DISCRETE-EVENT MODELING OF MISRECOGNITION IN PTZ TRACKING

Abstract

This paper introduces an approach to the problem of choosing when to zoom a moving camera so as to follow a designated video surveillance target. Rather than just trying to maintain a simple viewing constraint (e.g., target > 10% of image), the potential misrecognition of the target is also used to decide when to zoom. A discrete-event approach is used to develop two models of *appearance change* as well as a model that represents the *viewing constraints* for target surveillance. Disagreement between the appearance models represents a potential loss of target. Supervisory discrete event control theory is used *to automatically* construct a controller that selects zoom actions to prevent loss of target. The implementation of this controller is overviewed and results presented.

DISCRETE EVENT MODELING OF MISRECOGNITION IN PTZ TRACKING

Damian M. Lyons

Dept. of Computer & Information Science
Fordham University
Bronx NY 10458
dlyons@fordham.edu

ABSTRACT

This paper introduces an approach to the problem of choosing when to zoom a moving camera so as to follow a designated video surveillance target. Rather than trying to maintain a simple viewing constraint (e.g., target > 10% of image), the potential misrecognition of the target is also used to decide when to zoom. A discrete-event approach is used to develop two models of *appearance change* as well as a model that represents the *viewing constraints* for target surveillance. Disagreement between the appearance models is taken to indicate a potential loss of target. Supervisory discrete event control theory is used *to automatically* construct a controller that selects zoom actions to prevent loss of target. The implementation of this controller is overviewed and results presented.

1 INTRODUCTION

This paper introduces a new method to address the problem of choosing when and how much to zoom a PTZ camera so as to follow a designated target. Typically the PTZ control problem is specified as the control necessary to keep the target within certain PTZ bounds so that a security operator can easily observe the individual. For example, the automated tracking controller may be constrained to keep the target centered and sized to fit into a bounding box in the center of a display monitor. This establishes *conflict* between the need of the controller to uniquely recognize and "lock" onto the target and these PTZ bounds. This conflict is usually resolved in favor of the observer by performing recognition first on each frame followed by a PTZ motion to bring the target into viewing bounds. This sacrifices tracking reliability for good observability; the security guard may get a good view of the wrong target!

The paper presents two useful results. The first is the discrete-event modeling of both target recognition *and* target viewing constraints. The second is the automatic derivation of an optimal supervisory PTZ controller description from these models.

2 EXISTING WORK

The literature contains a number of approaches to the problem of tracking a single human surveillance target in a sequence of video images[1]. Pfinder [2] tracks a single person in a stationary camera (or stereo camera pair) using a multi-class statistical model of color and shape to segment the target from its background. W4 [3] tracks people in stationary monochromatic video using background subtraction for segmentation and making use of a cardboard shape model [4] that represents the relative position and sizes of body parts. The Nine-grid [5] algorithm employs line-scan measurements to label body features using a 2D model. Backpack [6] builds on [4] to determine whether a target is carrying anything. Some authors have addressed the issue of tracking the individual parts of the body of the human target [7, 8] using techniques usually applied to multiple target tracking [9]. Others have used flexible 2D contours to model the target shape [10]. In each case, since the data comes from one or more stationary cameras, the primary use of target model is to interpret the image data.

However, in the case of a PTZ camera [11], the model information can be used to attempt to improve the future tracking performance. For example, the camera can be moved so that the target is centered and zoomed and not clipped or too small. In general, the PTZ commands are generated purely to improve such a "viewing" constraint [12]. This constraint is really a specification of the desired *output* of the tracking system to an observer, e.g., a security guard. There is no guarantee that such a viewing constraint represents whether the tracking system can recognize the target. The objective of this paper is to develop an approach to modeling the target so that PTZ commands can be issued that improve tracking performance as well as deliver the required output to an observer.

This paper presents two state-based models of how the appearance of a human surveillance target can change over time, one based on shape and one based on color. The state of each model can be generated by looking at some measurements of the target in the current image while tracking. Disagreement between these models about what state the target is in is considered a potential loss of target. Operator viewing constraints are also represented with a state-based model. The discrete-event theory of Ramadge & Wohnam [13] is then used to construct a supervisory PTZ controller that obeys both the viewing and recognition constraints.

