

Monitoring human behavior from video taken in an office environment

Douglas Ayers, Mubarak Shah*

Computer Vision Lab, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA

Received 11 May 1999; revised 18 December 2000; accepted 20 January 2001

Abstract

In this paper, we describe a system which automatically recognizes human actions from video sequences taken of a room. These actions include entering a room, using a computer terminal, opening a cabinet, picking up a phone, etc. Our system recognizes these actions by using prior knowledge about the layout of the room. In our system, action recognition is modeled by a state machine, which consists of ‘states’ and ‘transitions’ between states. The transitions from different states can be made based on a position of a person, scene change detection, or an object being tracked. In addition to generating textual description of recognized actions, the system is able to generate a set of key frames from video sequences, which is essentially content-based video compression. The system has been tested on several video sequences and has performed well. A representative set of results is presented in this paper. The ideas presented in this system are applicable to automated security. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Video; Action recognition; Key frames; Context

1. Introduction

Human action recognition has become an important topic in computer vision. One of the most obvious applications of this technology is in security. In this paper we present a system for recognizing human actions, which is geared toward automated security applications. A human action recognition system is useful in security applications for two important reasons. The first is to detect the entrance of an unauthorized individual and monitor that individual’s actions. The second is to monitor the behavior of people who do belong in an area. By recognizing actions a person performs and using context, the behavior of the person can be determined. Some behaviors are inappropriate for certain persons. For example, someone using another person’s computer without them being in the room or taking objects they are not permitted to take. The ability to associate names with people in the scene would help achieve both of these goals.

The system described in this paper recognizes human action in an environment for which prior knowledge is available. Three low-level computer vision techniques are used in our system. These techniques are skin detection, tracking and scene change detection. All three techniques

use color images. Our system is capable of recognizing the actions of multiple people in a room simultaneously.

The system can recognize several actions: entering the scene, picking up a phone, putting down a phone, using a computer terminal, standing up, sitting down, opening something, closing something, picking up an object (specified as interesting in advance), putting down an object (previously picked up), leaving the scene and leaving the scene with an object. Several of the actions are fairly generic in nature. Objects that could be picked up by a person include briefcases, computer equipment, etc. Objects that can be opened and closed include cabinets, overhead compartments, doors, etc.

In addition to generating a textual description of recognized actions, our system reduces a video sequence into a smaller series of key frames, which concisely describe the important actions that have taken place in a room. Reduction of a video sequence to a series of key frames facilitates further analysis of the scene by computers or humans (such as deciding the name of the person who performed certain actions). Another advantage of key frames is the reduction of space required to store and time required to transmit them.

The rest of this paper is organized into six sections. Section 2 deals with related work in this area. In Section 3, we discuss the role of prior knowledge in our approach. Section 4 describes our system. In this section, we describe three low-level computer vision techniques, the high-level

* Corresponding author. Tel.: +1-407-823-2341; fax: +1-407-823-5419.
E-mail addresses: ayers@cs.ucf.edu (D. Ayers), shah@cs.ucf.edu (M. Shah).

method we use for action recognition and discuss strategies for determining key frames from video sequences. The low-level techniques are skin detection, tracking and scene change detection. All of the low-level techniques use color imagery and provide useful information for the action recognition, which is based on finite state machine model. In this section, we also discuss strategies for determining key frames from video sequences. Section 5 deals with the experimental results. We have tested our ideas with several video sequences and we provide a summary of our analysis. In Section 6, we comment on the limitations of our system. Finally, in Section 7 we provide conclusions and propose some future work in this area.

2. Related work

There is a large body of work on the analysis of human motion reported in the literature. Please see three excellent surveys: Cédras and Shah [16], Aggarwall and Cai [1] and Gavrilá [3] for a detailed treatment of this subject. For a sample of recent work, refer to special section of IEEE PAMI on Video Surveillance [2]. In the following, we briefly describe some sample work in this area, which in no way is exhaustive and complete.

Bobick and Davis [17] described a method to recognize aerobic exercise from video sequences. First they apply change detection to identify moving pixels in each image of a sequence. Then Motion History Images (MHI) and Motion Recency Images (MRI) are generated. MRI is the union of all images after change detection has been applied, which represents all the pixels, which have changed in a whole sequence. MHI is a scalar-valued image where more recent moving pixels are brighter. In their system, MHI and MRI templates are used to recognize motion actions (18 aerobic exercises). Several moments of a region in these templates are employed in the recognition process. The templates for each exercise are generated using multiple views of a person performing the exercises. However, it is shown that during recognition only one or two views are sufficient to get reasonable results.

Stauffer and Grimson [4] presented a probabilistic approach for background subtraction to be used in a visual monitoring system. Their main contention is to use a mixture of Gaussian models as compared to a widely used single Gaussian to model the color values of each pixel. They claim that each pixel is an independent statistical process, which may be a combination of several processes. For example, swaying branches of a tree result in a bimodal behavior of pixel intensity. The authors also discuss simple classifications based on aspect ratio of tracked objects. Their method involves developing a codebook of representations using an on-line vector Quantization on the entire set of tracked objects.

Shershah et al. [18] presented a method for modeling and interpretation of multi-person human behavior in real-time

to control video cameras for visually mediated interaction. They use implicit and explicit behaviors of people. Implicit behaviors are defined as body movement sequences that are performed subconsciously by the subjects. Explicit behaviors are performed consciously by the subjects and include pointing and waving gestures. Given a group behavior they introduce a high-level interpretation model to determine the areas where the cameras are focused.

Kojima et al. [14] proposed an approach to generate a natural language description of human behavior from real video images. First, a head region of a human is extracted from each frame. Then, using a model-based method, 3-D pose and position of the head are estimated. Next, the trajectory of the head is divided into segments and the most suitable verb is selected.

Davis and Shah [15] were the first ones to use a finite state machine approach to model different phases of gestures to avoid the time consuming step of warping. This paper demonstrated recognition of seven gestures which are representatives for actions of 'left', 'right', 'up', 'down', 'grab', 'rotate' and 'stop'. The system was able to recognize a sequence of multiple gestures. The vector representation for each of the seven gestures is unique. For the left gesture, the thumb and index fingers do not move, while the remaining fingers move from top to bottom. For the right gesture, all fingers move from right to left. The other five gestures are similar.

Intille and Bobick [12] and Intille et al. [13] discussed the use of context to enhance computer vision applications. In these articles, context is taken advantage of primarily to perform tracking.

The main goal of Rosin and Ellis's [10] system was to differentiate between humans and animals to reduce false alarms. This system is especially interesting because of its use of context to help improve recognition. To improve the performance of their intruder detection system, the authors include a map of the scene, which shows areas such as sky, fence and ground. This helps to differentiate between a person moving through the scene and an animal (such as a bird).

Nagai et al. [9] and Makarov et al. [11] have also written papers which involve intruder detection. Nagai et al. uses optical flow to find intruders in an outdoor environment. Makarov et al. focuses on intruder detection in an indoor scene. Comparison between edges in the current frame and edges in the background is used to perform intruder detection on image sequences with variant lighting.

