

# Robust Real-Time Upper Body Limb Detection and Tracking

Matheen Siddiqui, and Gérard Medioni  
Univ. of Southern Calif  
1010 Watt Way  
Los Angeles, CA  
{mmsiddiq,medioni}@usc.edu

## ABSTRACT

We describe an efficient and robust system to detect and track the limbs of a human. Of special consideration in the design of this system are real-time and robustness issues. We thus utilize a detection/tracking scheme in which we detect the face and limbs of a user and then track the forearms of the found limbs. Detection occurs by first finding the face of a user. The location and color information from the face can then be used to find limbs. As skin color is a key visual feature in this system, we continuously search for faces and use them to update skin color information. Along with edge information, this is used in the subsequent forearm tracking. Robustness is implicit in this design, as the system automatically re-detects a limb when its corresponding forearm is lost. This design is also conducive to real-time processing because while detection of the limbs can take up to seconds, tracking is on the order of milliseconds. Thus reasonable frame rates can be achieved with a short latency. Also, in this system we make use of multiple 2D limb tracking models to enhance tracking of the underlying 3D structure. This includes models for lateral forearm views (waving) as well as for pointing gestures. Experiments on test sequences demonstrate the efficacy of this approach.

## Categories and Subject Descriptors

I.4.9 [Image Processing and Computer Vision]: Applications;  
I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—  
*motion, video analysis*

## General Terms

Human Factors, Performance, Reliability

## Keywords

Body Pose Estimation, Limb Detection, Limb Tracking, Gesture and Interaction Analysis, Real-Time Vision

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VSSN'06, October 27, 2006, Santa Barbara, California, USA.  
Copyright 2006 ACM 1-59593-496-0/06/0010 ...\$5.00.



**Figure 1: The output of our tracking system. Information such as the location of the arms as well as whether they point towards or away from the camera is useful in human-machine interaction.**

## 1. INTRODUCTION

The study of automatic and natural modes of interaction between humans and machines is an important field of research. A major contributing factor to this is the prevalence of machines in everyday life. As society advances technology plays a larger supporting role and we thus increasingly find ourselves interfacing with machines and computers. This occurs at all levels ranging from video games, and cameras to more pedestrian activities such as interacting with ATMs or “check-in” terminals in airports.

In the same way computers have become part of everyday life, robots are finding their way into more applications. To become accepted, however, these “Social Robots” need to autonomously interact and communicate with ordinary people in a natural and social way [8] [2] [6]. This is especially true when the targeted users might not be technologically savvy, or even have some disabilities preventing them from using traditional control devices and modalities [12].

For this purpose, it is clearly important to not only locate potential users but also obtain information about their body posture and arm configurations as illustrated in Figure 1. Such information is not only useful for non-verbal communication but also in monitoring and activity recognition. However, a system designed to obtain such information needs to operate robustly and in real-time. This task is complicated by the high dimensionality of the upper-body, and the often unpredictable motion of the arms, so tracking can be difficult at video frame rates.

To contend with real-time issues, our approach to limb tracking employs a detection/tracking strategy where we first search for the face and limbs of users, and then track the position and orientation of the forearms of the detected limbs. While detection is exhaus-

tive and can be on the order of seconds, tracking is considerably faster than the frame rate, and on the order of milliseconds. By buffering frames during the detection process this framework can achieve real-time processing with low latency.

In this system, we make use of a 2D articulated upper body model for detection, and simple forearm models for tracking. Using simple 2D models for tracking people is useful because they have reduced dimensionality and few degeneracies with a single camera [13]. But along with this reduction in dimensionality, they also have limited expressive power. To address this, we make use of multiple 2D tracking models tuned for motions ranging from waving to pointing.

Despite recent advances, tracking people remains a difficult problem as trackers must contend with self occlusions, fast motion, and difficulties in detecting features. It is not uncommon for trackers to lose track or get stuck on limb-like image structures. Our system solves this by re-initializing the tracker when limbs are lost.

The rest of this paper is organized as follows: In section 2 we present an overview of relevant work. In section 3 we present the details of our system. In section 4 we demonstrate the effectiveness of this approach on test sequences. In section 5 we conclude and provide future directions of research.

