Detecting, Tracking and Interacting with People in a Public Space

Sunsern Cheamanunkul*, Evan Ettinger*, Matt Jacobsen*, Patrick Lai† and Yoav Freund*
{scheaman, eettinger, mdjacobs, yfreund}@cs.ucsd.edu, patrick.lai@stanford.edu

ABSTRACT

We have built a system that engages naive users in an audio-visual interaction with a computer in an unconstrained public space. We combine audio source localization techniques with face detection algorithms to detect and track the user throughout a large lobby. The sensors we use are an ad-hoc microphone array and a PTZ camera. To engage the user, the PTZ camera turns and points at sounds made by people passing by. From this simple pointing of a camera, the user is made aware that the system has acknowledged their presence. To further engage the user, we develop a face classification method that identifies and then greets previously seen users. The user can interact with the system through a simple hot-spot based gesture interface. To make the user interactions with the system feel natural, we utilize reconfigurable hardware, achieving a visual response time of less than 100ms. We rely heavily on machine learning methods to make our system self-calibrating and adaptive.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*artificial, augmented, and virtual realities; evaluation/methodology*

General Terms

Algorithms, Experimentation

Keywords

Machine Learning, Boosting, Real-Time Hardware

1. INTRODUCTION

Most human computer interaction is currently based on a keyboard, a pointing device (mouse, touch-pad or touch screen), and a computer screen. Multi-modal interfaces promise a future in which people will interact with computers much like people interact with each other - through speech and gestures. Working towards this goal, we follow the "smart room" approach where a set of cameras and microphones are used to capture users' speech and gestures. However, unlike most efforts to date, we installed our system in a pre-existing public space. In addition, we have built an autonomous system that engages people passing through the public space who are not immediately aware of the system's presence.

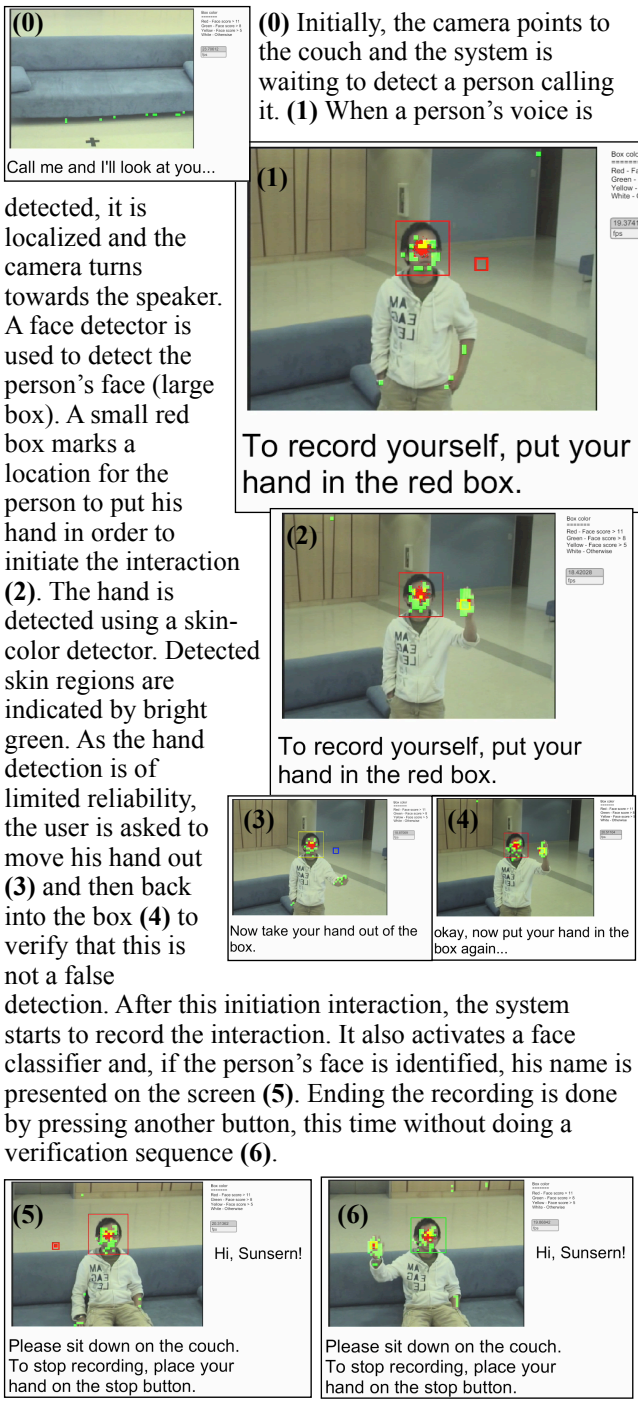
The main problem we have to contend with when placing a system in a public space is the many different interaction scenarios that can occur within the environment. First, the number and location of people are unconstrained. Second, the lighting and sound conditions are much worse than those that can be achieved in a controlled laboratory, and even worse, these conditions can vary with time. Therefore, a large amount of calibration and re-calibration is required to get to a level of acceptable accuracy and reliability. In our system we use machine learning methods to make our system adaptive to its environment and user-base.

