A Vision-based Remote Control

Björn Stenger, Thomas Woodley, Roberto Cipolla

Abstract This chapter presents a vision-based system for touch-free interaction with a display at a distance. A single camera is fixed on top of the screen and is pointing towards the user. An attention mechanism allows the user to start the interaction and control a screen pointer by moving their hand in a fist pose directed at the camera. On-screen items can be chosen by a selection mechanism. Current sample applications include browsing video collections as well as viewing a gallery of 3D objects, which the user can rotate with their hand motion. We have included an up-to-date review of hand tracking methods, and comment on the merits and shortcomings of previous approaches. The proposed tracker uses multiple cues, appearance, color, and motion, for robustness. As the space of possible observation models is generally too large for exhaustive online search, we select models that are suitable for the particular tracking task at hand. During a training stage, various off-the-shelf trackers are evaluated. From this data different methods of fusing them online are investigated, including parallel and cascaded tracker evaluation. For the case of fist tracking, combining a small number of observers in a cascade results in an efficient algorithm that is used in our gesture interface. The system has been on public display at conferences where over a hundred users have engaged with it.

1 Introduction

This chapter presents a vision-based gesture interface using a single camera on top of a display, allowing touch-free input at a distance. Figure 1 shows photos from a

Björn Stenger,
Computer Vision Group, Toshiba Research Europe, Cambridge, UK. e-mail: bjorn@cantab.net

Thomas Woodley
Department of Engineering, University of Cambridge, UK. e-mail: tew32@cam.ac.uk

Roberto Cipolla
Department of Engineering, University of Cambridge, UK. e-mail: cipolla@eng.cam.ac.uk

Fig. 1 Showcase of the proposed gesture interface. *(Left) A single camera on top of the display is directed towards the user, who is able to control a screen cursor with his fist. Videos can be selected by hovering over a button. (Right) The playback of a video can be stopped with an open hand gesture.*

showcase presented during the *Internationale Funk Ausstellung* IFA 2008 exhibition in Berlin. Such a gesture system may have several uses in practice: to remotely control a TV or other appliances, or browsing public information terminals in museums or shop windows. There are many factors that make hand tracking from a single view difficult in practice. Hands can exhibit a wide range of appearances, for example due to changes in pose and in scene lighting. Furthermore, the variability of shapes, poses, and color between different people is high. At a standard frame rates of 30 frames per second there may also be significant motion blur. Figure 2 illustrates this variability, showing examples of cropped hand regions, taken from image sequences of four different people. The system should also use minimal computational resources because any lag in interactive applications is very noticeable.

In addition to fast and robust tracking, a method for automatic initialization is required to find the hand at the beginning of the interaction and after tracking failure. Loss of track occurs regularly, for example every time the hand is outside the camera's view. The proposed system thus integrates an off-line trained detector to initialize and also to update the tracker in order to avoid drift. Building a general, robust hand detector is still a challenging problem, and we restrict ourselves to detecting a particular pose, in this case a fist pose. When viewed from the front, the visual appearance of a fist is characteristic and a robust detector has been trained for it.

In the following section we provide an overview of prior work on hand tracking and object tracking in general. Section 3 explains the design of the tracking algorithm and presents experimental results on test data. The full system and its components are described in Section 4.

Fig. 2 Appearance variation of hand regions. *Shown are cropped hand regions from test sequences. Motion blur, changing pose and other skin colored objects make tracking challenging.*

2 Prior Work

A large number of vision-based gesture interfaces have been proposed in the scientific literature and some systems have already been commercialized. This section provides an overview of hand tracking methods in image sequences. It also summarizes developments in the area of general object tracking as some of these methods are used in Section 3.

2.1 Hand Tracking for Human Computer Interfaces

Reviews on hand tracking have been published by Pavlović *et al.* [62], Wu and Huang [86, 88]. These papers contain a good taxonomy of early work on hand tracking and gesture interfaces. More recently, Erol *et al.* [25] published a review of full 3D hand tracking. Generally, there are a number of factors to consider when designing or describing a hand tracking system.

1. **The number of state parameters:** Methods differ by the number of parameters they estimate. This may range from the case where only the 2D location of a hand in an image is obtained to the case where the full articulated pose in 3D is estimated. In some cases a dynamic model is used, whose parameters are included in the estimation process.
2. **The estimation method and features used:** At the heart of each system is the algorithm which estimates the state parameters from the observed image data. Some methods employ an explicit geometric model and use a model-fitting approach, while others take the approach of learning from training data. Hybrid approaches exist too, which generate training data from a geometric model. Methods also differ in the types of features they use, which can be based on color, shape, or motion of the hand.
3. **The set-up and capture system:** There exist a wide range of set-ups, differing in the number and position of cameras, for example. Systems that use two or more cameras may compute a depth map or a visual hull as the input to the recognition system. Furthermore, active systems such as structured light or time-of-flight systems are becoming more common and allow depth estimation that is often

more robust than passive two-view stereo. Other important parameters are the camera's resolution, light sensitivity and frame-rate. Clearly the camera position also makes a difference: whether it is facing top-down towards a desktop, whether it is facing towards a user in front of an uncontrolled background, or whether it is mounted on a mobile robot platform.

These factors should be kept in mind whenever analyzing a gesture interface system, as the underlying assumptions differ in each case. In the following we introduce some hand tracking systems that have been prominent in the literature.

One of the first systems for markerless hand tracking was that of Cipolla and Hollinghurst [14, 15], who used a B-spline active contour as a 2D shape model to track a pointing hand from two uncalibrated views. In each view an affine transformation of the contour was estimated, and using a ground plane constraint the indicated target position could be found. The system required a simple background, but it could operate in real-time. Freeman and Weissman [27] introduced a single-camera system for television remote control by tracking an open hand. It used an image subwindow as a template for both detection and tracking. Matching was performed using local edge orientation in order to be robust to some illumination changes. In their *EigenTracking* work, Black and Jepson [10] proposed an eigenspace representation of a set of hand images. Using a coarse-to-fine matching strategy, both the affine transformation as well as the closest of four gestures were estimated. Isard and Blake [40] modeled the hand shape with a B-spline. The tracker combined color blob tracking with contour tracking in a particle filter framework. Later, MacCormick and Isard [52] presented a drawing system based on a this tracker together with independent sampling of the finger parameters. Tosas [76] recently presented a similar color-based contour tracker in a number of demo applications, such as a 'virtual turntable'. Wu and Huang [87] applied a learning based method, combining labeled and unlabeled data with the EM algorithm and using neural network classification to distinguish 14 hand postures in different views. The classification rate was 92% using features obtained by principal component analysis of the hand images. Triesch and von der Malsburg [78] built a hand gesture interface for robot control. Elastic 2D graph matching was employed to match templates of twelve different hand gestures to an image. Combined Gabor and color features made the system relatively stable to clutter, achieving a recognition rate of 93% in front of simple background and 86% for cluttered backgrounds. Bretzner *et al.* [11] used multi-scale blob detection of color features in order to detect an open hand pose with possibly some of the fingers extended, corresponding to different input commands. A simple 2D shape model was used for tracking with a particle filter. The method required a skin color prior, which was obtained by manually labeling 30 frames. The system tracked at 10 fps and was also demonstrated in a TV remote control application. Lockton and Fitzgibbon [50] built a real-time system that could recognize 46 different hand poses, including finger spellings in American Sign Language. Extracting hand silhouettes in each frame using color, their method was based on efficient template matching. Accurate registration was facilitated by a wrist band and led to a very high recognition rate. Krahnstoever *et al.* [47] presented a multi-modal vision and speech interface to interact with a large display. The hand

tracking component was based on finding location hypotheses in each frame and matching them over a time-window with the Viterbi algorithm. A spatial prior was used to associate blobs to hand and face. An interface based on tracking *multiple* skin colored regions was proposed by Argyros and Lourakis in [1]. The skin color model was obtained by manually labeling skin regions, but the color model was adapted during tracking. The tracker was used in a stereo-system in order to realize a 3D mouse application [2]. Extensions of cascaded detection using AdaBoost were proposed in [46, 60]. Whereas Kölsch and Turk [46] showed robust detection of specific hand poses, Ong and Bowden [60] trained a classifier hierarchy for multi-pose detection. Kölsch and Turk [45] further presented a multi-cue tracker that combined color and a number of local features under 'flocking' constraints. The color model was automatically initialized from hand detection. The system by Robertson *et al.* [67] used a trained detector followed by optical flow tracking and was employed in a 'virtual mouse' application. Ike *et al.* [37] presented a real-time system for gesture control that could detect three different hand poses independently in each frame. Due to the high computation requirement it was implemented on a multi-core processor.

To summarize, a large number of hand trackers for gesture recognition have been proposed in the literature. We have re-implemented some of these for evaluation.

One class of applications, which this chapter does not discuss in detail are *virtual desktop* applications [42, 58, 70, 83]. Originally, in most of these systems the camera and possibly a projector are mounted above, facing down towards the tabletop and a user is able to interact with real and projected virtual objects. Similarly, by placing the camera below a transparent tabletop a multi-touch interface can efficiently be implemented, such as in *Microsoft Surface* [53].