3 SUPERVISORY DISCRETE EVENT CONTROL

In a number of papers since 1982, Ramadge & Wonham (RW) [13, 14] and their students have developed a mathematical theory of control for discrete-event system (DES). This theory is based on the common concept of an automaton. RW introduce the concept of a controllable event; that is, an event that *can be prevented from occurring*. They add a mapping γ to the automaton description to represent such a *controllable automaton*. This mapping indicates for each state whether each controllable event is enabled or not.

The discrete event control problem they set themselves is called *Supervisory Control*: Can the controllable events be manipulated so that the events generated by a discrete event plant model stays *within* some defined specification, a subset of all possible events. The specification they use is the set of regular languages over the alphabet of the plant. Any such language can be implemented as an automaton [15].

There exists a least restrictive, non-blocking supervisory controller for a plant and specification provided they obey some basic constraints. These constraints are that the specification language is *controllable* and *prefix closed* with respect to the plant. If these hold, RW show that the controller CT for specification S and plant P is given by

$$L\,(CT) = \mathit{sup}\; C\,(\; L(S) \cap L(P)\;)$$

Where L(A) is the language generated by A, *C* (K) is the class of languages K, and *sup* is the supremal element of the class. RW provide an algorithm for generating CT.

Since both plant and specification are constructed as controllable automata, RW also provide a useful set of construction operators to build more complex automata from simpler ones, including the following:

- The *synchronous product* of two automata, A||B, is simply the shuffle product if their alphabets are disjoint. If they share any events in common, then A||B synchronizes on these events.
- The *projection operation* on an automaton, which we write here as A/Σ, is the automaton resulting from A if the events in Σ are not observable.

Supervisory control is typically about control by *disabling* events, rather than by generating them. To control a PTZ camera, it is necessary to generate control events. RW have also developed an approach to control problems requiring the generation (or “forcing”) of events. They refer to this as *timely preemption*.

The TCT [16] software tool allows a designer to create automata descriptions and apply the various operators in the formalism including the automatic generation of supervisory control automata.

4 RECOGNITION AND VIEW MODELS

The target model represents the target as seen by the PTZ camera. The target model will consist of two *cue* models – one will capture the way the target shape changes as the target moves, and the second will capture the way the target color changes as the target moves.

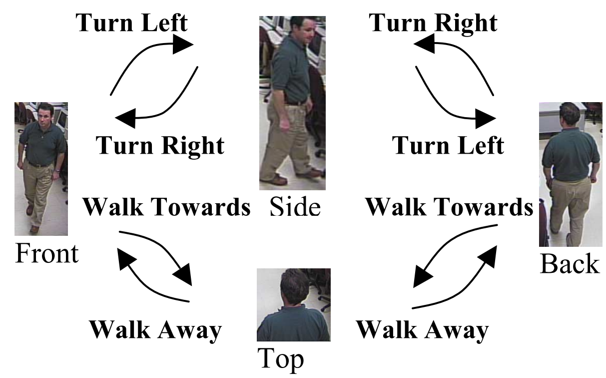


Figure 1: Target Model State Graph

The camera is assumed to be in an overhead location. Thus, the camera will view the target either from the front,

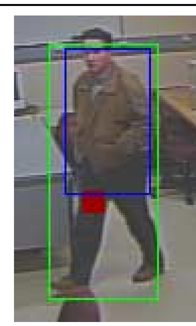


Figure 2: Shape Estimation

the side, the back, or the top (Figure 1). The target can transition from one orientation to another by turning and/or by walking towards or away from the camera. Both the shape and color cue models are based on this general, four-state target model.