In order to use the machine learning methods, we need to collect large amounts of training data. This brings us to one of the inherent problems in developing systems that are based on machine learning: on the one hand, one needs training data in order to train the system, but on the other hand, one needs an operational system in order to collect data. In our project, this problem manifested itself as follows: we use a PTZ camera to collect video recordings which we later use to train our computer vision algorithms. However if we recorded data continuously, then we would quickly fill our disk space with useless video material, since most of the time the public area is empty. In order to record useful footage, we need to detect people, localize them, and point the camera in their direction. Moreover, before we start making recordings of people we need to first get their permission! Recording people without their knowledge is both and invasion of privacy and a waste of valuable disk space. We thereby need a system that can localize people, engage them, and get their permission to record before we can do anything else.

Our solution to this cyclical problem is to adopt an evolutionary approach to system development. We first used a minimilistic calibration process to get the system roughly working. At first, the accuracy of the system was poor and it took significant effort to automatically record any useful data. Once we managed to record a sufficient amount of data, we retrained the system, which improved its accuracy and allowed us to collect useful data more easily. With

*Dept. of Computer Science and Engineering, UC San Diego
†Computer Science Dept., Stanford University

ICMI-MLMI'09, November 2–4, 2009, Cambridge, MA, USA.



(0) Initially, the camera points to the couch and the system is waiting to detect a person calling it. **(1)** When a person's voice is detected, it is localized and the camera turns towards the speaker. A face detector is used to detect the person's face (large box). A small red box marks a location for the person to put his hand in order to initiate the interaction **(2)**. The hand is detected using a skin-color detector. Detected skin regions are indicated by bright green. As the hand detection is of limited reliability, the user is asked to move his hand out **(3)** and then back into the box **(4)** to verify that this is not a false detection. After this initiation interaction, the system starts to record the interaction. It also activates a face classifier and, if the person's face is identified, his name is presented on the screen **(5)**. Ending the recording is done by pressing another button, this time without doing a verification sequence **(6)**.

Figure 1: An example interaction with TAC.

larger corpus of more useful video recordings, we were also able to add new features such as face recognition, making the interaction more interesting and engaging.

After about 22 weeks of data collection and retraining in this way, we now have a system which we call "the automatic cameraman" (TAC). TAC can detect and track people reliably throughout a large public space (of about 10×13 meters). Figure 1 is a summary of a typical interaction between a person and TAC. The images in the figure have been

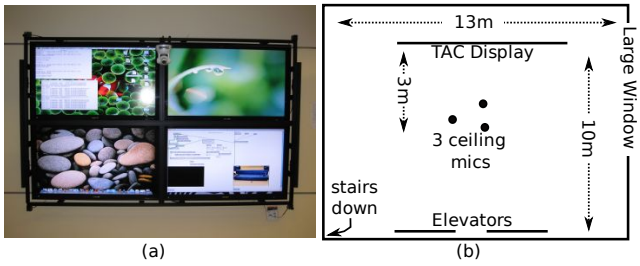


Figure 2: (a) Frontal view of TAC with PTZ-camera and four microphones visible on the corners of the display (b) The layout of the TAC lobby.

captured from the TAC display. A frontal view of TAC and the room layout can be seen in Figure 2 and on our web site [1].

This is a very simple interaction, but it has proven interesting and engaging enough that 3-8 recordings are made each day, ranging in length from 10 seconds to several minutes.

Many technical challenges had to be overcome in order to create a working system. These challenges are of two types: (1) calibration and adaptation and (2) instantaneous reaction.

Calibration is required to relate the measurements made by the microphones and the cameras to each other. Adaptation is required since the interaction space is large and has variable lighting and acoustics conditions. We develop an easy to use framework that allows us to continually improve the audio localizer, face detector, skin detector and face recognizer by adapting them to the characteristics of the camera, the lighting, and to the faces of the people that frequent the hallway. Our approach to adaptation is to adapt on the time scale of days. The system is operational for 12 hours each day from 8 am until 8 pm. During the time that it is not operational, TAC reviews the recordings collected in the previous day and uses machine learning algorithms to optimize the various signal processing and pattern recognition components.

We run the machine learning algorithms at night for two reasons. First, we do not want the system to adapt too quickly. Quick adaptation based on a single interaction is likely to cause an overall degradation in performance and make the system unstable. Second, the machine learning algorithms are computationally heavy and can take a few hours to complete. Running them when the system is interacting with a user is likely to increase the reaction time of the system. Keeping this reaction time low is an important component of any real-time HCI.