In addition to gesture-based control, a further target application of hand tracking is automatic sign language recognition. The pioneering recognition system by Starner *et al.* [69] modeled a hand by a skin-colored ellipse that was tracked in image sequences. A hidden Markov model was then used to recognize a 40 word vocabulary based on the shape and motion trajectory, obtaining a recognition rate of 98%. Much progress has been made recently, see [61] for a general survey, and [12, 19] for state-of-the-art results.

For some applications, such as motion capture for animation or biomechanical analysis, it is desirable to fully capture the hand motion in 3D. Currently these methods use either colored gloves, markers on the hand, or data gloves with built-in sensors. However, there has been progress in markerless 3D hand tracking [3, 21, 35, 66, 68, 72]. A common approach to full 3D hand tracking is to use a geometric hand model. The model is usually created manually, but can also be obtained by reconstruction methods. Models that have been used for tracking are based on planar patches [89, 90], deformable polygon meshes [35] or generalized cylinders [22, 66]. The underlying kinematic structure is based on the biomechanics of the hand. Each finger can be modeled as a kinematic chain with four degrees of freedom (DOF) attached to the palm, and the thumb may be modeled similarly with either four or five DOF. Together with rigid body motion of the hand, there are 27 DOF to be estimated. Working in such a high dimensional space is particularly

difficult because feature points that can be tracked reliably are sparse: the texture on hands is typically weak and self-occlusion occurs frequently. However, the anatomical design places certain constraints on the hand motion. These constraints have been exploited in different ways to reduce the search space. One type of constraint are the joint angle limits, defining the range of motion. Another very significant constraint is the correlation of joint movement. Angles of the same finger, as well as of different fingers, are not completely independent. These constraints can be enforced in different ways, either by parameter coupling or by working in a space of reduced dimension, e.g. one found by PCA [35, 89]. To summarize, while headway has been made on the problem of full 3D tracking, the task remains challenging. It has been shown to work in principle, but is often constrained to slow hand motion or controlled scenes. The problem is generally ill-posed using a single camera, because in some poses finger motion can be unobservable due to self-occlusion. The use of additional hardware, such as multiple cameras [20, 32], colored gloves [82] or depth-cameras [34], can therefore help significantly.

2.2 Commercial Gesture Interface Systems

A number of commercial systems for gesture recognition have already been made available and more are likely to follow.

Game console makers have shown interest in gesture recognition, while proposing different solutions. While Sony has been working with a passive vision system [26, 64], Nintendo is using a wireless controller with accelerometer and optical sensing technology [56] and Microsoft has shown technology based on active illumination [65]. One of the first widely known vision systems was the *EyeToy* camera in 2003 [26] that could be connected to the *Play Station 2* game console and its successor in 2007, the *PlayStation Eye* for the *PS3* [64]. Here the camera is placed on top of the screen, facing the user who can interact with menus and games using gestures. In gaming, it is clearly important to avoid any lag that can interfere with game play. Another key requirement is to handle dark or changing lighting conditions in living rooms. Until now, the algorithms used for gesture recognition used in *EyeToy* games have been fairly basic yet sufficiently robust. Two examples are optical flow estimation and frame differencing in order to find regions of object motion.

Oblong Industries, co-founded by John Underkoffler, commercialized *g-speak*, an interface using gloves and markers [57]. Underkoffler was also a scientific adviser for the 2002 science fiction movie *Minority Report* which featured a similar gesture interface for video navigation.

In order to overcome many of the robustness issues, active systems such as time-of-flight cameras, have been used for gesture recognition, e.g. [13]. These systems directly output a depth map, considerably simplifying feature extraction and segmentation. Companies such as *GestureTek* have commercialized a number of gesture recognition systems using time-of-flight cameras, for example systems for public exhibitions [28].

In 2008 the *Toshiba Qosmio* laptop first shipped with hand gesture control software that worked with the built-in webcam [77]. The gesture system works by integrating global hand detection [38, 54] with local tracking. Users are able to control a screen pointer with their hand and select objects by moving their thumb. The work presented this chapter mainly builds on this system.

2.3 Visual Tracking of a Single Object

In the following we take a more general view and consider the task of robustly tracking a single object in image sequences. This has been studied extensively in the computer vision literature [9, 18, 33, 36, 39, 75, 79, 85]. The task can be stated as follows: Given an initial state of the object in the first frame, estimate the state of the object in the subsequent frame, then do this sequentially on each frame of the sequence. The state of the object contains the parameters of the geometric transformation that we are interested in. This can be, for example, the target's *x*-*y*-position in the image or the full three-dimensional pose. Some methods also estimate the shape or articulation parameters of a target object. Tracking methods also differ in the image cues they employ to measure the similarity from frame to frame. These can be raw or normalized image pixels, edge contours, color histograms, outputs of oriented filters, or any other features computed from the object region. In order to successfully track the object in the next frame, the underlying assumption is that the features are still sufficiently characteristic for the object (*generative* approaches), or make it appear different from the background (*discriminative* approaches). The chance of finding discriminative features is clearly increased when combining multiple cues. Numerous papers on multi-cue tracking have demonstrated the concept of different cues complementing each other and overcoming the failure cases of individual cues [6, 9, 24, 31, 48, 55]. A typical example is a hand being tracked while it moves in front of the face. The hand may still be tracked based on shape while color features become less reliable. The most common approach to multi-cue tracking is to evaluate several observers in parallel and subsequently combine their output, by either switching between them [6] or by probabilistically merging them [24, 48, 55, 63]. A key issue when merging tracking results is how to obtain a good confidence measure for each cue. This is a tricky question since the performance of one cue may only be assessed by using a different cue or different representation of the target object. One answer is the discriminability between foreground object and background region. This is the basis of recent work on *discriminative* tracking [5, 16, 29], where tracking is formulated as a classification task. Collins *et al.* proposed a method for online feature selection which selects the most discriminative features from a pool of color-based features [17]. The ability to discriminate was evaluated as either the two-class variance ratio or the difference of the top two likelihood peaks. Avidan introduced *ensemble tracking*, where multiple (3-5) weak classifiers were combined via AdaBoost [5]. At each frame a new weak classifier was learned and the ensemble was updated by replacing the least reliable

classifiers in each time step. Grabner and Bischof applied online boosting to feature selection [29]. Features from a larger pool of 250 weak classifiers were evaluated and and a set of 50 selectors chose those that were combined into a strong classifier.

In practice, an issue with online adaptation is the *adaptability vs. drift* trade-off: Allowing the tracker to adapt to rapid changes of the object's appearance bears the risk of incorrectly adapting to the background. Ideally an object model is available that includes all possible variations. Such a model could then be used as an 'anchor' for the tracker. Obtaining such a model is challenging and different representations have been used, including the color distribution [6], a representation learned from a short initial sequence [30] or an off-line trained detector [4, 45, 84]. Detectors have been included into tracking systems, for example, by running them in tandem [45, 84] or by closely integrating them with the tracker's estimation procedure [4, 49, 59]. Indeed, a viable tracking solution is to use a detect-and-connect strategy, shown for example in [44]. In many cases this approach is not yet sufficiently fast for real-time tracking and the detectors lack sufficient flexibility, but this is a promising avenue for future research.

Existing algorithms that integrate multiple cues choose their component observers in a heuristic manner beforehand [6, 9, 49]. In the following we are proposing a method to make this choice in a more principled way.

3 Tracking with Multiple Observers

We now address the question of how to design a tracker using multiple observation models. The observation models are components from different stand-alone tracking algorithms such as single template matching, optical flow and online classification. The idea is to learn which of these are suitable components and how they should be arranged for efficient evaluation. We collect a set of training sequences and ground-truth label them by hand. On these sequences we evaluate error distributions for different observers. The ground truth labels allow us to evaluate combinations of observers on a test set. The term 'observer' in this chapter refers to an observation model, and can be seen as a component of a tracker, the other component being the dynamical model. However, in the evaluation we only rely on the observations within a search window around the previous estimate, thus both terms may be used interchangeably here. We consider the particular tracking scenario of tracking a fist using a static camera. However, the method is general and can be applied to other settings [73].