Olson and Brill [7] developed an automated security system for use in an indoor or outdoor environment. Their system can detect events such as entering, exiting, loitering and depositing (an object). A map of the room which may have special regions labeled, such as tables, is used by the system to keep track of a person's movement through the room. A log of important events is kept by their system and alarms can be associated with certain events occurring at certain times in certain regions. The system also learns the

location of entrances and exits and computes the depth of points in the scene from human motion through the scene.

Most previous work in the area of automated security has not taken advantage of context to the extent our system does. Our system is novel because it makes use of context to help recognize some actions, which are difficult to model. These include picking up a phone, using a computer terminal and opening a cabinet. Also, in this system, we extract key frames to concisely describe the actions visually, which provide content-based video compression.

3. Prior knowledge

An accurate description of the layout of the scene is critical to our system. With prior knowledge about the layout of the scene, our system is able to use context to help recognize actions that persons in the scene are performing. Our system needs information about the location of entrances (which also act as exits). Locations of objects of interest and information about how these objects are used are also required by the system. For instance, the system should track a phone or briefcase yet perform scene change detection on a cabinet or mouse. In Fig. 3 for example layouts of scenes are shown. Note that they are all part of the same room and could be handled simultaneously if the field of view permitted. In these scenes, the yellow boxes show entrances and exits; the green boxes show the areas where a person's head is supposed to be when performing an action; the red boxes show the areas, which may change; the blue boxes show the area, which should not be occluded in order to recognize the action being performed; finally the pink boxes show the objects, which are to be tracked.

The layout of the scene can be easily entered by using a graphical tool for specifying the location of features. For each of the features, the type of feature is selected then a box is drawn on the corresponding area of the scene. For large features on which the system will perform change detection, several rectangles may be specified within the larger region if the user desires (for example, the first image in Fig. 3). Each region is also given a unique label so it can be referenced. Specifying the location of features in the scene should not take more than a few minutes.

Prior knowledge allows the system to use context to help make decisions about which actions are occurring in a room, while reducing computation. For example, skin detection is only performed in areas that have been declared as entrances. It is not reasonable for a person to appear in the middle of a room. Also, scene change detection is only performed in areas that have been declared as interesting and only when a person is near enough to an object to have an effect on it. Similarly, tracking is only performed on objects when a person is close enough to the object and in some cases only after scene change detection has been performed. An additional constraint, that a person be either sitting or standing, is placed on some actions. For example,

a person is required to be sitting to use a computer terminal and a person is required to be standing to open a cabinet.

Eventually, the system may be extended to learn important locations in the environment over extended periods. Instead of assuming interesting places to be known a priori, we can find objects of interest in an environment by observing long-term human motion. Frequently visited locations in the environment will correspond to interesting objects. The frequency with which people visit particular locations in a room and the time they spend at those locations should let us distinguish between corridors and, for example, computer terminals. However, in this paper, we assume all the interesting places to be known through prior knowledge.

4. The system

In this section, we describe our overall system in detail. The system is made up of three low-level components and one high-level component. The low-level components are skin detection, tracking and scene change detection. The high-level component is a state machine model for action recognition. An interesting feature of the system is that the high-level component is not dependent on which low-level algorithms are used. This allows us to experiment with new low-level algorithms in the future. It is important to be able to integrate different low-level techniques in a system for automatic action recognition on real video sequences. The output of the system is a textual description along with the key frames extracted from the sequence. Some heuristics are used to help determine, which frames best represent the actions, these heuristics are discussed in a subsection on key frames.

Before the major components of the system are discussed in detail, some information about the equipment and color space used needs to be presented. The sequences were taken with a Sony Digital Handycam (DCX-1000) video camera. The standard view of a room for our system is from about 25 to 45° below the camera's horizontal when mounted on a tripod. The video board used to capture test sequences was a Sun Video Board. This board generates images in $Y'CbCr$ color space. All sequences used to test our system were digitized at half resolution (320×200) at approximately 30 Hz.

4.1. Skin detection

Skin-like colors are not typically present in an office environment. This fact makes color-based skin detection and tracking feasible in our system. In our system, we detect a person's entrance into a room using skin detection. The algorithm used to track the person is then initialized with the location of the skin area. Prior knowledge of the scene is used to insure that the skin area detected belongs to the person's head. The head is easier to track than other parts

of the body because the head is not occluded often and it does not deform as the person moves.

The skin detection technique used by our system is a modified version of the technique described by Kjeldsen and Kender [6]. This technique was implemented in the *HSV* color space by the authors. In our system, skin detection is performed in the native $Y'C_bC_r$ color space. The advantage of this is that the conversion between $Y'C_bC_r$ color space and *HSV* color space does not have to be computed. This conversion is non-linear and computationally expensive to perform.

Kjeldsen and Kender's method uses a histogram-based technique to do skin detection. During the training phase, images with skin areas cropped are input to the system. When skin pixels are encountered in the training image, the area around the point in the histogram, corresponding to the pixel's $Y'C_bC_r$ value, is incremented by a Gaussian. When points that are not skin are encountered, the area around the point is decremented by a narrower Gaussian. After the training images have been processed, the histogram is thresholded, so that all pixels are labeled as skin or not skin. To declare a pixel as skin or non-skin during runtime, all that needs to be done is to lookup the pixel's Y' , C_b and C_r values in an array. Note that skin detection is only performed in areas that have been declared as entrances.

A connected component algorithm is performed only if there are a substantial number of pixels labeled as skin after the skin detection step. The classical connected component algorithm has been modified to return the corners of a box enclosing the skin region and the total number of pixels in the skin region. If the size of the box is above a certain threshold and the number of skin pixels belonging to the region is significant, the centroid of the box is computed and is input to the tracking algorithm for initialization.

4.2. Tracking

The tracking component of our system uses a method developed by Fieguth and Terzopoulos [5]. We use this method in our system because of its ability to track multiple objects and its relatively low computational cost. Our system uses this method to track both people and objects in the scene. This method uses a goodness of fit criteria (Eq. (1)) in color space and a simple non-linear estimator to perform tracking of an object. In Section 5 this algorithm can be seen simultaneously tracking two people. Note that a method to handle occlusion was also presented in the paper by Fieguth and Terzopoulos. No such method is currently used by our system.

Our system operates in a different color space than Fieguth and Terzopoulos's tracking algorithm as it was originally proposed. Therefore, the goodness of fit equation needed to be modified. The modified equation used for the goodness of fit in $Y'C_bC_r$ color space is given by the

following equation:

$$\psi = \frac{\max\left(\frac{C_b}{\bar{C}_b}, \frac{C_r}{\bar{C}_r}\right)}{\min\left(\frac{C_b}{\bar{C}_b}, \frac{C_r}{\bar{C}_r}\right)}. \quad (1)$$

Only the chroma components were considered since invariance to luma is a desirable quality. In the equation, (C_b, C_r) is a measurement vector (average color of object at current time) and (\bar{C}_b, \bar{C}_r) is a target vector (average color at initialization). $\psi = 1$ implies a perfect fit and ψ increases as the fit becomes poorer. The tracking box is broken up into four rectangular regions for tracking. Using multiple regions provides a better estimate of the object's color. This improves the accuracy of the tracking algorithm.

