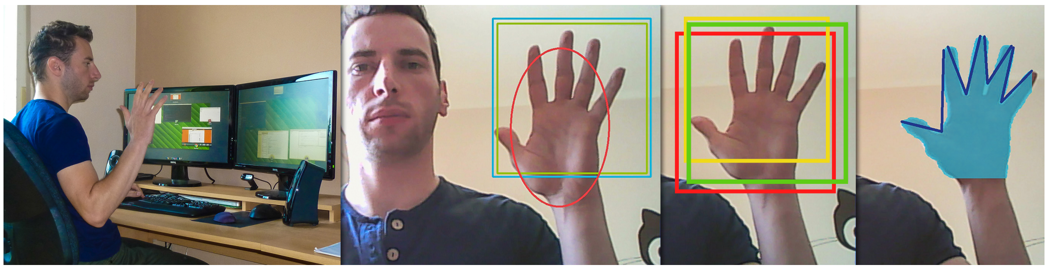

Figure 1: Hand detection and tracling. From left to right: overview of the setup; input frame with Haar-detected bounding box, CAMSHIFT-fitted-ellipse, and bootstrapped-Shape-bounding box; Haar bootstrapping (only once during initialization); final result with hand tracked and fingers counted, used to control applications.

1 INTRODUCTION

Robust, gesture based device control enables a plethora of possibilities to simplify our everyday life, as gestures are an important natural form of human expression. Gestures are a natural part of human interaction. With intrinsic form of communication for human-machine interaction one could intuitively control various devices.

A popular approach in gesture recognition is using special hardware. Gesture recognition was always highly motivated by gaming applications. One of the most prominent examples include Microsoft Kinect that utilizes depth-mapping to augment the vision process. In a contrast to such approaches, we utilize unaugmented live 2D input from a webcam. Our goal is to combine multiple gesture detection methods in a stable pipeline to achieve robustness and real-time capability.

This work focuses on hand movement. Our system falls in the category of dynamic action recognition – we consider the temporal aspect. However, before we can compute the hand movement, we need to secure that the object we are tracking is indeed user's hand.

We do not apply machine learning techniques, but combine and augment known gesture detection algorithms. Each method receives input frames and returns a detected shape (in some form) and a confidence level. These results are combined to maximize performance. We use the Haar classifier [11, 18], CAMSHIFT [5], Shape [4], and Skin [13, 16]. An overview is shown in Figure 1. To further increase the performance we have modified the algorithms, as detailed below.

Some of these algorithms (we denote *detectors* or *filters* from now on) require prior training. We solve the problem with *bootstrapping*. As soon as the simplest detector produces results with sufficient confidence, these are used to train the more sophisticated detectors. This is an iterative process, that nevertheless completes quite fast and does not interfere with the actual interaction session. The 'filter cascade' allows us to execute the final training on the actual data, hence no discrepancy arises between the training and interaction data sets.

The contributions of this paper include: An iterative bootstrapping approach to supply the filters with required initial data, basing on the same data stream the detection will subsequently operate; Modifications of the detector algorithms to optimize their performance for hand detection; Evaluation of the resulting software, especially in terms of real-time capability and robustness of the detection process.

2 RELATED WORK

Directly related to our approach is gesture recognition in games, like the Microsoft Kinect [20], however the Kinect uses additional depth information that we do not utilize. Wang et al. [19] compared various classifier combinations for hand gesture recognition. Their input was acquired with two cameras, we use only one webcam; in a further contrast we operate on the live webcam stream and continuously apply some temporal predicates.

We agree with the general state of the art (e.g. [15]) on the segmentation of the hand, however, we use a *combination* of various methods like Viola–Jones classifier [18] (from now on called Haar) and the CAMSHIFT algorithm [5] instead of their phases two and three of [15]. The advances in sign language recognition [1, 3, 9] are less relevant to us, as for these approaches two hand gestures need to be recognized; hand positions relatively to each other and to the face matter. Varying languages are a further difficulty level.

We refer to two surveys [8, 14] of recognition and tracking methods. Quite related to our work are various keypoint and feature detection algorithms [2, 10, 12]. A recent comparison of binary features is [7]. Vezhnevets et al. [17] survey early *skin detection* methods. Similar to many others we also use the HSV color space. Tripathi et al. [16] use YCbCr color model. Our Skin reimplements Pulkit and Atsuo [13].