## 2. RELATED WORK

Human body pose estimation and tracking from images and video has been studied extensively in computer vision. Of relevance to this approach are those methods that make use of articulated (or near articulated) models [11][7][13]. These methods align an articulated model of the human body using various image cues such as appearance, edges, color, and motion. A major component of such systems is devoted to the machinery used to perform this alignment. This includes statistical approaches, like [16], where a learned appearance based detector is used to aid in a sampling based tracking system. In [11], Lee uses data driven MCMC sampling to obtain likely poses. Other approaches offer a more bottom up framework [15][10], for example, by assembling low level features such as parallel pairs of edge segments.

Our strategy for limb detection is based on the work of [7]. Here human models consist of a collection of 2D part models representing the limbs, head and torso. The individual parts are organized in a tree structure, with the head at the root. Each part has an associated detector used to measure its likelihood of being at a particular location, orientation and scale within the image. Also, soft articulated constraints exist between part/child pairs in the tree structure. The human model is aligned with an image by first searching for each individual part over translations, rotations, and scales. Following this, the optimal pose is found by combining the results of each individual part detection result efficiently.

As with detection, human body tracking has been extensively studied. In this problem the task is to simply update a model's pose between successive image frames. Tracking limits the scope of the solution space, as continuity may be assumed. There are various kinds of tracking methods ranging from gradient based optimization methods [3] to sampling methods [16] and combinations of these [17] used in both single and multi-camera settings.

In this work we concentrate on tracking image regions corresponding to the hands and forearms rather than a complete articulated object. This allows for very fast processing. Our approach to tracking is more akin to the kernel based tracking methods [5]. However, we use a tracker that also accounts for the orientation of a region of interest such as in CAMShift [1]. Similar to [18], this is found via an optimization framework, however we perform this optimization directly using gradient based methods.

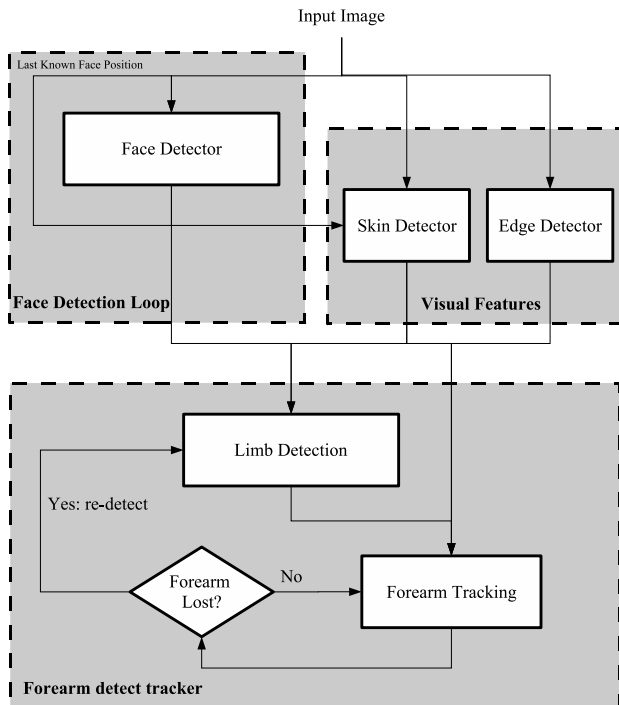


Figure 2: An overview of our approach

## 3. APPROACH

In this section we describe the details of this system. This system is designed to be robust and run in real-time. We thus make several simplifying assumptions. First we assume only one user is present. We also do not explicitly modeling clothing. Instead we assume the users wear short sleeve shirts and their forearms are exposed. This assumption simplifies the detection of good visual features, as skin regions and boundaries can be extracted with greater ease. The assumption is also fairly valid in warmer environments. We also assume that the scale of the objects does not change too dramatically. This greatly reduces the search space during detection as we only need to search over rotations and translations.