4.3. Scene change detection

The scene change detection technique used in our system is similar to the technique used in Pfinder [8]. The major difference is that it uses $C_b^*C_r^*$ color space exclusively. Pfinder uses this color space only when a pixel is not significantly brighter than expected, to avoid misclassification due to shadowing. Also, Pfinder uses a system based on a simple adaptive filter to update the statistics for a pixel. In our system, background differencing is only performed when a person is near an interesting area of the room. The following equation shows how $Y'C_bC_r$ is transformed into $C_b^*C_r^*$ color space.

$$C_b^* = \frac{C_b}{Y'}, \quad C_r^* = \frac{C_r}{Y'}. \quad (2)$$

Note that $Y'C_bC_r$ and YUV (referred to in Ref. [8]) are essentially referring to the same color space. $C_b^*C_r^*$ color space has been found to provide a stable chrominance measure despite shadowing [8]. It is especially important to deal with shadowing robustly in our system, since a person may often walk near an object casting a shadow on the object without using it.

In Pfinder, the equation for the log likelihood that a pixel belongs to the scene is computed as follows:

$$d_k = -\frac{(\mathbf{y} - \boldsymbol{\mu}_k)^T \mathbf{K}_k^{-1} (\mathbf{y} - \boldsymbol{\mu}_k) + \ln|\mathbf{K}_k| + m \ln(2\pi)}{2}, \quad (3)$$

for each pixel, k , in an interesting area, \mathbf{K} is a covariance matrix and $\boldsymbol{\mu}$ is a mean vector for C_b^* and C_r^* ; \mathbf{y} is a vector containing the current C_b^* and C_r^* values of a pixel; m is the dimensionality, in this case always 2. Sub-sampling of pixels in a region can be used to reduce computation to keep the system functioning in real-time.

Our scene change detection component takes advantage of areas that are expected to change and those that are expected not to change. An example of where this could be useful is performing scene change detection on a mouse to detect that a person is using a computer terminal. To help insure that the change in the mouse is not a result of

Table 1
List of actions and recognition conditions

Actions	Recognition conditions
Enter	Significant skin region detected for τ frames
Stand or sit	y-position of a person's tracking box increases or decreases significantly
Pick up object	The object's initial location has changed after the person and object have moved away
Put down object	Person moves away from object being tracked
Open or close	Increase or decrease in the amount of 'scene change' detected in object
Use terminal	Person sitting near terminal and scene change is detected in mouse, but not behind it
Pick up phone	Scene change in phone detected, phone has been tracked until it stopped moving in the y-direction
Put down phone	Phone has been tracked back to its original location
Leave	Person's tracking box is close to edge of the entrance area through which people enter and the person is not holding an object
Leave with object	Conditions for leave are met, the person has picked up an object, not put the object down and the object's tracking box is close to the edge of the entrance area.

occlusion, the area behind the mouse is required not to have changed in order for a person to be declared as using the terminal. It would be unlikely for a person to occlude the mouse without occluding the portion of the scene behind it.

Large objects can be broken up into several smaller sub-regions. This allows change in large objects to be detected more robustly. For example, a large change in a small region of the object can be ignored since it will not affect all the regions in the object. This helps reduce the likelihood of objects being flagged as having changed when only occluded by people moving around in the scene. Also, a time limit has been implemented for objects in order for a change to be flagged. Objects that are not significantly different from the background for τ or more frames are not flagged as having changed.

Before the system starts detecting actions, the statistical information about the color of each of the regions must be computed. This takes about a second of video. If a person is detected entering the area, the initialization stops until the person has left. The system can also update itself if no person enters the area for a period of time. This reduces the effect of lighting conditions and other factors that change over time.

4.4. Action recognition

The three components previously described, provide the information necessary for the final component of the system to make determinations about what actions have been performed by persons in the scene. Table 1 shows all of the actions that our system can recognize in the example room considered in this paper. These actions mainly involve person entering and exiting the room and manipulating objects (e.g. pick up, put down, etc.). Our approach for

action recognition makes extensive use of context. For instance, in the beginning the system looks for entry of person in designated areas. Once the person has entered, the person is tracked and depending on the person's interaction with the environment, the change detection or tracking is performed.

Fig. 1 describes graphically how action recognition is performed. There is some similarity between our action recognition method and artificial intelligence planning techniques, although our approach was not motivated by these techniques. The model presented in Fig. 1 had been designed for the example room used in this paper. It is slightly simplified in that it does not incorporate the 'pick up object', 'put down object' and 'leave with object'. A different model would likely be used for performing action recognition in a different setting. In the example room, we assume that three interesting objects are in a given field of view. These are a phone, a cabinet and a computer terminal. The arrows in Fig. 1 represent transitions that could be made based on the position of a person, scene change detection, or an object being tracked. A zero represents no output. Transitions that have a '/' depend on the state of objects in the room or whether a person is sitting or standing. The system stores this information. This makes our model slightly different from a traditional state model. For example, whether 'open cabinet' or 'close cabinet' is displayed depends on the current state of the cabinet. Also, if a transition is made from the 'near phone' state to the 'sitting' state, and the person is already sitting, no output is displayed. We assume that when a person enters, they are standing.

In the state model, shown in Fig. 1, the transition from the 'Start' state to the 'enter' state is made when a new person is detected as described in Section 4.1. If the person moves near the phone, the transition to the near phone state is made. If the person picks up the phone, the 'pickup phone' message is then displayed as the transition to the 'talking on phone' state is made. Eventually, the transition from the talking on phone state to the 'hanging up phone' state is made. Finally, the transition back to the near phone state is made. A transition will eventually be made back to the 'standing' or sitting state. The output of this transition will depend on whether the person was previously standing or sitting as discussed earlier. If the person is standing and moves near the cabinet, a transition, to the 'near cabinet' state is made. If the person opens or closes the cabinet, the 'open/close' cabinet message is displayed and the transition, is made to the 'opening/closing cabinet' state. Eventually, the person will return to the standing state. If the person is sitting near the terminal, the transition to the 'near terminal' state is made. If the person uses the terminal then the 'use terminal' message will be displayed and a transition will be made to the 'using terminal' state. Eventually the person will return to the sitting state. Finally, the person will walk out of the room and the 'leave' message will be displayed and a transition will be made to the 'end' state.

