A Modular Approach to Gesture Recognition for Interaction with a Domestic Service Robot

Stefan Schiffer   Tobias Baumgartner   Gerhard Lakemeyer

Knowledge-Based Systems Group
RWTH Aachen University, Aachen, Germany
tobias.baumgartner@rwth-aachen.de
(schiffer,gerhard)@cs.rwth-aachen.de

Abstract. In this paper, we propose a system for robust and flexible visual gesture recognition on a mobile robot for domestic service robotics applications. This adds a simple yet powerful mode of interaction, especially for the targeted user group of laymen and elderly or disabled people in home environments. Existing approaches often use a monolithic design, are computationally expensive, rely on previously learned (static) color models, or a specific initialization procedure to start gesture recognition. We propose a multi-step modular approach where we iteratively reduce the search space while retaining flexibility and extensibility. Building on a set of existing approaches, we integrate an on-line color calibration and adaptation mechanism for hand detection followed by feature-based posture recognition. Finally, after tracking the hand over time we adopt a simple yet effective gesture recognition method that does not require any training.

1 Introduction

In the development of domestic service robots easy and natural interaction is a vital property of a successful system. Intuitive control and interaction can, for example, be achieved with natural language. However, a huge part of meaning in communication is also transferred via non-verbal signals [1]. A very important mode of this non-verbal communication is using gestures. This is especially true in interaction with a domestic service robot, since controlling the robot often relates to entities in the world such as objects and places or directions. References to objects can conveniently be made by pointing gestures while other dynamic gestures can be used to indicate directions or commands.

Quite some approaches tend to pose undesirable requirements both, before the start of operation as well as within operation itself. For example, they may require a tedious calibration of hand colors or depend on initialization poses of the human. Also, some approaches are quite costly and demand high computational resources. We try to avoid these undesirable properties and aim for a flexible, modular, and robust system that is both easy to set up and easy to use while minimizing computational demands. We designed a modular architecture where gesture recognition is decomposed into sub-tasks orchestrated in a multi-step system. This enables a filter-and-refine like processing of the input where

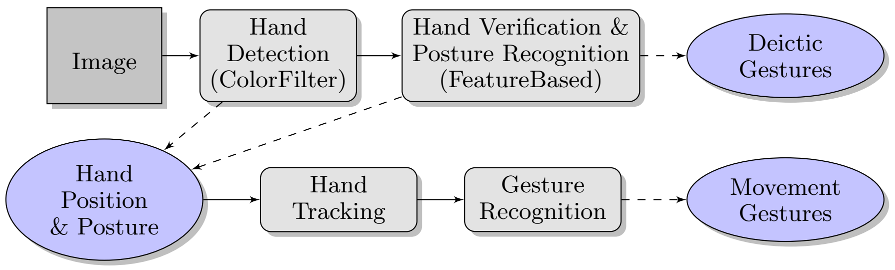


Fig. 1. Architectural overview of our approach

the single steps are as independent as possible to allow for an easy replacement of the particular method used for any of them. Further, the output of each step can already be used for specific purposes. That is, for example, pointing gestures can already be inferred from the position (and possibly the posture) of a hand and a face, while recognition of dynamic gestures additionally requires hand tracking to extract a trajectory. What is more, the overall computational demands are kept low because the amount of information to be processed gets reduced in each step.

2 A Multi-step Approach

The process of gesture recognition is subdivided into four main steps: *hand detection*, *posture recognition*, *hand tracking*, and finally *gesture recognition*. Hand detection is the task of finding the position of one or more hands in an image, where we follow a color-based approach. To increase robustness against false detections, we additionally apply a *hand verification* step. Posture recognition then is to determine the shape of the hand, that is to say the configuration of the fingers (e.g., a *fist* or an *open hand*) and the orientation of this posture. Both, hand verification and posture recognition are performed with a feature-based classification method. Binary classification is used to decide whether a candidate area actually contains a hand and a multi-class classification determines one of multiple possible postures. Hand tracking refers to recording the position of the hand (and its posture) over a sequence of images. Finally, gesture recognition is understood as identifying a specific dynamic movement of the hand from the trajectory formed over time. A graphical overview of our system's architecture is depicted in Figure 1.

