Multiple Target Tracking with Lazy Background Subtraction and Connected Components Analysis

Robert G. Abbott
Sandia National Labs
MS 1188
PO Box 5800
Albuquerque, NM 87185, USA
rgabbot@sandia.gov

Lance R. Williams
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131, USA
williams@cs.unm.edu

Abstract

Background subtraction, binary morphology, and connected components analysis are the first processing steps in many vision-based tracking applications. Although background subtraction has been the subject of much research, it is typically treated as stand-alone process, dissociated from the subsequent phases of object recognition and tracking. This paper presents a method for decreasing computational cost in visual tracking systems by using track state estimates to direct and constrain image segmentation via background subtraction and connected components analysis. We also present a multiple target tracking application which uses the technique to achieve a large reduction in computation costs.

1. Introduction and Background

Background subtraction, binary morphology, and connected components analysis are the first processing steps in many vision-based tracking applications. [16, 13, 9, 19, 8, 17] Segmentation by background subtraction is a useful technique for tracking objects that move frequently against a relatively static background. Although the background changes relatively slowly, it is usually not entirely static. Illumination changes and slight camera movements necessitate updating the background model over time, making background modeling an major consumer of computational resources in a tracking system.

This paper presents Lazy Background Subtraction and Connected Components Analysis (LBSCCA), a method for decreasing computational cost in tracking systems by using track state predictions to direct and constrain image segmentation via background subtraction and connected components analysis.

Some other recent research has incorporated top-down control of background subtraction. Harville [10] used high-level feedback to locally adjust sensitivity to background variation using application-specific high level modules. In this paper we offer a method to entirely avoid modeling the background outside regions of interest.

Various techniques for background subtraction have been explored including temporal differencing [13], median filtering [5], and mixture of Gaussians [7]. Each algorithm is a tradeoff between

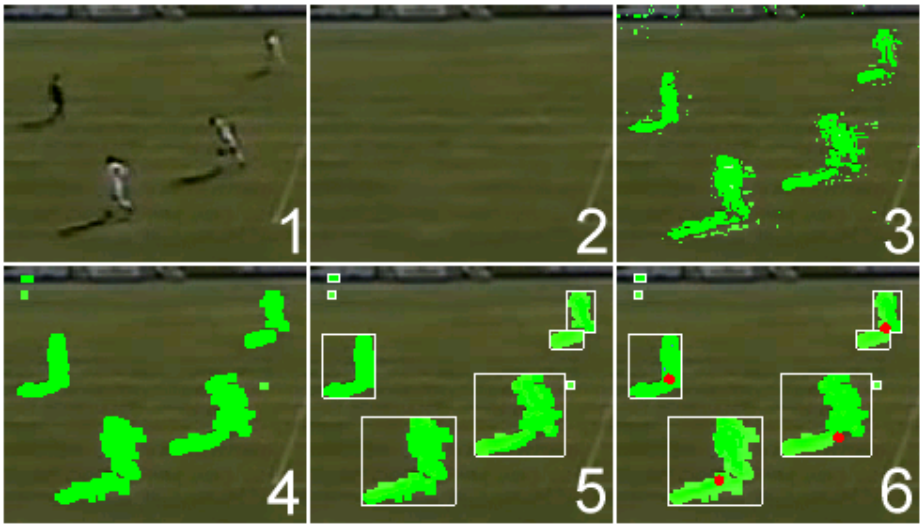


Figure 1. An illustration of video processing steps in a tracking application. The image shows 4 players in a game of soccer. 1) Original image. 2) Background image. 3) Foreground pixel mask (bright green). 4) Binary morphology. 5) Connected components analysis (each blob is bounded by a white rectangle). 6) Feature extraction (red dots show ground contact estimates). Section 3 gives more detail on our implementation of each stage.

effectiveness and computational cost. More efficient application of background subtraction will permit the use of more accurate but computationally costly background models.

1.1. Selective Attention in Computer Vision

Computational models of human selective visual attention [11, 15, 18] are grounded in biological attention mechanisms which allocate the limited processing resources of the brain. The models direct attention to highly salient low-level image features. Recent research is also beginning to propose models for top-down selective attention driven by cognitive processes. We hypothesize that some form of predictive state estimator analogous to the Kalman filter used in our implementation is necessary to effectively allocate attention for moving objects.

Most of the research on computational models of selective attention has focused on predicting which image regions are of interest to humans. There have been fewer attempts to apply selective attention to improve in the performance of computer vision applications. Selective attention has been used to improve object class learning in cluttered contexts [6] and to suppress extraneous features for more robust recognition [20]. However, little if any research has investigated using selective attention to improve throughput for an overwhelming flow of high-resolution visual information. In this respect, LBSCCA offers a new application for models of selective attention.

2. Lazy Background Subtraction and Connected Components Analysis

This section describes LBSCCA. We will first present the unoptimized, bottom-up process (Figure 1) which is similar to that used by many tracking applications. We will then present the top-down LBSCCA algorithm and show that it preserves the semantics of the full bottom-up approach by performing the same steps on selected portions of each frame.

First, coarse adjustments are applied to the incoming video stream (Figure 1.1). Examples include

frame rate reduction and cropping to exclude irrelevant pixels.

