Video Surveillance of Interactions

Yuri Ivanov
MIT Media Laboratory
Cambridge, MA 02139
yivanov@media.mit.edu

Chris Stauffer
MIT Artificial Intelligence Laboratory
Cambridge, MA 02139
stauffer@ai.mit.edu

Aaron Bobick
MIT Media Laboratory
Cambridge, MA 02139
bobick@media.mit.edu

W. E. L. Grimson
MIT Artificial Intelligence Laboratory
Cambridge, MA 02139
welg@ai.mit.edu

Abstract

This paper describes an automatic surveillance system, which performs labeling of events and interactions in an outdoor environment. The system is designed to monitor activities in an open parking lot. It consists of three components - an adaptive tracker, *an* event generator, *which maps object tracks onto a set of pre-determined discrete events, and a* stochastic parser. *The system performs segmentation and labeling of surveillance video of a parking lot and identifies person-vehicle interactions, such as* `pick-up` *and* `drop-off`. *The system presented in this paper is developed jointly by MIT Media Lab and MIT Artificial Intelligence Lab.*

1 Introduction

Research in visual surveillance is quickly approaching the area of complex activities, which are framed by extended context. As more methods of identifying simple movements become available, the importance of the contextual methods increases. In such approaches, activities and movements are not only recognized at the moment of detection, but their interpretation and labeling is affected by the temporally extended context in which the events take place (e.g., see [2]). For example, in our application, while observing objects of different classes, cars and people, the detection of the class of an object may be uncertain. However, if when participating in an interaction the object behaves like a car, entering interactions with other objects in a way characteristic to a car, then belief about its class label is reinforced.

The monitoring system we describe in this paper is an example of an end-to-end implementation, which is adaptive to the physical features of the monitored environment and exhibits certain contextual awareness. Adaptation to the environment is achieved by a tracker based on Adaptive Background Mixture Models ([18]). It robustly tracks separate objects in environments with significant lighting variation, repetitive motions, and long-term scene changes. The tracking sequences it produces are probabilistically classified and mapped into a set of pre-determined discrete events. These events correspond to objects of different types performing different actions.

Contextual information in the system is propagated via a stochastic parsing mechanism. Stochastic parsing handles noisy and uncertain classifications of the primitive events by integrating them into a coherent interpretation. Parallel input streams and consistency checking allow for detection of high level interactions between multiple objects. The system demonstrates integration of low level visual information with high level structural knowledge, which serve as contextual constraints. The system is capable of maintaining concurrent interpretations when multiple activities are taking place simultaneously. Furthermore, the system allows for interpretation of activities involving multiple objects, such as interactions between cars and people during `PICK-UP` and `DROP-OFF`.

The remainder of this paper is organized as follows: section 2 introduces the problem domain and gives a brief overview of previous and ongoing research in building automated surveillance systems. Section 3 gives an overview of the system and details of its implementation. The section describes the system components, a tracker (section 3.1), an event generator (section 3.2) and the parser (section 3.3) presenting the level of detail necessary for general understanding [1]. Results of the system working on the surveillance data are shown in section 4, which is followed by conclusions,

[1] Complete details on the tracker can be found in [11] and [18]. Parser is described in full in [2], [12] and [13].

presented in section 5.

2 Related Work

The work presented in this paper spans a broad range from low level vision algorithms to high level techniques used for natural language understanding.

The tracking component of the system uses an adaptive background mixture model, which is similar to that used by Friedman and Russell ([10]). Their method attempts to explicitly classify the pixel values into three separate, predetermined distributions corresponding to the color of the road, the shadows, and the cars. Unfortunately, it is not clear what behavior their system would exhibit for pixels which did not contain these three distributions.

Pfinder ([20]) uses a multi-class statistical model for the tracked objects, but the background model is a single Gaussian per pixel. After an initialization period without objects, the system reports good results indoors. Ridder et al. ([17]) modeled each pixel with a Kalman filter which made their system more robust to lighting changes in the scene. While this method does have a pixel-wise automatic threshold, it still recovers slowly and does not handle bimodal backgrounds well.