Note that the intermediate steps yield useful information already. The hand detection and verification provide us with hand positions and posture recognition adds the posture of a hand detected. Specific commands such as a lifted open hand can already be processed, for example, as an indication of a *stop*-request or the user's request for attention. Taking an existing face detector as a secondary cue also allows for identifying pointing gestures when the posture is, say, a lifted index finger. We refer to gestures at this stage as *deictic gestures*. Also, by iteratively refining the information we can dismiss wrongly detected hands and attach additional information thus enabling increased performance of later steps.

3 Related Work

Hand Detection A lot of the existing hand detection approaches are color-based, since skin-colored regions can quite easily be detected in an image [2, 3]. There are two main issues with (such static) color-based hand detection, however. For one, not every skin-colored region in an image is in fact a hand. For another, in dynamic environments the values of skin color may vary over time, for example, because of changes in lighting conditions or due to obstructions like shadows or colored reflections. A combined way to tackle the problems mentioned above is to make use of the assumption that an image containing a hand also contains a face. This does not seem to be too restrictive given a human-robot interaction scenario. If a face is detected, a skin color can be extracted from that face and then be used for hand detection. Such an approach was already presented in [4]. We follow similar directions but apply some variations. We operate on the HSV color space which, according to [5], seems most appropriate for skin color detection. We then use a modified scheme to dynamically adapt the color values over time. Another way to detect hands is to use features instead of color. In [6] a sliding window is used and a classifier is applied on this window to decide on whether this window contains an object of interest (i.e. a hand) or not. It is an extension of a method presented for object detection in [7], namely a haarcascade, which is a cascade of haar-like visual features. The classifier is built using the AdaBoost method [8] for boosting. An alternative to classical boosting could be random forests [9], especially for multi-class classification as we will employ for posture recognition. Random forests have successfully been applied to face recognition for multiple identities on a mobile robot [10] already.

A method to recognize hand postures is followed in [11]. The idea is to bring the fingers of a hand in a meaningful relation by considering the hand's convex hull and counting the defects of this hull. However, this is only done for planar hands and it is not generally applicable in its current form. Triesch and v.d. Malsburg present a system for posture recognition using elastic graph matching [12]. Several points on a hand are used to build a graph which is compared to graphs trained on a set of postures before. Although both these methods seem appealing and could potentially be plugged into our modular architecture, we opt for a feature-based posture recognition following the approach in [6] to be able to exploit the similarities in hand verification and posture recognition.

Hand Tracking Tracking can, for example, be realized with a method called "Flocks of Features" [13]. To track, the hand position is initialized by detecting one of a set of six trained postures. Then, features are chosen on this hand and followed with KLT feature tracking [14]. Another successful tracking mechanism is CAMSHIFT [15], a modification of the *mean shift* algorithm. The position of a fixed size window is iteratively computed. However, CAMSHIFT is color-based and it is thus subject to the drawbacks we mentioned earlier. That is why we follow the lines of [14] in tracking.

Gesture Recognition There is a huge body of work on gesture recognition, a survey can be found in [16], for example. A lot of the approaches use statistical methods such as Hidden Markov Models (HMMs) which we think are computationally too demanding for our setting. Also, statistical methods usually rely on (often huge amounts of) training data which the recognition then depends on. Instead, we plan on extending a simple approach to gesture recognition described in [17]. It is especially intriguing since it does not require any form of training and does not depend on any external library or toolkit either.

4 Hand Detection & Posture Recognition

In this section we detail our approach to hand detection, hand verification, and posture recognition. First, we employ a color-based approach for hand detection. A *ColorFilter* selects those parts of the image that are skin-colored. The extracted parts are then forwarded to a *Hand Verification* step which decides whether these areas actually contain a hand or not. All detected and verified hands are subsequently processed by a multi-class classifier determining one of a set of specific postures for each hand. The result of this procedure, that is, the hands in an image and their (possibly undefined) postures are then available for any later module. This multi-step processing tries to reduce the search space with every step. The cheapest, color-based classification is done first. The false positive rate may be comparatively high here, though. However, the subsequent step filters out further false positives, now with a feature-based method which is computationally more costly, but only has to consider fewer candidates. Only those hands are processed in the posture classification that passed the first two steps. This way, we proceed with less effort than it would have taken to do the classification of the different postures on the complete image instead of on candidate regions only.