Second, the frame is used to update the background model. The specifics of this step depend on the choice of background modeling algorithm. In the case of median filtering, each pixel of the new frame as added to a queue representing the most recent n pixel values at that location. The queue is copied, sorted, and the median value selected. The array of median values is an image of the background with all foreground objects removed (Figure 1.2), assuming the background color is the most prevalent over the n previous frames.

Third, each pixel is labeled as foreground or background. One simple approach is to compute the difference between the current image and the background model at each pixel. A threshold is applied to the array of differences to create a binary foreground mask (Figure 1.3).

Fourth, we apply binary morphological operations to the foreground mask (Figure 1.4). Erosion removes small connected components assumed to be noise, while dilation increases the likelihood that objects will be represented by single blobs.

Fifth, we apply connected components analysis to the foreground mask to find contiguous regions of pixels, or "blobs" (Figure 1.5). Ideally each blob corresponds to a single tracked object, but in practice, blobs may appear spuriously or go undetected due to observation noise. Furthermore, occlusion can cause two or more tracks to appear as a single blob.

Sixth, we extract features from each blob. The features include a point location (Figure 1.6), which may simply be the mean position of pixels in the blob. Other features such as hue or texture may also be extracted to improve tracking robustness.

The six image processing steps produce a set of observations. Each observation is a possible track location. The observations serve as input for the data association and state estimation algorithms. Finally, the state estimates are used to predict subsequent observations.

Lazy Background Subtraction and Connected Components Analysis performs the same steps, but in reverse order. State prediction directs attention to regions of interest. Image processing techniques are applied only to pixels within the regions of interest in order to generate observations. Finally, the observations and state predictions are used for data association and state estimation. Figure 2 contrasts the top-down and bottom-up approaches. Here we describe the steps LBSCCA in top-down order.

To use LBSCCA we first use state prediction to allocate attention to regions of interest. Regions of interest define the starting points for detailed analysis. The analysis will extract observations from the pixel data using image processing techniques. Our tracking application uses the Kalman Filter to define regions of interest (Section 3.1).

The next step in top-down order is feature extraction, which generates observations. Each observation consists of a point location (illustrated by the red dots in Figure 1.6), and potentially other information such as hue. Features are extracted from blobs (bright green in Figure 1).

In order to find the blobs, LBSCCA performs segmentation by connected components analysis of foreground pixels. The search for blobs is limited to regions of interest. The connected components analysis only extends beyond the region of interest if it finds a blob which is partially within the region. Blobs entirely outside the region of interest are not discovered.

However, connected components analysis must respect binary morphology. Dilation can form larger connected components from smaller disconnected components. The algorithm must not neglect pixels which *would* have been in the connected component, had binary morphology been carried out first, as in the bottom-up process (see Figure 3).

LBSCCA solves this problem with tentative dilation. During connected component analysis, a neigh-

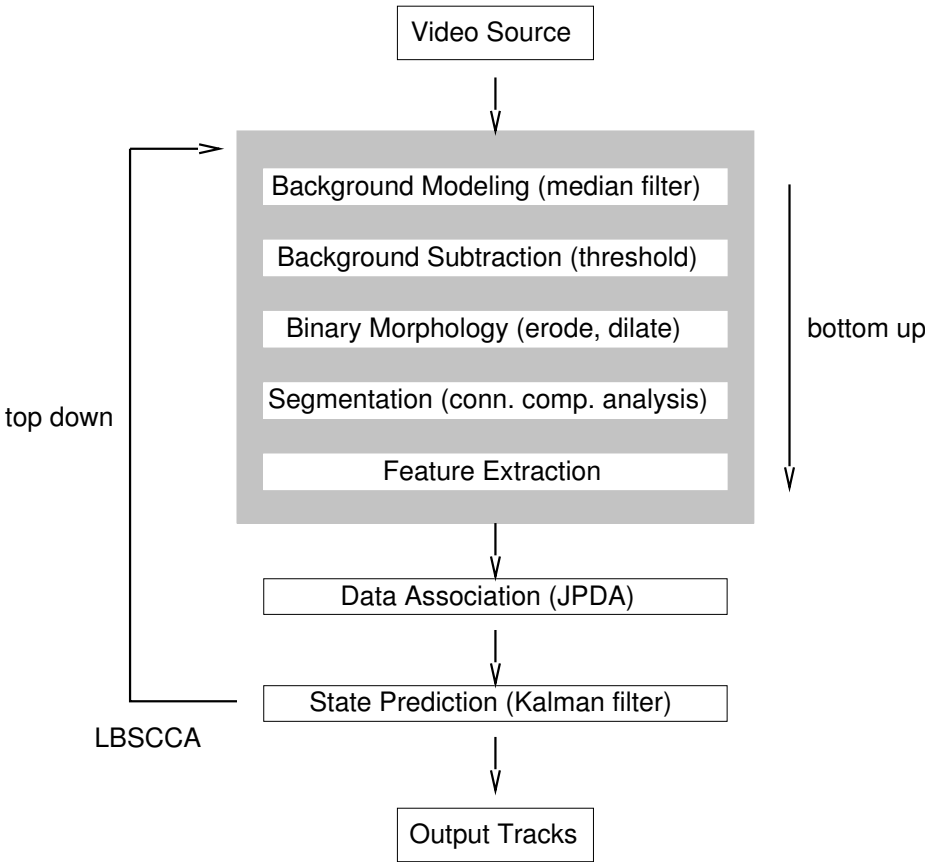


Figure 2. Schematic of a vision-based tracking system. Image processing stages, in gray, extract observations for the high-level data association and state estimation algorithms. In the conventional approach, each frame proceeds up through the stages. In top-down LBSCCA approach, the steps are reversed and control comes from the top down; based on state prediction, LBSCCA triggers per-pixel processing of regions of interest to extract observations for the tracking algorithm.