The Shape Cue Model. The foreground region in the image is identified by motion differencing. The PTZ camera is held stationary while this is done. A bounding box is inscribed around the foreground region. Depending on which state the target is in, it will present a different silhouette shape to the camera view. The state is estimated by measuring the dimensions of the bounding box as shown in Figure 2. The outer bounding box shown is the bounding box of the full foreground motion area. The inner bounding box covers the top half of the area only (to avoid walking legs). The aspect ratio of width (w) of the inner box to the height (h) of the outer box is used to estimate whether the target is facing the camera front, side or from the top. Since back and front have the same aspect ratio in general, the four-state model of Figure 1 collapses into the three state model of Figure 3. The aspect ratio conditions for each state were established empirically.
The three shape states are labeled SFront, STop and SSide. The target transitions between these states by moving. These motions are modeled by a set of events linking the three states in a completely connected state graph. Notice that none of these events are controllable. Figure 3 can be represented by the following automaton:

$$
\begin{aligned}
CM_{shape} &= (\Sigma_c, \Sigma_u, Q, \delta, q_0, Q_m), \quad \text{where} \\
\Sigma_c &= \varnothing \\
\Sigma_u &= \{t1, mt, ma\} \\
Q &= \{\text{SFront, STop, SSide}\} \\
\delta &= Q \times Q \\
q_0 &= \text{Front} \\
Q_m &= Q.
\end{aligned}
$$

The Color Model. The color cue model describes how the set of color regions identified with the target change. It will be assumed that the principal cause of change in the target color is the pose of the target with respect to the camera. It will also be assumed that there is a target identification phase prior to tracking: that is, a phase in which the system can take a set of characteristic color measurements of the target. These color measurements are made as follows. The target is divided spatially into a series of regions as shown in figure 4. In the implementation used in this paper, the operator uses the mouse to draw a bounding box over the intended target. The bounding box is then divided up automatically into four regions covering the hair, face, torso and legs of the target. The relative size of these regions comes from biometric average measurements [17]. It is assumed that the characteristic color in each region can be approximated by a normal distribution $\mathcal{N}(\mu,\sigma)$. The RGB color pixels in each region are normalized to RGY and the following means and standard deviations calculated:

$$\mathcal{N}_{fi}(\mu, \sigma) \text{ for } f \in \{Hair, Face, Torso, Legs\}, \; i \in \{R,G,Y\}$$

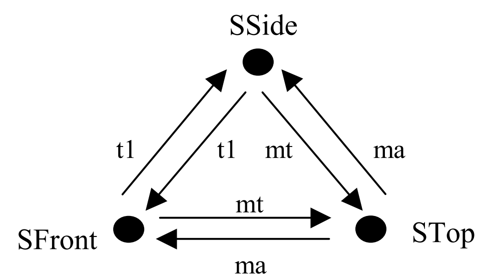


Figure 3: Shape Model

An example of the mean RGY color for each target region is shown on the left in Fig. 4.

During tracking, these color statistics are compared against the colors in the foreground bounding box.

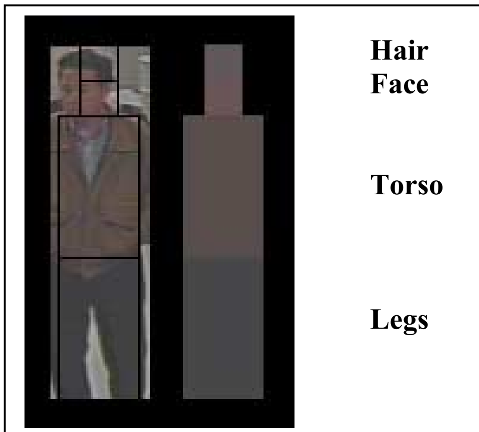


Figure 4: Spatial Color Regions

Because the motion region extraction produces a bounding box that does not always correspond well with the silhouette of the target, the following procedure is used to identify which parts of the foreground bounding box to test for which target region color.

When a target moves, part of the background is uncovered, which then becomes part of the foreground region extracted. To compensate for this, the direction of motion of the target is calculated in successive frames. A strip of length h and width $w/3$ is taken down the center of the bounding box. This strip is offset either to the left or right by $w/6$ depending on the direction of motion. The strip is divided into three portions vertically: Head and Face colors are matched in the upper region; Torso color in the middle region; and Leg color in the lower region.