In order for the gesture based HCI to feel natural and intuitive, the computer system needs to react quickly and accurately. Studies show that hand-eye coordination is effective if the delay between an action and its observation is shorter than 100ms [17, 18]. This places a very tight constraint on the complexity of the image processing algorithms that can be used. Moreover, our measurements show that the reaction time of a general purpose workstation is at best around 300ms, which is a long enough delay for the interaction to feel unnatural. To achieve a reaction time of less than 100ms we use a hardware solution, namely a field programmable gate array (FPGA).

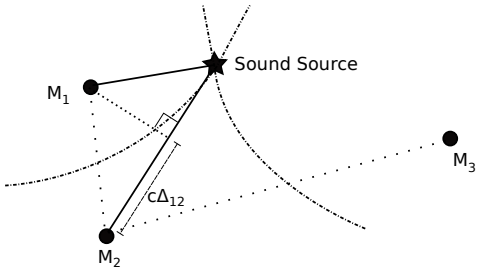


Figure 3: Audio localization in two dimensions. If Δ_{12} and Δ_{23} are known, then the intersection of the corresponding hyperbola of one sheet corresponds to the location of the sound source.

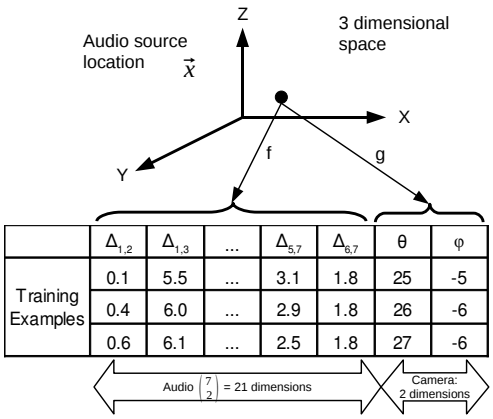


Figure 4: The sound manifold. f is a mapping from sound source location x to a set of TDOA measurements $\vec{\Delta}$. g is a mapping from x to a pan and tilt directive for the PTZ camera.

There is a very large literature on smart rooms, much too large and diverse to be described here, but popularized by many including Alex Pentland [20]. One line of work which has had a large influence on our work is that of William Freeman and collaborators [10, 11, 9]. The main idea that we took from Freeman's work is that relatively simple computer vision algorithms can be very effective for HCI applications if the feedback from the system is fast. A fast reacting interface can be effective even if it is not very accurate because it engages the hand-eye coordination capabilities of the human brain to compensate for the limitations of the computer vision algorithms. Our work also is also closely related to audio-visual fusion techniques, in particular the works concerning locating and tracking people [25, 21, 19].

The rest of the paper is organized as follows. Section 2 explains our approach to the speaker localization problem. Section 3 describes the methods we use for skin detection, face detection, face tracking and face recognition. Section 4 explains how we use FPGAs to reduce the reaction time of the system to below 100ms. Section 5 describes how the components are combined into a working system and section 6 discusses some experiments we have done to quantify the ability of the system to adapt.

2. AUDIO LOCALIZER

We use an audio-source localization method to direct the PTZ camera towards the speaker. The system is fast and reliable, pointing the camera in the direction of the sound source in 1-2 seconds with error smaller than one degree. In this section we describe our machine-learning based solution to this problem.

Audio localization using microphone arrays is a well developed technique [4]. The technique is based on the fact that sound produced at a given location will arrive at different times at spatially separated microphones. This difference is known as the time delay of arrival (TDOA) for a given pair of microphones and is depicted in Figure 3. Knowing the TDOA for a pair of microphones and the spatial locations of the microphones restricts the location of the sound source to a hyperboloid of one sheet in space. Knowing the relative positions and TDOAs for *four* microphones will uniquely identify the spatial location of the source.

The *accuracy* of the localization is restricted by the accuracy of the TDOA estimation and by the sensitivity of the TDOA to the location of the source. This last sensitivity is high if the distance between the microphones is of the same order of magnitude as the distance between the microphones and the sound source. In other words, if the microphones are close to each other the sensitivity to the location of the source is low. In order to have good sensitivity in a large room, we need to place the microphones far from each other. We also need a large number of microphones so that a sound source at any location in the room can be captured clearly on at least four microphones. Sensitivity and accuracy are also highly dependent on the acoustics of the room and on the particular location of the sound source and the microphones. Finding good locations for the microphones in a particular room is a lengthy process of trial and error.

Most audio source localization systems assume that the relative locations of the microphones are known in advance. This is achieved by mounting the microphones on a fixed frame at carefully selected locations. These frames are usually one dimensional, which is why these systems are usually called microphone arrays, but can sometimes be two or even three dimensional. The size of these microphone arrays are restricted by the need to transport them and are rarely more than 2×2 feet. This limits the localization accuracy to only those locations about 2-4 feet from the array. Moreover, even if we could use such a prefabricated microphone array, we would still have the problem of translating the location estimate into a pan-tilt directive for the PTZ camera, since it is not on the same physical frame as the array.