boring pixel p is added to the foreground mask if p is a foreground pixels, *or* if the dilation structuring element at p covers any foreground pixel. In the foreground mask, foreground pixels are denoted by the value 2, whereas pixels contained only in the dilation are denoted by 1. If a "close" operation (dilate followed by erode) is desired, we accept the tentative dilation and erode the mask, making no distinction between the values 1 and 2. If an "open" operation (erode followed by dilate) is desired, as in Figure 3, then the tentative dilation is discarded by thresholding the mask at a value of 2 to preserve only foreground pixels. We then erode and dilate the mask. The tentative dilation ensures that all pixels which will be connected to the component after the "open" operation are discovered during the connected component analysis. Finally, we must perform a second connected components analysis on the completed mask, since erosion may have disconnected one or more components.

The binary morphology operations must distinguish between foreground and background pixels, which implies background subtraction and an underlying background model. In LBSCCA the background model for a pixel is not updated until the top-down process needs to determine whether a pixel is foreground. In addition to the per-pixel background model $\mathbf{B}$, we maintain a background model state matrix $\mathbf{F}$ of the same dimension. Each element $\mathbf{F}_{ij}$ specifies the frame for which $\mathbf{B}_{ij}$ is valid. $\mathbf{F}$ is

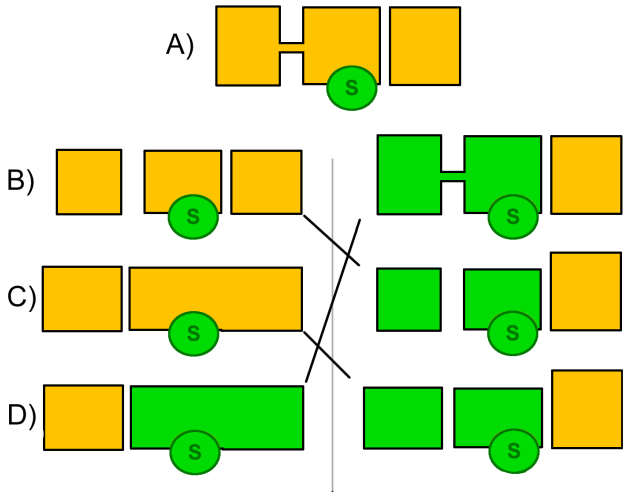


Figure 3. Binary morphology interacts with connected components analysis. In the left-hand column, the original image (A) undergoes vertical erosion (B), horizontal dilation (C), and connected components analysis (D) from search region S. In the right-hand column, connected components analysis is performed first to avoid full-image processing, and binary morphology is only performed on the initial connected component. The result is incorrect because it does not match (D); furthermore, it is not a connected component.

initialized only once, at the beginning of program execution, to some invalid frame index. Whenever a the background model for pixel p_{ij} is needed, we first consult $\mathbf{F}_{ij}$ to determine whether $\mathbf{B}_{ij}$ is current, and if necessary update $\mathbf{B}_{ij}$ and $\mathbf{F}_{ij}$. In this way, background modeling is constrained to the blobs found in the regions of interest.

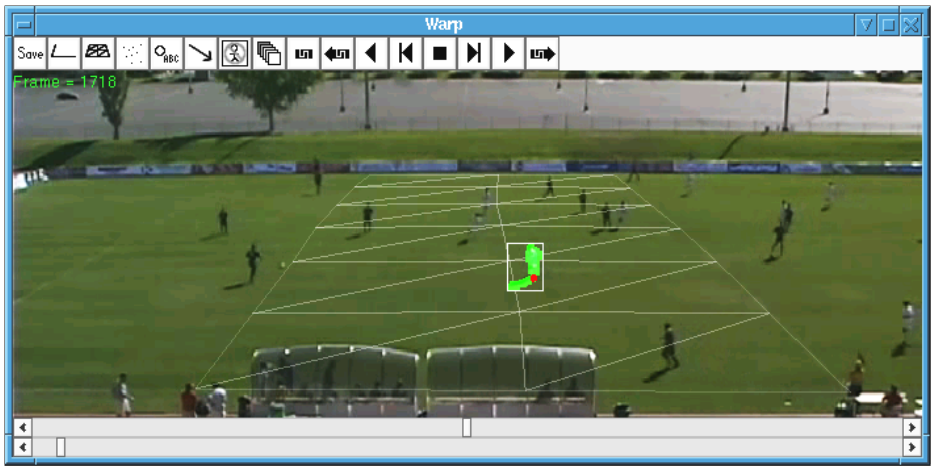
The result of this top-down process is equivalent to full-frame background subtraction and connected components analysis for the purposes of a data association and tracking algorithm which uses a constrained search region for each track. We now present such a system.