3.1 Observation Models

The goal is to find, for a given tracking scenario, the best observer or combination of observers. Our approach is to first evaluate each observer individually and from

Method	Observation	Estimate	Confidence value
NCC	Normalized cross correlation	max correlation	correlation score
SAD	Sum of absolute differences	min distance	distance score
BOF	Block-based optical flow of 3×3 templates	mean motion	mean NCC score
KLT [51]	Kanade-Lucas-Tomasi sparse optical flow using 50 features	centroid of good features	fraction of good features
FF [45]	Flocks of features: Tracking 50 local features with high color probability and 'flocking' constraints	centroid of good features	fraction of good features
RT [6]	Randomized templates: NCC track of eight subwindows, with motion consensus and resampling	centroid of good features	fraction of good features
MS [18]	Mean shift: Color histogram-based mean shift tracking with background weighting	min histogram distance	histogram distance
C [71]	Color probability map, blob detection	scale space maximum	probability score
M [71]	Motion probability map, blob detection	scale space maximum	probability score
CM [47]	Color and motion probability map	scale space maximum	probability score
OBD [29]	Online boosted detector: Classifier boosted from pool of rectangle features updated online	max classifier output	classifier margin
LDA [49]	LDA classifier computed from five rectangle features in the previous frame (Observer 1 in [49])	max classifier output	classifier margin
BLDA [49]	Boosted LDA classifier using 50 LDA classifiers from a pool of 150, trained on the previous five frames (Observer 2 in [49])	max classifier output	classifier margin
OFS [16]	Online feature selection of 3 out of 49 color-based features based on fg/bg variance ratio	centroid of top features	mean variance ratio of selected features

Table 1 Observers in the evaluation. *A diverse range of observers are tested in the experiments. They can be grouped into four types: single template matching, local feature matching, histogram-based region matching, and online classifiers. Between them they use a variety of cues, including image intensity, color and motion features. Some observers maintain a fixed representation while others are updated over time.*

these values measure the performance of combinations of observers. The observers we consider are those used previously in tracking algorithms, see Table 1. They can be classified into four types: single template matching, motion consensus of local features [6, 45, 51], histogram-based region matching [18] and online classification [16, 29, 49]. Note that the individual observers are not restricted to using a single cue.

Single Template Matching. The first two observers use normalized cross correlation (NCC), and sum of absolute differences (SAD), respectively. Given the subwindow of the most recent detection, the matching score in the search window is computed exhaustively. We also performed initial experiments using correlation of local orientations, used by Freeman and Weissman [27]. We found that the tracker works when the hand moves slowly, but edge features tend to be unstable when motion blur occurs.

Local Feature Matching. The first local feature method is block-based optical flow (BOF), where the object region is divided into a regular 3×3 grid of subwin-

dows which are matched independently in the next frame using NCC. The second method computes sparse optical flow on salient features, using the Kanade-Lucas-Tomasi (KLT) tracker with a set of 50 features [8, 45]. The third method is the 'flocks of features' tracker by Kölsch and Turk [46]. It combines motion cues with a learned object color distribution. KLT features are used to compute motion which has to satisfy the following constraints constraints: (1) all features maintain a minimum distance from each other, and (2) no feature is further from the current median than a maximum distance. The name of the method comes from the similarity to the motion pattern of flocks of birds. A further constraint is that the features need to be located in regions of high object color probability. Features that do not satisfy all three constraints are pruned and replaced with new ones. Finally, we have re-implemented the randomized template method of [6]. A set of eight templates that are randomly located and sized within the object region is tracked using NCC.

Region Matching. The first method in this category is color based mean shift [18], where the best match is found by minimizing the distance of the color distribution to the target. We use an RGB histogram representation, where each color channel is divided into 16 bins. In addition we use the background color weighting scheme that was proposed as an extension in [18]. The next method is based on pixel-wise color probabilities as proposed by Jones and Rehg for skin color detection [43]. Color distributions of the object and the surrounding background region are obtained during initialization and the probability for each pixel belonging to the foreground is computed. We then run a box filter over this probability image that finds blobs, i.e. regions of high object probability surrounded by regions of low object probability [71]. This is quite similar to the hand tracker by Bretzner *et al.* [11] where scale-space extrema in color feature space are found. The third method uses the motion cue in a similar way. It computes the pixel-wise probability of motion in a region. The distributions for moving regions and background are obtained off-line from a hand labeled sequence of frame difference images. Note that in general this cue is typically present near the object's boundary, but not necessarily inside the object for homogeneous surfaces. The final method combines both color and motion cues. The function combines three terms as a weighted sum as in [47]. The functions are smoothed spatially by Gaussians with a variance depending on the size of the previously detected hand. The pixel-wise probability density function of observing a hand at image location $\mathbf{y}$ is defined as

$$p(\text{hand}|\mathbf{y}) \propto w_c\, p(\mathbf{y}|\text{col}) + w_m\, p(\mathbf{y}|\text{mot}) + (1 - w_c - w_m)\, p(\mathbf{y}|\text{col})\, p(\mathbf{y}|\text{mot}), \quad (1)$$

where w_c and w_m are weights that are determined through experiments on a validation set (in our case $w_c = w_m = 0.1$).

Online Classification. The first method in this category is a re-implementation of online boosting as proposed by Grabner and Bischof [29]. A classifier is learned online by selecting a set of rectangle features (weak classifiers) from a pre-selected pool of 250 features. Assuming that the object location is known at frame t, the classifier is evaluated in the neighborhood of the previous location to create a confidence map. The new object location is moved to its maximum and, using the new la-

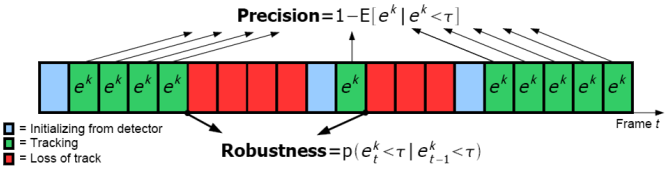


Fig. 3 Observer evaluation. *At each frame t of a test sequence an observer O^k outputs its position estimate $\hat{\mathbf{x}}_t^k$ and confidence value c_t^k. The position error e_t^k relative to the ground truth is calculated during successful tracking (represented by green cells). Loss of track occurs when the error exceeds a threshold τ (switch to red). Tracking is re-initialized from an off-line trained detector (blue). Precision and robustness metrics are calculated from the test results.*

bels for object and background, new training examples are sampled from the image to update the classifier. The second online method uses linear discriminant analysis (LDA), based on five rectangle features. The third method is Boosted LDA and combines 50 classifiers from a pool of 150 using AdaBoost. These two methods have been proposed by Li *et al.* [49] who used these as observation models in a particle filter trained over different time periods. While the LDA classifier is trained only on the previous frame, the Boosted LDA classifier is trained on the previous five frames. Finally, we have implemented the online feature selection (OFS) scheme by Collins *et al.* [17], where discriminative color features are found using the variance ratio criterion.

We train a detector to initialize and re-initialize each tracker. The detector was trained off-line using AdaBoost for feature selection [54, 81] from a dataset of approximately 5000 positive and negative examples. The version that we use [54] uses feature co-occurrence to increase the performance over the baseline method.

3.2 Evaluating Single Observers

The evaluation of observers proceeds as follows. Given an image sequence $I_t, t = 1,\ldots,T$, at every time step t each observer $O^k, k = 1,\ldots,K$ computes an estimate of the target location $\hat{\mathbf{x}}_t^k$ as well as the distance to the labeled ground truth location $\mathbf{x}_t^{gt}$ as error $e_t^k = d(\hat{\mathbf{x}}_t^k, \mathbf{x}_t^{gt})$. The estimate $\hat{\mathbf{x}}_t = (x, y, s)$ contains the center location x, y and scale estimate s. The error is computed as the the scale-normalized distance between the centers. Observers that do not estimate the scale s, obtain this value from the most recent detection and it remains constant during tracking. Every observer also outputs a confidence value c_t^k at each frame, which is computed depending on the type of observer, see the right column of Table 1. Loss of track occurs when the location error e_t^k is above a threshold value $\tau < 1$. In this case the tracker outputs τ as error and is re-initialized at the next successful detection. The detections are pre-computed by running the off-line detector over all sequences. In summary, the measurements for the individual observers O_k on a training sequence are given by

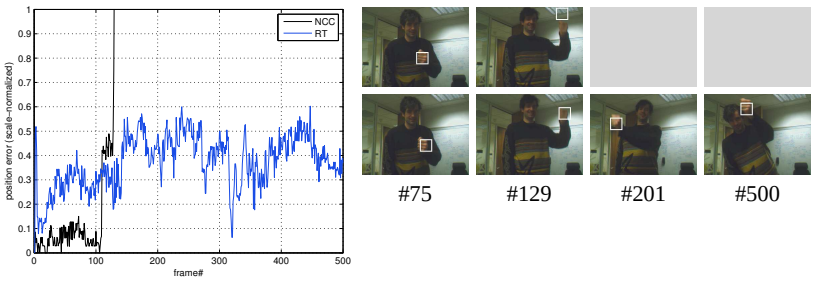


Fig. 4 Example of precision vs. robustness of trackers. *The plot shows the tracking error of two stand-alone trackers with different observation models: maximum correlation (NCC), and randomized template tracking (RT). In this example, NCC is more accurate but fails early on, while RT is able to track over a longer period with less precision.*

$$\mathscr{Z}^k = \{\hat{\mathbf{x}}_t^k, e_t^k, c_t^k\}, \;\; t = 1, ..., T\,. \tag{2}$$

This allows the evaluation of single observers on the complete sequence, not just on the first successfully tracked segment. Loss of track can occur at any time during the sequence when an observer's particular assumptions, e.g. slow motion or small pose change, do not hold. The number of tracked frames when running the tracker only once is dependent on when this event occurs: if it is near the beginning of a sequence the measured robustness is worse than when it is near the end. The performance of an observer is estimated as the expected error over all frames,

$$E[e^k] = \frac{1}{T}\sum_t e_t^k, \; k = 1, ..., K. \tag{3}$$

However, this function does not allow the comparison of observers when track is lost, because the error is meaningless in this case. In practice we are therefore interested in both the tracking error while the tracker is following the target as well as the probability of losing track. This motivates the distinction into two performance criteria, precision and robustness. Precision is related to the expected error only during successful tracking by

$$1 - E[e^k | e^k < \tau]. \tag{4}$$

The robustness is the probability of successful tracking as

$$E[e_t^k < \tau | e_{t-1}^k < \tau]. \tag{5}$$

See Figure 3 for a schematic of the evaluation on one sequence.