In order to overcome the limitations of pre-fabricated microphone arrays we use an "ad-hoc" array which consists of seven independently placed microphones. As a result we cannot pre-compute the mapping from TDOA measurements to pan-tilt commands, instead, we use a machine learning approach to learn this mapping. Our machine learning approach approach uses recently published methods for learning the structure of smooth low dimensional manifolds embedded in a high dimensional space.

Figure 4 describes how the audio source localization problem gives rise to a low dimensional manifold. Denote by $\vec{x} \in R^3$ the location of the sound source in space. As we are using 7 microphones there are $\binom{7}{2} = 21$ microphone pairs, resulting in 21 TDOA measurements for a particular loca-

tion.[1] We denote these 21 measurements by $\vec{\Delta} \in R^{21}$. We denote the pan-tilt angles of the camera by $(\theta, \phi) \in R^2$. The location $\vec{x}$ determines $\vec{\Delta}$ and (θ, ϕ). In other words, the locations of the microphones and of the camera define two mappings $f : \vec{x} \rightarrow \vec{\Delta}$ and $g : \vec{x} \rightarrow (\theta, \phi)$. The mappings f and g are not linear, but they are differentiable, meaning linear mappings give good approximations for small spatial regions.

Our localization method is based upon learning a regressor $L : \vec{\Delta} \rightarrow (\theta, \phi)$. It has previously been shown that restricting L to be globally linear or quadratic can lead to reasonable accuracies for the camera pointing problem [8]. We extend this work, by using a space partitioning tree wherein linear mappings in the partition cells are used. In recent work, random projection trees (RPTrees) have been shown to be adaptive to data that lies on a low-dimensional manifold [7, 12]. This is particularly applicable here since even though the ambient dimensionality of $\vec{\Delta}$ is 21, the intrinsic dimensionality is only 3. An RPTree is a space partitioning binary tree built very similar to that of the familiar k-d tree. Instead of making axis parallel splits like that of the k-d tree, RPTrees first project the data that fall in a node onto a random data direction and then split at the median of these projection values. See [7] for a review of the theory surrounding RPTrees and for a discussion of their utility in the regression setting see [16].

With inspiration from the RPTree work, we examine a variant that we call principal-direction trees (PDTree). Instead of using a random direction to projecting the data onto, we use the principal direction of the data as defined by a principal component analysis. An example of a PDTree on a toy dataset is shown in Figure 5. To learn L we grow PDTrees of fixed depth and fit least-squares linear regressors in the leaf nodes.

We can easily acquire a training set to learn L with help from the face detector. Training examples can be collected whenever a user speaks while their face is centered in the field of view, creating a stable measurement of the form $(\vec{\Delta}, \theta, \phi)$. Many such examples can be collected over time by having the PTZ-camera continually centering the user's face and the user continuing to speak. After collecting several new training examples, we then retrain a PDTree for L at the end of each week.

3. VIDEO ANALYSIS

TAC has four image analysis components: a skin detector, a face detector, a face tracker and a face recognizer. All four components are based on a combination of visual features and a machine learning algorithm. We describe each component in turn.

3.1 Skin-color detector

In TAC, skin color is used as a feature for face detection and as a control mechanism for on-screen buttons. We design a simple, yet effective skin-color detector that identifies each pixel as skin or non-skin. Per our design goals, we want

[1]In principle, all 21 delays can be computed from the delays of microphones 1-6 relative to microphone 0. However, microphone 0 might receive a very weak or corrupted signal, which would result in 6 incorrect measurements. Using all 21 delays significantly improves the accuracy and reliability of the localizer.

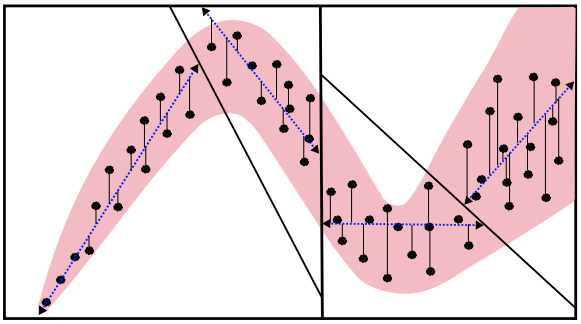


Figure 5: A toy dataset whose distribution is shown in pink. A PDTree of depth two is built with least-squares linear regressors fit in the leaves.

our skin-color detector to be efficient and adaptive to TAC's user-base and lighting conditions.

In order to minimize the response time, we avoid using any high-level contexts as detection features. Instead, the detector operates solely at the pixel level, in particular in the HSV color space.

One of the challenges of a pixel-based skin-color detector is that pixel values of the same object vary tremendously depending on the environment e.g. lighting conditions, camera model, etc. Thus, learning simple ranges on each color channel does not work well in practice. Instead, we train our detector using a discriminative approach, and by doing so, we make no explicit assumptions about the distribution of skin color in color space. In addition, since the detector is trained from actual video collected by TAC, the resulting detector is specialized to how skin appears in our given environment.