3. The MuTTSA System

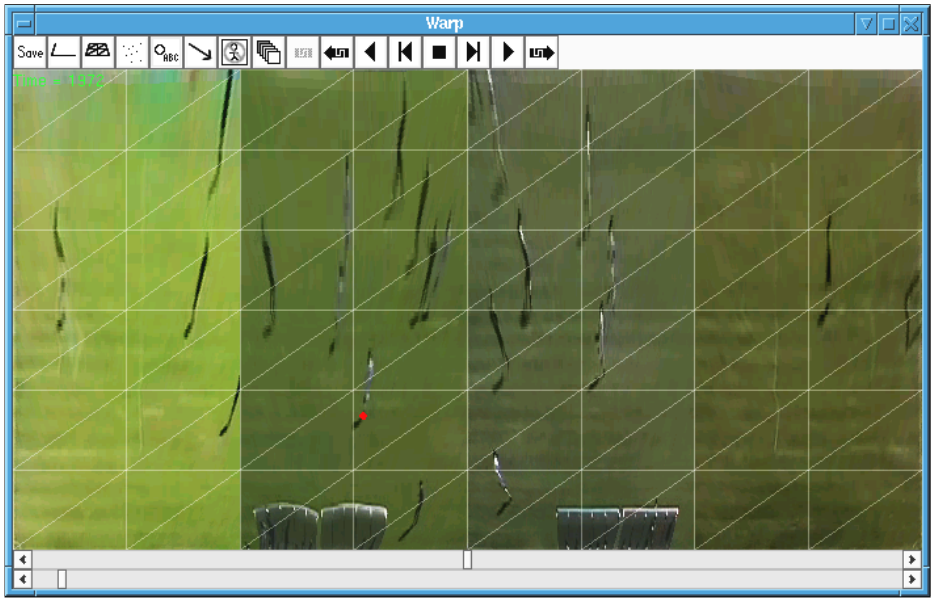
In this section we present the Multiple Target Tracker with Selective Attention (MuTTSA). This system implements LBSCCA and provides a platform for validating the lazy approach and quantifying the reduction in computational cost.

MuTTSA is a computer vision application for simultaneous tracking of multiple targets. It has been used to track players in soccer games. MuTTSA is a purely visual system, and the targets (soccer players) are not instrumented with any special optical targets or transmitters. The size of the soccer pitch is 231 by 218 feet. Due to the large tracking area, four cameras are used to obtain footage with adequate resolution to track all players simultaneously (Figure 3). After cropping and frame decimation the resolution of the combined video stream is 2720×240 pixels at 10 Hz. Although MuTTSA is an off-line tool, near real-time performance makes the data collection task more convenient, and provides the possibility of on-line operation in the future.

Because MuTTSA implements LBSCCA, our presentation of MuTTSA will follow top top-down order of processing steps shown in Figure 2. First, state prediction directs attention to regions of interest. Next, pixels within the regions are processed to generate observations. Finally, the observations drive data association and state estimation.



(a) Soccer footage from a single camera, 720 x 240 pixels. The system is tracking a single player. The player's estimated position is the red dot.



(b) Soccer footage from four cameras projected into a single coordinate system to simulate a bird's eye view. Note the red dot at the feet of the player track.

Figure 4. MuTTSA merges footage from multiple cameras into a synthesized viewpoint in a single world coordinate system. Image processing steps are performed on the original footage, but the resulting observations are projected into the world coordinate system for data association and tracking.

3.1. State Prediction

MuTTSA's tracking algorithm is Joint Probability Data Association (JPDA) with Kalman Filtering (KF). Both algorithms are recursive; each state estimate $\mathbf{x}_t = F(\mathbf{Y}_t, \mathbf{x}_{t-1})$ is a function of both the

current observations $\mathbf{Y}_t$ and the predicted state, which is a linear function of the previous state estimate. In MuTTSA the user provides the initial state $\mathbf{x}_0$ by interactively creating a track for each player. After initialization the tracks are automatically updated by the tracking system, but the user can intervene to correct tracking errors.

Space does not permit full derivation of the KF and JPDA in this paper. Rather, we will describe our application of KF and JPDA within MuTTSA. For introductions and overviews refer to Welch and Bishop [21] for KF, and Oh and Sastry [14] for JPDA. Refer to Bar-Shalom and Fortmann [2] for more detail about both KF and JPDA.

Kalman Filtering has two steps: predicting the current state based on the previous state, and using the current observations to update the prediction. LBSCCA uses the prediction to constrain the gathering of observations. Instead of scanning the entire visual field, we analyze only the portions where we expect to observe tracked objects. Top-down control extends down through connected components analysis and background subtraction.

In MuTTSA the state estimate for each track k consists of position and velocity. The a priori Kalman Filter prediction for the state of track k at time t is $\hat{\mathbf{x}}_t^k = \mathbf{A}\mathbf{x}_{t-1}^k$, specifically

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\dot{x}} \\ \hat{\dot{y}} \end{bmatrix}_t^k = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_t^k . \tag{1}$$

Kalman filtering also provides a principled estimate of the error covariance $\mathbf{P}$. Error covariance temporarily grows for tracks when their motion departs from the linear model $\mathbf{A}$ and when observations are noisy or missing. MuTTSA uses the error covariance to determine the size of the region of interest around each track. JPDA use the same technique for its observation validation step, discussed in Section 3.3. The a priori estimate of error covariance $\mathbf{P}$ at time t is

$$\hat{\mathbf{P}}_t^k = \mathbf{A}\mathbf{P}_{t-1}^k\mathbf{A}^{\mathrm{T}} + \mathbf{Q} \tag{2}$$

where $\mathbf{Q}$ is the variance of the process noise. In MuTTSA all nonlinear motion (acceleration) of the players is modeled as process noise. We use diagonal matrices $\hat{\mathbf{P}}_0^k = \mathrm{diag}(1, 1, 0.5, 0.5)$ and $\mathbf{Q} = \mathrm{diag}(2, 2, 1, 1)$. In both $\mathbf{P}$ and $\mathbf{Q}$ the first two values are in units of feet, and the second two values are in units of feet per 0.1s because the frame rate is 10Hz.