Figure 5: Color Matching

Samples are taken from each region classified to see if they lie within σ_{fi} as follows:

$$C_{fi}(x) = \begin{cases} 1 \text{ if } (x-\mu_{fi})^2 < \sigma_{fi}^2 \\ 0 \text{ else} \end{cases}$$

$$\sum_i \sum_{x \in r} C_{fi}(x) > \frac{1}{t} A(r)$$

where *f*∈*{Hair, Face, Torso, Legs}* *i*∈*{R,G,Y}*, *r*∈*{upper, middle, lower}, and A(r)* is the area of portion of the strip, and *t* is a confidence threshold.

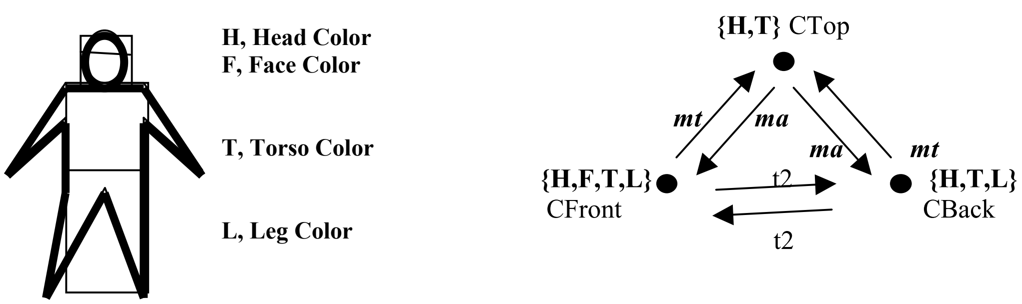


Figure 6: Color Cue Model

Depending on the target state, a different selection of the four color regions will be seen at any time. In the top view, the Hair and Torso colors dominate. In the front and side views, all four color regions can be seen. In the back view, the Face cannot be seen.

This generates the color cue model shown in Figure 6. Note that again this is a three state model, but this time the Front and Side target states collapse into CFront. The target can transition between these states by moving and turning, and these are represented by the transitions that link the three states in the completely connected graph shown. Although the move events are the same in both cue models, the turn events are different, due to the different mappings of the cue model states to the target model states. An automata description of CM_{color} can be constructed in the same manner as for CM_{shape}.

The two cue models share some events: This expresses a coupling between the models. For example, if both cue models are used in the plant model, then it should not be possible to be in both the STop and CFront states. To construct a plant model from both of these cue models, they are combined using RW's sync operator, which ensures that the models coordinate on events:

$$CM = CM_{shape} \,||\, CM_{color}$$

and define

$$Q_{ur} = Q_{shape} \times Q_{color} - Q_{cm}$$

which is the set of "illegal" state combinations such as (STop,CFront).

<u>Recognition model.</u> The target model represents the "real" motions and state of the target. However, the observation of the target with a camera may prevent the "real" state of the target from being measured. A key problem in observing the shape and color information used in the target model is the image resolution: the target may be too small or partially out of view. The solution is to devote more image pixels to the target; that is to center and zoom into the target. We will assume that re-attaining the target area observed in the target identification phase is the best zoom choice to identify the target.

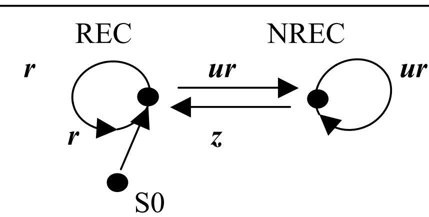


Figure 7: RM with forced zooming

To formalize the recognition model we introduce two image interpretation functions. Let *I* be the set of all images. The shape interpretation function R_{shape} maps an image $i \in I$ onto a state $q \in Q_{shape}$ and R_{color} maps an image to a state $q \in Q_{color}$. The multi-cue recognition function R is defined as:

$$R: I \rightarrow Q_{shape} \times Q_{color} \quad s.t. \quad R(i) = (R_{shape}(i), R_{color}(i))$$

We will define two events based on this recognition function.