Contextual labeling in our system is performed by a stochastic parser, which is derived from that developed by Stolcke in [19], as was previously described in [2]. We extended standard Stochastic Context-Free Grammar (SCFG) parsing to include (1) uncertain input symbols, and (2) temporal interval primitives that need to be parsed in a temporally consistent manner. In order to allow for noisy input we used the robust grammar approach, similar to that of Aho and Peterson ([1]). We extended the SCFG parser to accept a multi-valued input strings which allow for correction of the substitution errors, which is similar to the work done by Bunke([6]).

[8] is fully devoted to syntactic analysis of interactions and cooperative deterministic processes. It is related to our solution, formulating problems similar to ours, which are cast in terms of cooperative grammatical systems. In contrast, we describe interactions by a single stochastic grammar and use a single parser in an attempt to avoid the computational complexity of the complete Cooperative Distributed (CD) Grammar Systems.

In the area of monitoring long term complex activities, Courtney ([7]) developed a system, which allows for detection activities in a closed environment. The activities include person leaving an object in a room, or taking it out of the room. Perhaps the most complete general solution is described in Brill at al. ([5]), who are working on an Autonomous Video Surveillance system. Brand ([3]) showed the results of detecting manipulations in video using a non-probabilistic grammar. This technique is non-probabilistic and requires relatively high quality low-level detectors.

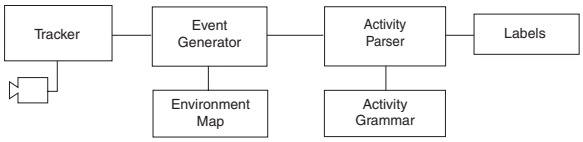

Figure 1. System architecture. The video from a camera is processed by the tracker module, which outputs tracks and candidate class labels to the event generator. Event generator maps the partial tracks onto a set of events, interpreted by the parser, according to the action grammar.

The work of Remagnino, Tan and Baker ([16]) shows an elegant solution to the parking lot monitoring task. Behavior and situation agents model object interactions using Bayesian networks. It is not clear, however, how well the system performs in a situation with occlusions where objects get completely lost for periods of time.

Morris and Hogg ([14]) show a technique which describes the events in the scene in terms of distance to the closest landmark and the object speed. The system estimates probabilities of these events and then reasons about typicality of their occurrences. This approach implicitly represents interactions between cars (landmarks) and people. In our system we attempt direct description of behaviors and interactions to produce textual most likely interpretations of high level events in the scene.

Oliver and Rosario ([15]) developed a system for detecting people interactions, which modeled interactions using Coupled Hidden Markov Model ([4]). In the course of the latter research, a multi-agent simulation was used to produce synthetic training data to train the CHMM modeling the interaction. The relation to our work is that their representation of the multi-agent simulation can be viewed as a structured, stochastic grammar-like description of the interactions.

3 Monitoring System

The architecture of the monitoring system described in this paper is shown in figure 1. The system consists of three components. The *tracker* processes the data from a camera and identifies moving objects in the camera view. The objects are tracked and the data about their movement are collected into partial tracks.

The partial tracks are then passed to the *event generator*, which generates discrete events for the beginning and the end of each track according to a simple environment model. This model encodes the knowledge about the environment

and can be learned. The map helps the generator to differentiate between tracks that correspond to objects entering and leaving the scene from objects which are lost by the tracker due to occlusion.

The *parser* analyzes the events according to the grammar which structurally describes possible activities. The grammar represents the knowledge about structure of possible interactions, making it possible to enforce structural and contextual constraints. We presently describe each component in detail.

3.1 Tracker

The tracker used in our system is based on a adaptive mixture of Gaussians technique described in detail in [18]. In this approach each pixel is modeled by a separate mixture of K^2 Gaussians as follows:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \qquad (1)$$

where $\omega_{i,t}$ is an estimate of the i^{th} mixture coefficient for time t, X_t is the current pixel value, and $\mu_{i,t}$ and $\Sigma_{i,t}$ are the parameters of the corresponding component.

If the current pixel value, X_t, is found to be well modeled by one of the mixture components (X_t is within 2.5 standard deviations from the mean), the weights $\omega(i,t)$, and parameters of the corresponding component are re-estimated. If the former is not true, the least likely component of the mixture is replaced by a new one, with the mean $\mu_(i,t)$ set to X_t and high initial variance, $\Sigma_{i,t}$.