3.2. Pixel Processing Steps

In MuTTSA, the primary feature of a blob is its point location. This must correspond to the (x, y) location of a player on the ground plane. Given our cameras' oblique viewing angle and the inclusion of shadows in the extracted blobs, a simple measure such as the median of the blob will not determine where the players' feet make contact with the ground. Instead we use an instance-based regression algorithm [1] to estimate the point of contact based on other blob features. In particular, the regression algorithm computes the distance from the top of the player's head to the point of contact with the ground. We have found the tops of players' heads to be a relatively stable feature for tracking, compared to *e.g.* the torso or feet which are more frequently obscured by other players and shadow.

In order to populate the knowledge base of the instance-based regression algorithm, MuTTSA supports interactive blob classification. Whenever MuTTSA incorrectly estimates the position of a player's

feet within a blob, the user can select the blob and indicate the feet on an enlarged depiction of the blob. This example is added to the knowledge base of the regression algorithm. Each blob produced by connected components analysis is compared to the instances in the knowledge base, and the point location of the most similar blob in the knowledge base is adapted to the new blob. The apparent height of a player varies greatly between the near and far fields, so the most important feature in the similarity metric is the y component of the location of the player's head. The other features are blob height, area (pixel count), and the x component of position (with a small weight). An entry in the knowledge base need not specify a point location. In this case, matching blobs will not generate observations for data association and tracking. This allows the system to learn to reject false positives which arise from noise in the background subtraction process. For our tests we populated the classifier with 67 training examples.

The next step is connected components analysis, which closely follows Section 2. Only the portions of each frame that might generate observations of interest to the tracking mechanism are analyzed. The user may also specify maximum and minimum limits on blob area, height, and width. Blobs which are too large or small are discarded immediately in order to bypass feature extraction. Very large blobs can result when a camera is bumped, causing the background to change suddenly.

MuTTSA uses background difference thresholding to classify foreground pixels. Choosing a threshold is a tradeoff between false positives (spurious blobs) and false negatives (missing blobs). In practice the soccer field is a relatively stable background and we get good results by simply starting with a small threshold and raising it until false positives are reasonably sparse. We use a distance threshold of 7 in RGB color space where each component ranges from 0 to 255.

Median filtering is used to model the background. Cheung and Kamath report that median filtering to be competitive with more computationally expensive approaches [3]. Other per-pixel background models such as Mixture of Gaussians [7] are equally compatible with LBSCCA. Our median algorithm has two parameters: T specifies how often to add a sample to the background model, and N how many samples to retain. As a result the background model changes only every T frames. In contrast, background subtraction and thresholding must be performed for each time step because the foreground changes more rapidly. In our experiments we used a value of 10 for both T and N, so the background is a median of frames from a sliding 10 second window.

3.3. Data Association and State Estimation

After extracting observations using pixel-based operations under the direction of Kalman state prediction, the final steps are data association and state estimation. Our presentation of JPDA follows Oh and Sastry [14].

The problem of data association arises because we do not know the mapping from tracks to observations, nor even which observations are valid and which are noise. JPDA implements probabilistic assignment; several observations can influence the estimate for a single track, each in proportion to the likelihood that the observation was caused by the track under the assumed model of Gaussian noise.

Using the Kalman state prediction $\hat{\mathbf{x}}_t^k$ from Equation 1 and Kalman error covariance prediction $\hat{\mathbf{P}}_t^k$ from Equation 2, we define the predicted observation for each target

$$\hat{\mathbf{y}}_t^k = \mathbf{C}\hat{\mathbf{x}}_t^k \quad \text{with} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3}$$

because only positions (and not velocities) are observed. For every pairing of an observation j with a

track k define the innovation

$$\mathbf{v}_t^k(j) = \mathbf{y}_t^j - \hat{\mathbf{y}}_t^k \tag{4}$$

and its covariance

$$\mathbf{B}_t^k = \mathbf{C}\hat{\mathbf{P}}_t^k\mathbf{C}^{\mathrm{T}} + \mathbf{R}^{\mathrm{T}} \tag{5}$$

where $\mathbf{R}$ is the variance of the observation error. We used $\mathbf{R} = \mathrm{diag}(1, 40)$ with the y component of observation error covariance much larger than the x component because our cameras' oblique view of the soccer field and the natural vertical motion of running both contribute significantly. Observation j is valid for track k only if

$$\mathbf{v}_t^k(j)^{\mathrm{T}}(\mathbf{B}_t^k)^{-1}\mathbf{v}_t^k(j) < \delta \tag{6}$$

with the threshold δ a user-specified parameter. We used $\delta = 1$. If observation j falls outside the validation ellipse for track k, then the probability of association between the track and observation is forced to 0, i.e., it is assumed that j is not an observation of k. Discarding low-probability associations in this way reduces the combinatorial complexity of calculating the joint probability with minimal effect on the results. Observations not valid for any track are discarded.

