Object Tracking in Video Sequences

Johns Hopkins University

Image Compression / Packet Video / Video Processing

Chris Shirk

cshirk@ieee.org

Abstract

Object tracking in video sequences increases the effectiveness of existing video surveillance networks, many of which have no statistically significant impact on crime.

Problem Statement

Rapid technological progress in concert with recent world events have given rise to the sudden spike in the adoption of surveillance systems. The United Kingdom has installed over 1.5 million cameras in response to terrorist bombings [4]. As an outgrowth of the 9/11 attacks, the Washington DC Metropolitan Police Department operates a network of almost a thousand cameras through the Joint Operations Command Center, though official MPD numbers identify a mere 16 cameras across 15 sites [5]. However, the explosion of CCTV public surveillance has not been accompanied by a statistically significant reduction in crime [6], and has otherwise had its value proposition thrown into question. It is the purpose of this paper to suggest object tracking in video sequences as a method to increase value in existing unintelligent CCTV installations.

Objective

The objective of this paper is two-fold: highlight the questionable value of existing CCTV public surveillance networks, and detail a method for object tracking in video sequences.

Approach

In order to question the value of existing CCTV public surveillance networks, we will briefly examine various studies that question their effectiveness. Then, we'll review an algorithm that provides the beginnings for actionable intelligence from video, and ultimately suggest that object tracking be utilized for existing surveillance networks to help sift through the torrents of data and reduce the amount of video passed on to analysts.

Analysis

CCTV public surveillance networks can be characterized by one main problem: it is impossible for even a skilled observer to provide full attention to more than two things at a time [1]. Thus, considering that each scene can have multiple objects of interest, and the scene / observer ratio is usually high, an analyst's ability to perform effective real-time security is thrown into question. An example of a classic CCTV command center is shown below – note the high scene / observer ratio (screens / potential chairs).

An August 2002 meta-study by the British Home Office reviewed 46 studies, including five from the United States, that analyzed the effectiveness of surveillance in fighting crime [6]. For every CCTV public surveillance network in the United States, no desirable impact on crime could be measured.

One major problem is that there is simply too much raw video for analysts to observe to provide real-time intelligence that can be translated to action on the ground. Additionally, for crimes that have already occurred, gathering information – such as what people a suspect came into close proximity with for a given week across a dozen security cameras – can be incredibly laborious. Searching through vast archives of surveillance video could be vastly aided by additional video metadata. It is the purpose of object tracking to provide some of this metadata.

Basic object tracking has numerous intelligence applications, including counting the number of people in a scene, tracking a given subject and anyone he comes into close proximity with, and tracking collisions between vehicles. The algorithm we will review has four main stages: frame wavelet transform, camera motion normalization, object isolation, and tracking conflict resolution.

To effectively track objects, camera motion must be factored out from the video sequence. The projective model can describe, and thus factor out, translation, rotation, zooming, panning and tilting.

The equation for the projective model follows [3]:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Simple matrix multiplication yields:

$u = m1 * x + m2 * y + m3 * 1$

$v = m4 * x + m5 * y + m6 * 1$

$w = m7 * x + m8 * y + 1 * 1$

Given that x' = u / w, y' = v / w, it follows that:

$$x' = \frac{m_1 x + m_2 y + m_3}{m_7 x + m_8 y + 1}$$

$$y' = \frac{m_4 x + m_5 y + m_6}{m_7 x + m_8 y + 1}$$

A less computationally intense model that approximates the projective model is the bilinear model. The equation for the bilinear model follows [3]:

$$x' = q_1 xy + q_2 x + q_3 y + q_4,$$

$$y' = q_5 xy + q_6 x + q_7 y + q_8;$$

Starting with the 2-D optical flow equation [3],

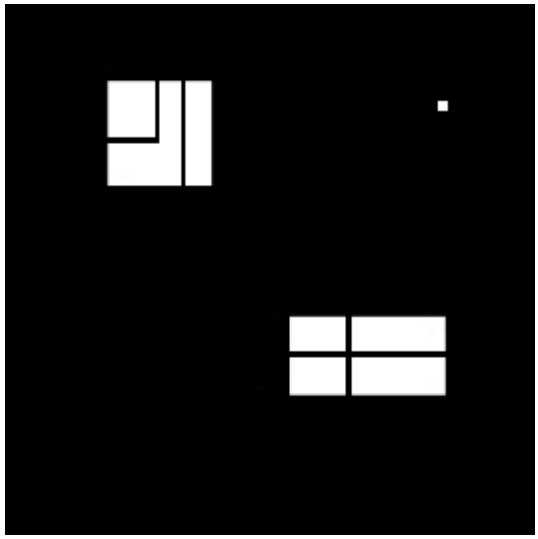
$$u_f E_x + v_f E_y + E_t \approx 0$$

where u_f and u_y are the x and y velocities, and E_x, E_y and E_t are the partial derivatives of the gray scale values with respect to x, y, and t, we can obtain the parameters for the bilinear model by minimizing the following equation [3]:

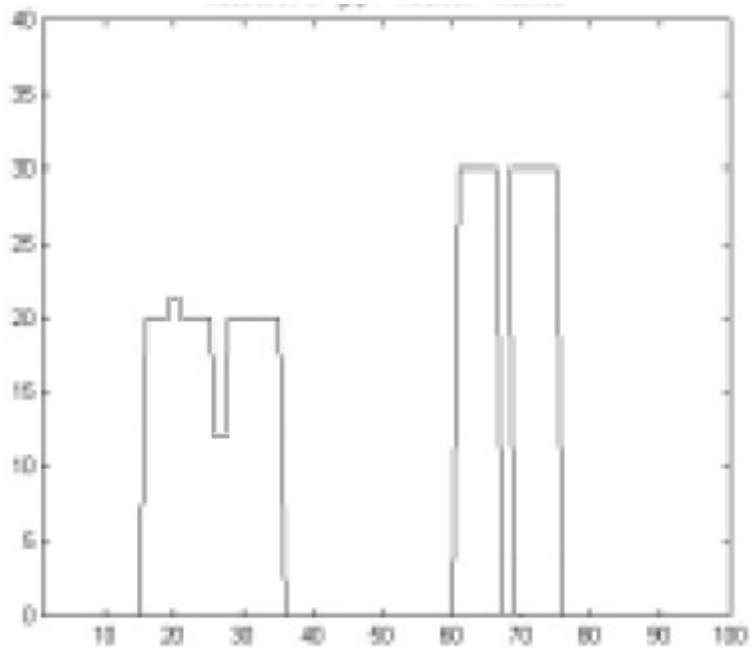
$$\epsilon = \sum_{(x,y)} (u_m E_x + v_m E_y + E_t)^2$$

Once we have solved for the parameters of the bilinear model, we can proceed to object isolation. Object isolation analyzes two frames, which we call the current frame and the reference frame. It employs the wavelet transform to extract edge information from the frames, and segments the current frame into background and object regions by thresholding the edges for camera motion. Effectively, within a given tolerance, edges moving more than the bilinear model would predict correspond to object motion. This procedure produces object motion blocks of various sizes which may correspond to noise, or even the same object. In order to resolve this, an approach called the “gap/mountain” method [2] is utilized.

For example, assume the following image represents a 100x100 pixel image that contains what appears to be eight objects. Background pixels – everything outside of the object motion blocks as determined by thresholding the edges from the wavelet transform – have a value of zero and are shown as black. Object pixels – which is everything else – are given a value of one and are shown as white.

