

# Target tracking and localization using infrared video imagery \*

Alex Barsamian and Vincent H. Berk and George V. Cybenko

Thayer School of Engineering, Dartmouth College, Hanover, NH 03755 USA  
*firstname.lastname@Dartmouth.EDU*

## ABSTRACT

One of the significant problems in visual tracking of objects is the requirement for a human analyst to post-process and interpret the data. For instance, consider the task of tracking a target, in this case a moving person, using video imagery. When this person hides behind an obstruction, and is therefore no longer visible by the camera, conventional tracking systems quickly lose track of the target and are no longer able to indicate where the target is or where it was headed. A human interpreter is then needed to conclude that the person is hiding, and probably (with certain probability) is still there.

A Process Query System (PQS) is able to track and predict the path of arbitrary objects, based only on a description of their dynamic behavior, thus eliminating the need for precise identification of each object in every frame. The PQS is therefore able to draw human-like conclusions, allowing the system to track the person even when he/she is out of view. Additionally, using dynamic descriptions of tracked objects allows for low-quality video signals, or even infrared video, to be used for tracking.

In this paper we introduce a novel way of implementing a video-based tracking system using a Process Query System to predict the position of objects in the environment, even after they have disappeared from view. Although the image processing pipeline is trivial, tracking accuracy is remarkably high, suggesting that overall performance can be improved even further with the use of more sophisticated video processing and image recognition technology.

**Keywords:** Thermal Imaging, Visual Tracking, Process Query Systems

## 1. INTRODUCTION

Currently, most crowded, public locations are monitored by dozens, sometimes even hundreds of cameras. Airports, shopping malls, casinos, stores, and even traffic intersections are populated with digital imaging equipment, usually for security, but sometimes for marketing or simple behavioral observation. Since the cost of this infrastructure has dropped significantly with the introduction of digital video systems, their proliferation has exploded beyond the point where humans can realistically observe all feeds concurrently. (In many cases, the observation data is never seen until or unless an incident occurs, prompting subsequent review of a particular segment of old footage.)

It is therefore necessary to devise novel and comprehensive methods for maintaining a basic level of situational awareness in any given observed environment. Unfortunately, simple tracking is not enough to deduce possible intent from tracked objects. For instance, a group of people does not necessarily constitute a threat; however, if all individuals are moving in your direction, then it may be. Similarly, when tracking many individuals, it may be that we are more interested in certain individuals than in others. This distinction, for example, could be made based on their individual behaviors such as direction and speed of movement, or other dynamic characteristics.

Two key elements to achieving full situational awareness are: (1) the ability to track objects when they move out of sight, or are obscured by other objects, and (2) the ability to identify key patterns of behavior based on the dynamics of an object. For instance, imagine a vehicle in an urban environment, tracking other vehicles and persons in its vicinity. As this vehicle moves, buildings will obscure other objects both temporarily as well as permanently. Additionally, to determine if a person is a threat, it may be necessary to deduce if the person is moving towards the vehicle, for example. When such a situation is detected, a human operator can be notified of the threat. By using a Process Query System (see,<sup>1</sup> and<sup>2</sup>) as our situational awareness engine, we are able to satisfy both key elements efficiently and generically.

---

\*Supported under DOJ Award No. 2000-DT-CX-K001 and DHS Award No. 2005-DD-BX-1091 Points of view in this document are those of the author(s) and do not necessarily represent the official position of DHS.