The next step is to determine if the pixel X_t belongs to the background. In order to do that, we sort all the components in the mixture in the order of decreasing ratio $\omega / |\Sigma|^2$. This ratio effectively assigns higher importance to the mixture components that received the most evidence and have the lowest variance. The intuitive meaning of this ratio is that the components which correspond to background typically have more observations attributed to them and those observations vary little.

Then, after the components are sorted, we can set a threshold, T, which will separate components responsible for background pixels from the ones modeling foreground as follows:

$$B = argmin_b \left(\frac{\sum_{k=1}^{b} \omega_k}{\sum_{k=1}^{K} \omega_k} > T \right) \qquad (2)$$

where the meaning of the value B is that the first B components of the sorted mixture are found "responsible" for background. Now, if the pixel X_t is best modeled by one of the "background" components, it is marked as belonging to the background.

Finally, foreground pixels are segmented into regions by a two-pass, connected components algorithm.

Establishing correspondence of foreground regions between frames is accomplished using a linearly predictive multiple hypotheses tracking algorithm which incorporates both region position and size. We have implemented an on-line method for seeding and maintaining sets of Kalman filters, modeling the dynamics of foreground regions. Details of this process can be found in [18]. Essentially, for each frame, the parameters of the existing dynamical models are estimated; those models are used to explain observed foreground regions, and, finally, new models are hypothesized based on foreground regions which were not explained by any existing model.

Our system adapts to robustly deal with lighting changes, repetitive motions of scene elements, tracking through cluttered regions, slow-moving objects, and introducing or removing objects from the scene. Slow moving objects take longer to be incorporated into the background because their color has a larger variance than the background. Also, repetitive variations are learned, and a model for the background distribution is generally maintained even if it is temporarily replaced by another distribution which leads to faster recovery when objects are removed.

3.2 Event Generator

The event generator in our system is responsible for mapping the tracks produced by the tracker onto a set of predetermined discrete events. These events form the basis for the syntactic constraints that the parser enforces on the input stream. The events in our system are not *object-centered*, but rather are formulated in terms of *tracker states*. While identity of an object is unknown to the tracker, it remains the same throughout the detected object trajectory. The tracker can "lose" an object and then "find" it again later due to occlusion or sudden change of lighting, in which case the identity of the object is not preserved. Reasoning about object identity in such situations is deferred to the parser which can enforce contextual information. In this reasoning, exact configuration of the object trajectories is not important. Only the endpoints of the tracks influence how the identity is computed.

Every time the tracker reports beginning or the end of a track, the event generator produces an event. This set of primitive tracker states, such as `object-lost`, `object-found`, forms the alphabet of interactions, presented to the parser in form of a grammar.

In order to generate events, the event generator is given a simple *environment map*. Based on this map the event generator determines if the events occurred in "special lo-

[2]For the system described in this paper we use a mixture of 4 components.

cations", where the objects tend to enter or leave the scene. In addition, if the tracker does not have sufficient degree of confidence about the object's class, multiple candidate events are generated, one per candidate class with likelihoods assigned to each according to what's reported by the tracker. In the system there are total of 9 events, which are generated according to the following rules:

1. If the track began in an area where objects tend to enter the scene, `car-enter` and `person-enter` events are generated. The events are marked with the corresponding likelihoods to account for errors in classification. For instance if a beginning of a person track is reported by the tracker and the likelihood of that event is 0.7, a `person-enter` event with likelihood 0.7 is posted to the parser. Along with it, a complementary event `car-enter` is posted in the same time slot, with the likelihood of 0.3.

2. If the track did not begin in one of the "entry" areas, `car-found` and `person-found` events are generated.

3. If the track ended in one of the "exit" areas, `car-exit` and `person-exit` events are produced.

4. If the track did not end in one of the "exit" areas, `car-lost` and `person-lost` events are posted.

5. If an object's velocity dropped below a certain threshold, an `object-stopped` event is generated.

The process of generating events is illustrated in figure 2. Note that some of the track endpoints are mapped onto a pair of concurrent events, which accounts for classification errors. The parser will select one or the other, depending on which one results in the overall parse with maximum probability. Typically, at the beginning of each track, the tracker has not observed the object long enough to be certain about its class membership. Therefore, `x-enter` and `x-found` events have likelihoods close to 0.5. In contrast, by the time the object disappears or is lost, there is enough data to make more accurate classification decision. Consequently, class likelihoods of `x-exit` and `x-lost` events are typically higher than those of `x-enter` and `x-found`.