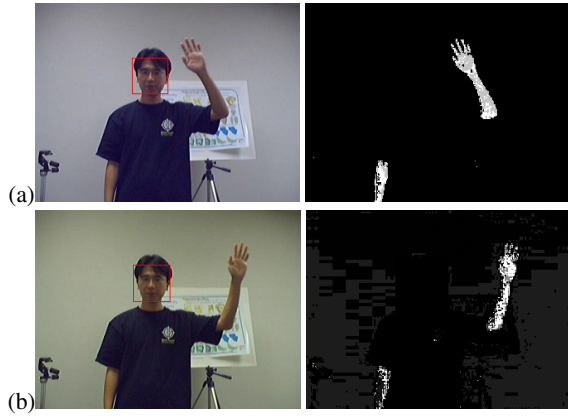
The overall approach is illustrated in Figure 2. The system contains a face detection module, and a visual feature extractor, in addition to the forearm detector/tracker. Knowing the face location constrains the possible locations of the arms (i.e. they need to be close to the head) as well as dynamically provides information about skin color, an important visual cue used for both detection and tracking. This module is described in section 3.1.

The forearm detector/tracker module contains a limb detector and forearm tracker. Limb detection is described in section 3.2. The detected forearm locations can then be used to initialize the tracking system described in section 3.3. Here, we use multiple 2D limb tracking models to enhance tracking of the underlying 3D structure. This includes models for lateral views (waving) as well as for pointing gestures as shown in Figure 5. The switch between these two tracking models is described in section 3.4.1.

### 3.1 Face and Visual Feature Extraction

This module searches for key image features used in the limb detection and tracking modules.

In this module we make use of a Harr face detector as implemented in OpenCV [1]. To increase computational speed and, to



**Figure 3:** Use of the results of the face detector in skin detection as described in section 3.1. Between (a) and (b) the illumination and white balancing of the camera changes, but skin pixels are still properly detected.

a lesser extent, robustness we embed this process in another detection/tracking scheme. Initially, we search for a face in the entire image. In subsequent frames, we only search in a neighborhood around the previous face result. If a face is not found in this limited image area, we search the entire image again. If the face is still not found, we use information from the last found face.

The color information in the image region of the detected face can then be used to initialize a hue-saturation space histogram. This histogram can then be used to assign each pixel in the entire image a likelihood of being a skin pixel. Following this we create a histogram of skin likelihoods and zero out those pixels falling in the least likely bin. This is done to eliminate pixels that constitute the least likely 40% of skin pixels. The likelihood scores of the remaining pixels are then rescaled to be between 0 and 1.

Examples of this process are shown in Figure 3. From this we see that adapting the skin color histogram to the pixels in the face region increase the robustness of skin detection to changes in illumination. In addition to skin color information we make use of a Canny edge detector for boundary information.

### 3.2 Limb Detection

Given the face, skin, and edge information we can then search for the arms. This is accomplished with the use of the upper body model used shown in Figure 4. This model encodes the upper arms, lower arms and head. Between each limb are soft constraints that bias the model to a rest state shown. These constraints allow the limbs to easily spin about their joints, while making it more difficult for the limbs to move away from the shown articulated structure.

In general each limb can be associated with a detector that grades its consistency with the underlying image. In this work, we only attach feature detectors to the lower arms as they are likely to be the most visible part of the arm during a gesture, and the head position is constrained to be at the position found by the face detector. The lower arm detector used in this work is based on both edge and skin color information. In particular, a given translation,  $\mathbf{t}$ , and orienta-



**Figure 4:** Articulated upper body model used for limb detection. The limbs are arranged in a tree structure as shown, with the head as the root. Between each parent/child pair their exists soft constraints. In our work we anchor the head at the location found from the people detection (section 3.2) module and only incorporate visual features of the forearm.

tion,  $\theta$ , within the image is graded according to the following:

$$y(\mathbf{t}, \theta) = \sum_{\mathbf{x} \in BP} D(R(\theta)\mathbf{x} + \mathbf{t}) + \lambda \sum_{\mathbf{x} \in SP} -\log P_{skin}(R(\theta)\mathbf{x} + \mathbf{t}) \quad (1)$$

where  $R(\theta)$  is a rotation matrix,  $BP$  consists of boundary points,  $SP$  consists of skin colored regions such as the hand/forearm,  $D(\mathbf{y})$  is the distance to the closest edge point in the image (computed using a distance transform [7]) and  $P_{skin}(\mathbf{y})$  is the probability of the pixel  $\mathbf{y}$  being skin color.