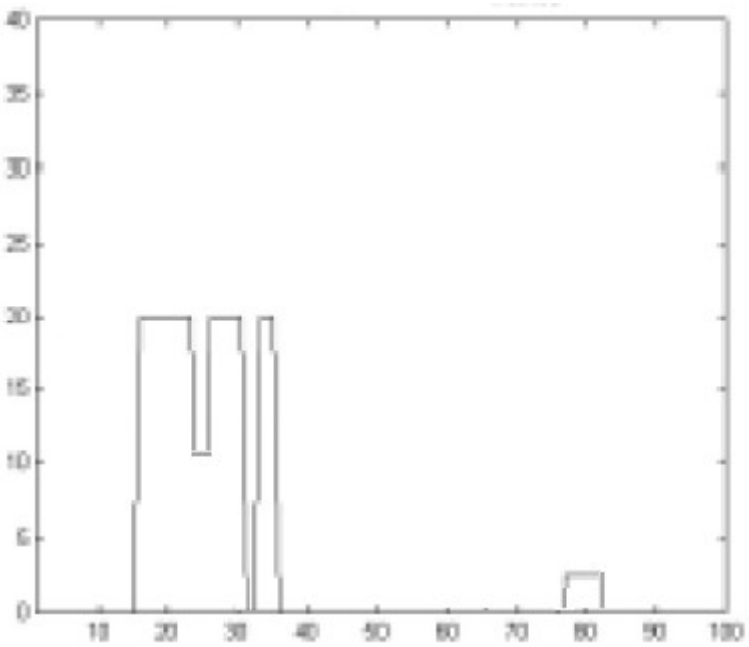


The “gap/mountain” algorithm is configured with what we assume to be intelligent but for our purposes arbitrary thresholds for gaps and mountains. In this example, we use a value of 10 pixels for both. Initially, the algorithm scans the rows, adding up object pixels, from which we achieve the following result:



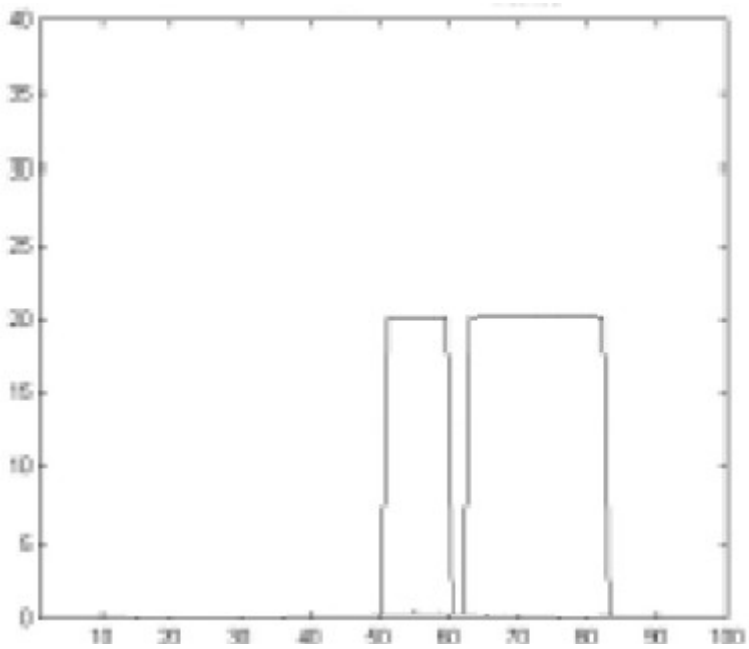
From the result we can determine there are two mountains and three gaps (the tiny space at rows 68-69 is less than the minimum gap distance, and is not counted). We divide this result into two sub-regions that each contain a mountain, and repeat the procedure on the columns.

Sub-region 1:



From sub-region 1 we can determine there are two gaps and one mountain. Specifically, the block of pixels around column 80 is less than the minimum mountain distance, and is ignored. Additionally, the small breaks in the main object are less than the minimum gap distance. At this point, we have narrowed down sub-region 1 to one object. After performing one more pass (switching back to the rows) to ensure that sub-region 1 only contains one object, we can record the location of the object.

Sub-region 2:



From sub-region 2 we can determine there are two gaps and one mountain. Specifically, the break in the main object is too small to be considered a gap. Before recording the object location, we perform an additional pass – switching back to the rows. Thus, from sub-region 2, we have extracted the location of an object.

In order to obtain the true locations for the objects in the current frame, we apply the "gap/mountain" algorithm to a second reference frame. If the original reference frame occurred earlier than the current frame, we select a new reference frame that occurs later. After running the "gap/mountain" algorithm on the current frame and the new reference frame, we obtain a second set of object locations. The true object locations in the current frame are the overlapping regions from the two sets of locations.

Once we have achieved the true object locations for any given frame, we can derive track information for the current objects. Track information simply allows us to predict the location of an object in subsequent (future) frames. Effectively, from three frames we solve for velocity and acceleration for the equation $S = vt + \frac{1}{2} at^2$.

Aside from the first three frames of a video sequence, for any given frame, we have m objects and n existing tracks that might extend to the objects. Tracking objects becomes difficult when objects stop moving, new objects start moving, objects collide or simply become occluded by features in the scene. In order to reliably track objects in these complex situations, we must first establish a set of tracking variables to identify objects and resolve the potential conflicts.