It is interesting to look at the relationship between precision and robustness. Observers with a fixed spatial model tend to be more precise than observers where the spatial arrangement is more flexible, see for example Figure 4, which shows tracking without re-initialization using single template matching (NCC) and local feature matching (RT) on one sequence. In this example NCC is more precise than RT, but

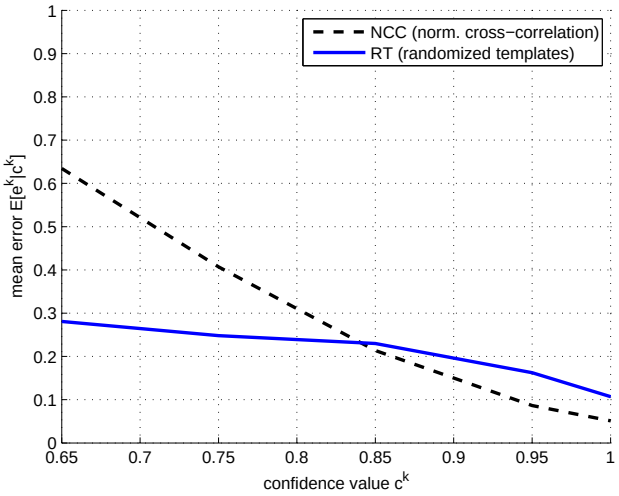


Fig. 5 Expected error as function of confidence. *This data is obtained from training sequences and allows the direct comparison of observers given their confidence values. Shown here are the results for observers NCC (normalized cross-correlation) and RT (randomized templates).*

the tracker loses lock earlier. Note that similar ideas have recently been explored in the visual object classification literature, where a representation's invariance vs. discriminative power trade-off was explored [80].

In order to evaluate each observer individually in a tracking algorithm, we use a threshold value of $\tau = 1$ on the tracking error (in Equations 4 and 5) to determine loss of track. When this value is exceeded, the tracker is re-initialized at the next detection. The value of $\tau = 1$ corresponds to the case where track has clearly been lost. Other threshold values could be used, where a smaller value enforces higher precision and lower robustness, and vice versa. Precision and robustness are then computed by taking expectations over all frames of the test sequences.

3.3 Evaluating Multiple Observers

This section deals with the question of how to evaluate the performance that can be achieved by combining multiple observers. Ideally, we would like to select the observer O^k with the lowest error e_t^k at each time step. This information is not available at test stage, so instead the observer's confidence value c_t^k is used. Confidence values have often been used to compare the results of multiple observers and combine them [6, 9, 48]. However, most observers have a relatively simple object representation and thus the confidence value itself cannot be expected to be perfectly reliable. For example, an observer may have a high confidence value at an incorrect location

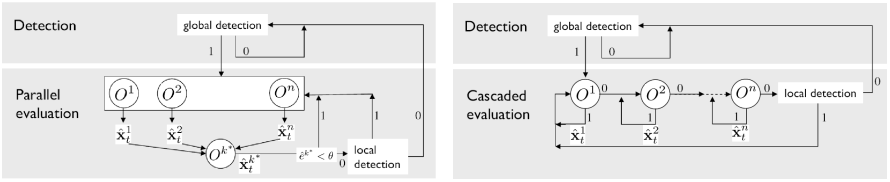


Fig. 6 Evaluation schemes. *(a) In the parallel evaluation, the output from the observer with the lowest expected error is chosen. (b) In the cascaded evaluation, the next observer is only evaluated if the expected error is above a threshold. An off-line trained detector is used to re-initialize. The binary tests in this schematic represent threshold tests on the expected error.*

if there is an object close-by that is similar to the target in the observer's feature space. The observer confidence values cannot be compared directly in our case, so they are simply regarded as features computed by each observer. In order to make them comparable we estimate the distributions $p(e^k|c^k)$ from the training data, i.e. the error distribution of observer O^k given its confidence value. To use the finite data set we discretize the range of the c^k values and compute $p(e^k|c^k)$ in each partition. For the evaluation we represent it by the mean of each distribution, $E[e^k|c^k]$. Thus the estimated error for an observer O^k at time t is $\hat{e}_t^k = E\left[e^k|c_t^k\right]$. Figure 5 shows two of these functions for the normalized cross-correlation (NCC) and randomized templates (RT) observers. For example, if both observers return a confidence value of 0.9, the expected error of NCC is lower than that of RT.

We distinguish two different approaches of combining observers: parallel and sequential, respectively. In parallel evaluation, the estimates of multiple observers are available at each time step and the output of the most reliable observer is selected. Note that alternative fusion methods could be used, for example weighting the observer estimates. In sequential (or cascaded) evaluation observers are evaluated in sequence: If the first observer returns a high confidence, then no other observer is evaluated. Otherwise, the evaluation continues with the next observer. The advantage of sequential observation is that on average significantly less computation is required. However, the order of evaluation as well as the thresholds on the expected error are critical for good performance.

3.4 Parallel Evaluation

The parallel evaluation scheme selects the observer with the lowest expected error given its confidence value at each time step, i.e. $k_t^* = \text{argmin}_k\, \hat{e}_t^k$, see Figure 6(a). If this error is above a certain threshold, then the tracker is re-initialized at the next successful detection.

Fig. 7 Fist data set. *The evaluation is performed on a data set of 12 sequences of 500 frames each, (top) six used for training and (bottom) six for testing. The sequences are taken with a static camera on a display showing people moving their hand in front of the screen. The dataset contains motion blur, skin-colored objects in the background, and occasionally other people in the scene.*

The running of tests consisting of all possible combinations of all trackers on all test sequences would be very time consuming. We therefore run all the observers individually on the test sequences and record the results on all frames. These are then used in the combination tests as the result from each component observer. In order to confirm the validity of such a set-up, we subsequently perform tests using the complete tracking framework for a few combinations of observers.

3.5 Cascaded Evaluation

Although the combined estimate is generally expected to be better than individual estimates, the main disadvantage is the increased execution time. In cascaded evaluation observers are evaluated in sequence, starting with the first observer, and continuing with the next observer only if the expected error is above a threshold value, see Figure 6(b). If no observer returns a sufficiently low expected error, the algorithm attempts to jump to the top of the cascade using local detection. In the evaluation, as in the parallel case, the output of the individual observers is used to estimate the performance of different combinations of observers as well as threshold values for switching observers.

3.6 Dynamic Model Discussion

A dynamic model is an integral component in every tracking algorithm as it can enable tracking through short periods of occlusion or weight the observations according to the most likely target motion. However, for the evaluation we aim to be independent of the dynamics, which are difficult to model in the case of rapid hand motion. Instead we sample the observation space densely at each pixel location in a neighborhood around the previous estimate and rely only on the observations without prediction, corresponding to a maximum likelihood location estimate.

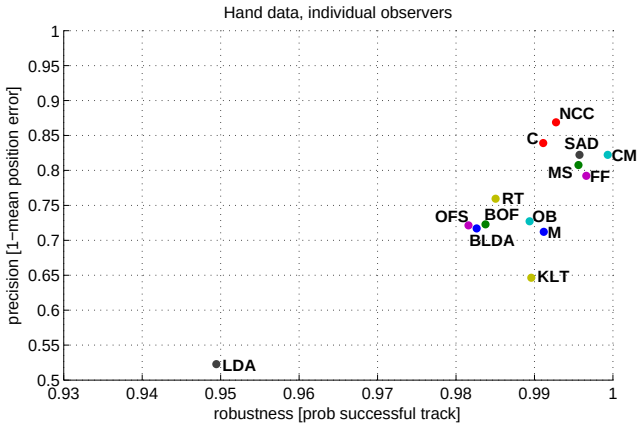


Fig. 8 Evaluation of individual observers. *This plot shows precision and robustness values on the test data. NCC is the most precise observer, the color-motion observer (CM) is the most robust.*

This methodology is consistent with the observation made in the particle filtering literature that the performance largely depends on the proposal distribution [23]. If good motion models are available, these should certainly be integrated in the final system [41]. The final tracking algorithm used in the gesture interface does employ prediction based on a constant velocity model for defining the search region and an auto-regressive filter for temporal smoothing.

3.7 Experimental Results

We evaluate the method on a hand dataset containing 12 sequences (10 with rapid motion, 2 with slower motion) of 500 frames each of size 320×240, recorded at 30 fps. The sequences are taken indoors with a static camera on top of a screen showing different people pointing their fist towards the camera in order to control a screen pointer. Frames of the dataset are shown in Figure 7. Half of the sequences are used to learn the expected errors $E[e^k|c^k]$ for each observer O^k, the other half is used for performance evaluation.