To align this model with an image, we seek to optimize the response of (1) subject to the soft constraints present in the model and the given location of the head. This can be formulated as an optimization problem:

$$\Theta^* = \arg \min y(\Theta) + \lambda^{(2)} c(\Theta) \quad (2)$$

where  $\Theta$  is the translation and rotation of each part shown in Figure 4,  $y(\Theta)$  represents the image matching score shown in (1),  $c(\Theta)$  quantifies the deviation from the relaxed limb configuration shown in Figure 4. This term is detailed in [7].

Observing that the constraints between each limb forms a tree structure, we can solve (2) using the method of [7]. This is accomplished by first computing (1) over translations and rotations (except over the face) for each of the forearms which are at the leaves of the tree. We can then *assemble* the configuration that minimizes (2) by considering a discretized set of possible locations of successive limbs in the tree structure. At the end of this process is an array defined over the range of poses of the root limb (i.e. the head). In each element of the array is the optimal configuration of the child limbs along with its overall score. The process is described in detail [7].

Rather than selecting the overall optimal pose in this array, we simply use the configuration attached to the head location found by the face detector. Also, instead of treating the upper body as a single entity, we treat left and right arms separately. This allows us to detect and track them separately. The underlying model disambiguates them.

### 3.3 Limb Tracking

Detection is useful when we have no prior knowledge of the limb location. In this case we need to search the entire image to find

potential limbs. After detection, however, we know the limb in the next frame must be near the limb found in the previous. Using this smoothness assumption, we can track the forearms of the user using local information. This is more computationally efficient than a full detection and is often more robust. In general, however, it is possible that one can move faster than frame rate, and thus cause the tracker to lose its object. It is thus imperative that a tracker knows when it loses track so that a re-detect can be executed.

In this approach we use a new efficient tracker. We model regions of interest as a collection of *feature sites* that indicate the presence of a skin colored pixel or a boundary pixel. This is illustrated in Figures 5.

Tracking is achieved by maximizing the consistency of the feature sites with the underlying image. This can be posed as an optimization problem over translation and orientation as expressed in the following equation:

$$\theta^*, \mathbf{t}^* = \arg \min_{\theta, \mathbf{t}} \sum_{\mathbf{x} \in BP} D_{dist}(R(\theta)\mathbf{x} + \mathbf{t}) + \lambda \sum_{\mathbf{x} \in SP} F_{skinScore}(R(\theta)\mathbf{x} + \mathbf{t}) \quad (3)$$

where  $R(\theta)$  is a rotation matrix,  $BP$  consists of boundary points,  $RP$  consists points in what should correspond to skin, and  $D_{dist}()$  yields the distance to the nearest boundary point within the region of interest. This is efficiently calculated using a distance transform of the detected edge points[7]. In addition, the term  $F_{skinScore}(x, y)$  represents a function that is zero when the image has a skin colored pixel at location  $(x, y)$  and is large otherwise.

While a natural choice for  $F_{skinScore}(x, y)$  is the negative logarithm of the skin pixel probability,  $(-\log P_{skin}(y))$  we solve (3) using gradient based methods. Thus we need the  $F_{skinScore}(x, y)$  to be *smooth*. This is achieved by using its continuous distance transform [7]:

$$F_{skinScore}(\mathbf{y}) = \min_{\mathbf{x}} (\|\mathbf{x} - \mathbf{y}\| + -\alpha \log(P_{skin}(\mathbf{x}))) \quad (4)$$

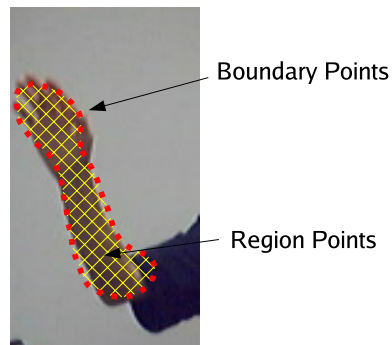
This transform tends to give smoother images that have basins around regions of high skin probability as shown in Figure 9. This serves to improve both the speed of convergence when solving (3) as well as the range of convergence.

We solve (3) directly by using a Levenberg-Marquardt optimizer [14] with a fixed number of iterations. We also only compute  $D_{dist}$  and  $F_{skinScore}$  in a fixed size region about the previous pose of the forearm. This region is large enough to accommodate movement while keeping computational costs low. Finally the face is masked out to prevent regions from being attracted to it.