3.3 Parser

Our SCFG parser is an extension of that by Earley ([9]) and Stolcke ([19]). For each input symbol, the parser keeps a *state set*, a set of parser *states* that collectively describe current pending derivations. A state is a production rule, augmented with two additional markers, i and k. Marker k indicates where in the input string the rule is applied, and marker i shows where in the rule the parser is currently located. The position of the parser inside the rule that corresponds to i is shown by the symbol ".". We denote a state as:

$$i : X_k \rightarrow \lambda . Y \mu \ \ [\alpha, \gamma] \tag{3}$$

where X and Y are *non-terminals* and λ and μ are arbitrary sequences of *terminals* and *non-terminals*. In the SCFG parsing algorithm ([19]), each state also carries forward and inner probabilities denoted in (3) by α and γ, respectively. α, also called a prefix probability, is the probability of the parsed string up to position i, and γ is a probability of the sub-string starting at k and ending at i.

Parsing begins with initializing the first state set with an initial state. The parsing proceeds as an iteration between three steps - *prediction*, *scanning* and *completion* until the final state is reached. In this paper we only give a minimal exposition of the parsing algorithm, as it is presented in full elsewhere (eg. see [19], [2] and [12]).

In order to propagate track data through the parse we modify each state to include two additional auxiliary variables - $\mathbf{l}$ and $\mathbf{h}$ (a *low mark* and a *high mark* of the state). These variables hold the data about endpoints of the corresponding track:

$$\mathbf{l} = \begin{pmatrix} f_l \\ t_l \\ x_l \\ y_l \\ dx_l \\ dy_l \end{pmatrix} \qquad \mathbf{h} = \begin{pmatrix} f_h \\ t_h \\ x_h \\ y_h \\ dx_h \\ dy_h \end{pmatrix} \tag{4}$$

where f is a frame number, t - a time stamp, x and y are object coordinates in the image, and dx and dy are object velocity components.

These data are used to compute the penalty function of equation (5), which weighs total probability of the parse by joining two partial tracks with endpoints at $\mathbf{h}_1$ and $\mathbf{l}_2$. This penalty function ensures that the tracks, considered for joining, are not too far apart and are temporally consistent.

$$f(\mathbf{r}_p, \mathbf{r}_2) = \begin{cases} 0, & if \ \ (t_2 - t_1) < 0 \\ \exp\left(\frac{(\mathbf{r}_2 - \mathbf{r}_p)^T (\mathbf{r}_2 - \mathbf{r}_p)}{\theta} \right), & o/w \end{cases} \tag{5}$$

where $\mathbf{r}_p$ is computed from $\mathbf{r}_1$, based on a constant velocity assumption: $\mathbf{r}_p = \mathbf{r}_1 + d\mathbf{r}_1(t_2 - t_1)$, $\mathbf{r}_1$ and $\mathbf{r}_2$ correspond to the track endpoint positions at the track break, and $d\mathbf{r}_1$ is the instantaneous velocity of the object at position $\mathbf{r}_1$.

When an event from the event generator is received, the parser advances by one step, producing a new state set and searching it for the final state. If the final state is found, the parser traverses the parsing queue and assembles the most likely parse. After the parse is assembled, the parser outputs the resulting interpretation. Note, however, that since this