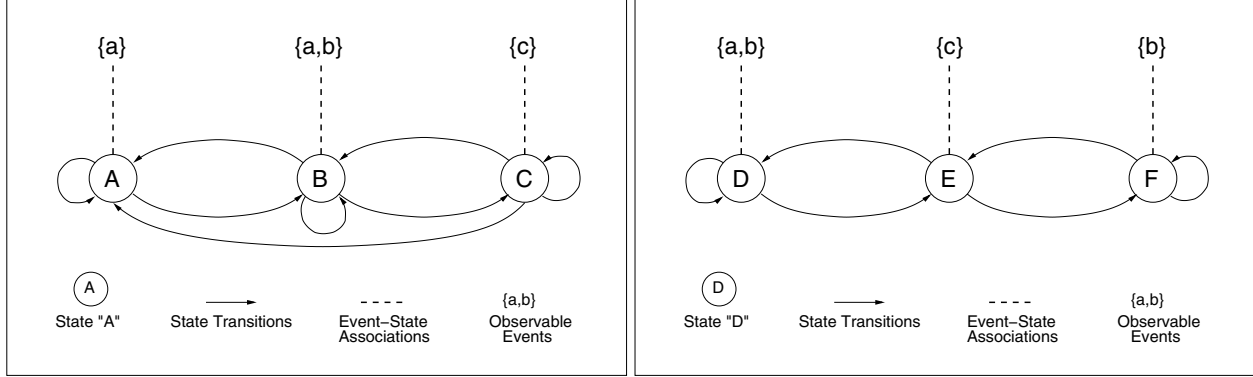


Figure 1. Two process models,  $\mathcal{M}_1$  and  $\mathcal{M}_2$ .

## 2. BACKGROUND

### 2.1. Visual Tracking

For as long as video imagery has been used in surveillance of remote locales, there has been an interest in automated tracking and classification of visual objects. It is both a means to assist humans who are viewing video streams and making decisions based upon their observations, and an important component technology to other automated systems. Recently, a fairly advanced level of scene processing has become commonplace. Much modern video recording equipment, for instance, has as a standard feature: The ability to only record frames which contain motion. That is, frames which are significantly different from a background scene. And a number of object-tracking methodologies have emerged based upon this: A static, unchanging background against which objects to be tracked move, such as the approach taken by Wang and Doherty.<sup>3</sup>

A drawback to such systems is their poor ability to cope with dynamic scenes of changing viewpoints. Infrared sensors partially address this problem, by helpfully classifying objects against an arbitrary background based on their relative energy output; this leaves only the work of tracking them. Reid, drawing on the best ideas of a number of others in the field, proposed a system<sup>4</sup> in 1979 that addresses aircraft tracking from radar sensors with a hypothesis refining algorithm that initiates tracks and then either discards or reports them as more observations flow in.

### 2.2. Process Query Systems

Process Query Systems are a new paradigm in which user queries are expressed as process descriptions. This allows a PQS to solve large and complex information retrieval problems in dynamic, continually changing environments where sensor input is often unreliable. The system can take input from arbitrary sensors and then form hypotheses regarding the observed environment, based on the process queries given by the user. Figure 1 shows a simple example of such a model. Model  $\mathcal{M}_1$  represents a state machine  $S_1 = (\mathcal{Q}_1, \Sigma_1, \delta_1)$ , where the set of states  $\mathcal{Q}_1 = \{A, B, C\}$ , the set of observable events  $\Sigma_1 = \{a, b, c\}$ , and the set of possible associations  $\delta_1 : \mathcal{Q}_1 \times \Sigma_1$  consists of  $\delta_1 = \{\{A, a\}, \{B, a\}, \{B, b\}, \{C, c\}\}$ . Notice how this process is able to produce observed event  $a$  in both state  $A$  and state  $B$ . A possible event sequence recognized by this model would be:

$$e_1 = a, e_2 = a, e_3 = b, e_4 = c, e_5 = b$$

which we will write as  $e_{1,5} = aabcb$  for convenience. Possible state sequences that match this sequence of observed events could be  $AABCB$ , or  $ABBCB$ , both of which are equally likely given  $\mathcal{M}_1$ . A rule-based model would need a lot of rules to identify this process, based on all the possible event sequences. Below is a set of all the rules necessary for detecting single transitions:

$$\begin{aligned}
AA &\rightarrow \{aa\} \\
AB &\rightarrow \{aa\}, \{ab\} \\
BB &\rightarrow \{aa\}, \{ab\}, \{ba\}, \{bb\} \\
BA &\rightarrow \{aa\}, \{ba\} \\
BC &\rightarrow \{ac\}, \{bc\} \\
CC &\rightarrow \{cc\} \\
CB &\rightarrow \{ca\}, \{cb\} \\
CA &\rightarrow \{ca\}
\end{aligned}$$

Needless to say, the list of possible event sequences for double transitions is massive (eg. transitions  $AAA$ ,  $AAB$ ,  $ABB$ ,  $ABC$ , ... etc.) and only gets bigger when we consider dealing with the possibility of missed observations. Rules would then have to include sequences that have two transitions for a single observed event, albeit with a lower priority, accounting for the fact that we expect only few observations to go missing.