Event ur occurs if $R(i) \in Q_{ur}$
Event r occurs if $R(i) \in Q_{cm}$

Intuitively, the event *ur* happens if the recognition function reports a state that is "illegal" with respect to the cue models, and *r* if the state is legal. The state representation for the recognition is relatively simple, as shown in Figure 7.

The model's REC and NREC states represent a consistent (with the cue models) and inconsistent interpretation of the target respectively. When an inconsistent interpretation of the target is obtained (the event *ur*) the only way to recover is to attempt to devote more image pixels to the target by *recentering* and *rezooming*. Let A_m be the area of the bounding box of the target in the image when the color model was originally gathered. Let z_m be the zoom setting necessary to drive the area of the bounding box from its current value to A_m in the next image, if this is possible, otherwise it is the maximum zoom setting. The zoom event should produce *at least* z_m. This zooming is the event *z* shown in Figure 7. A controller can choose to issue a recenter and zoom event whenever it pleases. RW implement these forced events such as these using event preemption. The recognition model including RW preemption is show in Figure 8 below.

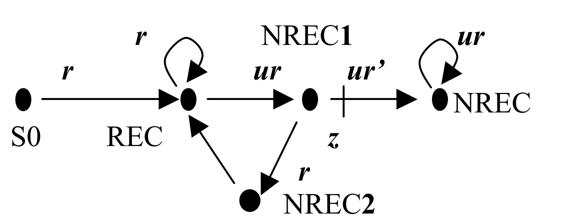


Figure 8: RM with preemption

A new event *ur'* is defined. This has the same definition as *ur* but it is classified as a controllable event. The intermediate states NREC1 and NREC2 are also introduced. Disabling the controllable event *ur'* will be considered the same as requesting that the forced event *z* should occur.

The state and transition structure shown for the recognition model RM in Figure 8 is a good general template for a controllable degree of freedom of the camera. Therefore we will define an *automaton template* DOF(a,b,c) as follows:

$$RM = DOF(r, ur, z)$$

This means that replacing the generic events a, b and c in the automaton template with the events *r, ur,* and *z* then we get the automaton RM shown in Figure 8. The state names are irrelevant for our purposes and they can simply be referred to as numbers.

<u>View model</u>. Whereas RM represents recognition constraints on the tracking process, the view model will represent the "output" constraints on the tracking process – the constraints about what constitutes a good image for an observer such as a security guard. We will impose two straightforward constraint: In each image *i,* the bounding box area of the target *BA(i),* and the location of the target in the image *PT(i),* should always be bounded as follows:

$$A1 < BA(i) < A2$$

$$| C - PT(i) | < N$$

where *A1* and *A2* are size bounds (e.g., 25% of the total image), where *C* is the image coordinates of the center of the image, and *N* is a positive constant less than or equal to half the smallest dimension of the image. The view model requires a rezoom action when *BA(i)* is too small or too big, and a recenter only when *PT(i)* becomes too far from the center of the image.

We will introduce an event *oz,* to happen when (11) does not hold on the image, and an event *ov* when (12) does not hold. The event *ok* happens only when (11) & (12) hold. These are *uncontrollable* events. For preemption purposes, the events *oz'* and *ov'* will also be introduced, with the same definition, but classified as controllable. Disabling the first will be understood to mean specifying a forced rezoom and recenter event *rz*, the second, a forced recenter event *rc* (with no zooming).

Using (10) we can now define the view model as a synchronous product of two automata generated from the DOF template as follows:

$$VM = DOF(ok, ov, rc) \,||\, DOF(ok, oz, rz)$$

<u>Plant Model.</u> The Plant model can now be defined as the synchronous product of the Cue Model, the Recognition Model and the View Model:

$$PLANT = CM \,||\, RM \,||\, VM$$

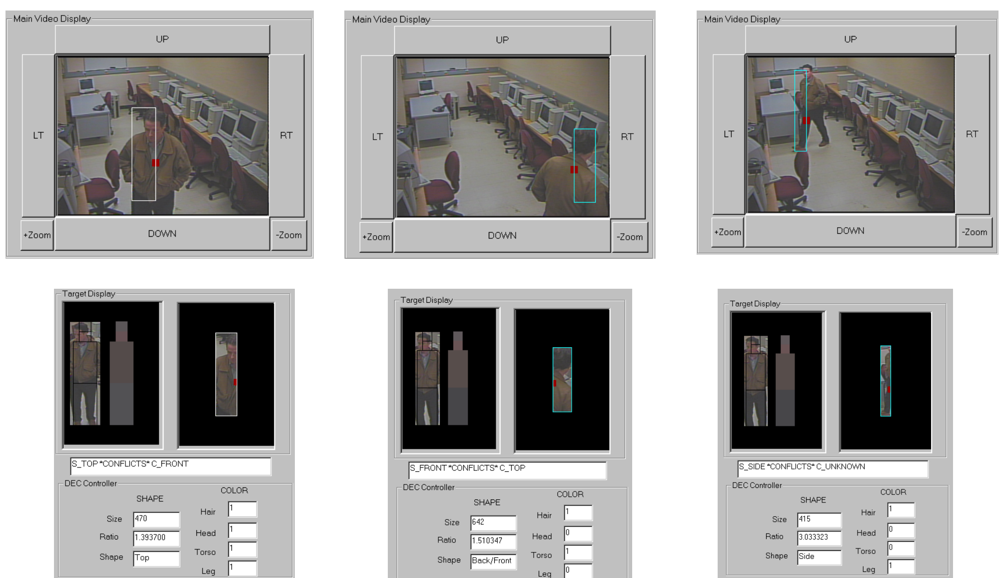
Since no events are common between the models, this is simply the shuffle product of the three automata and has $Q_{cm} \times Q_{rm} \times Q_{vm}$ states. We can simplify this a little. First, note that the physical actions of the target in CM (i.e., *t1, mt,* etc.) are not directly observable; their effect can only be seen via RM. Thus, we can project Σ_{cm}, the modeled physical motions of the target, out. Secondly, the forced control events $\Sigma_{fc} = \{z, rc, rv\}$ used in the RM and VM models are only controllable via their respective preemption events: *ur', oz', ov'*. Thus, these can also be projected out to simplify the model.

$$PLANT = (CM \,||\, RM \,||\, VM) \,/\, \Sigma_{cm} \cup \Sigma_{fc}$$

5 CONTROLLER SYNTHESIS

The supremal controllable sublanguage (SCS) approach will be used to automatically design a controller for the plant (14). The first step is to specify the desired behavior of the controlled plant as a regular language over the event set of the plant. Informally, the desired behavior is that the automatic tracker maintains the target in a recognized state and maintains the target in view. In event terms, this means the controlled event language should contain strings of the event *r* of any length. However, if *ur* appears in any string, it should be immediately followed by *r*. The automaton, RC is a generator for this language:

$$RC = DOFSPEC(r, ur)$$



(a) **(b)** **(c)**

Figure 9: Misrecognition Examples

With respect to the viewing constraints: The language should include strings of *ok* events of any length. If an *oz* or *ov* happens, then they should be immediately followed by an *ok* event. Finally, any synchronous combination of these strings is also a valid string in the language.

$$VC = DOFSPEC(ok, ov) \,||\, DOFSPEC(ok, oz)$$

The shuffle product of these automata will allow any shuffle of the strings, as desired, giving us the final control criterion:

$$SPEC = RC \,||\, VC$$

The SCS approach can now be used to automatically generate a non-blocking, minimally restrictive controller that controls PLANT to ensure *only* the language generated by SPEC is generated:

$$L(CONTROL) = \sup C\ (L(SPEC) \cap L(Plant))$$

The TCT development environment [16] was used to evaluate (17) above, to ensure that (16) is controllable and prefix closed, and to produce the control enablement mapping γ. This resulting controller had 15 states and 47 transitions. The control disablement mapping, γ, associated with CONTROL was generated as follows (the prefixed numbers are the state names)