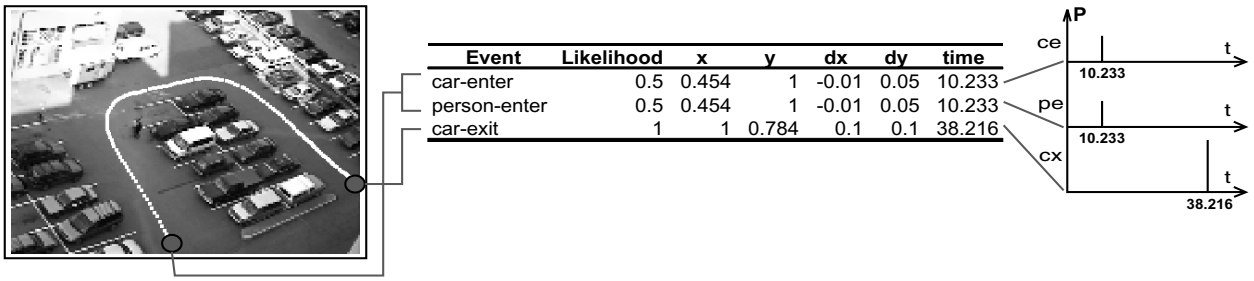

Figure 2. Illustration of a process of mapping tracks onto discrete events. The tracker reports the beginning and the end of the track. In this example, the beginning of the track corresponds to an object entering the scene. At that point the class label of the class cannot be determined. This results in generation of two concurrent events - one per class (cars and persons) with probability of the label being 0.5.

operation is local, it is possible that it will be subsumed at a later time by a more general interpretation. This is the case when interpreting such interactions as `DROP-OFF` and `DRIVE-IN`, where in our grammar (shown later) the latter is a subset of the former (e.g. see figure 4).

In the course of developing the parser, we implemented a mechanism which allows the parser to deal with parallelism in primitives and interpretations. These two kinds of parallelism normally present a significant obstacle for traditional parsers, as they are strictly sequential machines. Parallelism in interpretations, where several derivations, related to different activities are being traced concurrently, is accounted for by traditional error recovery methods ([1, 6]). In the spirit of this technique we replace the original grammar G by a robust one $\hat{G}$, formed as follows:

1. Each terminal, say b, appearing in productions of grammar G is replaced by a pre-terminal, e.g. $\hat{B}$, in $\hat{G}$:

$$\begin{array}{lcl} G: & \Rightarrow & \hat{G}: \\ A \rightarrow bC & & A \rightarrow \hat{B}C \end{array}$$

2. Each pre-terminal of $\hat{G}$ is expanded to the corresponding terminal augmented by a $SKIP$ rule. For instance:

$$\begin{array}{l} \hat{G}: \\ \hat{B} \quad \rightarrow \quad b \quad | \quad SKIP \quad b \end{array}$$

3. `SKIP` rule is added to $\hat{G}$, which includes all repetitions of all terminals:

$$\begin{array}{l} \hat{G}: \\ SKIP \quad \rightarrow \quad b \quad | \quad \ldots \quad | \quad b \quad SKIP \quad | \quad \ldots \end{array}$$

The robust grammar will consume the erroneous symbols by the $SKIP$ production. During the parse, while analyzing the current symbol, the parser will simply ignore all the symbols unrelated to the current derivation by collecting them in $SKIP$ rules.

The second kind of concurrency relates to the fact that an interaction involves at least two objects, subject to their own independent consistency constraints ([2, 12]). [8] shows a complete multi-agent solution to this problem. We chose not to follow that route because it requires perfect detection of the objects in the scene. Instead, we developed a multi-class interleaved consistency mechanism ([13]), which allows to achieve similar goals within a single parser.

4 Experimental Results

Here we show results of the system run on a data collected on a parking lot at Carnegie Mellon University. The system runs in real time processing data from a live video feed or a video tape. The tracker and the event generator run on an 175 MHz R10000 SGI O2 machine. The parser runs on an 200 MHz R4400 SGI Indy.

The tracker runs at approximately 12 fps on 160x120 images. It generally exhibited unbroken tracks except in cases of occlusions and extreme lighting changes. The events were mapped using a hand-coded, probabilistic classifier for object type (e.g. car or person), which used the aspect ratio of the object.

The parser requires the interaction structure described to it in terms of Stochastic Context Free Grammar. A partial listing of the grammar employed by our system for the parking lot monitoring task is shown in figure 3. Labels in capitals are the *non-terminals* while the *terminals*, or primitives, are written in small letters. Square brackets enclose probabilities associated with each production rule. These probabilities reflect the typicality of the corresponding production rule and the sequence of primitives, which it represents.

