A Visual Tracking Framework for Intent Recognition in Videos

Alireza Tavakkoli, Richard Kelley, Christopher King, Mircea Nicolescu, Monica Nicolescu, George Bebis

Department of Computer Science and Engineering, University of Nevada, Reno

{tavakkol,rkelley,cjking,mircea,monica,bebis}@cse.unr.edu

Abstract. To function in the real world, a robot must be able to understand human intentions. This capability depends on accurate and reliable detection and tracking of trajectories of agents in the scene. We propose a visual tracking framework to generate and maintain trajectory information for all agents of interest in a complex scene. We employ this framework in an intent recognition system that uses spatio-temporal contextual information to recognize the intentions of agents acting in different scenes, comparing our system with the state of the art.

1 Introduction

Recognizing the intentions of others is an important part of human cognition, especially in activities in which collaboration or competition play an important role. To achieve autonomy, robots will have to have similar capabilities. This will be especially critical in tasks requiring substantial human-robot interaction.

Any system that uses visual clues to infer the intent of active agents in a scene requires a robust mechanism to track these agents at pixel level, to accurately generate and maintain their trajectories at world level, and to pass this world-level information to higher level stages of the process. In this paper, we propose a system that performs these tasks and is robust to illumination changes, background clutter, and occlusion.

In order to detect humans in videos several methods have been proposed. Dalal and Triggs in [1] proposed a method to detect human bodies in images by employing Histogram of Oriented Gradients (HoG) in conjunction with a linear SVM trained for the class of human bodies. Ramanan *et al.* in [2] proposed a geometric and probabilistic approach to detect human bodies in video sequences. However, these methods are very time consuming, which makes them undesirable for applications with real-time constraints.

We demonstrate our vision system both in isolation and as a part of a larger intent recognition system. That system uses hidden Markov models of activities in a Bayesian framework that uses context to improve the system's performance.

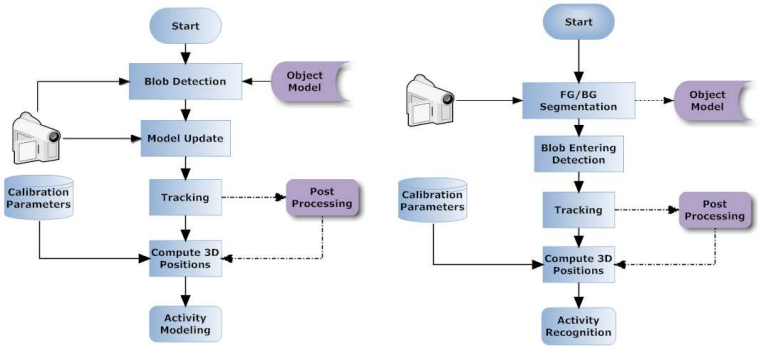


Fig. 1. The two object tracking frameworks for (a) *activity modeling* using a modified mean-shift tracker and (b) *intent recognition* using a spatio-spectral tracker.

2 Object Tracking Module

We provide a set of vision-based perceptual capabilities for our robotic system that facilitate the modeling and recognition of actions carried out by other agents. As the appearance of these agents is generally not known a priori, the only visual cue that can be used for detecting and tracking them is image motion. Although it is possible to perform segmentation from an image sequence that contains global motion, such approaches – typically based on optical flow estimation [3]– are not very robust and are time consuming. Therefore, our approach uses more efficient and reliable techniques from real-time surveillance, based on background modeling and segmentation:

- During the *activity modeling* stage, the robot is moving while performing various activities. The appearance models of other mobile agents, necessary for tracking, are built in a separate, prior process where the static robot observes each agent that will be used for action learning. The robot uses an enhanced mean-shift tracking method to track the foreground object.
- During the *intent recognition* stage, the static robot observes the actions carried out by other agents. This allows the use of a foreground-background segmentation technique to build appearance models on-line, and to improve the speed and robustness of the tracker. The robot is stationary for efficiency reasons. If the robot moves during intent recognition we can use the approach from the modeling stage.

Fig. 1 shows the block diagram of the proposed object tracking frameworks.

2.1 Intent Recognition Visual Tracking Module

We propose an efficient Spatio-Spectral Tracking module (SST) to detect objects of interest and track them in the video sequence. The major assumption is that

the observer robot is static. However, we do not make any further restrictions on the background composition, thus allowing for local changes in the background such as fluctuating lights, water fountains, waving tree branches, etc.