### 3.4 Tracking Models

We use the tracker described in section 3.3 to track the forearm of the user. For this purpose an appropriate model is required. The advantage of using simple 2D models to track the 3D forearm is that they can be used efficiently and robustly so long as they *match* the underlying 3D structure. However, a single 2D model is not ideal for all views. For example a lateral, waving forearm in which the profile of the forearm is visible, is significantly different from tracking an arm where a user is pointing near the camera and only the hand is visible.

To effectively track the forearm as it moves in 3D we utilize multiple 2D tracking models as shown in Figure 6. The model shown in (a) is designed to track laterally viewed forearms which often occurs in waving gestures. In (c) the circular shaped model is designed to track forearms pointing towards the camera. In this case only the hand is visible and the circle effectively tracks this



**Figure 5: The tracking model represents an object as a collection of feature sites that correspond to either skin colored (at the intersection of the yellow lines) or a boundary pixel (red squares). The consistency of the model with the underlying image is measured as a weighted sum of the distance of the boundary pixels to the nearest edge pixel and the skin color scores under the skin feature sites.**



**Figure 6: Tracking models used for forearms moving in 3D. The tracking model used for laterally viewed forearms is shown in (a), pointing forearms is shown in (c). The model in (b) is for forearms viewed between (a) and (c).**

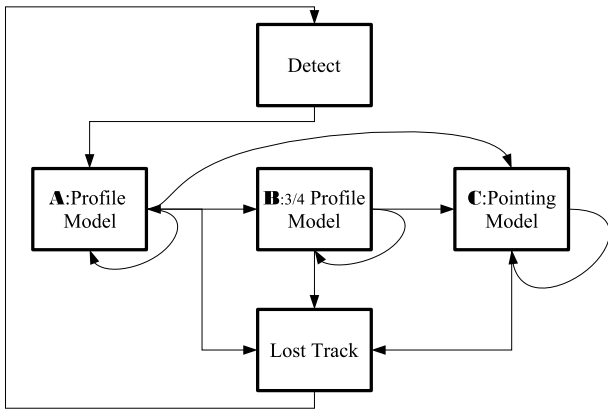
hand. The model in (b), which is just a shorter rectangle, is useful for situations between (a) and (c).

#### 3.4.1 Model Switching

After a forearm is detected using the method of section 3.2 we must track the forearm using one of the models described in section 3.4. This choice is based on intuitive understanding of how to switch between models as well as how well each model accounts for the underlying visual features. This is quantified using the percent of pixels in the model's interior that are skin colored (i.e. the *fullness* score), as well as the percent of skin pixels covered by the model in an expanded region of interest (i.e. the *coverage* score). These scores, as illustrated in Figure 8, correspond to the skin pixels accounted for by the model and those that are missed by model.

The transitions between models are summarized by Figure 7. This transition diagram was designed to help track the forearm as it switches from waving to pointing at the camera. Initially we consider the fully lateral forearm model. This model is selected because frequently a gesture is initiated by waving to a device. If the fullness score is or falls below a threshold we can switch to 3/4 profile model or the pointing model. We switch to the 3/4 profile model if its fullness score is above 90% and it accounts for 90% of the nearby skin pixels and we can switch back to the profile model if its fullness is above 90%. From either profile model, we can switch to the point model when its fullness is above 90% and only miss 10% of the skin pixels around it.

We also note that in transitioning from pointing to a profile view



**Figure 7: Summary of tracking model selection state transitions.**

a re-detect must be issued. This is because without any orientation information it is easier to just re-detect.

In addition to switching between the models we must also detect when the tracker simply loses track of the underlying forearm. This can occur, for example, if the user moves too quickly for the given frame rate. We detect this by simply keeping track of the number of underlying skin pixels in the currently used tracking model. If the total number of skin pixels falls below a threshold for any model, the system resets and re-detects using the method of 3.2.

## 4. RESULTS

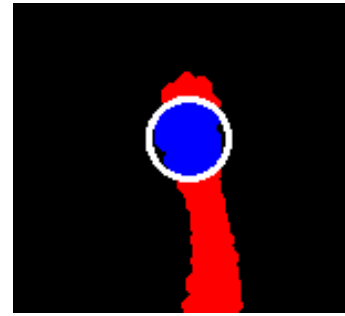
As illustrated in the following sequences acquired at 15fps, this system has been extensively tested with various users and indoor settings. Although these experiments were executed off-line, they illustrate the effectiveness of our approach. Real-time performance is discussed in section 4.1.