In our system high-level non-terminals (`CAR-THROUGH`,

```
TRACK:          CAR-TRACK                [0.5]
              | PERSON-TRACK             [0.5]

CAR-TRACK:      CAR-THROUGH              [0.25]
              | CAR-PICKUP               [0.25]
              | CAR-OUT                  [0.25]
              | CAR-DROP                 [0.25]

CAR-PICKUP:  ENTER-CAR-B  CAR-STOP
             PERSON-LOST  B-CAR-EXIT     [1.0]

ENTER-CAR-B:    CAR-ENTER                [0.5]
              | CAR-ENTER  CAR-HIDDEN     [0.5]

CAR-HIDDEN:     CAR-LOST  CAR-FOUND      [0.5]
              | CAR-LOST  CAR-FOUND
                CAR-HIDDEN               [0.5]

B-CAR-EXIT:     CAR-EXIT                 [0.5]
              | CAR-HIDDEN  CAR-EXIT     [0.5]

CAR-EXIT:       car-exit                 [0.7]
              | SKIP car-exit            [0.3]

CAR-LOST:       car-lost                 [0.7]
              | SKIP car-lost            [0.3]

CAR-STOP:       car-stop                 [0.7]
              | SKIP car-stop            [0.3]

PERSON-LOST:    person-lost              [0.7]
              | SKIP person-lost         [0.3]
```

Figure 3. A `CAR-PICKUP` branch of a simplified grammar describing interactions in a parking lot.

`PERSON-THROUGH`, `PERSON-IN`, `CAR-OUT`, `CAR-PICK` and `DROP-OFF`) have *semantic action* blocks associated with them, which are not shown in the figure for brevity. Each such action is a simple script which outputs the corresponding label (such as `DROP-OFF`), and all the available data related to the non-terminal (e.g. starting and ending video frame or time- stamp). The semantic action is invoked when the final state is reached and the resulting maximum probability parse includes the corresponding non-terminal.

The production rule probabilities have been manually set to plausible values for this domain. Learning these probabilities is an interesting problem, which is planned for future work. However, our observations showed that the grammatical and spatial consistency requirements eliminate the majority of incorrect interpretations. This results in our system being quite insensitive to the precise values of these probabilities.

The test data consisted of approximately 15 minutes of video, showing several high level events such as `drop-off` and `pick-up`. The events were staged in the real environment, where the real traffic was present concurrently with the staged events. The only reason for staging the events was to have more examples within 15 minutes of video. The drop-offs and pick-ups were performed by people unfamiliar with the system. The resulting parses were output in the real time. In figures 5 a) - e) we show a sequence of 5 consecutive detections of high level events. The sequence shown in the figure, demonstrates the capability of the system to parse concurrent activities and interactions. The main event in this sequence is the `DROP-OFF`. While monitoring this activity, the system also detected unrelated high level events: 2 instances of `CAR-THROUGH` and a `PERSON-THROUGH` event. The figure 5 f) shows the temporal extent of activities, shown iconically in figures 5 a)-e). The figure also illustrates the selectivity of the parser granted by the use of `SKIP` productions. Having found the interpretation `PERSON-THROUGH` (figure 5 b)) consisting of events shown in the second line of figure 5 f) (lines 5 and 9 of figure 4), the parser consumes the events `car-enter`, `person-found` and `car-exit` (lines 6-8 of figure 4) by the `SKIP` rule.

In the example grammar shown in figure 3, the non-terminal `CAR-HIDDEN` covers the case when it is necessary to join two partial tracks in order to produce a plausible interpretation. The occurrence of such a break might be due to an occlusion or a sudden lighting change. This feature makes the tracker tolerant to occlusions when an object completely disappears from the camera view for some period of time.

All of these parses can be traced down to the primitives, which hold the track data. Consequently, the complete track can be reconstructed, as shown by white traces in figures 5 a) - e). In the longest segment of video,

the event generator produces between 150 and 200 events; the exact count depends upon the reaction of the tracker to video noise. After tuning the environment map used by the event generator to convert tracks to events, all the high level interactions were correctly detected.

5 Conclusions

In our future work we will address more accurate modeling of the environment. Currently, tracks are mapped onto events with a non-probabilistic map of the environment. This results in high sensitivity of the event generation to subtle changes in timing of the tracker. The work presented in [18] suggests that such maps can be learned automatically.