The proposed system models the background pixel changes using an Incremental Support Vector Data Description module. The background model is then used to detect foreground regions in new frames. The foreground regions are processed further by employing a connected component processing in conjunction with a blob detection module to find objects of interest. These objects are tracked by their corresponding statistical models which are built from the objects' spectral (color) information. A laser-based range finder is used to extract the objects' trajectories and relative angles from their 2-D tracking trajectories and their depth in the scene. However, the spatio-spectral coherency of tracked objects may be violated in cases when two or more objects occlude each other.

A collision resolution mechanism is devised to address the issue of occlusion of objects of interest. This mechanism uses the spatial object properties such as their size, the relative location of their center of mass, and their relative orientations to predict the occlusion (collision).

Incremental Support Vector Data Description Background modeling is one of the most effective and widely used techniques to detect moving objects in videos with a quasi-stationary background. In these scenarios, despite the presence of a static camera, the background is not completely stationary due to inherent changes, such as water fountains, waving flags, etc. Statistical modeling approaches estimate the probability density function of the background pixel values. Parametric density estimation methods, such as Mixture of Gaussians techniques (MoG) are proposed in [4]. If the data is not drawn from a mixture of normal distributions the parametric density estimation techniques may not be useful. As an alternative, non-parametric density estimation approaches can be used to estimate the probability of a given sample belonging to the same distribution function as the data set [5]. However, the memory requirements of the non-parametric approach and its computational costs are high since they require the evaluation of a kernel function for all data samples.

Support Vector Data Description (SVDD) is a technique which uses support vectors in order to model a data set [6]. The SVDD represents one class of known data samples in such a way that for a given test sample it can be recognized as known, or rejected as novel. Training of SVDDs is a quadratic programming optimization problem. This optimization converges by optimizing only on two data points with a specific condition [7] which requires at least one of the data points to violate the KKT conditions – the conditions by which the classification requirements are satisfied– [8]. Our experimental results show that our SVDD training achieves higher speed and require less memory than the online [9] and the canonical training [6].

A normal data description gives a closed boundary around the data which can be represented by a hyper-sphere (i.e. $F(R, a)$) with center a and radius R, whose volume should be minimized. To allow the possibility of outliers, slack

variables $\epsilon_i \geq 0$ are introduced. The error function to be minimized is:

$$F(R,a) = R^2 + C\sum_i \epsilon_i \|x_i - a\|^2 \tag{1}$$

subject to:

$$\|x_i - a\|^2 \leq R^2 + \epsilon_i \quad \forall i. \tag{2}$$

Kernel functions $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ are used to achieve more general descriptions. After applying the kernel and using Lagrange optimization the SVDD function becomes:

$$L = \sum_i \alpha_i K(x_i, x_i) - \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \qquad \forall \alpha_i : 0 \leq \alpha_i \leq C \tag{3}$$

Only data points with non-zero α_i are needed in the description of the data set, therefore they are called *support vectors* of the description. After optimizing (3) the Lagrange multipliers should satisfy the normalization constraint $\sum_i \alpha_i = 1$.

Our incremental training algorithm is based on the theorem proposed by Osuna *et al.* in [8]. According to Osuna a large QP problem can be broken into series of smaller sub-problems. The optimization converges as long as at least one sample violates the KKT conditions.

In the incremental learning scheme, at each step we add one sample to the training working set consisting of only support vectors. Assume we have a working set which minimizes the current SVDD objective function for the current data set. The KKT conditions do not hold for samples which do not belong to the description. Thus, the SVDD converges only for the set which includes a sample outside the description boundary. The smallest possible sub-problem consists of only two samples [7]. Since only the new sample violates the KKT conditions at every step, our algorithm chooses one sample from the working set along with the new sample and solves the optimization on them.

Solving the QP problem for two Lagrange multipliers can be done analytically. Because there are only two multipliers at each step, the minimization constraint can be displayed in 2-D. The two Lagrange multipliers should satisfy the inequality in (3) and the linear equality in the normalization constraint.