Now let's introduce a second model  $\mathcal{M}_2$  in Figure 1 defined by state machine  $S_2 = (Q_2, \Sigma_2, \delta_2)$ . Consider that both processes are regularly and concurrently occurring in the observed environment. Note that the process states are labelled differently, i.e.  $Q_1 \cap Q_2 = \emptyset$ , although both processes produce the same set of observable events, i.e.  $\Sigma_1 \cap \Sigma_2 \equiv \Sigma_1 \equiv \Sigma_2$ . Now consider the following sequence of events:

$$e_{1:24} = abaacabbacabaccabacbbc$$

where each observation may have been produced by instances of model  $\mathcal{M}_1$ , model  $\mathcal{M}_2$ , or be totally unrelated. It must be noted that any modelled process may very well be occurring multiple times concurrently. A Process Query System uses multiple hypotheses, multiple model techniques to disambiguate observed events and associate them with a "best fit" description of which processes are occurring and in what state they are. In comparison, a rule-based system would get impossibly complex for the above situation. Additional problems with rule-based approaches arise when probabilities are assigned to the transitions, and/or the event productions.

Consider the following example. Assume the first model describes the dynamics of a propeller plane, and the second model describes the dynamics of a fighter jet, both observed by radar. It may very well be possible that there are several propeller planes and a group of jet fighters in the same airspace, all within radar range. The PQS will use the radar data as input observations together with the two models to disambiguate which radar observations were triggered by which aircraft, by associating radar observations using the models. Subsequently, the hypothesis will be that there are several instances of the model  $\mathcal{M}_1$  (the propeller plane) and a group of instances of model  $\mathcal{M}_2$  in the observed environment. Since the environment is dynamic, the *top hypothesis* will be changing continuously as planes move in and out of radar range.

A PQS is a very general and flexible core that can be applied to many different fields. The only things that change between different applications of a PQS are the format of the incoming observation stream(s) and the submitted model(s). Compare this with a traditional DBMS; inventory tracking systems, accounting, customer databases, etc. are all different applications that, at the core, are all based on the same DMBS. Likewise we have implemented vehicle tracking systems, server farm monitoring applications, enterprise network security trackers, and covert timing channel detectors using the same PQS software core by simply supplying a different observation stream and a different set of models. Internally a PQS has four major components that are linked in the following order:

1. Incoming observation handling and sensor subscription.
2. Multiple hypothesis generation.
3. Hypothesis evaluation by the models.
4. Selection, pruning, and publication.

To conclude, the big benefits of a PQS are its superior scalability and applicability. The application programmer simply connects the input event streams and then focuses on writing process models. Models can be constructed as state machines (above), formal language descriptions, Hidden Markov Models, kinematic descriptions, or a set of rules. All these different

model types are first compiled down to the fundamental PQML (*Process Query Modeling Language*) representation and then submitted to the PQS. The PQS is now ready to track processes occurring in a dynamic environment and continuously present *the best possible explanation of the observed events* to the user.

### 3. EXPERIMENTAL SETUP

Our data source for infrared tracking and classification experiments is a Thermal-Eye TSCss Long Wave Infrared camera produced by L3 Communications Infrared Products. It comes in many configurations; the camera used for these experiments has a fixed focus lens with a 50° field-of-view and an operating range of +6° to +40°C. The output is a grayscale image of the scene in NTSC-compatible format and captured by a Linux PC with an off-the-shelf framegrabber video card.