In Figure 10 we show an example in which the tracker loses track and must be re-initialized. Adjacent to each figure is its frame number. In Figure 10 the initial pose for each limb is correctly detected at frame 0. From this, each tracker can be initialized and successfully track the limbs until frame 6. In frames 6, 7 and 8, the subject moved significantly faster than the acquisition rate. The trackers loses track and are re-initialized with another detection. In the remaining frames (9-14) limbs are tracked correctly again.

In Figure 11(a) we show the results of the limb detection and tracking module when model switching is employed. In this 32 frame sequence the user transition from waving to pointing. In frame 0 the initial pose of each limb is correctly detected. From this, each forearm tracker can be initialized and they successfully track the limbs until frame 14. Between frames 14 and 15, the subject's right forearm transitions from waving to pointing. The corresponding tracker successfully switches to the pointing model. In frame 18 the left forearm follows suit. Over the remaining frames the user lowers his arms. The tracking model subsequently switches via re-initialization and the limbs are tracked correctly again.

In Figure 11(b) we show an additional example in which the user moves his arms up and down in a 20 frame sequence. This requires the system to switch between models and re-initialize itself. From this figure we see the system is able to keep reasonable pace.

In Figure 12 we show the system running on a PETS sequence. Note that this environment is very different than that of the other test sequences. To run the system, the scale was manually adjusted



**Figure 8: Fullness and coverage scores in the tracking system for the pointing model. The blue pixels correspond to skin colored pixels the model accounts for, while the red colored pixels correspond to skin colored pixels the model misses. The fullness score corresponds to the percent of pixels in the interior that are skin colored. This is just the ratio of the blue pixels to the total number of skin sites in the model. The coverage score correspond to the percent of skin pixels covered in the expanded region of interest. This is the ratio of the number of blue pixels to the total number of skin colored pixels ( blue + red).**

and attention was focused to the user to the far left. The system was able to detect and track both his arms.

### 4.1 Real-Time Implementation

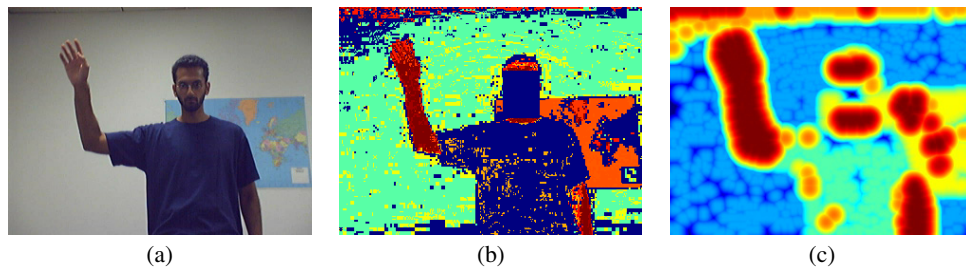
This system has been implemented on a Dual 3GHz Xeon with hyper-threading enabled. To make full use of this multi-processor platform, we use the multi-threaded programming models offered by the Software Architecture for Immersipresence (SAI)[9]. Using SAI we can create separate processing centers *known as cells* which we arrange in a chain. Each cell has its own static data contained in *Node* structures. Data can also be passed down this chain (via *pulses*) to facilitate the passing of runtime results between cells.

In our system we have separate cells for image acquisition, and face detection and feature extraction. We also have separate cells for the detection/tracking of each individual arm. While data is passed between cells serially each cell is allowed to process data as soon as it is available, thereby enabling pipelined processing. Buffering of data between cells is implicit in SAI via critical sections surrounding the static data.