The state model is currently read from a text file so it can

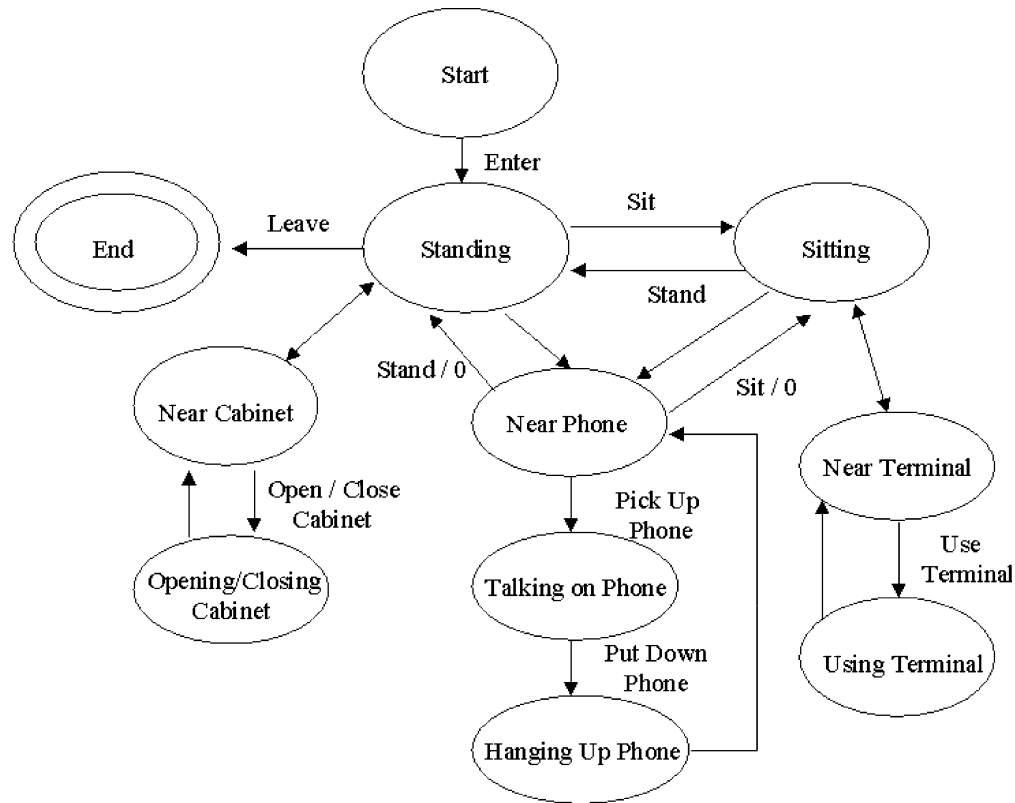


Fig. 1. State models, for example room, which contains a phone, a cabinet and a computer terminal. Words above transition arrows represent the output of the system. The system has no output for transitions, which have a zero or no text over them.

be modified easily, although a graphical tool could be developed in the future. The model is represented internally by an array of states. The first state in the array is the start state of the model. Each state has a pointer to every state it is connected with by a transition, the conditions under which text may be displayed in this state and the information necessary to determine if a transition occurs. The information necessary to determine if a transition occurs varies based on the type of transition as shown in Fig. 2. One type of transition is for a person entering one or more areas. For this type of transition, all that is required is the location of the areas. For a transition that requires some regions to change and some not to change, statistics about the regions are computed at initialization. To determine if text should be output, the conditions under which text is output for the current state are constantly checked. For example, when in the talking on phone state, the text 'put down phone' is displayed when the phone has returned to its original condition. Information about the previous state is saved for some of the conditions. To determine the current state, each transition is constantly checked and when the conditions are met, the transition to the next state is made. More types of transitions and text output conditions can be easily added and then used in the text file to create more complicated models.

In Table 1, all of the actions require the use of some

thresholds. These thresholds are specified in the same file as the state model. For actions involving scene change detection, a threshold must be provided for the probability that a region has not changed. We compute an average d_k for each region and compare it to the logarithm of the desired probability. For sitting and standing, we use a total change in a person's tracking box's current y -position from the tracking box's y -position when the person's entrance was first detected. For actions, which involve being near an object or moving away from an object, we consider a person to be near when the tracking box is within the large square around the object (see Fig. 3). A threshold for the distance from the edge of the entrance box (see Fig. 3) must be defined for the leave action. Also, in the case of putting down the phone, some small difference from exact initial position is allowed. An additional requirement for the 'open', 'close', pick up object and 'pick up phone' actions, not described in the table for simplicity, is that a person be near the object. It is not difficult to determine good values for these thresholds. Once these thresholds have been set, they do not need to be changed for different lighting conditions or camera positions. The system has been shown to be unaffected by small changes in these thresholds.

In the model presented in Fig. 1, there is the possibility of getting 'stuck' in a state if an error occurs. If a tracking error occurs while in the talking on phone state, the system should

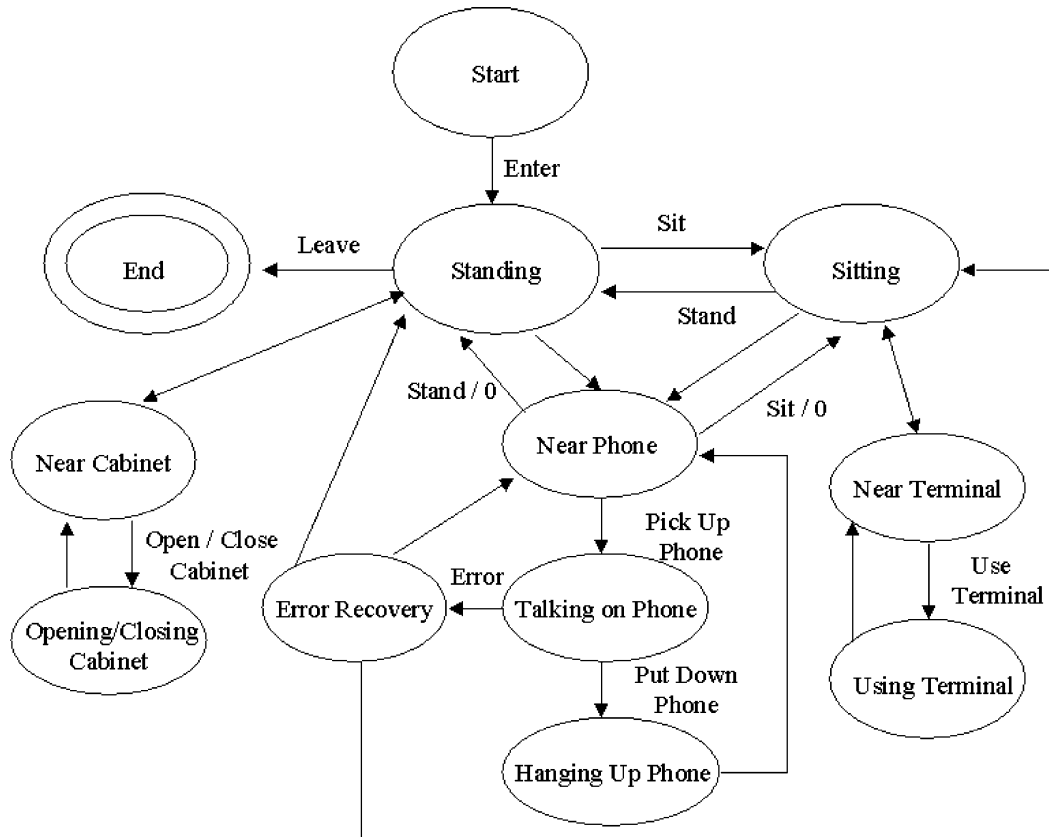


Fig. 2. State models, for example room, which contains a phone, a cabinet and a computer terminal. Words above transition arrows represent the output of the system. The system has no output for transitions, which have a zero or no text over them.

not remain stuck in an invalid state. The system should be able to continue to detect other actions the person performs. The state labeled ‘error recovery’ in Fig. 2 is entered when an error is detected while in the talking on phone state. A transition is then made to the sitting, standing, or near phone state depending on the type of error that occurred and whether the person was last in the standing or sitting state. If the box tracking the phone moves away from the person, a transition is made to the near phone state. If the person moves away from the telephone, a transition is made to the sitting or standing state. Notice that talking on phone is the only state in Fig. 2 for which the system needs an error recovery state. This is because this state depends on tracking the phone in order to transition to another state.