3: $\gamma(ur')=1$ 5: $\gamma(oz')=1$
6: $\gamma(ov')=1$ 7: $\gamma(ur')=1$
8: $\gamma(oz')=1$ 9: $\gamma(ov')=1$
10: $\gamma(oz')=1$ $\gamma(ov')=1$ 11: $\gamma(ur')=1$ $\gamma(oz')=1$
12: $\gamma(ur')=1$ $\gamma(ov')=1$ 13: $\gamma(oz')=1$ $\gamma(ov')=1$
14: $\gamma(ur')=1$ $\gamma(oz')=1$ $\gamma(ov')=1$

Notice that in states 14 and 11, the controller will attempt to reset zoom for the recognition model and also the viewing model. To resolve this conflict, we add the constraint that $A_m \leq A2$.

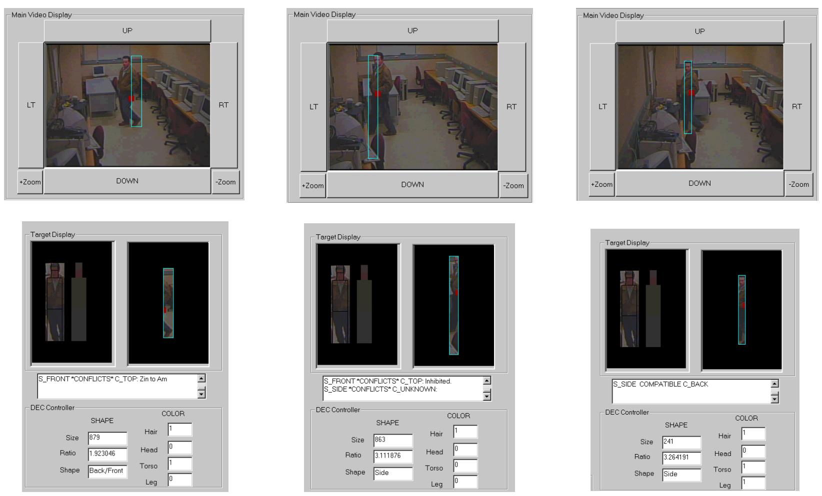
6 RESULTS

In the implementation used in this paper, image differencing is used to locate the foreground region. The area and location of the foreground bounding box is calculated as are the recognition measurements as described in the shape and color cue model sections. Based on these measurements, the input events to the CONTROL automaton are generated. The output from the automaton is a command to recenter, a command to recenter and rezoom to the viewing constraints, or a command to rezoom to the original target area A_m or some combination of these.

Figure 10 shows an example of a misrecognition (a) followed by a zoom resulting in recognition of the target (c). The middle frame (b) shows an example of temporal noise being filtered.

7 DISCUSSION

This paper has presented an approach to the design of an automatic tracking controller for deciding when to zoom, so as to keep a target under video surveillance using a PTZ camera. A problem was identified with the current approach to PTZ tracking: namely, that control of the



(a) (b) (c)

Figure 10: Misrecognition followed by Zoom

Each of these commands resulted in a signal that moved the pan, tilt and zoom motors a portion of the distance towards their goal, unless the camera was already within a threshold distance of this. This simplification weakened the specification in (15); allowing multiple misrecognition events in a row, but eliminated the need for extensive camera calibration.

Zoom commands were filtered by requiring that two successive zoom commands agree on the zoom value, and inhibiting further zoom for a short period after a zoom. This was necessary to handle the temporal noise in the foreground extraction.

Figure 9 shows three examples of disagreement between shape and color models during tracking. Note in 9(c) that the color information does not place the color model into any of its states. A fourth color cue model state was added, CUnk, to capture this case.

camera is directed by operator viewing constraints, and only indirectly on improving the tracking performance. This paper has concentrated therefore on developing a model of potential misrecognition of the target from the image data, and using this model to decide when to zoom. The approach is based on formalizing models of the target to represent the recognition and viewing criteria necessary to conduct tracking. Supervisory discrete event control theory is used to automatically construct an optimal controller that controls the camera both to improve tracking performance and to improve the operator viewing conditions. The implementation of this controller was overviewed and some results were presented of the controller operating as described.