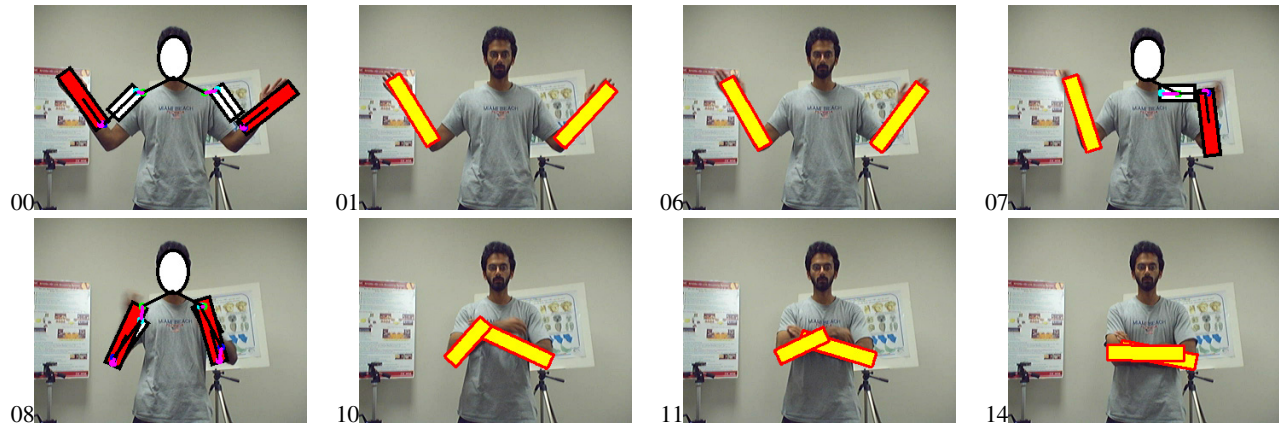
On this platform face-detection and visual feature extraction takes about 50ms per frame, while limb detection takes 400-700ms (per limb) and forearm tracking takes about 50ms per frame. Clearly detection is the bottleneck in this system. We reduce its impact by preventing the system from performing successive detection on a given limb until at least 5 frames have passed. This prevents excessive buffering of frames and keeps the latency low. When users are moving at normal speeds, detection is not issued too frequently and the dropped frames are not a noticeable issue. The system currently runs at about 10 frames per second.

## 5. CONCLUSION AND FUTURE WORK

We have described the design and implementation of a limb detection and tracking system. The system works robustly and in real-time as demonstrated by the examples. We have successfully implemented this system in real-time processor on a dual Xeon 3GHz machine with hyper-threading technology. The system works robustly and efficiently, and has been extensively tested qualitatively.



**Figure 9: The negative log of Skin color probability before (b) and after (c) applying the continuous distance transform. The figure in (c) is more suitable for the optimization of equation (3) as the skin regions have smoother boundaries. As reference, the original frame is shown in (a). In these figures a red color indicates a smaller value while a blue color indicates a larger value.**



**Figure 10: Results of the limb detector and tracker when automatic re-initialization is required. In frame 0 the initial pose of each limb is detected and successfully tracked until frame 6. Here the trackers lost tracker and the system re-initialized itself.**

So far, we have successfully been able to detect and track arms in a single sequence. Our next steps, include feeding these results to a gesture recognition module[4]. We also are interested in making use of multiple cameras and extending this formulation to 3D. Finally, more quantitative analysis of system performance will be performed using annotated test sequences.

## Acknowledgements

This work was conducted in the context of a collaborative project with the Electronics and Telecommunications Research Institute (ETRI), a Korean non-profit, government-funded research organization.

## 6. REFERENCES

- [1] G. R. Bradski. The OpenCV Library. *Dr. Dobb's Software Tools for the Professional Programmer*, November 2000.
- [2] C. Breazeal. Socially intelligent robots. *ACM Interactions*, 12(2):19–22, 2005.
- [3] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *CVPR*, Washington, DC, USA, June 1998.
- [4] I. Cohen and H. Li. Inference of human postures by classification of 3d human body shape. In *IEEE Workshop on Analysis and Modeling of Faces and Gestures*, pages 74–81, Nice, France, 2003.
- [5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.
- [6] D. J. Feil-Seifer and M. J. Mataric. Socially assistive robotics. In *Proceedings of the International Conference on Rehabilitation Robotics*, Chicago, IL, July 2005.
- [7] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.
- [8] T. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems, Special issue on Socially Interactive Robots*, 42(3-4):143–166, 2003.
- [9] A. R. François. Software architecture for computer vision. In G. Medioni and S. Kang, editors, *Emerging Topics in Computer Vision*, pages 585–654. Prentice Hall, 2004.
- [10] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *International Journal of Computer Vision*, 43(1):45–68, June 2001.
- [11] M. W. Lee and I. Cohen. Human upper body pose estimation in static images. In *ECCV*, pages II:126–138, 2004.
- [12] D. P. Miller. Assistive robotics: An overview. In *Assistive Technology and Artificial Intelligence, Applications in Robotics, User Interfaces and Natural Language Processing*, pages 126–136, London, UK, 1998. Springer-Verlag.
- [13] D. D. Morris and J. M. Rehg. Singularity analysis for articulated object tracking. In *CVPR*, Santa Barbara, CA, June 1998.
- [14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.
- [15] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *CVPR*, pages II: 467–474, 2003.
- [16] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard. Tracking loose-limbed people. In *CVPR*, pages I: 421–428, 2004.
- [17] C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3d body tracking. In *CVPR*, Kauai Marriott, Hawaii, December 2001.
- [18] H. Zhang, W. Huang, Z. Huang, and L. Li. Affine object tracking with kernel-based spatial-color representation. In *CVPR*, pages I: 293–300, 2005.