3 FILTERS

Our system consists of several image filters used as hand detectors. We divide the detectors into position and the feature cascades. Our approach combines multiple detectors: Haar, CAMSHIFT, Shape, and Skin, an overview is in Figure 2. The first cascade is responsible for obtaining the location of a hand in an image, the second exposes hand features in a given area of an image. It requires the data from the first, position cascade to function properly. Even further, the CAMSHIFT and Shape filters from the position cascade are *bootstrapped* using the data from the Haar filter. Here we describe the separate filters, Section 4 discusses their combination.

Our Haar filter was introduced as Viola–Jones cascade-classifier [18], with further extensions (e.g. rotation) [11]. It produces a magnitude of features, similar to Haar bases and then utilizes AdaBoost [6] to select the few meaningful specimen. Haar combines multiple weak classifiers to a stronger one [11]. The *cascade of classifiers* is a chain, where each classifier detects almost all positive features and removes a significant part of negative features – each classifier removes a different kind of negative features, thus the chain yields highly accurate results. Haar is typically used for face recognition, the detection rate for frontal faces can be as high as 98%. While we also used it to eliminate the face in the input image in the pre-processing stage, *our* critical usage of Haar stems on a training set with hand images.

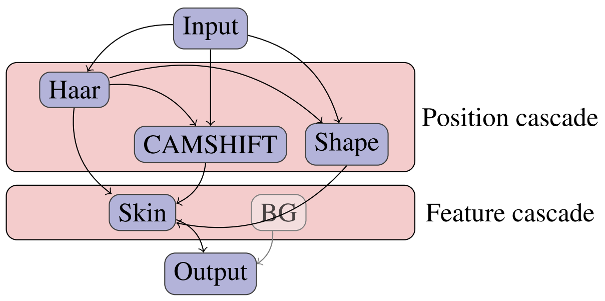

Figure 2: A schematic representation of our approach. First cascade filters provide information to second cascade. Background subtraction (BG) is future work.

Continuously Adaptive Mean-Shift (CAMSHIFT) [5] basically computes a hue histogram of the tracked area and, in the new frame, *shifts* the tracked area based on probabilities of the pixels to be part of the tracked area. The shift goes to the center of gravity of most probable pixels. Then the histogram is updated. CAMSHIFT requires the initial area to track. We basically use the part of the image for that Haar is confident enough and *track* it not merely with Haar, but also with CAMSHIFT and Shape, see Section 4 for details.

A further (very popular) filter is template matching [4]. Given an base image (the *haystack*) and a smaller template image of size $w \times h$, (the *needle*), we compare patches of size $w \times h$ with the template image, the needle appears at the patch position with the maximum value. Our Shape filter detects the hand position in this manner. We utilize only the red channel and use the normalized cross-correlation metric. The shape of the hand is obtained from the Haar filter.

A single skin color model can be successfully used for most human beings in HSV color space [5]. The hue value won't vary much, only the saturation needs to be adjusted individually. We adjusted saturation and brightness for the varying real-life lightning configurations, our Skin detector utilizes these data.

4 TRACKING SYSTEM

We aim for a real-time capable system that is very robust in terms of detection performance and is easily usable as a software component in larger applications. The system should work without the need for long calibration steps, it should be platform independent and mostly unobtrusive. After our system has registered the gesture, an action is triggered.

Given an average frame rate of 24 fps, a calculation time of about 40 ms sets a challenging boundary. Even more, as delays would dramatically affect user experience. We optimized the code for real-time usage. The number of false positives needs to be kept to a minimum. Extensibility is also required. We used a modular plugin system for the detection algorithms and a high-level scripting language (Python) with the highly optimized C++ library OpenCV.

All filters were adjusted to improve hand detection. Haar has proven very robust in our tests, we use it to detect the initial hand position. During early tests, we experienced many false-positives with a face. Therefore, a cascade trained on frontal and profile faces was used to remove the face in the image (if any), improving the hand detection rate significantly. We further use two separate hand cascades. One detects open hands and the other one fists. All three cascades run inside one detector. Haar often detects the same object at slightly different positions, called *neighbors*. The more neighbors, the more likely is the correct detection. We dynamically adjust the minimum number of neighbors to retain a candidate.