3.8 Individual Observers

The precision and robustness measurements on the unseen test data are shown in Figure 8. A number of observations can be made. First, single template matching has high precision, with NCC being the most precise, and SAD the third most precise. Observers that include color also score highly, including the color probability (C),

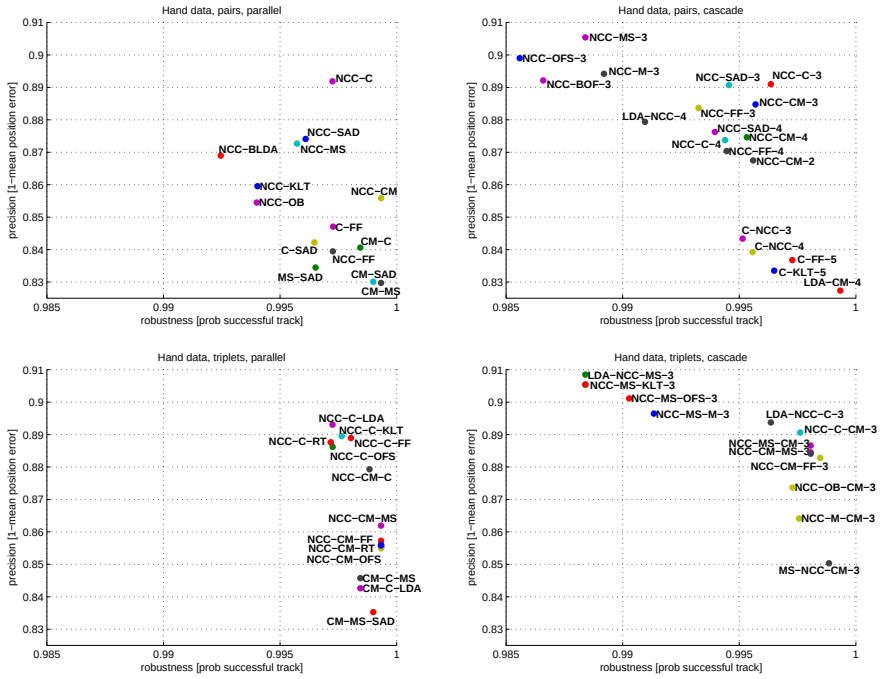


Fig. 9 Evaluation of observer combinations. *These plots show the precision and robustness measured on the test data. (top) pairs, (bottom) triplets, (left) parallel evaluation, (right) cascaded evaluation. Only a small subset of data points near the right upper corner with both high robustness and precision are shown in these graphs.*

color-motion probability (CM), mean shift (MS) and flocks of features (FF) observers. Among the online classifiers, the online boosting (OB) observer shows the highest precision. Observers using local features generally perform slightly worse, with KLT and LDA observers ranking lower in terms of accuracy. In terms of robustness, the color-motion (CM) observer comes out on top, followed by the flocks of feature (FF) observer. Color based observers (MS, C) as well as single template observers (SAD, NCC) also perform well. Color and motion probability individually show similar robustness. The regular block-based optical flow algorithm showed to be more robust than the KLT tracker, but both had difficulties handling rapid hand motion. The LDA observer shows significantly less robustness.

The performance of the off-line trained detector is not included in the evaluation. According to our definition of robustness it does not perform well because every missed detection is counted as loss of track. The percentage of correct detections on the test set is 48.4%, but it varies significantly across the sequences. On some sequences there are fewer detections due to blur and pose changes.

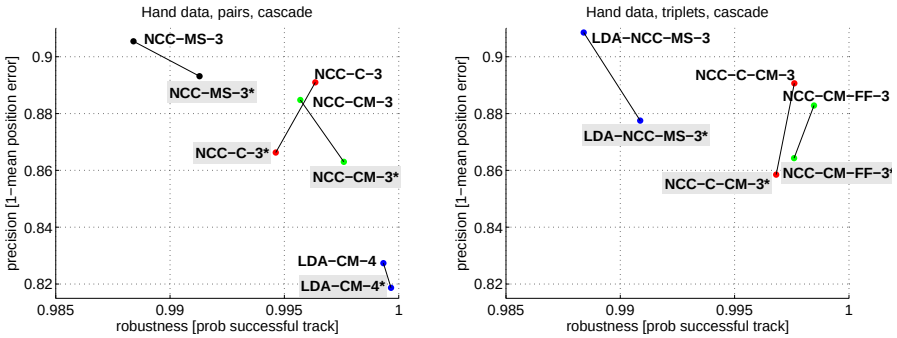


Fig. 10 Comparison with real tracking results. *These plots show the precision and robustness measured for selected combinations of observers. It compares the results by theoretical combination (as in Fig. 9) and real tracking results obtained for selected combinations of observers, shown here with gray background. The left plot show the results on pairs and the right plot on triplets. The agreement is reasonable, although there is inherently some variation between the results.*

3.9 Observer Combinations

Parallel Evaluation. We evaluate all pairs of observers using a threshold value of $\tau = 1$ on the expected error, resulting in a total of 91 combinations. Subsets of the results are shown in the top left of Figure 9. The graphs only show combinations that are near the right upper corner of high robustness and high precision. The combination of NCC with one of the color-based observers CM, C and MS shows good performance. In the videos the hand occasionally moves rapidly, resulting in significant motion blur. These cases tend to be failure modes for intensity or gradient based methods. On the other hand, the color distribution is less affected by motion blur. The robustness of these color-based observers is increased by most of the other observers that help to bridge the frames where the color cue is unreliable. The analysis also shows how observers using different cues complement each other. For example, the NCC-C combination has robustness-precision values of (0.997, 0.892), better than either NCC (0.992, 0.869) or C (0.991, 0.839) alone.

The evaluation of the 364 combinations of triplets shows a further improvement in performance, see the bottom left of Figure 9. Most noticeably, the best performance is achieved with combinations that include the NCC observer together with a color-based observer, C or CM. A local feature based observer such as LDA, KLT, or RT, can help too. Note, however, that the performance relative to the pairwise evaluation does not always change significantly. For example, by adding LDA to the NCC-C combination, the precision only increases slightly and robustness remains unchanged. Sometimes the precision can even decrease while robustness increases, such as in the case of NCC-C-FF. This means that on some occasions the additional observer helps to bridge gaps, but its estimate is otherwise not used.

Fig. 11 Hand tracking results using NCC and color-motion (CM) observers. *Shown are results of individual observers, a detector and parallel and cascaded evaluation. Colors indicate which estimate is used. In this sequence the hand is tracked successfully by both pair-wise schemes with a lower error than with either of the observers.*

Cascaded evaluation. We compared all ordered combinations of pairs at five different threshold levels (0.1, 0.2, 0.3, 0.4, 1.0) resulting in a total of 912 evaluations. Subsets of the results are shown in the top right plot in Figure 9. Most of the results with the highest precision employ NCC at the beginning of the cascade. High robustness is achieved when at least one of the observers uses the color cue, such as C or CM. The combination of NCC and CM (NCC-CM-3, i.e. a threshold value of 0.3) that was proposed in [74] performs well in terms of precision, only slightly slightly worse compared to evaluating the same observers in parallel. Some combinations show higher precision, e.g. NCC-MS-3, however, this comes at the cost of lower robustness. It is also interesting to note that the performance of LDA in combination with other observers shows significantly improved robustness, e.g. LDA-CM-4, compared to its individual result.

We also evaluated all triplets of observers at five different threshold levels, a total of 4468 combinations. Subsets of the results are shown in the bottom right plot in Figure 9. As a general observation, the results are further improved. Successful combinations frequently include different types of observers, typically a single template, a color-based observer and either motion or local features. If one component is reliable over a long time period, the overall performance changes only little.

The results also suggest that in many cases arranging the observers in the order of their individual precision leads to good performance. Combinations that include NCC as first or second component perform consistently high. One idea is therefore

Fig. 12 Hand tracking using NCC-CM-M observers in parallel. *The NCC observer (blue) is used initially. During motion blur the tracker switches to the CM observer (red). For a couple of frames the M observer (purple) is used, while the light is turned off, before switching back to CM.*

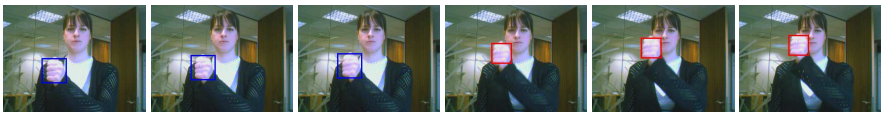


Fig. 13 Hand tracking using NCC-CM-FF observers in a cascade. *The NCC observer (blue) is used initially, switching to the CM observer (red) during motion blur.*

to estimate using the most precise observer at each time step. If the expected error falls below the threshold, the next observer acts as a fallback method. Note that in some cases the cascaded tracker may have switched to an observer that is less precise during a difficult part of the sequence. It is therefore worth checking regularly if it is possible to jump to the top of the cascade again via local detection in order to increase tracking precision.