The camera recalibrates its output in real time in much the same way that visible-light cameras with auto-exposure adjust the exposure of scenes. As the total amount of IR energy captured by the sensor changes, the brightness of the pixels rendered by the camera changes to compensate. This prevents a too-hot scene from being washed-out, and helps identify objects emitting a relatively low level of IR radiation.

For the purposes of our experiments, we assume that the objects to be tracked will be emitting significantly more IR energy than their surroundings; because the camera's rendition of a scene depends upon contrast, we count on the fact that interesting objects will be rendered in bright white ( $r + g + b \geq 600$ ). We process the video stream frame-by-frame, and generate observations which the PQS tracks.

In processing a frame, we thus classify each pixel as being "in" or "out" — that is, part of an object worth tracking, or not — based on its brightness. Upon classifying pixels as "in" or "out", we process the frame with a standard Union Find algorithm that locates connected groups of "in" pixels, and computes the centroid (in screen coordinates) and approximate mass of the group (in pixels). It is these two pieces of information that comprise the observation that will be processed by the PQS:

$$Position = \{x, y, mass, t, f\}$$

Since PQS is a real-time tracking system, the time  $t$  when the observation occurred is reported as well. And for the process model's internal bookkeeping, the frame number  $f$  in the video stream is also reported.

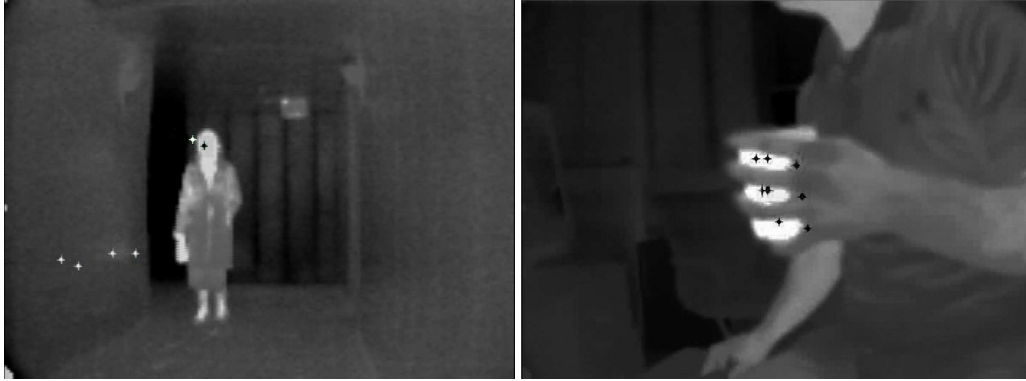
The PQS organizes the observations that are sent to it into tracks by maintaining a set of hypotheses. If observations  $A$  and  $B$  together comprise a potential track in the PQS, and observation  $C$  arrives some time  $t$  later, we can compute the likelihood of observation  $C$  being generated by the same object that generated  $A$  and  $B$  by computing (from  $A_{x,y,t}$  and  $B_{x,y,t}$ ) the instantaneous momentum of the object, and thus an estimate of where it should be; observation  $C$  "fits" into the current hypothesis if it is sufficiently close to our prediction of where the object should be, and is thus scored accordingly. The closer, the better. Of course, if we only have observation  $A$ , we know nothing of the object's direction or speed, and so we admit a large number of shorter tracks. These, however, are quickly pruned as more observations arrive and the system gathers more information about the scene. (Note the method described above is reminiscent of a simplified Kalman filter.<sup>5</sup>)

To compensate for the fact that objects may change direction or accelerate or decelerate very quickly relative to the sample rate of the sensor (in this case the standard 29.97 frames per second specified by NTSC), the algorithm that computes momentum weights observations based on their recency. In particular, when a new observation is added to the track  $t$  seconds after the last one and its centroid is  $p$  pixels from the last observation's centroid, the new momentum is

$$m = m_{prev} + C_m \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2},$$

where  $C_m$  is the dropoff factor, a tunable constant. In our experiments, we used  $C_m = 2.0$ . This essentially provides a filter for smoothing out sudden changes. In addition to tracking centroids, the PQS also uses its tracks to compute the "mass-momentum" of the object; that is, the instantaneous rate of change of the object's mass, as given by the number of continuous pixels. For the mass momentum,  $C_m = 30.0$  to account for the step factor in the Union Find algorithm.