We are also planning on learning the rule probabilities,

	Event	UID	Avg. Size	Class	P	x	y	t	frame	
DROP-OFF	ENTER	724	0.122553	0	0.5	0.450094	0.938069	917907137.8	1906	DRIVE-IN
	ENTER	665	0.046437	1	0.5	0.6107	0.94674	917907122.5	1799	
	PERSON-LEAVE	665	0.045869	1	0.997846	0.648089	0.98855	917907142.7	1938	
	STOPPED	724		0	0.995784	0.348569	0.345513	917907146.5	1964	
	ENTER	780	0.034293	1	0.5	0.74188	0.980292	917907151.3	1998	
	ENTER	790	0.069093	0	0.5	0.814565	0.032611	917907153.4	2012	
	FOUND	787	0.033573	1	0.5	0.297585	0.357887	917907153.1	2010	
	CAR-LEAVE	790	0.061263	0	0.997285	0.975971	0.211984	917907155.3	2025	
	PERSON-LEAVE	780	0.038616	1	0.999923	0.974494	0.865237	917907158.6	2047	
	PERSON-LEAVE	787	0.032045	1	0.999997	0.296519	0.183704	917907158.7	2048	
	ENTER	813	0.034776	1	0.5	0.012821	0.348379	917907160.9	2063	
	ENTER	816	0.093513	0	0.5	0.960425	0.793899	917907161.9	2070	
	CAR-LEAVE	724	0.097374	0	0.993211	0.972272	0.693728	917907165.2	2091	
	CAR-LEAVE	816	0.089424	0	0.99023	0.693699	0.990798	917907165.2	2091	

Figure 4. Results of track mapping on one of the runs of the system. Two subsets of events, outlined in the picture, correspond to `DRIVE-IN` **and** `DROP-OFF`**. Interpretation of this data is shown in figure 5.**