Unfortunately, our action recognition scheme will be susceptible to two types of errors. The first type of error occurs when one of the low-level computer vision techniques fails. For example, when the tracking algorithm loses an object. The second type of error occurs because not enough information is available. For example, when a person puts down an object while completely blocking this action from the camera’s view. Even if the low-level algorithms provided truth data, the second kind of error would still exist in the system. Whenever possible, we

would like our system to be able to detect these errors, note that an error has occurred, and continue to process the video correctly. Without modifying our action recognition approach, we can add states to our model, which will catch some of the errors.

Action recognition is a high-level process, which needs to interpret low-level information regarding tracking, scene change detection, skin detection, etc. Since our aim is to monitor human behavior, which is a qualitative process, we are particularly concerned with quantitative information. Therefore, low-level information does not need to be very precise in order to perform action recognition. For instance, there is no need to track a person’s face precisely. As far as we are able to determine the person has entered the room and is moving towards some interesting area, it is sufficient for action recognition. Similarly, we do not need to know the location of scene change detection with sub-pixel accuracy. If we can qualitatively determine a gross change in certain area that is sufficient for our high-level process.

As stated earlier, the finite state model is currently entered by a user. It will be interesting to explore ways to determine the finite state model automatically by watching the environment for an extended period. The system will start with no model and incrementally build a model by



Fig. 3. Five example scene configurations. The first scene on the left is set up to show a person sitting down at a computer or opening a cabinet. The second scene is set up to show a person picking up and putting down a phone. The third scene is set up to show a person picking up a file box. The fourth scene is set up to show a person sitting down to use a computer terminal. The final scene is set up to show a person picking up a thermos.

observing the scene. We believe this may be one of the possible ways children learn to recognize human actions.

4.5. Determining key frames

In addition to recognizing actions and generating textual descriptions the system is able to generate key frames from a video sequence. Key frames are very important because they take less space to store, less time to transmit and can be viewed more quickly. The generation of key frames from a video sequence is essentially a content-based video compression, which can provide tremendous amount of compression. In particular, in the context of video surveillance application instead of saving all the frames, only key frames can be saved, which can provide relevant information about important events in the video. Similarly, the human observer can review only key frames, instead of all the frames, which can be accomplished more quickly. In the literature there is large amount of work on camera shot detection. A shot is defined as a continuous sequence captured by a single camera [19]. A video can be represented by a sequence of camera shots. One popular method for shot detection is histogram intersection. It is common to represent each camera shot by an arbitrary selected key frame, for example the first or last frame of a shot.

Another widely used method for detecting key frames in a video is to identify a frame, which significantly differs from the previous frame using image differencing. The methods based on simple image difference are sensitive to camera motion and lighting conditions. Our method for detecting key frames is based on understanding of events in the video.

In our system, key frames are not always output at the time when an event is detected. Some key frames are output from a time prior to when the event was detected. Other key frames are output at a time after the event was detected. Our system uses some simple heuristics to output key frames, which most accurately describe the actions that take place in the room. ‘opening’ and ‘closing’ key frames are taken when the d_k values for the object stabilize for τ frames. Standing and sitting key frames are taken when a person’s tracking box stops moving up or down respectively. Enter key frames are taken as the person leaves the entrance area. Leave key frames are taken as the person enters the exit area. Note that at the time when the person enters the exit area the system is not sure if they are going to leave. The frame at that time is saved and is output as a key frame if the person actually leaves the room.



Fig. 4. One person opens a cabinet, while another sits down (350 frames, 8 key frames).

5. Results

Figs. 4–10 show the results.¹ We have performed many tests, however the sequences shown here are representative of the system's capabilities. All of the actions that were mentioned in Section 1 are shown in these sequences. Also, one of the sequences has two people acting in the same scene.

In the first sequence, shown in Fig. 4, person one enters the room, sits down on a chair, stands up and leaves the room. While person two enters the room, opens the file cabinet, closes the file cabinet and leaves. In the second sequence, shown in Fig. 5, a person enters the room, picks

up the phone, puts down the phone, and then leaves. In the third sequence, shown in Fig. 6, a person enters the room, picks up an object, puts down an object some other place and then leaves. In the fourth sequence, shown in Fig. 7, a person enters the room, sits down on a chair, uses a terminal, stands up and leaves. Finally, in the fifth sequence, shown in Fig. 8, a person enters the room, picks up an object and leaves with the object. In all sequences, our system was able to recognize all of these actions automatically, the textual descriptions of recognized actions are shown on the top of each image in the figure.

As mentioned earlier, in addition to textual output, shown on the transitions in Fig. 1, a sequence of key frames is generated by this system. The first (Fig. 4) sequence had 350 frames, out of which 8 key frames were taken. The second and third sequences (Figs. 5 and 6) consisted of

¹ MPEG movies showing the results are located at <http://www.cs.ucf.edu/~ayers/research.html>.