4.1 Hand Detection

The maybe most obvious feature of hands in an image is color. Skin color may easily be detected [2, 3], but especially a static color model suffers from dynamic changes in the environment such as lighting, shadows, and inter-person skin color variations. Further, static color models typically require a training which has as many examples for skin color as possible. Any hand exhibiting a color that is not covered by the training set will not be detected by this color model. That is why we opt for a dynamically adapting skin color model with almost no prior information. Instead, we exploit the following assumption: If the robot is to interact with a person, this person is most likely facing the robot. So, similarly to [4], we first detect faces in the image to extract parameters for our color model from the face region. For one, the face detection module is active on our robot already, so no additional effort is required. For another, there are properly working face detection methods [18] readily available in OpenCV[1]. After a sufficiently reliable detection of a face in x subsequent images we compute expectation values σ_c and standard deviation μ_c for each of the HSV color space

[1] `http://opencv.willowgarage.com/`

Table 1. Results of Color Evaluation

Cover Percentage[2]		True positive	False positive	Mean iterations
Face	Image			
0.3	0.6	0.83	0.19	0.71
...				
0.5	0.8	0.90	0.19	1.03
...				
0.6	0.8	0.93	0.24	3.42
0.7	0.5	0.96	0.33	11.25
...				
0.9	0.9	0.99	0.44	54.10

channels c, i.e., hue, saturation, and value. Pixels in the image with a value of v_c in color channel c are then classified according to the formula $\mid \sigma_c - v_c \mid \leq \alpha \cdot \mu_c$ where α is a positive real-valued factor allowing us to control the adaptivity.

Additionally, we compute two control values for every image that help in determining when and how we need to adapt our model. The *image cover percentage* indicates how many percent of the image are classified to be skin-colored. Analogously, the *face cover percentage* tells us how many pixel of the face region were classified to be of skin color. If the *face cover percentage* is too low, this indicates that only few pixels were considered to be skin, which obviously is undesirable. In that case we have to increase α to make our skin color classifier more lenient. If, on the other hand, the *image cover percentage* is too high, too much of the overall image is considered to be skin. In that case we need to lower our α for the model to be more restrictive.

Evaluation To determine appropriate values for the *image cover percentage* and *face cover percentage* we constructed a database of around 800 pictures of typical application scenario settings. To do so, we placed faces and hands with variations in lighting and brightness on indoor background images. Faces and corresponding hands were varied in size to mimic a user standing in front of the robot at distances of between two and three meters. For every combination of *image cover percentage* between $[0.5 \ldots 0.9]$ and *face cover percentage* between $[0.3 \ldots 0.9]$ in steps of 0.05 we computed the true positive and false negative rate as well as the number of iterations it took the color model to reach a fixed value as *mean iterations*. Selected results are listed in Table 1.[2]

The final ColorFilter then used a threshold of 0.5 for the *face cover percentage* and 0.8 for the *image cover percentage*. We chose to do so because at these values the true positive rate of 90% was reasonably high while the false positive rate of 19% was tolerable, especially since further false detections can still be sorted out in later steps. A mean number of 1.03 iterations also indicates that these values are quite stable and do not require too many adaptations. Combinations beneath the horizontal line after the values 0.6 and 0.8 have a very high number of iterations so that it would take too long to compute color models with these.

4.2 Hand Verification & Posture Recognition

Since the previous step, the color-based hand detection, results in a non-negligible amount of false positives, the next step is to verify the candidate regions. Also,

[2] Percentages are given as values between 0.0 for 0% and 1.0 for 100%, respectively.

Table 2. Results for *Haarcascade* trainings in overview

	Training Images		True Positive	False Detections	Stages	Training Time
	Positive	Negative				
All Hands	1000	700	0.15	71.95	11	21h
All Upright	1000	1000	0.36	3.06	18	42h
Open Own	3000	2000	0.98	12.71	12	7h
Open Kumar	1000	700	0.93	2.37	14	9h
Fist	130	100	0.57	0.23	4	1h
Fist + Open	500 + 500	1000	0.28	0	19	41h
L	100	100	0.25	0	10	3h

we want to determine the posture of every hand verified. For both these tasks we choose a feature-based approach. A basic technique in feature-based detection is to use a so-called *sliding window*. A window of fixed size is slided over the image and for each position of the window a decision rule (i.e. a classifier) decides whether the window contains an object of interest or not. Since the classifier used with the sliding window can be a binary or a multi-class classifier and the window does not need to be slided over the whole image but can also be only slided over regions of interest (ROIs) we opt to use feature-based classification (with haar-like features) for both, the hand verification and the posture recognition.