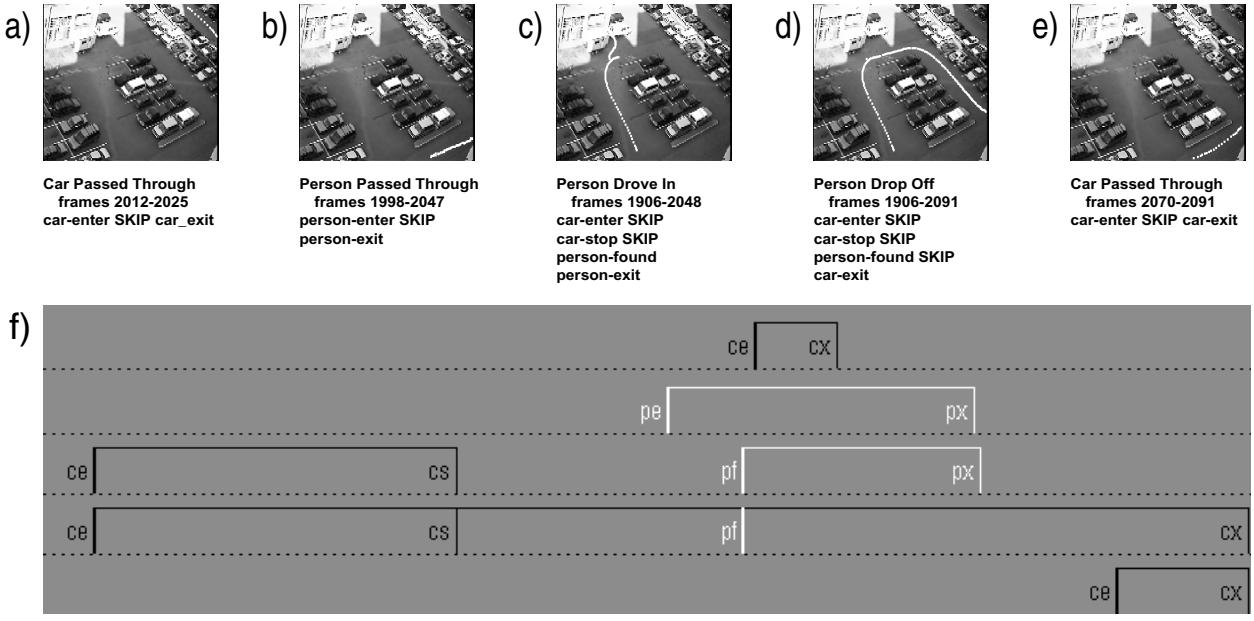

Figure 5. a) A car passed through the scene, while `DROP-OFF` **was performed. Corresponding track is shown by a sequence of white pixels. b) Person passing through. c) A person left the car and exited the scene. At this moment the system has enough information to emit the** `DRIVE-IN` **label. d) The car leaves the scene. The conditions for** `DROP-OFF` **are now satisfied and the label is emitted. e) Before the car performing the** `DROP-OFF` **exits the scene, it yields to another car passing through, which is shown here. f) Temporal extent of the actions shown in a)-e). Actions related to people are shown in white. Top line of the picture corresponds to the label a), the bottom one - e). Car primitives are drawn in black. The figure clearly demonstrates concurrency of events. In this figure, primitive events are abbreviated as follows: ce -** `car-enter`**, cs -** `car-stop`**, cx -** `car-exit`**, pe -** `person-enter`**, pf -** `person-found`**, px -** `person-exit`**.**

observing the environment for extended period of time. This will help more accurate modeling the traffic patterns as well as performance of the tracker.

We are planning to better utilize object correspondences. In the current implementation, partial tracks are joined only based on the object's position and velocity, as reflected by the penalty function in equation 5. In the future, change in object appearance will also be considered in computing this penalty.

References

[1] A. V. Aho and T. G. Peterson. A minimum distance error-correcting parser for context-free languages. *SIAM Journal of Computing*, 1, 1972.

[2] A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *Proc. Comp. Vis. and Pattern Rec.*, pages 196–202, Santa Barbara, CA, 1998.

[3] M. Brand. Understanding manipulation in video. In *AFGR96*, pages 94–99. 1996.

[4] M. Brand and N. Oliver. Coupled hidden markov models for complex action recognition. In *CVPR*, pages 994–999, Puerto Rico, 1996. IEEE.

[5] Frank Z. Brill, Thomas J. Olson, and Christopher Tserng. Event recognition and reliability improvements for the autonomous video surveillance system. In *Image Understanding Workshop*, pages 267–283, Monterey, CA, 1998.

[6] H. Bunke and D. Pasche. Parsing multivalued strings and its application to image and waveform recognition. *Structural Pattern Analisys*, 1990.

[7] J. D. Courtney. Automatic video indexing via object motion analysis. *PR*, 30(4):607–625, 1997.

[8] E. Csuhaj-Varj. *Grammar systems : a grammatical approach to distribution and cooperation*. Topics in computer mathematics ; v. 5. Gordon and Breach, Yverdon, Sitzerland ; [Langhorne, Pa.?] USA, 1994.

[9] Jay Clark Earley. *An Efficient Context-Free Parsing Algorithm*. Ph.d., Carnegie-Mellon University, 1968.

[10] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach. In *Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[11] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities. In *Comp. Vis. and Pattern Rec.*, pages 22–29, Santa Barabara, CA, 1998. IEEE.

[12] Y. A. Ivanov and A. F. Bobick. Probabilistic parsing in action recognition. Technical Report TR 450, MIT Media Lab, Vision and Modeling Group, 1997.

[13] Yuri Ivanov and Aaron Bobick. Parsing multi-agent interactions. Technical Report 479, MIT Media Lab, December 1998 1998.

[14] R. J. Morris and D. C. Hogg. Statistical models of object interaction. In *ICCV'98, Workshop on Visual Surveillance*, 1998. 2.

[15] Nuria Oliver, Barbara Rosario, and Alex Pentland. Statistical modeling of human interactions. In *CVPR, The Interpretation of Visual Motion Workshop*, pages 39–46, Santa Barbara, CA, 1998. IEEE.

[16] P. Remagnino, T. Tan, and K. Baker. Agent orientated annotation in model based visual surveillance. In *ICCV*, pages 857–862, Bombay, 1998.

[17] Christof Ridder, Olaf Munkelt, and Harald Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In *International Conference on recent Advances in Mechatronics*, pages 193–199, 1995.

[18] Chris Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR '99*, pages 246–252, Ft. Collins, CO, 1999.

[19] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2), 1995.

[20] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.