Validation is the link between JPDA and LBSCCA. Because invalid observations do not affect data association or tracking, LBSCCA avoids the computational cost of finding invalid observations by only searching the valid region for each track.

The next step in JPDA is to compute β, the probability of association between each observation and each track. This calculation depends on the distance from each observation to each track, the error covariance of the tracks, and also the positions of other observations and tracks. We will forgo describing the equations for calculating β and refer the reader to Oh and Sastry [14] for more detail. We used the parameters $p_d = 0.8$ for the probability of detection and $\lambda_f = 0.01$ the false alarm rate.

Finally, state estimation updates the state predictions using the observations and data association probabilities β. These are analogous to the standard Kalman Filter state estimation equations but incorporate all observations for each track through β.

$$\mathbf{v}_t^k = \sum_{j=1}^{n_t} \beta_{jk}\mathbf{v}_t^k(j) \tag{7}$$

is the combined innovation, and

$$K_t^k = \hat{\mathbf{P}}_t^k\mathbf{C}(\mathbf{B}_t^k)^{-1} \tag{8}$$

is the Kalman gain. The a posteriori state and error covariance estimates for track k at time t are then given by

$$\mathbf{x}_t^k = \hat{\mathbf{x}}_t^k + K_t^k\mathbf{v}_t^k \tag{9}$$

$$\mathbf{P}_t^k = \hat{\mathbf{P}}_t^k - \left(\sum_{j=1}^{n_t}\beta_{jk}\right)K_t^k\mathbf{B}_t^k{K_t^k}^{\mathrm{T}} + K_t^k\left(\sum_{j=1}^{n_t}\beta_{jk}\mathbf{v}_t^k(j)\mathbf{v}_t^k(j)^{\mathrm{T}} - \mathbf{v}_t^k{\mathbf{v}_t^k}^{\mathrm{T}}\right){K_t^k}^{\mathrm{T}}. \tag{10}$$

4. LBSCCA Experimental Results

In order to measure the tracking capability of MuTTSA we selected a 100 frame sequence of soccer play. In addition to automatic tracking, MuTTSA supports interactive manual specification of track

positions. We used this feature to specify the ground truth for all 22 players through the 100 frame sequence.

We then tested automated tracking on the same sequence. We did not track the ball, which is an extremely difficult target due to its small size, low contrast, and high acceleration. Parameters for tracking and data association were as listed in the previous section. We computed the error for each track as the Euclidean distance from the same track at the same time in the ground truth file. The time-dependent error for each track is shown in Figure 5.

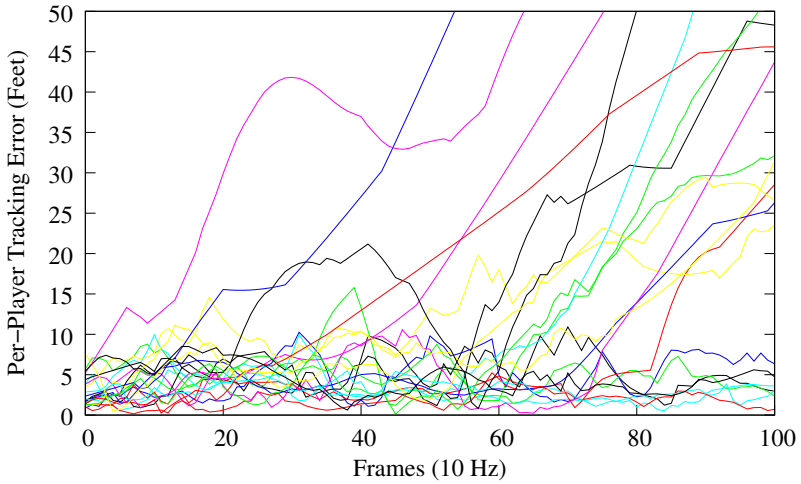


Figure 5. Tracking error for 22 soccer players over 100 frames.

We then computed the mean time to failure for individual tracks. We defined failure as a deviation of at least 10 feet from the track's actual location. This distance is less than 10% of the soccer field which measures 231 by 218 feet, but allows the algorithm to lose some accuracy while a target is occluded and reacquire the target when it becomes visible. The mean time to failure was 54.7 frames with a standard deviation of 34.0.

Next we measured the computation cost of MuTTSA with and without LBSCCA. The overall speedup of the tracking application depends on two factors: first, the reduction in background subtraction costs using LBSCCA compared to full background subtraction, and second, the cost of background subtraction relative to other tasks within the application.

We measured the reduction in background subtraction costs by instrumenting MuTTSA with a background counter. The counter is incremented for each background model update, and is shared between the background models for all pixels. The counter is only incremented for model updates and not for queries.

Using full background subtraction, the number of model updates depends only the resolution of the video. Using LBSCCA, the number of updates is data-dependent because large blobs cover more background. The cost of LBSCCA also depends on the number of tracks, because each track contributes a region of interest.

We profiled MuTTSA with a varying number of tracks $0 \leq K \leq 22$. The maximum number of tracks was 22 because each soccer team has 11 players. For each K we conducted 10 trials, each with a random selection of K tracks.