Data Sets A major question in building feature-based classifiers in general, and for binary classifiers for hand verification in particular, is the selection of an appropriate training set. In contrast to faces, the variations found with many different hand postures are considerably larger. To determine a good classifier for hand verification we trained and evaluated several haarcascades (using OpenCV) and random forests with different training sets compiled from collections available on the Internet, namely one with high resolution images of a single posture[3], a set with lower resolution images of ten different postures[4] and a collection by Ajay Kumar[5] used as training data for a touchless palmprint authentication [19]. Additionally, we constructed our own data set from hand images extracted from video sequences and placed on complex backgrounds. The video sequences were recorded by gesturing in front of our robot. We intend to make our data set publicly available soon. Negative examples were taken from a data set collected by Natoshi Seo.[6]

Evaluation We conducted extensive evaluation series with both, classical haarcascades and random forests. Results from different trainings for haarcascades and random forests are given in Table 2 and Table 3, respectively. The evaluation suggests that training does not seem suitable to generate a single cascade capable of detecting arbitrary hands with any of the available training sets. An in-depth investigation revealed that the choice of haar-like features available to the classifier influences the recognition results. This finding is supported by results from [6], where instead of traditional haar-like features the authors use

[3] `http://www2.imm.dtu.dk/~aam/datasets/datasets.html`
[4] `http://www.idiap.ch/resources/gestures/`
[5] `http://www.comp.polyu.edu.hk/~csajaykr/IITD/Database_Palm.htm`
[6] `http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/`

Table 3. Results for *Random Forest* classifiers

	TP rate	FP rate	Samples	Trees	Time (h:m)
All hands, old feat.	0.73	0.25	2×190	10	0:36
All hands, 4 orient.,	0.79	0.25	5×200	3	0:20
5 classes	0.63	0.03	5×2000	20	16:02
Open	0.73	0.07	2×250	5	0:09
Fist	0.78	0.07	2×250	7	0:21
Open + Fist	0.77	0.21	2×250	9	0:12
Closed	0.60	0.12	2×1000	2	1:18
Victory	0.61	0.16	2×180	1	0:06
Classify Postures	0.19	-	5×150	5	0:31

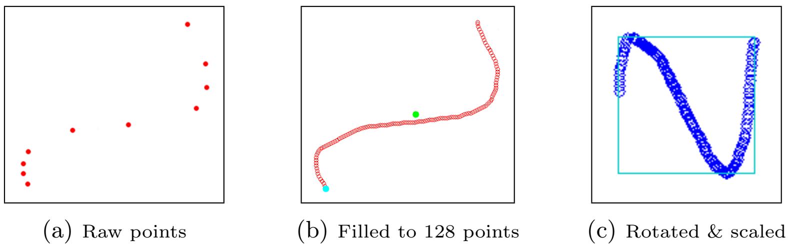
so-called "*four box*" features that are more fine-grained. Despite the unsatisfactory results for both, haarcascades and random forests, it is worth noting, that random forests have considerably lower training times while providing better results. Moreover, it is possible to extend our current implementation of random forests to use the "*four box*" features to improve on our results. Until we finish to do so, we make use of a haarcascade created by Daniel Baggio.[7] Evaluation on our test data yielded a true positive rate of 74% and a false negative rate of $9.1822 * 10^{-8}$, which corresponds to one false detection every six frames. Hence, in the current system it may happen, that the user has to repeat a gesture as a result of false detections.

Besides classifying between "hand" and "no hand" only, an additional dimension is to differentiate between several hand *postures*. As Tables 2 and 3 indicate, it is possible to train haarcascades as well as random forests to recognize specific single postures with true positive rates of up to 98% for haarcascades and at least 60% for random forests. Training times for random forests were considerably lower than for haarcascades. However, we did not succeed in building a single random forest classifier that was able to tell apart five different postures with a sufficiently high true positive rate. We hope to eventually achieve more accurate posture classification once we fully integrate the "*four box*" features used in the classifier from [6] into our random forest classifiers. Until then, we restrict ourselves to a few postures that we could train sufficiently accurate single classifiers for. Those classifiers can then be applied independently in parallel.