The detector is learned using an active training methodology. The detector is trained in rounds, where in each round labeled examples that are classified incorrectly by the current detector are added to the training set. A new classifier is then generated by feeding the updated training set to the learner. This process is repeated. We use AdaBoost on decision stumps as our learning algorithm [13, 14]. The features that we use are the hue, saturation and volume (HSV) representation of the pixel color. Figure 6 shows the improvement of the skin detector over multiple rounds of re-training. Four retraining iterations were needed for our skin detector to stabilize.

Since the skin detector must operate on each pixel in the image, it is important that it be as efficient as possible. To speed up the detector we use the early rejection method method proposed by Viola and Jones [24].

3.2 Face localization

When a sound is detected by the audio localizer, TAC points the camera to the sound source and then activates the face localization unit. Using the live video stream as input, the face localizer outputs the location and size of the user's face in the current frame. Once the location and size of the face is known, TAC can then issue commands to the PTZ camera in order to both keep the user's face in the center of the field of view and keep his/her face large enough for reliable face recognition.

The face localizer unit is comprised of a face detector and a face tracker. In principle, having only the face detector is

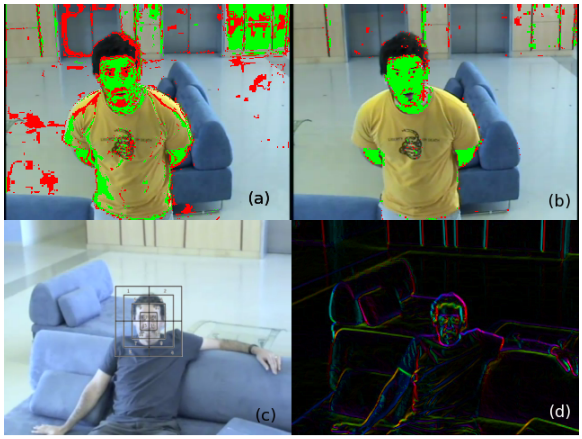


Figure 6: (a) Output from the first skin-color detector (b) Output from the skin-color detector after several iterations of re-training. Green pixels indicate skin regions with high confidence and red pixels indicate predicted skin regions, yet with lower confidence (c) The detection mask (d) The gradients used in HOG where colors indicate directions and intensity indicates magnitude.

enough for the task. However, the detector does not leverage temporal knowledge about previous detections. Employing a face tracking architecture on top of the detector exploits this temporal knowledge making a much more efficient face localization unit. In this section, we will describe first the face detector and then the face tracker.

3.2.1 *Face detector*

We started by using the implementation of the face detector of Viola and Jones in OpenCV [3]. After collecting a sufficient amount of images of faces we replaced the OpenCV detector with our own detector that is based on a somewhat different set of features. Our classification features are based on the skin scores from the skin detector and simple edge detectors. The edge features are based on the histogram of gradients (HOG) approach to edge detection [6]. Figure 6 shows the mask used to extract a feature vector from a square image patch. In each region of the mask, a histogram of skin scores and of HOGs are calculated and concatenated into a single feature vector.

We use active learning to collect a large and diverse training set with only minimal manual labeling. The active learning process is somewhat more elaborate than the one used for retraining the skin detector. We use the fact that a boosted detector outputs a real-valued score. This score can be interpreted as a measure of the confidence that the detector assigns to its prediction. We use this score in the retraining process: if the confidence of the detector is high - we add the example as a training example without further consideration, but when the confidence is low we require a human decision as to whether or not the detection box contains a face.

In the first round of training, the OpenCV face detector was used on the recordings from TAC to extract square patches of images that contain a face, giving us an initial set of positive examples. Randomly selected patches from TAC's video are added as negative examples. After training

Component	Exec. time per frame(ms)	No. of exec. per second
Face Tracker	10	25
Skin Detector	40	25
Face Detector	250	2
Face Recognition	800	1

Table 1: Response time of TAC

an initial detector, we then used it on new videos from TAC, adding new training examples to the training set according to the following three rules: (1) if the patch has high boosting score it is assumed to be a face and added as a positive example, (2) if the patch has large negative score it assumed a non-face and is added as a negative example, and (3) if it has score near zero then an operator must hand label it as face or non-face. After completing this process on a few new videos, the face detector was re-trained using the new training set. Since many examples are automatically labeled by the previous round's detector, the labeling workload is significantly reduced compared to labeling all examples. The current face detector on TAC is obtained by repeating this training process until improvement is insignificant, which was five iterations.

When the detector is deployed, one major computational bottleneck is in calculating the feature vector for each image patch. In particular. we need to compute skin scores for each pixel, HOG values, and collect their histograms within each region of the detection mask. Normally, we need to do this for every location-size pair in the image, quickly becoming very computationally expensive. If the face was detected in a previous frame we can use tracking which is described in the next section. The computational time for a full scan can be reduced greatly using the integral image technique presented in [24]. This technique leverages a nice mathematical trick to avoid the repeated computation of feature values across patches and quickly returning the histogram values needed for transforming an image patch into a feature vector for input into the boosted face detector.