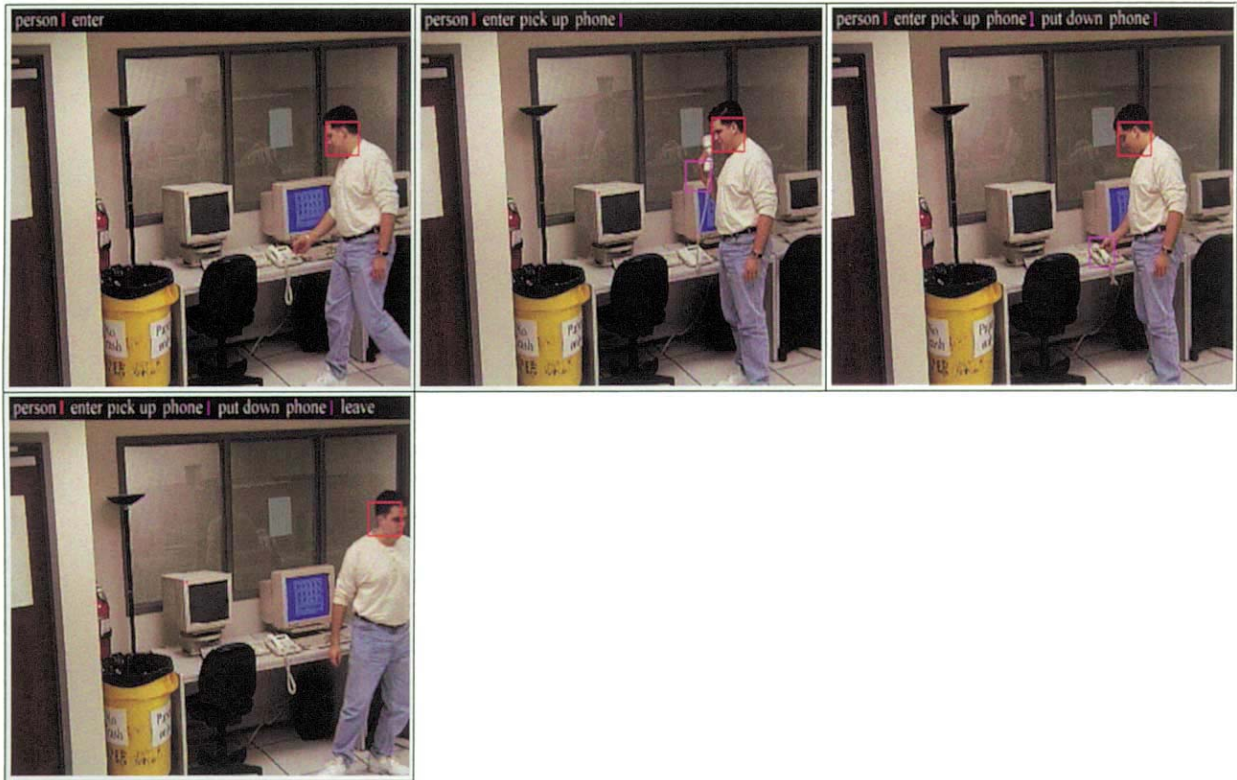


Fig. 5. A person picks up the phone, talks on the phone and then hangs up the phone (199 frames, 4 key frames).

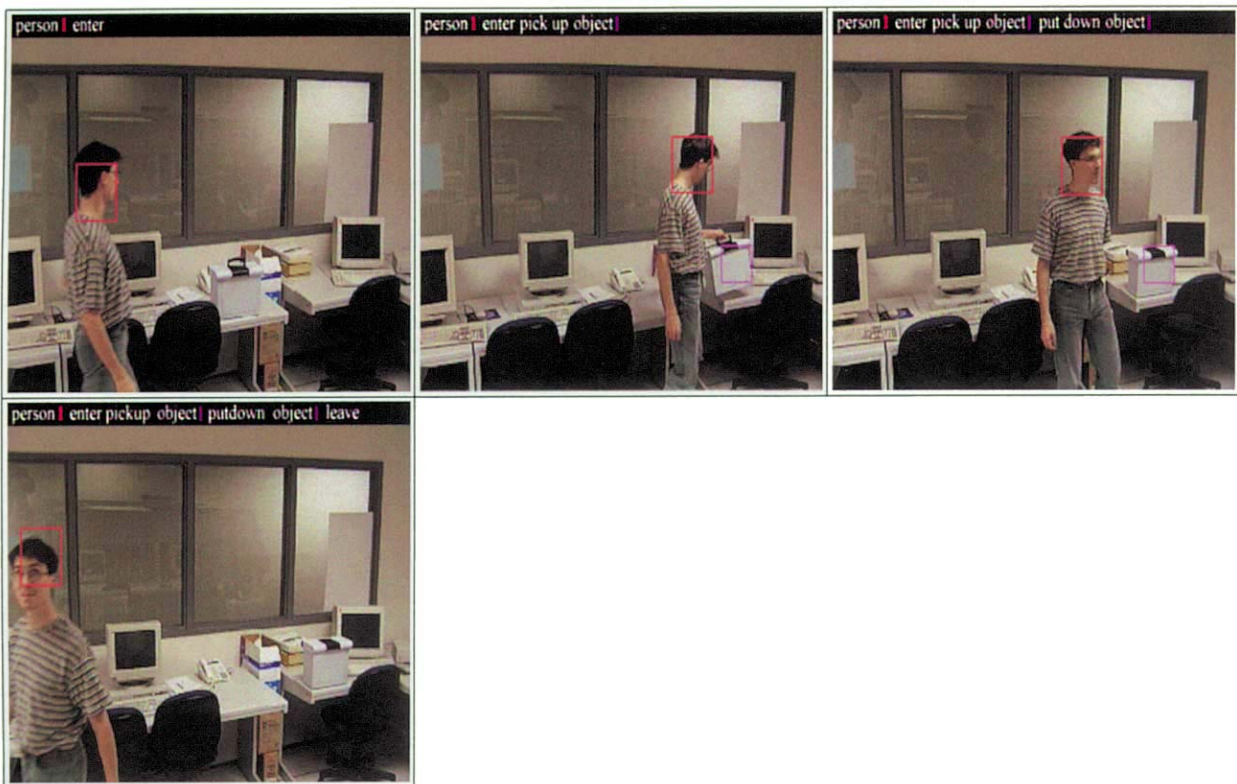


Fig. 6. A person picks up a box and puts it down at a different location (199 frames, 4 key frames).

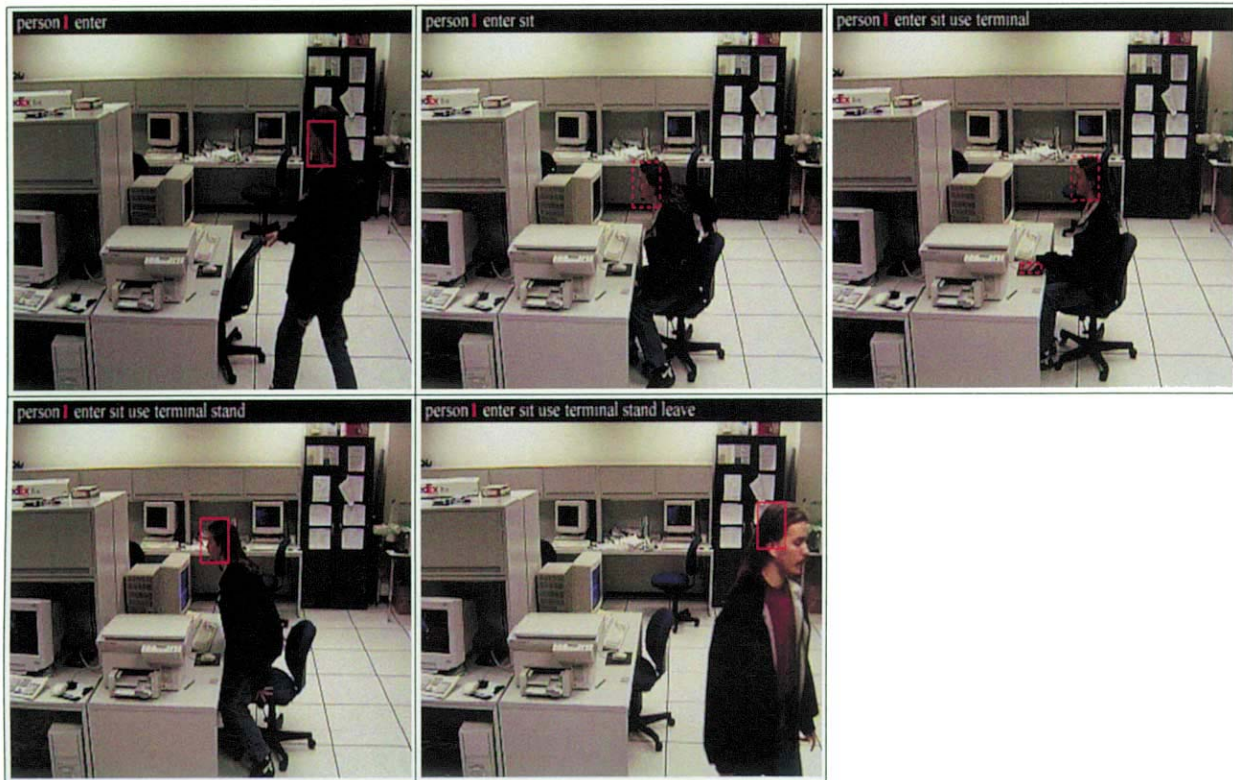


Fig. 7. A person sits down to use a terminal then stands up and leaves (399 frames, 5 key frames).