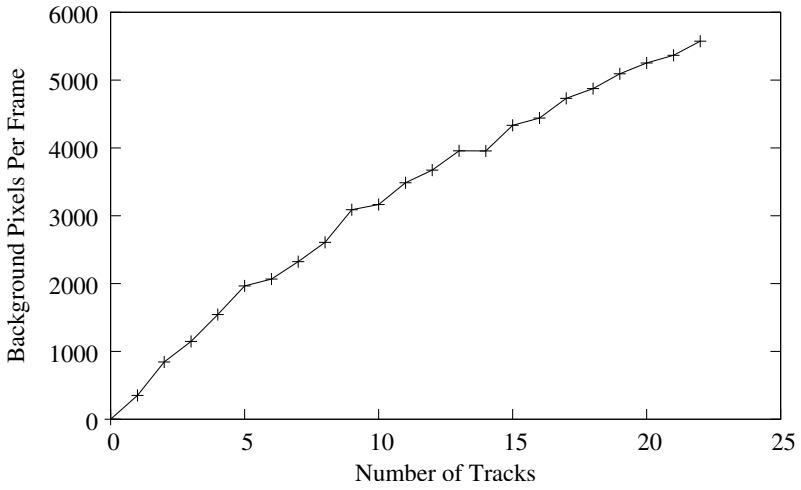


Figure 6. Average number of background pixel updates per frame when tracking up to 22 soccer players.

K	Full	LBSCCA	Reduction
1	78336	351	99.6%
5	78336	1965	97.4%
10	78336	3165	96.0%
15	78336	4332	94.5%
20	78336	5252	93.3%

Figure 7. Compared to full background subtraction, LBSCCA greatly reduces the average number of background model updates per frame, even when the number of tracks K is large.

Figure 6 shows the average number of background model updates per frame for each K. As K increases, the number of background model updates increases almost proportionally. The increase is somewhat less than proportional because regions of interest are more likely to overlap as K grows. In contrast, full background modeling requires 78336 background updates regardless of K (Table 7).

Because background modeling and connected components analysis are only part of a complete tracking application, the reduction in background updates does not directly correlate to an overall speedup of the application. We measured the update rate of MuTTSA in frames per second (FPS), with and without LBSCCA. As in our other tests, each frame is composed from 4 cameras for a total resolution of 2720×240. This test was conducted on a 1600 MHz Pentium-M laptop computer.

For small K, MuTTSA with LBSCCA sustained 6 FPS (Figure 8). With full background subtraction the maximum was 0.57 FPS. The speedup varies from 13.5 with $K = 1$ to 7.9 for $K = 10$. The near-flat profile of the curve for full background modeling when $K \leq 15$ shows that background modeling, not data association or state estimation, dominates the calculation for small K.

As K surpasses 15, the exponential JPDA calculation hits a wall and the cost of background modeling becomes less significant. This is because calculating the data association probabilities in JPDA is NP-hard [4]. JPDA validation prunes the calculation to some degree, but when K is large the number of overlapping validation regions increases, triggering a combinatorial explosion.

In our soccer application, accuracy takes priority over full automation, and so tracking is verified by

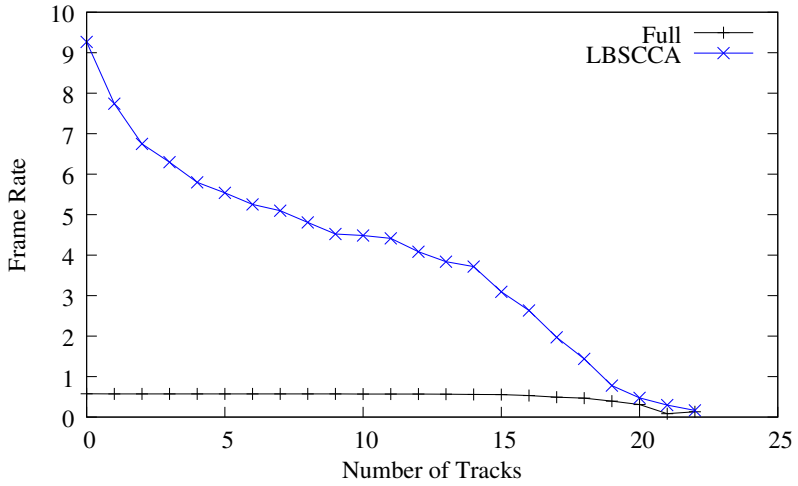


Figure 8. Average frame rate using full background subtraction vs. LBSCCA as a function of K, the number of players tracked. For fewer than 15 tracks, background subtraction dominates the computation costs and LBSCCA reduces this load dramatically.

the user. In practice, we have found it infeasible for a user to simultaneously verify more than a few tracks, so a limit of 15 is not problematic.

For applications requiring a large number of simultaneous data associations, several cheaper approximations to JPDA have been proposed. Kiril and Konstantinova [12] propose a scheme for hypothesis pruning. Oh and Sastry [14] present a polynomial Markov Chain Monte Carlo (MCMC) approximation to JPDA and derive the likelihood of close approximation. These techniques are compatible with LBSCCA.

5. Discussion

Because background subtraction with LBSCCA is data-dependent, the decrease in background computation costs is application specific. The benefit is small for frames nearly covered with foreground objects, but in this case background subtraction serves little purpose.