Despite these optimizations, a full scan of the image can be performed on a the workstation at a speed of around 2-3fps. As the grid of locations and sizes at which the detections are calculated is fixed, the detections are of limited accuracy and jittery. A better use of resources is to use an adaptive grid where the range of locations and sizes that are measured is based on detections at previous frames. This tracking based approach is described in the next section.

3.2.2 *Face tracker*

Translating the output of the detector into predictions of the location of a face requires solving two related problems. The first problem is peak-finding, in practice, a reliable detector would not detect a face only in a single location and size but in a range of locations and sizes. It is therefore necessary to identify the location of the "best" detection. Using the location with the maximal score is reasonable, but results in a very noisy and jittery detection. It is better to smooth the scores before finding the maximum, but then the question becomes how much to smooth. The second problem is that of tracking, i.e. taking into account that the face is likely to either stay at one location or else move at slowly varying speed (recall that the location of our PTZ

Figure 7: A user moving her arm rapidly upwards towards her body. The pixels classified as skin are marked green. The two frames were displayed simultaneously and captured with a digital camera. The left frame was generated by the FPGA while the right was generated by the work station. The first advantage of the FPGA is the much higher resolution of the skin detection. The second advantage is response time. The FPGA generates the image less than 100ms after the user moved her arm and the skin color is right where it should be. The workstation generates the image about 300ms after the user moved her arm and the skin detection appears after an additional 200ms.

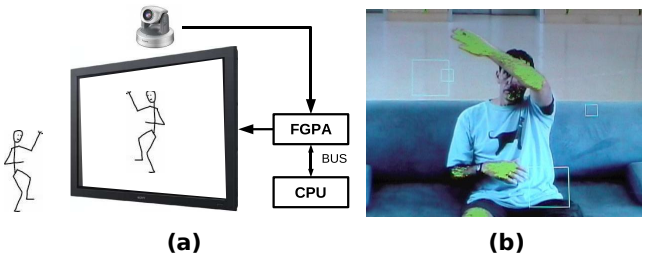


Figure 8: (a) The architecture of an FPGA based interactive system (b) A user playing with the "bubbles" (thin white squares) FPGA based interactive game.

camera is fixed and that the pan and tilt are known and can be subtracted away). The second problem is usually solved by Kalman filters or by particle filters [2].

We developed a variant of particle filters that has strong theoretical guarantees [5]. We use this new tracking algorithm to track faces in TAC. The new algorithm has a great computational advantage over face detection without tracking. Without the tracking algorithm, we need to calculate the score of 75,000 boxes in the video frame. As a result, we can perform face detection only 2-3 times per second and the resolution of our detections is not very high, resulting in jittery behavior. On the other hand, when we use our variant of the tracking algorithm, we calculate scores for only 500 boxes per frame. As a result we can track at 30 frames per second, which is the rate of the video camera, and as the locations of the detection boxes is adaptive, we get a much smoother and less jittery detection. On the other hand, the tracker can get stuck in local maxima, i.e. locations that are not faces but seem similar to faces and have the correct dynamics. We therefore perform a complete scan of the video frame every second to detect the actual location of the face and get the tracker out of local maxima.

The speed of each video component is shown in Table 1.

3.3 Face recognition

Once TAC has localized a face, the next step is to identify the person. The face recognition algorithm on TAC is based on the popular notion of eigenfaces [22, 23]. This method has proven to work well when the face is captured in a frontal view and carefully registered. On TAC, we perform face recognition when these conditions hold true, which hold quite often since we are engaging users causing them to look directly at the camera.

To detect whether the face is frontal and registered, we first convolve the region around the face detection with a face template. The template face is obtained by averaging over the frontal and registered faces that the system has seen thus far. TAC will try to recognize a detected face only when this correlation is high.

After a registered face is obtained, TAC computes its projection onto each eigenface and feeds them into a series of boosted binary classifiers. Each binary classifier gives a score corresponding to how likely the detected face comes from a particular person in the TAC database. A person in the database is recognized when the score from his/her boosted classifier are consistently high for several seconds. Since our user-base is not too large, this scheme works well to recognize regular users and further engage them in the interaction process.

TAC updates its database nightly. This includes adding newly recognized people to its database, updating the average face, updating the eigenfaces, and retraining the per-person classifiers.

4. RECONFIGURABLE HARDWARE

Studies in human-computer interaction have shown that in order for the response of the computer to appear instantaneous the response time has to be less than 100ms [17, 18]. It is surprisingly difficult to achieve this response time using standard computer architecture. Using a 210 FPS camera we measured the response time for displaying captured video using apple's quicktime software on a 2.66GHz Apple G5 workstation. The average response time between the change in the scene and the change in the image displayed on the screen is 220ms!