199 frames each, out of which 4 key frames were taken from each. The fourth sequence (Fig. 7) consisted of 399 frames, out of which 5 key frames were taken. The fifth sequence (Fig. 8) consisted of 170 frames out of which 3 key frames were taken. The sixth sequence (Fig. 10) consisted of 325 frames, out of which 4 key frames were taken. This is a tremendous amount of compression, less than 5% of frames are detected as key frames.

Fig. 4 shows several of the heuristics discussed in Section 5 in action. The enter and leave key frames are taken some distance away from the end of the entrance area. The person has finished sitting or standing, when the ‘stand’ and ‘sit’

key frames were taken. The cabinet is fully opened and fully closed when the open and close key frames were taken. The key frames have these characteristics despite the fact that the system actually detected the action at a different time. Similarly, in Fig. 5, notice that the phone is not tracked until the person actually starts to pick it up. Scene change detection is used to make the determination about when to start tracking the phone. When the phone reaches its apex, the pick up phone message is displayed. When the phone has returned to its original position, the put down phone message is displayed. In Fig. 6, the box is not tracked until the person approaches it. The box is not declared as picked up

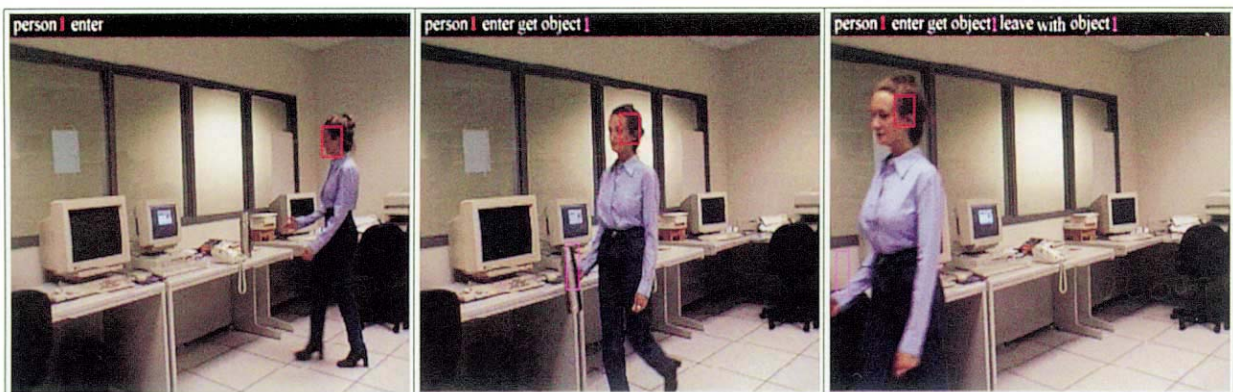


Fig. 8. A Person enters the room, picks up a thermos and leaves the room (170 frames, 3 key frames).



Fig. 9. A Person enters the room, picks up a thermos, but the tracking algorithm loses the thermos so an error is output, finally the person leaves the room (170 frames, 4 key frames).

until the person is far enough from the box's original location that scene change detection can be used to make sure that the box was actually picked up. When the person has moved a significant distance from the box, the system declares that the box has been put down. When the box is put down the system reinitializes the model for scene change detection on the box. Fig. 7 shows the 'use terminal' action.

In Fig. 10, we see the system 'breaking out' of a state, when an error is detected. First, the system correctly detects enter and pickup phone actions. However, shortly after the phone is picked up, the tracking algorithm fails and the phone is no longer correctly tracked. The system fails to recognize that the person put the phone down because of the failure of the tracking algorithm. When the person walks away from the phone, it is clear that an error has occurred. The system is unable to determine whether initially entering the talking on phone state was an error or whether the system missed a transition to put down phone. Therefore, a transition is made to the error recovery state and an error key frame is output. The system then recognizes the leave event.

Fig. 8 shows the leave with object action. In Fig. 9, we see the same sequence as in Fig. 8; however, the location of the thermos has not been input as accurately. This, combined with the fact that the thermos undergoes a lot of scaling because it is getting closer to the camera, causes the algorithm to lose the thermos quickly. By the time the person is far enough away from the initial position of the object for

the system to declare the 'pickup object' action, the tracking algorithm has already lost the thermos. When the thermos's tracking box has moved a significant distance away from the person, the 'error' message is displayed and the system stops tracking the thermos. Finally, the system detects the person leaving the room and the leave action is output.

6. Limitations

The system is limited by several factors. The first is the effectiveness of the three low-level computer vision techniques it utilizes. For example, if two people get too close, the tracking algorithm can become confused. Also, if the skin area of the face becomes occluded, the tracking algorithm may not function properly. Some objects are also difficult to track, these include objects that are too small or are easily confused with the background. The skin detection algorithm can have difficulty in a dark entrance, or if no clear view of a skin region is available as the person enters the room. Also, the skin detection algorithm is limited by the quality of the data it is trained on.

Another factor that affects performance is the quality of the prior knowledge that is provided to the system. For example, if an entrance is not declared large enough, people can enter the room undetected. Also, some actions are difficult or impossible to model in this approach. For