The PQS reports the top hypotheses (the tracks most likely to actually correspond with real, moving objects), along with a snapshot of their state (including their momentum and mass-momentum) back to the video processing system, which then annotates the video image with tracking information as well as indicators for whether a tracked object is growing in size (a large, positive mass momentum) and whether it seems to have a momentum that will put it on a collision course with another point. From the perspective of a camera pointed at a 3D scene, this can tell us whether an object is merely milling around aimlessly, or seems to be purposefully moving toward either the camera—if it is growing in mass—or towards some other sensitive zone (as indicated by its momentum).



**Figure 2.** Two images of thermally distinctive objects being tracked by the PQS. The left image shows a person in a hallway walking with a bottle of hot water. The bottle is tracked below, while the face of the person is tracked separately. The person is walking backward. The right image shows a person holding a cup of coffee. Notice how the fingers obscure and segment the cup into three parts that are all tracked separately.



**Figure 3.** Two images showing identification of behaviors. The images are in time sequence, the left being before the right. The left-hand person walking away from the camera, while the right-hand person is approaching the camera. The track on the left-hand person is marked with small squares indicating departure, while the track on the right-hand person is marked with large squares to indicate approach.

## 4. RESULTS

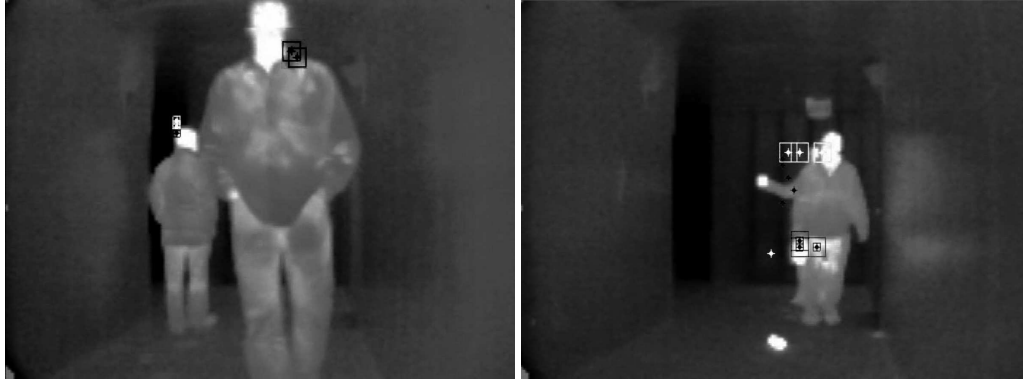
### 4.1. General discussion

We tested our models and processing pipeline by arranging the thermal camera to record a scene taking place in a hallway. The members of ISTS PQSNET lab and their coffee cups and water bottles were the test subjects. For each object count, we ran five trials of 30 seconds of free-form movement, including translation and rotation of the test subjects, moving in and out of the camera’s field of view, and test subjects obscuring one another. Ground truth for the trials was determined by humans by visual (and, in some cases, slow-motion) inspection. In most experiments either one or two persons were used.

### 4.2. Tracking Results

For tracking, we consider a track that corresponds to the movement of an actual object radiating IR energy a true positive, a track which does not correspond to an object of interest a false positive, and a track that ends while the same object is still in the experiment a false negative (for instance, when the view of an object is obscured and the tracker assigns it a new ID when it returns to view).

In measuring simple tracking results, our approach proved to differentiate and track the movements of objects very reliably even as they moved erratically, left the field of view, or obscured each other, to a point. The most notable tracking



**Figure 4.** The left image is a continuation of the time sequence of images from Figure 3. The right-hand person has now fully approached the camera and the PQS correctly indicates approach with the large squares. The right image shows two persons obscuring each other. Both are correctly tracked. They are moving around each other quickly and holding thermally distinctive objects. The system is correctly able to track persons that are obscured by other objects in the scene.