In order to implement the hand-in-the-box protocol depicted in figure 1 we need to add to the display a skin detector that would overlay the skin colored pixels on the image. Our skin detection and display software has a delay of 430ms between the change of the image and the change in the green overlay that identifies the skin colored regions (Figure 7.) An additional compromise we had to make is that the overlay pixels are corresponds to a square of 5x5 image pixels. The result of the 500ms delay and the low resolution of the skin detection overlay is that pressing the virtual button is awkward and error prone. To defend against errors we increased the complexity of the protocol, requiring the user to put their hand in and out of the box two times, resulting in a cumbersome interaction lasting 2-5 seconds.

We realized that in order to achieve a response time that is shorter than 100ms we need a different hardware architecture. The architecture that we came up with is described in Figure 8. The main idea is that the video stream does not go through the CPU. Instead, it passes through dedicated FPGA hardware that performs the skin detection, overlay

generation and pixel-counting. After each frame the FPGA transmits to the CPU a short message containing the number of pixels in each box and gets back a message with the location in which the boxes should be displayed in the next frame.

The video stream passes through the FPGA and back to the video display without *any* buffering. This point is worth emphasizing, we discovered that buffering a video frame in memory that is external to the FPGA chip results in prohibitive delays. Instead, we buffer only a 2 video scan lines on memory that resides inside the FPGA chip. The result is that the delay between the video input and video output is equivalent two the time of two scan lines, which is a fraction of a ms. The delay that we observe, which is around 70ms is all a result of delays in other units: the PTZ camera, the A/D converter and (probably most significantly) the LCD display.

We have yet to integrate the FPGA-based skin detector into TAC. In order to demonstrate the superior responsiveness of the FPGA skin detector we constructed a simple game called "bubbles". The game is based on the same idea as the virtual button. However, in this case there are one to six buttons on the screen at any time and the bubbles move across the screen and bounce against the screen boundary in a way familiar from simple computer games such as "pong". Pressing a button by placing your hand inside it causes the button, or "bubble" to pop and be replaced by a new button at a different place. The result is a rather entertaining game (see figure 8 and videos on [1]). The main point we wish to make with this game is that reducing response time from 500ms to 70ms results in a very significant qualitative improvement in the usability of the interface.

5. COMBINING THE COMPONENTS

In this section we discuss how the on-screen buttons operate and how we integrate the audio and face localization units to control the PTZ camera.

5.1 On-screen buttons

On-screen buttons are buttons that appear spliced into the video displayed on one of the displays similar to "hot spot" work of [15]. TAC uses these buttons as a means to interact with the user. A user can simply place their hand hovering over the virtual buttons in order to activate them. In the current implementation, the system uses one of these buttons to get authorization from the user before recording a video.

The on-screen buttons rely on the skin-color detector. The activation of each buttons is determined by the amount of skin pixels contained within the button's boundaries. When a user places his/her hand on the button, a score is computed. This score is given by the number of skin pixels in the area of the button. The activation of a button is determined by comparing it against a threshold. Using the FPGAs we are able to create much more responsive on-screen buttons with higher resolution. This opens the door to adding many more buttons to the interface and making the interaction much richer.

5.2 Controller

The controller integrates information from each of TAC's components and is solely responsible for controlling the PTZ camera. From the audio localization unit, it receives audio events in the form of a pan and tilt directive for the PTZ camera. An audio event is sent to the controller whenever an interval of sound exceeds an energy threshold, which during extended speech can be up to 40 audio events per second. Similarly the face localization unit sends face events in the form of locations and sizes to the controller whenever a strong face detection is found. When a face is present and strongly detected, the controller can receive face events at a speed near 25 events per second.

At first, the camera is in its resting state, pointing downwards, and the controller is waiting for a series of audio events to arrive to which it can direct the camera towards. Afterwards, if a face is detected, the camera disregards all subsequent audio events and tries to keep the face centered and properly zoomed within the camera's field of view. When a face event hasn't been received in a while, the audio events can again cause the camera to move.

Since neither the face nor the audio localization unit give perfect accuracy, some basic median filtering and clustering techniques are used to filter these input streams and create final directives for the camera.

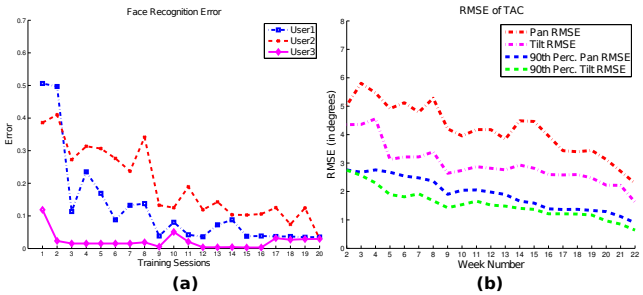


Figure 9: (a) Error rate of the face recognition unit on TAC over multiple sessions (b) RMSE for pan and tilt of a PDTree trained each week with new data acquired by TAC.