3.10 Tracker Evaluation On Selected Combinations

Given that the above analysis of observer combinations is based on the analysis of individual observers, an obvious question is how this result varies when the full combination is tested in a tracking framework. The two set-ups are not expected to give identical results because in the combined case the observer estimates are dependent on each other. Testing all combinations of observers becomes prohibitively expensive, thus we use the results on independent observations as a method to select promising combinations to evaluate. Figure 10 shows results on pairs and triplets using cascaded evaluation. On all examples the precision in the real tracking result is slightly smaller than to the results obtained with the simplified analysis, while the robustness values are very similar.

Figures 11 shows example results of two different observers individually, their pairwise combination, as well as the detector output. It can be seen that NCC is more precise, but in the end loses track due to fast motion. CM is less precise, but tracks the complete sequence successfully. The detector only fires in one frame in this example. Both pairwise schemes work well. In some cases the parallel and cascaded evaluation select different estimates, as in the second column of the figures.

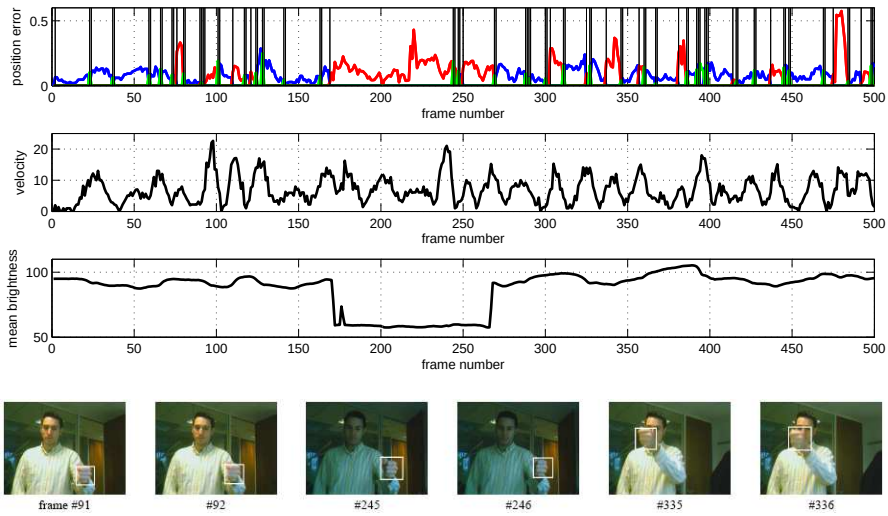
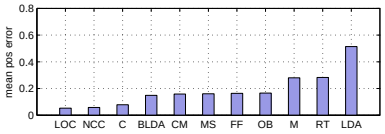


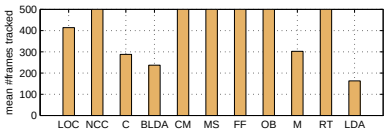
Fig. 14 Switching trackers over time. *This figure shows the tracker's switching behavior, colors in the plot indicate the component at each frame (blue=NCC, red=CM, green=detector). The hand velocity in pixels is shown in the second plot. During this sequence the light was turned off and on as can be seen in the mean brightness plot (third from top). Example frames where transitions occur are shown below (first and third pair from NCC to CM due to motion blur, middle pair from CM to NCC via local detection).*

Figures 12 and 13 show example frames from two test sequences using different observer triplets. Figure 12 shows results of the combination NCC-CM-M, evaluated in parallel. The NCC observer is used initially, but during fast motion the tracker switches to the CM observer. For a short while the motion (M) observer is used while the light is turned off. Figure 13 shows another case where the tracker switches from NCC to CM during fast motion.

The switching behavior of the NCC-CM tracker is illustrated in Figure 14, the same sequence as in Figure 12. During this sequence the light is turned off and on, as shown in the mean brightness plot in Figure 12. As the light is switched off, the template used by NCC is no longer suitable and the tracker switches to the CM observer.

When examining the performance during slow object motion, it becomes clear that in these cases the NCC tracker has a very low error, while the color based trackers can be distracted by other skin-colored objects such as the arms. See Figure 15 for a comparison on two sequences with slow hand motion. In the comparison, local orientation correlation (LOC) matching [27] is included, which shows the same precision as NCC, but slightly lower robustness on this data set.





	NCC	LOC	RT	FF	MS	C	M	CM	OB	LDA	BLDA
mean pos error	0.05	0.05	0.28	0.16	0.16	0.08	0.28	0.16	0.17	0.51	0.15
mean #frames tracked	500	212	500	500	500	288	302	500	500	163	237

Fig. 15 Results on two sequences with slow target motion. During slow hand motion the NCC and LOC observers which both use single templates are the most precise, however, LOC showed lower robustness.

4 Gesture Interface System

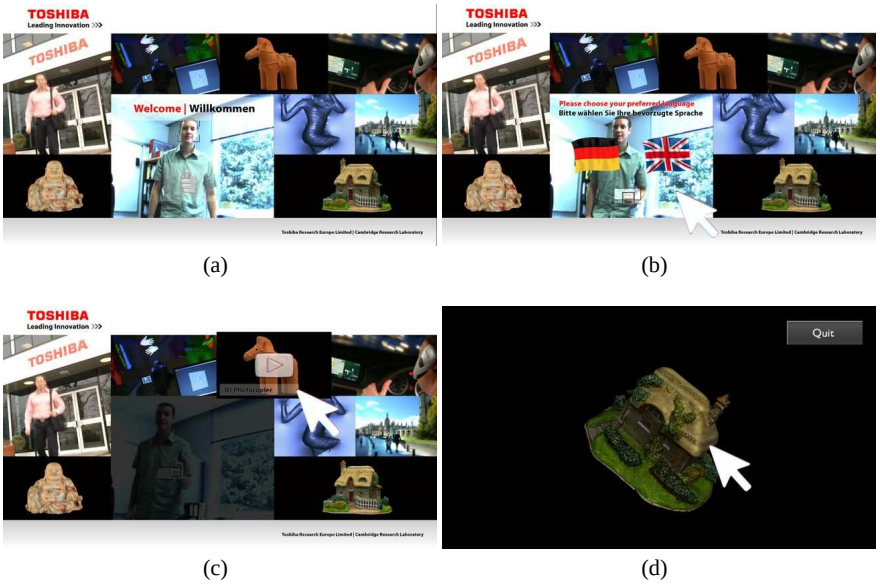
This section describes the components and the operation of the proposed gesture interface.

4.1 Visual Attention Mechanism

One goal of this work is being able to set up the system in an arbitrary environment, such as a living room or a public space, where multiple people may be within the camera's view. For some periods there may be no interaction at all, until one person initiates the interaction in order to achieve a specific task. Initially, our system finds faces using a boosted detector [54]. Once a face is detected, the user is prompted to hold up their fist within a rectangular input area below their face, see Figure 16(a). This also works for multiple users in the scene as the input areas are ordered according to scale of the face detections, giving priority to users who are closer to the camera. When a fist is detected, an interaction area is defined, within which the fist is tracked. The area is placed at the detected fist location and is scaled proportional to the detected size, meaning that the range of hand motion is largely independent of the distance to the camera. In tracking mode the user is then able to browse content shown on the display.

4.2 Tracking Mechanism

We take the results from the experiments in Section 3, which showed that an NCC-CM cascade gives good performance and use this as our fist tracker. The interface consists of a grid of windows, one of which shows the camera output. The 'active' region of fist tracking is shown as an overlay on the camera output window, see Figure 16(b). This window has the same aspect ratio as the overall screen, so the



(a) (b)

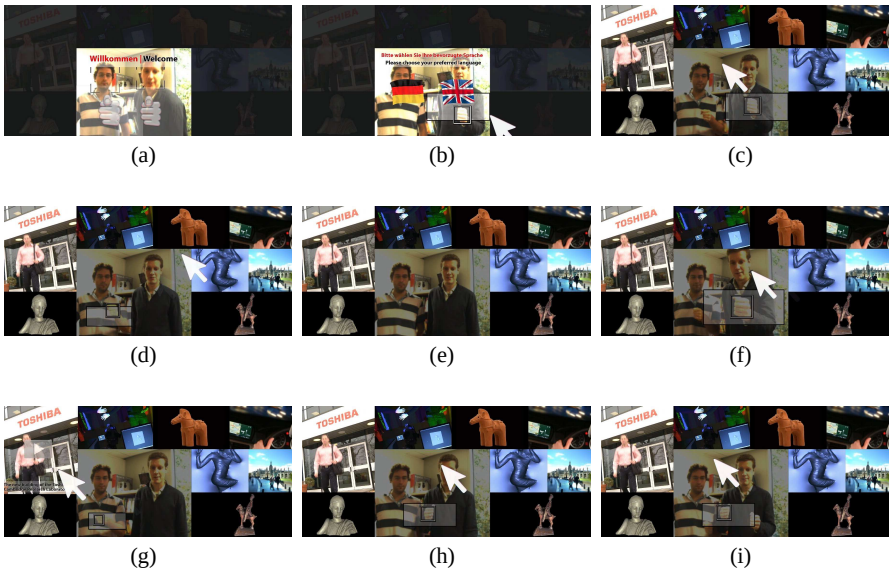
(c) (d)

Fig. 16 Gesture interface showcase. (a) *During the attention phase, a face detector is run continuously and, as soon as a person looks towards the screen, a welcome message is displayed.* **(b)** *If the person holds up their fist within the shown rectangle placed below the detected face, tracking begins and the user can move a screen pointer. Shown here is the language selection screen.* **(c)** *Video content can be selected by hovering over the buttons that appear when the arrow is over the corresponding thumbnail image.* **(d)** *In the 3D model viewer application, the user can rotate a 3D model with their hand motion.*

tracking result can be scaled up to give a cursor position on the screen, indicated with an arrow icon. The user can move their fist out of the interaction area at any time. In this case the tracker will fall back to global detection mode and re-calibrate the active tracking region to around the location of a newly detected fist.