Number of Objects	True Positives	False Positives	False Negatives
1	12	2	3
2	23	2	11
3	9	0	4

**Table 1.** PQS tracking results for scenes with 1, 2 and 3 moving objects.

problem (causing virtually all of the false negatives) was the displaying of single-observation tracks. This meant that spurious observations that could not be correlated to existing tracked objects, or possibly new objects, were displayed regardless. An easy fix for this problem is not to display tracking information until an object has been positively identified by the system. Several other false negatives were due to an overly aggressive mass threshold. This meant that smaller objects were easily lost due to their relatively insignificant size in the overall image. This is a tunable parameter.

### 4.3. Approach Behavior Results

Number of Objects	True Positives	False Positives	False Negatives
1	15	1	1
2	20	4	7
3	3	0	2

**Table 2.** Approach behavior detection for 1, 2, and 3 approaching (or departing) objects in a scene.

For measuring the classification of movement toward or away from the camera, we assign true positives when a significant object is correctly detected moving towards the camera, or correctly detected moving away from the camera. A false negative is noted when an object that was moving away is identified as moving closer, or when an approaching object is identified as moving away. A false negative is also considered the failure of the system to detect an obvious motion towards or away from the camera. Finally, a false positive is noted if a static object is identified as moving closer or farther away.

The overall performance of this behavioral detection is amazingly robust, although very fragmented. For instance, an object moving closer to the camera is identified quickly, then tracked normally for awhile, before it is once again identified as approaching the camera. This fragmentation can be easily filtered by smoothing the frame-to-frame conclusions. False negatives were mostly due to an object being identified as approaching, then for a brief second as departing, and then continuously as approaching again, and vice versa. A smoothing filter will remove those false negatives in the future also.

#### 4.4. Future work

Challenges facing tracking involve missed observations and false observations. Beyond the “in” or “out” and Union Find classification schema, our system makes no effort to determine the likelihood of whether a mass of brightly-colored pixels actually represents a bona-fide target. By performing more advanced video preprocessing to assign such a likelihood to each observation based on its infrared signature, the system will be better able to differentiate different types of targets and be more robust against false positives; for instance, the performance of a person-tracking system may be improved by only tracking person-shaped masses.

Experiments with large numbers of artificial objects have shown significant promise in the ability to track many objects at once ( $\geq 20$ ). To further the relevance of this application we will focus on tracking larger numbers of actual people and vehicles, as well as tuning parameters specifically for use with those types of objects.

Finally, an important improvement to the system is the ability to detect more complex behaviors of objects. Currently the system is able to identify approaching and departing behaviors; however, it is a trivial extension, using a PQS, to identify behaviors such as convergence on a central point, formation of a group, unit, or team, and obscuring and re-emerging of objects. The value of the system would be greatly benefitted by an extended range of behavioral models. Additionally, the output from the current system can be used to identify even more complex behaviors still. For instance, when a group of individuals is identified quickly departing from a central location, this may be classified as fleeing behavior and should focus the attention on the central location that the persons are fleeing from.

#### REFERENCES

1. V. Berk, W. Chung, V. Crespi, G. Cybenko, R. Gray, D. Hernando, G. Jiang, H. Li, and Y. Sheng, “Process query systems for surveillance and awareness,” in *Proceedings of the 7th World Multifconference on Systems, Cybernetics and Informatics (SCI 2003)*, (Orlando, Florida), July 2003.
2. G. Cybenko, V. H. Berk, V. Crespi, R. S. Gray, and G. Jiang, “An overview of process query systems,” in *Proceedings of SPIE Vol. 5403 Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, Orlando, Florida, April 2004.
3. R. E. V. D. Yiwei Wang, John F. Doherty, “Moving object tracking in video,” *29th Applied Imagery Pattern Recognition Workshop (AIPR’00)* **123**, p. 95, 2000.
4. D. B. Reid, “An algorithm for tracking multiple targets,” *IEEE Transactions on Automatic Control* **AC-24**, pp. 843–854, December 1979.
5. R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME—Journal of Basic Engineering* **82**(Series D), pp. 35–45, 1960.
6. C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic, 1999. ISBN 0-7923-8609-4.