4.3 Deictic Gestures

As already mentioned, we can make use of the intermediate results after the hand detection and posture recognition steps already. These static gestures are referred to as *deictic gestures*. We trigger detection either when a specific posture (such as a pointing) is recognized or by means of an external trigger such as keywords like "*stop*" or "*there*" spotted by the speech recognition module running on the robot. As an additional cue to extract information from for pointing gestures we make use of the face detection module we used before already. We extract the face's position and compute a vector from the center of the face position to the center of the hand's position to identify a pointing target.

[7] `http://code.google.com/p/ehci/wiki/HandTracking`



(a) Raw points (b) Filled to 128 points (c) Rotated & scaled

Fig. 2. Processing steps on a *Snake* trajectory for gesture recognition [17].

5 Tracking & Gesture Recognition

After the position of a hand has been determined, its trajectory has to be tracked over time to enable gesture recognition. *Tracking* and *Gesture Recognition* are thus closely connected.

5.1 Hand Tracking

The general idea with tracking is that a hand's position will not change too rapidly from one image frame to the next. A naive method to track a hand is to detect all hands in the image and associate the closest hand in the subsequent frame. However, this presupposes a very reliable hand detection. A different technique is applied in [6], where hands with the same posture are tracked over time. The underlying assumption that the posture of the hand stays the same throughout a gesture does not seem to allow for too natural gestures, though.

In our system, we use a tracking scheme called "Flocks of Features" as proposed in [13]. Components to realize the tracking are readily available in OpenCV. After detecting a hand, features are computed and then tracked using KLT feature tracking [14]. In a test series of 20 "tries-to-escape" where the user erratically moved around the hand within the camera's field of view, the selected tracking scheme proved sufficiently robust. Hands were tracked over periods of between 1.5 and 34.98 seconds with an average of 18.48 seconds. This is sufficient for most gestures in a domestic setting.

5.2 Gesture Recognition

The final step in our system is to recognize a gesture from following the hand's position over time. First, the trajectory of the hand is recorded. Then, this trajectory is compared to a set of known gestures. Since we want to keep our system as free from training and as computationally inexpensive as possible, we opt to adapt a method introduced by Wobbrock et al. [17] initially designed for one-stroke hand-written gestures. It is fast to implement and still yields reliable results. The basic principle is to norm trajectories to a certain scheme and then to compare performed gestures to a set of known gestures by computing their individual points' distances.

The preparation of trajectories before the comparison to known gesture templates is as follows. Since pairwise comparing all points is too costly only points with the same index are compared. For this to work, the trajectories have to contain the same number of points, which is why every trajectory is filled to have equidistant points. Here we chose 128 as the number of points following results given in [17]. Afterwards, the trajectory is rotated to a standard orientation, i.e., an orientation of $0°$ between the first point and the centroid, to achieve rotational invariance. Then, the resulting point set is scaled to fit a box of 100×100 pixels. Preparation steps are shown in Figure 2

Basic movements like "left to right" correspond to lines in a trajectory. Unfortunately, the detection of straight horizontal and vertical lines is problematic with the method from [17]. This is why we need to slightly modify the approach by applying a different scaling mechanism. If the ratio between width and height of the unscaled trajectory is below a certain threshold (empirically determined to be 0.3) we only scale the larger side of the box. This prevents the scaled line-trajectories from being too scattered to be recognizable. We adjust the gesture templates to compare with accordingly.

Evaluation We conducted a separate mouse-based evaluation of the gesture recognition component to yield results independent from the hand detection itself. We recorded a total set of 314 gestures (one of *Line*, *Wave*, *Square*, *Circle*, and *Triangle*) where 85.67% were recognized correctly. More than 50% of the false detections were confusions between *Square* and *Circle* gestures. One reason could be imperfections in the mouse-based input. Preliminary findings in the analysis of human gestures suggest that humans perform gestures surprisingly precise. If we then assume the trajectories from human gestures to be more precise than the ones created artificially, these confusions might drop down already. Moreover, we additionally applied a corner detection on the trajectory to clear up these confusions. With this extension in effect we could raise the detection rate to above 92%.