6. QUANTIFYING ADAPTIVITY

In the following sections we the results of some analysis we have done of the changes of performance of TAC as a function of time. This analysis demonstrates the efficacy of the machine learning algorithms in improving the performance of the system over time.

6.1 Audio based control of camera

Recall that we record new observations for the audio localizer's regressor as users interact with the system. We took all the observations TAC has seen up to a fixed date (∼3000 observations), and split this randomly into a 70/30 training and test set. We then examined how TAC can improve its localization accuracy by retraining a regressor for pan and tilt each week on the data from the training set seen to that point. We averaged root-mean squared error (RMSE) calculations over 20 such random training/test splits. Figure 9 shows the improvement of this regressor in terms of RMSE. Also shown is the RMSE when the top 10% of squared-residuals are removed from the RMSE calculation. The improvement is near-linear from week to week. Moreover, many of the errors are near or below one degree in both pan and tilt.

6.2 Face recognition

To show how well the face recognizer improves as users interact with the system more and more, we performed the following analysis: for three frequent users, we took 25 different video recordings of each user interacting with the system. Among the 25 recordings, we randomly selected 5 recordings for testing and used the rest for training. The training videos were ordered, and initially the training set consisted of only the registered faces from the first video. Subsequently, the training set was grown incrementally by adding new registered faces from the next video. In doing so, we simulated repeated interactions from the user with TAC over time. The results of 3 different users is shown in Figure 9. The error rate of the face recognizer goes down as the number of training sessions increases. This indicates the improvement of the face recognizer as more data is collected over time. Moreover, the final error of each classifier is less than 10% which allows for accurate recognition of these users over multiple frames of video.

7. CONCLUSION

We have described an audio-visual touchless interface called the automatic cameraman. Currently it is used to engage users in a public space by allowing them to record videos of themselves. These videos are then used to further train and improve the system on the scale of days and weeks, making TAC more responsive, accurate and personalized.

8. REFERENCES

[1] The ucsd automatic cameraman. Google search: "UCSD Automatic Cameraman".

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, Feb 2002.

[3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, October 2008.

[4] M. Brandstein and D. Ward, editors. *Microphone Arrays: Signal Processing Techniques and Applications*. Springer, 1st edition, June 2001.

[5] K. Chaudhuri, Y. Freund, and D. Hsu. Tracking using explanation-based modeling. arXiv:0903.2862v1, 2009.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[7] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 537–546, New York, NY, USA, 2008. ACM.

[8] E. Ettinger and Y. Freund. Coordinate-free calibration of an acoustically driven camera pointing system. In *ICDSC 2008: Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–9, Sept. 2008.

[9] W. Freeman, P. Beardsley, H. Kage, K. Tanaka, C. Kyuman, and C. Weissman. Computer vision for computer interaction. In *ACM SIGGRAPH*, 1999.

[10] W. Freeman and M. Roth. Orientation histograms for hand gesture recognition. In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.

[11] W. Freeman and C. D. Weissman. Television control by hand gestures. In *Intl. Workshop on Automatic Face and Gesture Recognition*, 1995.

[12] Y. Freund, S. Dasgupta, M. Kabra, and N. Verma. Learning the structure of manifolds using random projections. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

[13] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[14] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, Sept. 1999.

[15] A. Jaimes and J. Liu. Hotspot components for gesture-based interaction. In *INTERACT 2005*, pages 1062–1066, 2005.

[16] S. Kpotufe. Escaping the curse of dimensionality with a tree-based regressor. In *COLT '09: Proceedings of the 22nd annual workshop on computational learning theory*, 2009.

[17] J. D. Mackinlay, J. D. Mackinlay, G. G. Robertson, and S. K. Card. The information visualizer: A 3d user interface for information retrieval. In *Advanced Visual Interfaces, AVI*, pages 173–179, 1992.

[18] R. B. Miller. Response time in man-computer conversational transactions. In *AFIPS '68 (Fall, part I): Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277, New York, NY, USA, 1968. ACM.

[19] K. Nickel, T. Gehrig, R. Stiefelhagen, and J. McDonough. A joint particle filter for audio-visual speaker tracking. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 61–68, New York, NY, USA, 2005. ACM.

[20] A. Pentland. Looking at people: Sensing for ubiquitous and wearable computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):107–119, 2000.

[21] S. T. Shivappa, M. M. Trivedi, and B. D. Rao. Person tracking with audio-visual cues using the iterative decoding framework. In *AVSS '08: Proceedings of the 2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance*, pages 260–267, Washington, DC, USA, 2008. IEEE Computer Society.

[22] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, 1987.

[23] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[24] P. Viola and M. Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.

[25] D. N. Zotkin, R. Duraiswami, and L. S. Davis. Joint audio-visual tracking using particle filters. *EURASIP J. Appl. Signal Process.*, 2002(1):1154–1164, 2002.