One concern with LBSCCA is that the sudden appearance of new tracks may go unnoticed. If the number of tracks is fixed, or changes infrequently enough that tracks can be conveniently created manually, track creation is not an issue. For instance, the number of players in a soccer team is fixed by the rules. In this context, dynamic track creation is undesirable because the fixed number of tracks helps eliminate low-confidence tracks.

If new tracks can suddenly appear but only at known locations (portals), then partial background subtraction still provides a large benefit so long as the portals cover a relatively small area of the frame. For instance, areas near the edge of the frame may be portals where passing people or cars can suddenly come into view. In a surveillance applications, doorways and edges of obscuring buildings are also portals.

In some applications new tracks can be created by disaggregation. For example, in a parking lot vehicles may stop and unload passengers. In this case, partial background subtraction will still detect the new pedestrians if the vehicle is a tracked object. In such applications, detection may be more robust if the search region is extended slightly beyond existing tracks.

The most difficult applications are those in which new tracks may appear at unpredictable locations. In this case partial background subtraction has a negative impact on track acquisition. However, if slightly delayed detection of new tracks is acceptable, partial background modeling can be used to scan only a fraction $\frac{1}{n}$ of the frame in each time step, in addition to the area of interest for each existing target. This strategy incurs a constant cost per frame, provides updates for existing tracks at frame rate, and detects new tracks after $\frac{n}{2}$ frames in expectation. If computation resources are a limiting factor, this strategy might result in overall higher tracking accuracy than full background updates at roughly $\frac{1}{n}$ the frequency, although we leave validation of this hypothesis to future experiments.

6. Summary and Conclusion

In this paper we have presented Lazy Background Subtraction and Connected Components Analysis (LBSCCA), a technique for avoiding unnecessary background modeling and image segmentation costs by restricting image processing to regions of interest. LBSCCA is compatible with a broad range of algorithms for attention allocation, background modeling, and tracking. The main requirements for LBSCCA are 1) only certain regions of the screen are important, 2) a per-pixel background model is used, and 3) connected components analysis is used for segmentation.

We have also presented a Multiple Target Tracker with Selective Attention (MuTTSA) which implements LBSCCA. We tested MuTTSA on soccer video footage, and found that LBSCCA provides a significant reduction in computation. For instance, when tracking five targets LBSCCA accelerates the update rate of MuTTSA by a factor of 10.

References

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997. 7

[2] Y. Bar-Shalom and T. Fortmann. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987. 7

[3] S.-C. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In S. Panchanathan and B. Vasudev, editors, *Proceedings of Electronic Imaging: Visual Communications and Image Processing 2004 (Part One)*, volume 5308, pages 881–892. SPIE, 2004. 8

[4] J. Collins and J. Uhlmann. Efficient gating in data association with multivariate gaussian distributed states. *IEEE Trans on Aero and Elec Sys*, 28(3):909–916, 1992. 11

[5] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1337–1342, 2003. 1

[6] C. K. D. Walther, U. Rutishauser and P. Perona. On the usefulness of attention for object recognition. In *2nd International Workshop on Attention and Performance in Computational Vision at ECCV04*, 2004. 2

[7] A. M. Elgammal, D. Harwood, and L. S. Davis. Non-parametric model for background subtraction. In *ECCV (2)*, pages 751–767, 2000. 1, 8

[8] L. M. Fuentes and S. A. Velastin. People tracking in surveillance applications. In *2nd IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2001)*, Hawaii, 2001. 1

[9] I. Haritaoglu, D. Harwood, and L. S. Davis. W^4s: A real-time system detecting and tracking people in 2 1/2d. In *ECCV (1)*, pages 877–892, 1998. 1

[10] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 543–560, London, UK, 2002. Springer-Verlag. 1

[11] L. Itti and C. Koch. Computational modeling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, Mar 2001. 2

[12] A. Kiril and P. Konstantinova. Hypotheses pruning in jpda algorithm for multiple target tracking in clutter. In *International Conference Automatic and Informatics*, pages 45–48, 2003. 12

[13] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In *Proc. of the 1998 DARPA Image Understanding Workshop (IUW'98)*, November 1998. 1

[14] S. Oh and S. Sastry. A polynomial-time approximation algorithm for joint probabilistic data association. In *Proc. of the American Control Conference*, 2005. 7, 8, 9, 12

[15] A. Oliva, A. B. Torralba, M. S. Castelhano, and J. M. Henderson. Top-down control of visual attention in object detection. In *ICIP (1)*, pages 253–256, 2003. 2

[16] N. M. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000. 1

[17] J. Orwell, P. Remagnino, and G. Jones. From connected components to object sequences. In *First IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 72–79, Grenoble, France, 2000. 1

[18] N. E. Parkhurst D, Law K. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002. 2

[19] R. Rosales and S. Sclaroff. Improved Tracking of Multiple Humans with Trajectory Predcition and Occlusion Modeling. Technical Report BUCS-TR-1998-007, CS Department, Boston University, March 2 1998. 1

[20] L. Stark, C. Privitera, H. Yang, M. Azzariti, Y. Ho, T. Blackmon, and D. Chernyak. Representation of human vision in the brain: How does human perception recognize images? *Journal of Electronic Imaging*, 10(1):123–151, 2001. 2

[21] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, Dept. of Computer Science, Univ. N. Carolina at Chapel Hill, 1995. 7