There are two areas of future work. The first is in relaxing the assumption that the principal cause of color change is the pose of the target; Occlusion is also a major cause of color change. Indeed it affects both of the cue models used, but can most easily be represented by the color cue model. Since occlusion can result in model disagreement and an inappropriate choice of zoom, it would be advantageous to include it in the model. Occlusion causes additional legal subsets of the color regions in the template to become valid. For example, allowing the legs to be occluded generates additional states in the cue model.

The second area is in the selection of zoom actions. A very simple and inflexible approach was used in this initial work – zooming in or out to re-attain the original target size in the image. Ultimately, a more flexible approach calls for a hybrid control solution.

7 REFERENCES

1. Gavrila, D., The Visual Analysis of Human Movement: A Survey. Computer Vision and Image Understanding, 1999. 73(1): p. 82-98.
2. Wren, C., Azarbayejani, A., Darrell, T., Pentland A., Pfinder: Real-time tracking of the human body. IEEE PAMI, 1997. 19(7): p. 780-785.
3. Haritaoglu, I., Harwood, D., Davis, L. W4: Who, When, Where, What: A Real-time System for Detecting and Tracking People. in 3rd Face and Gesture Recognition Conference. 1998.
4. Ju, S.X., Black, M. J., and Yacoob, Y. Cardboard people: A parameterized model of articulated motion. in 2nd Int. Conf. on Automatic Face- and Gesture-Recognition. 1996. Killington, Vermont.
5. Lyons, D., and Pelletier, D., A Line Scan Computer Vision Algorithm for Identifying Human Body Features, in Lecture Notes in AI #1739, A. Braffart, et al., Editor. 2000, Springer-Verlag. p. 87-99.
6. Haritaoglu, I., Cutler, R., Harwood, D., Davis, L. Backpack: Detection of People Carrying Objects Using Silhouettes. in IEEE Int. Conf. on Computer Vision. 1999. Kerkyra Greece.
7. Cham, T., Rehg, J. A Multiple Hypothesis Approach to Figure Tracking. in IEEE Int. Conf. on Computer Vision and Pattern Recognition. 1999. Fort Collins, USA.
8. Rasmussen, C., Hager, G.,. Joint Probabalistic Techniques for tracking Multi-Part Objects. in Proc. Computer Vision & Pattern Recognition. 1998. Santa Barbara CA.
9. Rao, B., Data Association Methods for Tracking Systems, in Active Vision, A. Blake, Yuille, A.,, Editor. 1992, MIT Press. p. 91-106.
10. Baumberg, A., Hogg, D.,. Learning Flexible Models from Image Sequences. in Proc. 3rd European Conf on Computer Vision. 1994. Stockholm Sweden: Springer-Verlag.
11. Brodsky, T., Cohen, R., Cohen-Solal, E., Gutta, S., Lyons, D., Philomin, V., Trajkovic, M. Visual Surveillance in Retail Stores and in the Home. in Proc. European Workshop on Advanced Video Surveillance. 2001. Kingsten on Thames UK.
12. Stillman, S., R. Tanawongsuwan, and I. Essa. A System for Tracking and Recognizing Multiple People with Multiple Cameras. in Proceedings of Second International Conference on Audio- Vision-based Person Authentication. 1999. Washington, DC.
13. Wonham, W., Notes on Control of Discrete Event Systems. 2001.
14. Ramadge, P., Wohnam, W., Supervisory Control of a Class of Discrete Event Processes. SIAM Journal on Control & Optimization, 1987. 25(1): p. 206-230.
15. Hopcroft, J., Ullman, J., Introduction to Automata Theory, Languages and Computation. 1979, New York: Addison Wesley.
16. Wonham, W., Notes on Control of Discrete Event Systems. 1997 - 2002, University of Toronto: Toronto Canada.
17. Andersson, D.B.C.a.G.B.J., Occupational Biomechanics (2nd Edition). 1991: Publisher: John Wiley & Sons.