Figure 17 shows the system with two users in the field of view. The system only allows interaction of one user at a time. Priority is given using a simple first-come-first-serve policy. The figure shows one user taking control first and as he drops his hand, the other user's fist is detected.

We use a Point Grey Flea2 camera, connected via a IEEE 1394b cable, to capture images of resolution 320×240 pixels at 30 fps. The system has been implemented and tested on different platforms, including (i) a desktop with an eight-core Intel Xeon E5345, 2.33 GHz with 2GB RAM, (ii) a laptop with a dual-core Intel CPU T2600, 2.16 GHz with 1GB RAM. (iii) a laptop with a dual-core Intel Core 2 Duo T9800, 2.93 GHz with 3GB RAM. The system runs in real-time on these platforms and on the Core 2 Duo laptop it uses 30-50% of CPU cycles in tracking mode.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Fig. 17 Interaction example with two users. *This figure shows the system's behavior when two users are using the interface in turn.* **(a)** *Two faces are detected, both users are prompted to show a fist.* **(b)** *User on right raises hand, defining new interaction region, shown as a rectangle.* **(c)** *User on left shows fist, but user on right stays in control.* **(d)** *User on right lowers fist out of camera's view, fist of user on left is detected.* **(e)** *Both users lower their hands, no fist detection.* **(f)** *When fist is shown closer to the camera, the interaction region becomes larger.* **(g)** *User on right lowers fist, control switches.* **(h)** *User on right takes over again, this time with his other hand.* **(i)** *User on left raises fist, user on right shows both fists, control stays with current hand.*

4.3 Selection Mechanisms

In order to activate a screen icon, we need to define a selection mechanism equivalent to a mouse click. Solutions that have previously been proposed include changing hand pose, finger or thumb extension, and simply hovering over an icon for a short time period [11, 27, 37, 45, 52, 67]. We have implemented these by training separate detectors, see Figure 18, (a) an open hand detector, (b) a 'thumb up' detector, and (c) hovering over an icon for a short period of time (0.5 seconds). Additionally, we propose the following method: (d) detecting a quick left-right shake gesture. The shake gesture is detected by recording the hand motion over a sliding time window of 20 frames and classifying this vector. In experiments linear discriminant analysis (LDA) and k-nearest neighbor classifiers were tested, but more reliable results were obtained by computing the distance to the closest positive training example (among a small set of 75 examples) and thresholding this value. The activation mechanism can be set according to the user's preference, however, selection by hovering has been used as default setting during exhibitions.

(a) (b) (c) (d)

Fig. 18 Different gestures for selection. (a) *Open hand pose,* **(b)** *thumb up pose,* **(c)** *hovering for a short time period, and* **(d)** *a shake gesture.*

A video can be played by selecting the button that appears when the cursor is over the corresponding thumbnail image, see Figure 16(c). Another sample application is a 3D model viewer, shown in Figure 16(d), where the user can rotate a displayed 3D model with their hand motion. A video showing the system in operation can be viewed at `http://www.youtube.com/watch?v=RL9MpXhWCrQ&fmt=18`.

5 Summary and Conclusion

This chapter has addressed the task of selecting component observers for particular tracking scenarios. To this end, a set of 14 observers has been evaluated on test sequences. A framework was proposed that evaluates the robustness and precision of observers, allowing the user to choose a profile suitable for a given application. The measurements of individual components were used to exhaustively evaluate combinations of components. We have shown results on observer pairs and triplets only, but the analysis can be applied to larger numbers of components.

The observers that were used in this paper have been used in stand-alone trackers. Some of these trackers themselves employ online feature selection. Here, instead of switching between relatively simple features from a pool [5, 17, 29], we propose switching online between observers that may use different cues and estimation schemes. Our evaluation framework allows combining arbitrary components that output an estimate and a confidence value. Direct comparison is possible because we estimate the observers' error distribution given their confidence.

In our experiments, cascaded evaluation gives similar performance to parallel evaluation at much higher efficiency. One suggested strategy is to use the most precise tracker if possible and use more robust ones as a fallback mechanism, with an off-line trained detector for re-initialization. This architecture allows for long term operation, which is required in many applications.

The proposed gesture interface works by tracking a pointing fist with a single camera facing the user. The system includes an attention mechanism that allows one user at a time to be in control. Note that face recognition could be employed for customizing the interface, as done in a previous version of our system [74]. An active region is defined for the current user, within which the hand is tracked. We propose

a multi-cue method that switches trackers over time and is updated continually by an off-line trained detector. Current sample applications include browsing videos as well as viewing a gallery of 3D models of sculptures. The system allows the user to view the 3D model from different directions by rotating it by hand. This can also be seen as a step towards manipulation of virtual objects, which is still an active research area [7]. The system has been successfully used by over hundred people at conferences and public exhibitions.

References

1. A. A. Argyros and M. I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *Proc. 8th ECCV*, pages 368–379, May 2004.
2. A. A. Argyros and M. I. A. Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Proc. HCI workshop*, LNCS 3979, pages 40–51, Graz, Austria, May 2006.
3. V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In *Boston University Computer Science Tech. Report No. 2003-023*, Nov. 2003.
4. S. Avidan. Support vector tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 26(8):1064–1072, 2004.
5. S. Avidan. Ensemble tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 29(2):261–271, 2007.
6. V. Badrinarayanan, P. Pérez, F. Le Clerc, and L. Oisel. Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *Proc. ICCV*, Rio de Janeiro, Brazil, October 2007.
7. M. Billinghurst, H. Kato, and I. Poupyrev. The MagicBook - moving seamlessly between reality and virtuality. *IEEE Computer Graphics & Applications*, 21(3):6–8, 2001.
8. S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. http://www.ces.clemson.edu/∼stb/klt/.
9. S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *CVPR*, pages 232–237, Santa Barbara, CA, June 1998.
10. M. J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. ECCV*, pages 329–342, Cambridge, UK, 1996.
11. L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *Proc. Face and Gesture*, pages 423–428, Washington, DC, 2002.
12. P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language tv broadcasts. In *Proc. BMVC*, 2008.
13. Canesta. http://canesta.com. Accessed on 19 October 2009.
14. R. Cipolla, P. A. Hadfield, and N. J. Hollinghurst. Uncalibrated stereo vision with pointing for a man-machine interface. In *Proc. IAPR Workshop on Machine Vision Applications*, pages 163–166, Kawasaki, Japan, 1994.
15. R. Cipolla and N. J. Hollinghurst. Human-robot interface by pointing with uncalibrated stereo vision. *Image and Vision Computing*, 14(3):171–178, Apr. 1996.
16. R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, October 2005.
17. R. T. Collins, X. Zhou, and S. K. Teh. An open source tracking testbed and evaluation web site. In *Intl. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, January 2005.
18. D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–575, 2003.