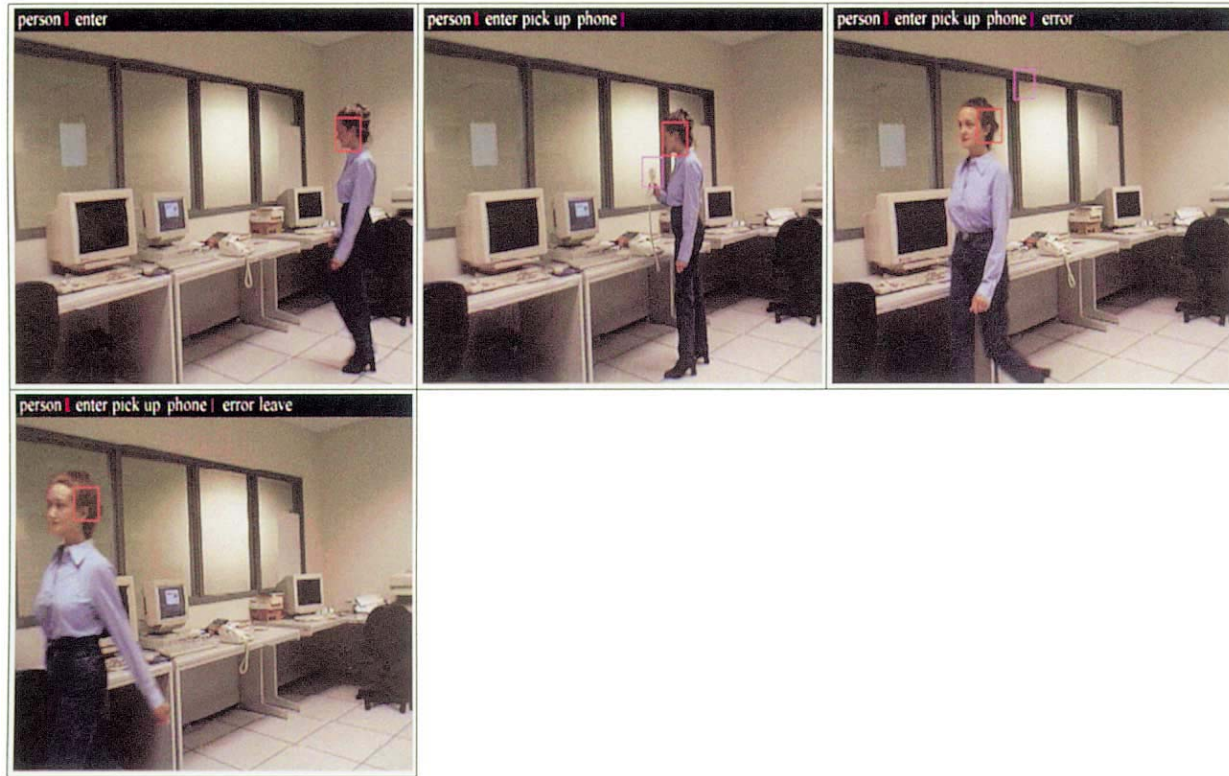


Fig. 10. A Person enters the room, picks up a phone, but the tracking algorithm loses the phone so an error is output, finally the person leaves the room (325 frames, 4 key frames).

example, it is impossible to model a new object entering the scene.

7. Conclusion and future work

We have proposed a system for recognizing human actions such as entering, using a terminal, opening a cabinet, picking up a phone, etc. The system uses the three low-level techniques of skin detection, tracking and scene change detection. We have successfully recognized these actions in several sequences, some of which had more than one person performing actions.

In the experiments shown in this paper, our field of view was limited. We have performed some experiments with a wide-angle lens (Figs. 9 and 10), which increased the field of view. This improvement, however, is still not satisfactory. It would be interesting to perform some experiments in a situation where the system could view most of a room. This would allow the system to be tested on sequences with several people in the scene performing a variety of actions simultaneously. We are experimenting with the use of multiple cameras to provide a view that encompasses most of a small room. We feel that this is one of the most important extensions that should be made to this system. Extending

our system to multiple cameras is conceptually straight forward, although the implementation details may be quite complex.

As mentioned in Section 3, it would be desirable for our system to incorporate some method for learning the layout of the scene automatically. Obviously, increasing the number of actions recognized by the system is another avenue for future work. Detecting the arrival of new objects into the scene is an action that will be added soon. Also, future work should include implementing the system in real-time and testing the system on a large number of long sequences. Another interesting area for future work would be determining the identity of a person who has entered the room. This could be done, off-line by processing the key frames. Also, any advances in the three techniques that the system relies on (skin detection, color-based tracking and scene change detection) would help improve the system. As mentioned earlier, changing the low-level techniques would not affect our high-level model for action recognition.

The system presented in this paper is capable of recognizing a number of interesting human actions. Future improvements to the low-level algorithms will make the system even more robust. The addition of multiple cameras and real-time analysis of video will allow the system to monitor an entire room over an extended period of time.

Acknowledgements

The authors want to thank Paul Fieguth for providing source code used for tracking.

References

- [1] J.K. Aggarwal, Q. Cai, Human motion analysis: a review, *Computer Vision Image Understanding* 73 (3) (1999) 428–440.
- [2] R.T. Collins, A.J. Lipton, T. Kanade, Special section on video surveillance, *Pattern Analysis and Machine Intelligence* 22 (8) (2000) 745–887.
- [3] D.M. Gavrila, The visual analysis of human movement: a survey, *Computer Vision and Image Understanding* 73 (1999) 82–88.
- [4] C. Stauffer, W.E.L. Grimson, Learning patterns of activity using real-time tracking, *Pattern Analysis and Machine Intelligence* 22 (8) (2000) 747–757.
- [5] F. Paul, T. Demetri, Color-based tracking of heads and other mobile objects at video frame rates, *CVPR*, 1997, pp. 21–27.
- [6] R. Kjeldsen, J. Kender, Finding skin in color images, *International Workshop on Automatic Face and Gesture Recognition*, 1996, pp. 312–317.
- [7] T.J. Olson, F.Z. Brill, Moving object detection and event recognition for smart cameras, *Proceedings of IU*, 1997, pp. 159–175.
- [8] C. Wren, A. Azarbayejani, D. Trevor, A.P. Pentland, Pfinder: real-time tracking of the human body, *Pattern Analysis and Machine Intelligence* 19 (7) (1997) 780–784.
- [9] A. Nagai, Y. Kuno, Y. Suirai, Surveillance systems based on spatio-temporal information, *International Conference on Image Processing*, 1996, pp. 593–596.
- [10] R.L. Rosin, T. Ellis, Detecting and classifying intruders in image sequences, *Proceedings of British Machine Vision Conference*, 1991, pp. 24–26.
- [11] A. Makarov, J.-M. Vesin, F. Reymond, Detecting and classifying intruders in image sequences, *Proceedings of SPIE: Real-Time Imaging* 2661 (1996) 44–54.
- [12] S.S. Intille, A.F. Bobick, Closed-world tracking, *CVPR*, 1996 pp. 672–678.
- [13] S.S. Intille, J.W. Davis, A.F. Bobick, Real time closed-world tracking, *ICCV*, 1997, 697–703.
- [14] M. Izumi, A. Kojiam, Generating natural language description of human behavior from video images, *ICPR* 4 (2000) 728–731.
- [15] J. Davis, M. Shah, Visual gesture recognition, *IEE Proceedings Vision, Image and Signal Processing* 141 (2) (1994) 101–106.
- [16] C. Cédras, M. Shah, Motion-based recognition: a survey, *Image and Vision Computing* 13 (2) (1995) 129–155.
- [17] B. Aaron, J.W. Davis, Motion-based recognition, in: M. Shah, R. Jain (Eds.), *Action Recognition Using Temporal Templates*, Kluwer Academic Publishers, Dordrecht, 1997, pp. 125–146.
- [18] S. Shershah, S. Gong, A.J. Howell, H. Buxton, Interpretation of group behavior in visually mediated interaction, *ICPR* 1 (2000) 226–269.
- [19] H.J. Zhang, S.W. Smoliar, J.H. Wu, Content-based video browsing tools, in: *SPIE Proceedings of Storage and Retrieval for Image and Video Databases*, 1995, pp. 389–398.