Given an acceptable frame-rate, object size should not change significantly between frames. One tracking variable in our set of criteria that tracks size is called dispersion, and is calculated by the equation below [3]:

$$disp = \left(\sum_{(i,j) \in \mathbf{O}} \sqrt{(i - c_x)^2 + (j - c_y)^2} \cdot p_{i,j} \right) / \left(\sum_{(i,j) \in \mathbf{O}} p_{i,j} \right)$$

In this equation, c_x and c_y represent the object centroid, and $p_{i,j}$ represents the value of the edge image at location *i, j*. The object centroid can be calculated from the following equations [3]:

$$c_x = \left(\sum_{(i,j) \in \mathbf{O}} p_{i,j} \cdot i \right) / \left(\sum_{(i,j) \in \mathbf{O}} p_{i,j} \right)$$

$$c_y = \left(\sum_{(i,j) \in \mathbf{O}} p_{i,j} \cdot j \right) / \left(\sum_{(i,j) \in \mathbf{O}} p_{i,j} \right)$$

Assuming that scene lighting remains relatively constant, another object aspect that does not change appreciably between consecutive frames is grayscale composition. We employ three measures of grayscale composition – the mean of all object pixels, the mean of 10% of the most intense (i.e. high valued) pixels, and the mean of 10% of the least intense pixels. The rationale behind three sets of variables for grayscale composition tracking is that should the object become occluded or otherwise modified in the current frame with respect to a reference frame, it will most likely maintain one or more of the grayscale composition variables within a specified tolerance. It may, for example, have a radically changed mean of 10% of the most intense pixels, and a slightly changed mean of all object pixels, but we will still be able to track the mean of 10% of the least intense pixels. By itself, this would not be sufficient, but in concert with the other tracking variables, it allows us to select the best-fit matches in each frame between *n* existing tracks and *m* current objects.

The final tracking variable is based on object texture. Grayscale distributions across objects are usually different, which allows us another method to distinguish objects from each other, and thus determine which tracks extend to which objects. The texture variable t_x is obtained from the wavelet transformed edge image [3], and is the mean of 10% of the pixels with the largest value, which roughly translates into texture.

For a given frame, each object and each track is associated with the five tracking variables – dispersion, mean grayscale value, mean of 10% of the most intense grayscale values, mean of 10% of the least intense grayscale values, and object texture. We extract the maximum analytic value from the video when we extend existing tracks to the appropriate objects. However, as mentioned previously, new tracks, dead tracks, and object collisions complicate this process.

Given that we have five tracking variables, we only consider extending tracks to objects that have at least four tracking variables in common, within some tolerance. Given *m* objects and *n* sets of tracks, we might have an *m x n* matrix like the following, where each element represents the number of matching variables between the object and track.

$$\begin{bmatrix} 5 & 3 & 0 & 2 & 2 \\ 2 & 4 & 3 & 5 & 1 \\ 3 & 0 & 5 & 1 & 4 \\ 0 & 2 & 2 & 4 & 3 \\ 2 & 3 & 4 & 0 & 4 \\ 0 & 2 & 0 & 0 & 3 \end{bmatrix}$$

In this matrix, we have six objects and five existing tracks. Any element greater than three denotes a possible track extension. As it is obvious that there won't always be a 1-1 correspondence between tracks and objects, we eliminate the trivial cases for which an element is the only candidate in its row and column for a track extension before delving into the problem of determining the best-bit given multiple options. For this matrix, element *(1,1)* is the only value that exceeds three in its row and column, so we eliminate row 1 and column 1 from the matrix.

$$\begin{bmatrix} \cancel{5} & \cancel{3} & \cancel{0} & \cancel{2} & \cancel{2} \\ \cancel{2} & 4 & 3 & 5 & 1 \\ \cancel{3} & 0 & 5 & 1 & 4 \\ \cancel{0} & 2 & 2 & 4 & 3 \\ \cancel{2} & 3 & 4 & 0 & 4 \\ \cancel{0} & 2 & 0 & 0 & 3 \end{bmatrix}$$

Element *(4,4)* of the original matrix is the only element in its row and column that is a possible track extension, so we eliminate row 4 and column 4 from the matrix.

$$\begin{bmatrix} \cancel{5} & \cancel{3} & \cancel{0} & \cancel{2} & \cancel{2} \\ \cancel{2} & 4 & 3 & 5 & 1 \\ \cancel{3} & 0 & 5 & 1 & 4 \\ \cancel{0} & \cancel{2} & \cancel{2} & \cancel{4} & \cancel{3} \\ \cancel{2} & 3 & 4 & 0 & 4 \\ \cancel{0} & 2 & 0 & 0 & 3 \end{bmatrix}$$