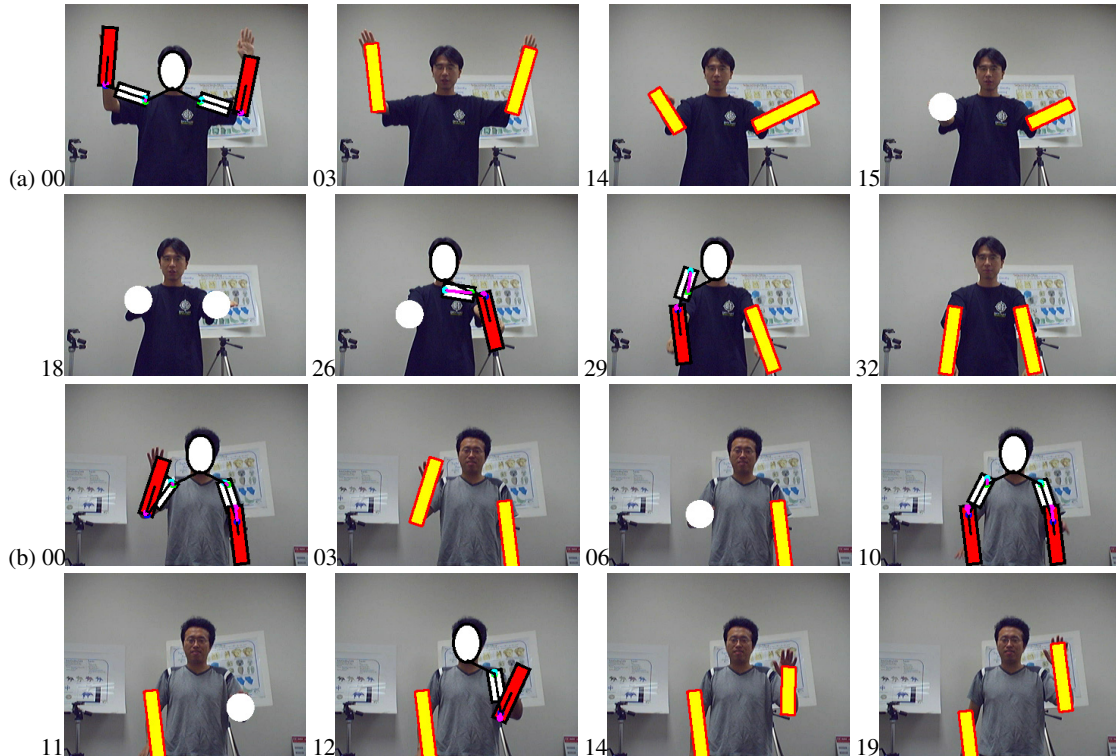


Figure 11: The results of the limb detection and tracking module when model switching is employed. In (a) the tracking models for each forearm switched from the profile (rectangular) model to that of the pointing (circular) model between frames 14 and 18. The tracking models switch back to profile model in the remaining part of this sequence. In (b) the user moves his arms up and down requires the system to switch between models and re-initialize itself.



Figure 12: The results of the limb detector and tracker on a PETS sequence. In frame 0 the initial pose is detected and successfully tracked through frame 36.