Haar performs very good on simple, plain-colored backgrounds, but it falls short on complex backgrounds and in difficult lighting conditions. It is quite vulnerable to hand rotations. CAMSHIFT can adapt to changing brightness and hand rotation with backprojection. It provides stable results for complex backgrounds. CAMSHIFT is prone to errors, when other skin-colored objects (e.g. the face) are visible. Shape shows a similar behavior: Our modification made it agnostic to lightning. Like Haar, it won't detect tilted hands. To compensate for varying light conditions, Haar and Shape work on normalized grayscale images; CAMSHIFT and Skin filter operate in HSV.

We assume that the bounding boxes of the hand in two consecutive frames need to overlap. We use a memory buffer for a few previous detected hand positions for each filter and discard outliers. This approach facilitates a *temporal link* between separate frames.

CAMSHIFT and Shape are *bootstrapped* with data from Haar filter. Haar bootstraps itself – we use an initial calibration phase when the hand is not moved. The consecutive bounding boxes, detected by Haar should overlap (see Figure 1). Such bootstrapping functions surprisingly good and is one of the novelties of our approach. Haar uses cascade data, pre-trained for hand detection, but Haar is not training its cascades on the hand of the user! Just holding the hand in front of the webcam for few seconds suffices to produce then confident results with Haar. The hand region is passed to CAMSHIFT and Shape, so they can perform the tracking. In other words, with our bootstrapping technique we are able to track not *some* hand, but exactly the hand of the current user. This ensures fast and consistent operation.

5 RESULTS

We evaluate both the robustness and the real-time capability of our implementation. We use a MacBook Pro with a 2.4 GHz Intel Core 2 Duo, 4 GB of RAM and its integrated camera, Python 2.7, and OpenCV 2.4.6.

We test our system in various repeatable conditions of different 'difficulty grade.'We recorded our typical gestures in various light conditions, with varying complexity of the background and at various speed of gesturing. The videos are publicly available under http://bit.ly/R6Owu6. The background complexity varies between simple background, skin-colored background, complex background, and mirroring. For the tests below we used two gestures: 'Exposé' and 'Move.' Table 1 presents the assessment. We saw a good performance almost everywhere; underexposure and highly specular background were expectantly problematic.

#	Background	Lighting conditions	Gesture	Speed	Result
1	Simple	Normal	Exposé	Slow	$+$
2	Simple	Normal	Exposé	Fast	$+$
3	Simple	Overexp.	Move	Slow	$+$[1]
4	Simple	Overexp.	Exposé	Slow	$+$
5	Simple	Underexp.	Exposé	Slow	$+$
6	Skin-colored	Underexp. (Noise)	Exposé	Fast	$+$[2]
7	Moving	Changing	Exposé	Slow	$+$
8	Reflections	Underexp.	Exposé	Fast	$+$[2]
9	Reflections	Underexp.	Move	Slow	$-$[3]
10	Reflections	Normal	Move	Slow	$+$[1,2]

Table 1: Results of the robustness evaluation.

The *real-time capability* was achieved. In our visual tests, the system worked fluidly, at significantly faster-than-normal frame rate. We noticed no disturbance when the system was working on live video feed. We used the machine's integrated camera and also experimented with an external webcam. The video was sampled at 24 frames per second.

We have benchmarked a typical video sequence for a drag and drop 'Move' gesture (video #3 from above); the results are shown in Table 2. This table serves as a quantifiable comparison of our method and existing work: how would the filters perform solely? We observe that all filters are highly real-time capable.

Combining the *worst case* timings of all filters, we theoretically achieve at least 18 fps on a quite dated machine

[1] With prior bootstrapping.
[2] After some tweaking.
[3] Haar did not correctly identify the hand.

Filter	Frames per second			
	mean	median	variance	min.
Haar	107.74	109.98	331.46	60.51
CAMSHIFT	394.4	403.9	1402.99	161.5
Shape	125.62	130.11	198.81	36.98
Skin	774.4	732.5	48210.8	157.7

Table 2: Benchmark results. We show the hypothetical frame per second rate for each filter, if executed solely.

using a single threaded implementation. Note that the actual minimum frame rate is even higher (45 fps), so the method is definitively real-time capable.