19. H. M. Cooper and R. Bowden. Large lexicon detection of sign language. In *Workshop HCI at ICCV*, pages 88–97, 2007.
20. T. E. de Campos and D. W. Murray. Regression-based hand pose estimation from multiple cameras. In *Proc. CVPR*, New York, NY, June 2006.
21. M. de la Gorce, N. Paragios, and D. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Proc. CVPR*, Anchorage, AK, 2008.
22. Q. Delamarre and O. D. Faugeras. Finding pose of hand in video images: a stereo-based approach. In *Proc. 3rd Int. Conf. on Automatic Face and Gesture Recogntion*, pages 585–590, Nara, Japan, Apr. 1998.
23. A. Doucet, N. G. de Freitas, and N. J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.
24. W. Du and J. Piater. A probabilistic approach to integrating multiple cues in visual tracking. In *ECCV*, pages 225–238, Marseille, France, 10 2008.
25. A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. In *Computer Vision and Image Understanding*, volume 108, pages 52–73, 2007. Special Issue on Vision for Human-Computer Interaction.
26. EyeToy. http://www.eyetoy.com. Accessed on 19 October 2009.
27. W. T. Freeman and C. D. Weissman. Television control by hand gestures. In *Intl. Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, June 1995.
28. GestureTek. http://www.gesturetek.com/. Accessed on 19 October 2009.
29. H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. CVPR*, volume 1, pages 260–267, 2006.
30. H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. ECCV*, Marseille, France, October 2008.
31. H. P. Graf, E. Cosatto, D. Gibbon, and M. Kocheisen. Multi-modal system for locating heads and faces. In *Proc. of the Second Intl. Conference on Automatic Face and Gesture Recognition*, pages 88–93, 1996.
32. H. Guan, J. Chang, L. Chen, R. Feris, and M. Turk. Multi-view appearance-based 3d hand pose estimation. In *IEEE Workshop on Vision for Human Computer Interaction*, New York, NY, June 2006.
33. G. D. Hager and P. N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proc. of the IEEE CVPR*, pages 403–410, 1996.
34. H. Hamer, K. Schindler, E. Koller-Meier, and L. van Gool. Tracking a hand manipulating an object. In *Proc. ICCV*, Kyoto, Japan, 2009.
35. A. J. Heap and D. C. Hogg. Towards 3-D hand tracking using a deformable model. In *2nd International Face and Gesture Recognition Conference*, pages 140–145, Killington, VT, Oct. 1996.
36. D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. 4th Int. Conf. on Computer Vision*, pages 93–101, Berlin, May 1993.
37. T. Ike, N. Kishikawa, and B. Stenger. A real-time hand gesture interface implemented on a multi-core processor. In *Proc. Machine Vision Applications*, pages 9–12, Tokyo, Japan, May 2007.
38. T. Ike, N. Kishikawa, and B. Stenger. A real-time hand gesture interface implemented on a multi-core processor. In *Proc. Machine Vision Applications*, pages 9–12, Tokyo, Japan, May 2007.
39. M. Isard and A. Blake. Condensation — conditional density propagation for visual tracking. *Int. Journal of Computer Vision*, 29(1):5–28, Aug. 1998.
40. M. Isard and A. Blake. ICondensation: Unifying low-level and high-level tracking in a stochastic framework. In *Proc. 5th European Conf. on Computer Vision*, volume I, pages 893–908, Freiburg, Germany, June 1998. Springer-Verlag.
41. M. Isard and A. Blake. A mixed-state condensation tracker with automatic model-switching. In *Proc. 6th Int. Conf. on Computer Vision*, pages 107–112, Bombay, India, Jan. 1998.
42. S. Izadi, A. Agarwal, A. Criminisi, J. Winn, A. Blake, and A. Fitzgibbon. C-slate: Exploring remote collaboration on horizontal multi-touch surfaces. In *Proc. IEEE Tabletop*, 2007.

43. M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *Int. Journal of Computer Vision*, 46(1):81–96, Jan. 2002.
44. R. Kaucic, A. G. A. Perera, G. Brooksby, J. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and sensor gaps. In *Proc. CVPR*, pages 990–997, San Diego, June 2005.
45. M. Kölsch and M. Turk. Fast 2D hand tracking with flocks of features and multi-cue integration. In *Workshop on Real-Time Vision for HCI*, Washington DC, July 2004.
46. M. Kölsch and M. Turk. Robust hand detection. In *Intl. Conf. Autom. Face and Gesture Recognition*, pages 614–619, Seoul, Korea, May 2004.
47. N. Krahnstoever, E. Schapira, S. Kettebekov, and R. Sharma. Multimodal human-computer interaction for crisis management systems. In *Proc. WACV*, pages 203–207, Orlando, FL, December 2002.
48. I. Leichter, M. Lindenbaum, and E. Rivlin. A generalized framework for combining visual trackers – the black boxes approach. *IJCV*, 67(2):91–110, 2006.
49. Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans. In *CVPR*, Minneapolis, MN, June 2007.
50. R. Lockton and A. W. Fitzgibbon. Real-time gesture recognition using deterministic boosting. In *Proc. British Machine Vision Conference*, volume II, pages 817–826, Cardiff, UK, September 2002.
51. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
52. J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In *Proc. 6th European Conf. on Computer Vision*, volume 2, pages 3–19, Dublin, Ireland, June 2000.
53. Microsoft Surface. http://www.microsoft.com/surface/, Accessed on 19 October 2009.
54. T. Mita, T. Kaneko, B. Stenger, and O. Hori. Discriminative feature co-occurrence selection for object detection. *PAMI*, 30(7):1257–1269, July 2008.
55. F. Moreno-Noguer, A. Sanfeliu, and D. Samaras. Dependent multiple cue integration for robust tracking. *PAMI*, 30(4):670–685, 2008.
56. Nintendo Wii. http://www.nintendo.com/wii. Accessed on 19 October 2009.
57. Oblong Industries. http://oblong.com/. Accessed on 19 October 2009.
58. K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.
59. K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe. A boosted particle filter: Multitarget detection and tracking. In *ECCV*, volume I, pages 28–39, Prague, Czech Republic, May 2004.
60. E.-J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Intl. Conf. Autom. Face and Gesture Recognition*, pages 889–894, Seoul, Korea, May 2004.
61. S. C. W. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *TPAMI*, 27(6):873–891, 2005.
62. V. Pavlović, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Analysis and Machine Intell.*, 19(7):677–695, July 1997.
63. P. Pérez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of the IEEE*, 92(3):495–513, March 2004.
64. Playstation Eye. http://www.us.playstation.com/ps3/accessories/scph-98047. Accessed on 19 October 2009.
65. Project Natal. http://www.xbox.com/en-us/live/projectnatal/. Accessed on 19 October 2009.
66. J. M. Rehg. *Visual Analysis of High DOF Articulated Objects with Application to Hand Tracking*. PhD thesis, Carnegie Mellon University, Dept. of Electrical and Computer Engineering, 1995.
67. P. Robertson, R. Laddaga, and M. Van Kleek. Virtual mouse vision based interface. In *Intl. Conf. on Intelligent User Interfaces*, pages 177–183, Funchal, Portugal, January 2004.

68. N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-D hand posture estimation based on 2-D appearance retrieval using monocular camera. In *Proc. Int. WS RATFG-RTS*, pages 23–30, Vancouver, Canada, July 2001.
69. T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Trans. Pattern Analysis and Machine Intell.*, 20(12):1371–1375, December 1998.
70. N. Stefanov, A. Galata, and R. Hubbold. Real-time hand tracker using variable-length markov models of behaviour. *CVIU*, 108(1-2):98–115, 2007.
71. B. Stenger. Template-based hand pose recognition using multiple cues. In *Proc. ACCV*, pages 551–560, Hyderabad, India, January 2006.
72. B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Trans. Pattern Analysis and Machine Intell.*, 28(9):1372–1384,, 2006.
73. B. Stenger, T. Woodley, and R. Cipolla. Learning to track with multiple observers. In *Proc. CVPR*, Miami, FL, June 2009.
74. B. Stenger, T. Woodley, T.-K. Kim, C. Hernández, and R. Cipolla. AIDIA: adaptive interface for display interaction. In *Proc. BMVC*, Leeds, UK, September 2008.
75. C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.
76. M. Tosas. *Visual Articulated Hand Tracking for Interactive Surfaces*. PhD thesis, University of Nottingham, 2006.
77. Toshiba Qosmio Press Release. http://laptops.toshiba.com/pressrelease/423413. Accessed on 19 October 2009.
78. J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Trans. Pattern Analysis and Machine Intell.*, 23(12):1449–1453, 2001.
79. N. Ueda and K. Mase. Tracking moving contours using energy-minimizing elastic contour models. In *Proc. ECCV*, pages 453–457, 1992.
80. M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *Proc. ICCV*, Rio de Janeiro, Brazil, October 2007.
81. P. Viola and M. J. Jones. Robust real-time face detection. *Int. Journal of Computer Vision*, 57(2):137–154, May 2004.
82. R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3), 2009.
83. P. Wellner. Interacting with paper on the digitaldesk. *Comm. ACM*, 36(7):87–96, July 1993.
84. O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *PAMI*, 27:1292–1304, 2005.
85. J. Woodfill and R. D. Zabih. An algorithm for real-time tracking of non-rigid objects. In *Proc. American Association for Artificial Intelligence*, 1991.
86. Y. Wu and T. S. Huang. Vision-based gesture recognition: A review. In *Gesture-Based Communication in Human-Computer Interaction*, volume 1739 of *Lecture Notes in Artificial Intelligence*, pages 103–116, 1999.
87. Y. Wu and T. S. Huang. View-independent recognition of hand postures. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume II, pages 88–94, Hilton Head, SC, June 2000.
88. Y. Wu and T. S. Huang. Human hand modeling, analysis and animation in the context of human computer interaction. *IEEE Signal Processing Magazine, Special issue on Immersive Interactive Technology*, 18(3):51–60, May 2001.
89. Y. Wu, J. Y. Lin, and T. S. Huang. Capturing natural hand articulation. In *Proc. 8th Int. Conf. on Computer Vision*, volume II, pages 426–432, Vancouver, Canada, July 2001.
90. H. Zhou and T. S. Huang. Tracking articulated hand motion with eigen-dynamics analysis. In *Proc. 9th Int. Conf. on Computer Vision*, volume II, pages 1102–1109, Nice, France, Oct. 2003.