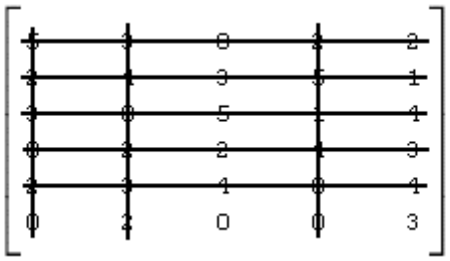
Similarly, element *(2,2)* of the original matrix is the only element in its row and column that is a possible track extension, so we eliminate row 2 and column 2 from the matrix.

$$\begin{bmatrix} \cancel{5} & \cancel{3} & \cancel{0} & \cancel{2} & \cancel{2} \\ \cancel{2} & \cancel{4} & \cancel{3} & \cancel{5} & \cancel{1} \\ \cancel{3} & 0 & 5 & 1 & 4 \\ \cancel{0} & \cancel{2} & \cancel{2} & \cancel{4} & \cancel{3} \\ \cancel{2} & 3 & 4 & 0 & 4 \\ \cancel{0} & 2 & 0 & 0 & 3 \end{bmatrix}$$

As no further trivial cases exist, we must use a weighted cost function to determine whether to extend any of the two remaining tracks to the three remaining objects. The weighted cost *dif* ties together the five object tracking variables, comparing the frame *(f)* to its track *(t)* counterpart. The equation is shown below [3]:

$$\begin{aligned} dif &= w_{tr}(|c_x^f - c_x^t| + |c_y^f - c_y^t|) \\ &+ w_{disp}(|disp^f - disp^t|) \\ &+ w_{gr}(|gr_l^f - gr_l^t| + |gr_m^f - gr_m^t| + |gr_h^f - gr_h^t|) \\ &+ w_{tx}(|tx^f - tx^t|), \end{aligned}$$

The constants w_{tr}, w_{disp}, w_{gr}, w_{tx} reflect the weights for the object location (centroid), object dispersion, object grayscale composition, and object texture, respectively. Thus, for our two remaining tracks, the object that has the smallest *dif* is selected for track extension. In our example, assuming that $dif_{5,3} < dif_{3,3}$ and $dif_{3,5} < dif_{5,5}$, the third track is extended to the fifth object and the fifth track is extended to the third object. The remaining matrix is shown below:



The sixth object has no corresponding track, and thus represents a new track. Through this procedure we can extend existing tracks to objects for any frame of the video sequence, and track objects as they move throughout the camera's field of vision.

Conclusion

As evidenced by the meta-study by the British Home Office, surveillance networks in the United States do not have a desirable impact on crime reduction. There is simply too much raw video for analysts to observe to provide real-time intelligence that can be translated to action on the ground. Object tracking could offload some of the work and reduce the amount of video sent to analysts. For example, through object tracking, we could cull a large set of video streams for potential robberies by only displaying scenes with two or more moving objects in close proximity in which one object stops moving and the other continues to approach and then flees at a higher velocity. Or, we could detect loitering by tracking when multiple objects cluster together for more than a specified duration. Ultimately, object tracking can enrich the information we have on a scene and allow us to expend the most expensive resource – the attention and time of an analyst – on information that's most relevant to the task at hand.

References

[1] A.J. Lipton. “Critical Asset Protection, Perimeter Monitoring, and Threat Detection Using Automated Video Surveillance“

http://www.objectvideo.com/objects/pdf/products/onboard/OV_WP_IVS.pdf

[2] Yiwei Wang. “Moving Object Tracking in Video”

http://w3.antd.nist.gov/pubs/aipr00.pdf

[3] Yiwei Wang. “Tracking Moving Objects In Video Sequences”

http://w3.antd.nist.gov/pubs/ciss00.pdf

[4] “EPIC Video Surveillance Information Page” Oct 2006;

http://www.epic.org/privacy/surveillance/

[5] “Police Surveillance Cameras, Washington, DC”

http://www.notbored.org/dc-police.html

[6] B.C. Welsh. “Crime prevention effects of closed circuit television: a systematic review”

http://www.homeoffice.gov.uk/rds/pdfs2/hors252.pdf