6 CONCLUSION

We have presented a new combination of several detectors for the robust, real-time hand gesture recognition. We have modified and adapted multiple methods from computer vision: Haar, CAMSHIFT, Shape, and Skin filters. They are typically used for face detection, but we adapted them to hand recognition. We have *composed* them into a working and robust system. Our contribution includes: Chaining the filters into two cascades: one for position detection and one for the shape of the hand; Temporal heuristics and position consensus remove detection outliers, thus improving robustness. Our system has a short-time memory. All filters need to agree on the roughly the same area, a complete outlier is discarded; The filters are bootstrapped, hence we always operate on recent and relevant data.

The *bootstrapping* is an important trait of our system. Most filters require some initial images to track and/or compare with. We iterate one filter until it reaches sufficient confidence levels and then use the successfully detected hand position as input for the further, advanced filters. Such a technique proves to be very stable, as it adapts the whole chain to the particularities of the current user (hand shape, hand size, skin color, etc.) and the current setting (e.g. background, light conditions, white balance). This also increases the robustness.

Our system works with a stock webcam on an inexpensive consumer computer hardware. The average filter performance was well real-time on a quite outdated machine. One important future goal is to port our software to systems with little processing power (such as embedded devices or smartphones). With background subtraction we would be able to to reduce the search window and hence to achieve better performance. Threading would greatly improve the real-time performance by executing independent filters in parallel.

A crucial issue is the minimization of false positives. We have already improved this point, but it would be possible to improve even more. On that account, we want to implement more heuristics and algorithms, which help the system track hands despite the occlusion. Of course, simply detecting more gestures is also important.

7 REFERENCES

[1] O. Aran, T. Burger, L. Akarun, and A. Caplier. Gestural interfaces for hearing-impaired communication. In D. Tzovaras, editor, *Multimodal User Interfaces*, pages 219–250. 2008.

[2] H. Bay, T. Tuytelaars, and L. Gool. SURF: Speeded up robust features. In *ECCV '06*, LNCS 3951, pages 404–417. Springer, 2006.

[3] B. C. Bedregal, A. C. R. Costa, and G. P. Dimuro. Fuzzy rule-based hand gesture recognition. In *Artificial Intelligence in Theory and Practice*, IFIP 217, pages 285–294. Springer US, 2006.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE T. Pattern. Anal.*, 24(4):509–522, 2002.

[5] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, (Q2), 1998.

[6] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, LNCS 904, pages 23–37. Springer, 1995.

[7] J. Heinly, E. Dunn, and J.-M. Frahm. Comparative evaluation of binary features. In *ECCV '12*, LNCS 7573, pages 759–773. Springer, 2012.

[8] E. Hjelmås and B. K. Low. Face detection: A survey. *Comput. Vis. Image. Und.*, 83(3):236–274, 2001.

[9] A. A. Kindiroglu, H. Yalcin, O. Aran, M. Hrúz, P. Campr, L. Akarun, and A. Karpov. Automatic recognition fingerspelling gestures in multiple languages for a communication interface for the disabled. *Pattern Recognit. Image Anal.*, 22(4):527–536, 2012.

[10] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. ICCV '11, pages 2548–2555. IEEE, 2011.

[11] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. LNCS 2781, pages 297–304. Springer, 2003.

[12] D. Lowe. Object recognition from local scale-invariant features. volume 2 of *ICCV '99*, pages 1150–1157. IEEE, 1999.

[13] K. Pulkit and Y. Atsuo. Hand gesture recognition by using logical heuristics. Technical Report 25, Japan Advanced Institute of Science and Technology, School of Information Science, 2012.

[14] S. S. Rautaray and A. Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artif. Intell. Rev.*, pages 1–54, 2012.

[15] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Eng. Appl. Artif. Intel.*, 22(8):1141–1158, 2009.

[16] S. Tripathi, V. Sharma, and S. Sharma. Face detection using combined skin color detector and template matching method. *Int. J. Comput. Appl.*, 26(7):5–8, 2011.

[17] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *GraphiCon*, ICCGV' 03, 2003.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR '01, pages I:511–518. IEEE, 2001.

[19] G.-W. Wang, C. Zhang, and J. Zhuang. An application of classifier combination methods in hand gesture recognition. *Math. Probl. Eng.*, 2012.

[20] Z. Zhang. Microsoft Kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, 2012.