6 Conclusion

In this paper, we proposed a modular system for visual gesture recognition for interaction with a domestic service robot. The processing pipeline is organized to reduce the amount of information to process at every step, starting with an inexpensive on-line adapting color-based method for hand detection, leaving only parts of the image to be processed by the more expensive feature-based posture recognition. The actual gesture recognition adopts a fast approach originally designed for hand-written gesture recognition that does not require any training. Results from intermediate processing steps can already be used for interaction, for example, to react on deictic pointing or stop gestures.

Although our system suffers from shortcomings in the hand verification and posture recognition steps yet, the overall performance is sufficient for it to be applicable in a domestic setting with some minor restrictions. While we successfully used the system at a recent major robotics competition already, each individual step can still be improved. These improvements, however, can be realized relatively easily due to the modular design. Likewise, anyone replicating the system can exchange any of the proposed components to his or her liking. Our next step is to fully integrate the "*four box*" features from [6] in our random forest implementation. Future work further includes the extension to work on 3D position information for hands and for gestures, respectively.

References

1. Engleberg, I.N., Wynn, D.R.: Working in Groups: Communication Principles and Strategies. 4 edn. Allyn & Bacon (2006)
2. Saxe, D., Foulds, R.: Toward robust skin identification in video images. In: Proc. 2nd Int'l Conf. on Automatic Face and Gesture Recognition. (1996) 379
3. Jones, M., Rehg, J.: Statistical color models with application to skin detection. International Journal of Computer Vision **46** (2002) 81–96
4. Francke, H., Ruiz-del Solar, J., Verschae, R.: Real-time hand gesture detection and recognition using boosted classifiers and active learning. In: Proc. 2nd Pacific Rim Conf. on Advances in Image and Video Technology, Springer (2007) 533–547
5. Zhu, X., Yang, J., Waibel, A.: Segmenting hands of arbitrary color. In: Proc. 4th Int'l Conf. on Automatic Face and Gesture Recognition. (2000) 446
6. Kölsch, M., Turk, M.: Robust hand detection. In: Proc. of the 6th IEEE Int'l Conf. on Automatic Face and Gesture Recognition. (2004) 614–619
7. Viola, P.A., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: Conf. on Computer Vision and Pattern Recognition (CVPR 2001). Volume I., Los Alamitos, CA, USA, IEEE Computer Society (2001) 511–518
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci. **55** (1997) 119–139
9. Ho, T.K.: Random decision forests. In: Proc. of the 3rd Int'l Conf. on Document Analysis and Recognition (ICDAR'95). Volume 1. (1995) 278–282
10. Belle, V., Deselaers, T., Schiffer, S.: Randomized trees for real-time one-step face detection and recognition. In: Proc. of the 19th Int'l Conf. on Pattern Recognition (ICPR'08), IEEE Computer Society (2008) 1–4
11. Panin, G., Klose, S., Knoll, A.: Real-time articulated hand detection and pose estimation. In: Proc. of the 5th Int'l Symposium on Advances in Visual Computing (ISVC '09), Berlin, Heidelberg, Springer-Verlag (2009) 1131–1140
12. Triesch, J., von der Malsburg, C.: Robust classification of hand postures against complex backgrounds. In: Proc. of the 2nd Int'l Conf. on Automatic Face and Gesture Recognition (FG '96), Washington, DC, USA (1996) 170–175
13. Kölsch, M., Turk, M.: Fast 2d hand tracking with flocks of features and multi-cue integration. In: IEEE Workshop on Real-Time Vision for Human-Computer Interaction at Conf. on Computer Vision and Pattern Recognition. (2004) 158
14. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94). (1994) 593–600
15. Bradski, G.R.: Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal **1** (1998) 1–15
16. Mitra, S., Acharya, T.: Gesture recognition: A survey. IEEE Transactions on Systems, Man, and Cybernetics (Part C) **37** (2007) 311–324
17. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: Proc. of the 20th ACM symposium on User Interface Software and Technology. (2007) 159–168
18. Viola, P.A., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision **57** (2004) 137–154
19. Kumar, A.: Incorporating cohort information for reliable palmprint authentication. In: Proc. 6th Indian Conf. on Computer Vision, Graphics & Image Processing. (2008) 583–590