We first compute the constraints on each of the two multipliers. The two Lagrange multipliers should lie on a diagonal line in 2-D (equality constraint) within a rectangular box (inequality constraint). Without loss of generality we consider that the algorithm starts with finding the upper and lower bounds on α_2 which are $H = min(C, \alpha_1^{old} + \alpha_2^{old})$ and $L = max(0, \alpha_1^{old} + \alpha_2^{old})$, respectively. The new value for α_2^{new} is computed by finding the maximum along the direction given by the linear equality constraint. If the new value for α_2^{new} exceeds the bounds it will be clipped ($\hat{\alpha}_2^{new}$). Finally, the new value for α_1 is computed using the linear equality constraint:

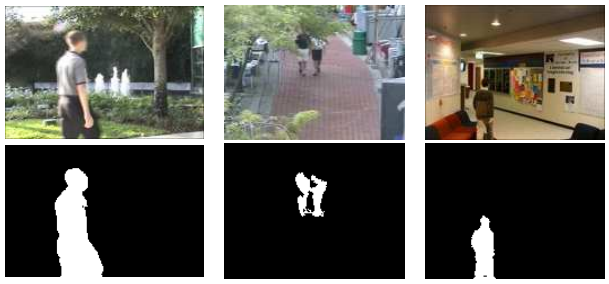
$$\alpha_1^{new} = \alpha_1^{old} + \alpha_2^{old} - \alpha_2^{new} \tag{4}$$

(a) (b) (c) (d)

Fig. 2. Comparison of methods in presence of irregular motion in *water surface* video: (a) Original frame. (b) MoG results. (c) AKDE results. (d) INCSVDD results.

The algorithm for background modeling using the incremental SVDD is composed of two modules. In the background modeling module the incremental learning scheme uses the pixel values in each frame to train their corresponding SVDDs. In the background subtraction module the pixel value in the current frame is used in the trained SVDD to label it as a known (background) or novel (foreground) pixel.

Background Modeling Results We compare the results of background modeling and foreground detection using our method with the AKDE [5] and MoG [4] on the *water surface* video. The results of MoG, the AKDE and our technique are shown in Figure 2 (b), (c), and (d), respectively.



(a) (b) (c)

Fig. 3. Background modeling using the proposed method.

Figure 3 shows the results of foreground detection in videos using our method. The fountain, waving branches, and flickering lights in Figure 3(a), (b), and (c) pose significant challenges.

Blob Detection and Object Localization In the blob detection module, the system uses a spatial connected component processing to label foreground regions from the previous stage. However, to label objects of interest a blob

refinement framework is used to compensate for inaccuracies in physical appearance of the detected blobs due to unintended region split and merge, inaccurate foreground detection, and small foreground regions. A list of objects of interest corresponding to each detected blob is created and maintained to further process and track each object individually. This raw list of blobs corresponding to objects of interest is called the spatial connected component list.

Spatial properties about each blob such as its center and size are kept in the spatial connected component list. The list does not incorporate individual objects' appearances and thus is not solely useful for tracking purposes. The process of tracking individual objects based on their appearance in conjunction with their corresponding spatial features is carried out in the spatio-spectral tracking mechanism.

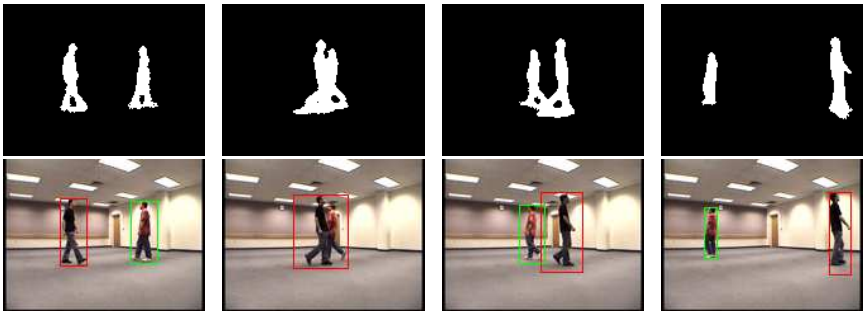
Spatio-spectral Tracking Mechanism A system that can track moving objects (i.e. humans) requires a model for individual objects. These appearance models are employed to search for correspondences among the pool of objects detected in new frames. Once the target for each individual has been found in the new frame they are assigned a unique ID. In the update stage the new location, geometric and photometric information for each visible individual are updated. This helps recognize the objects and recover their new location in future frames.

Our proposed appearance modeling module represents an object with two set of histograms, for the lower and upper half of the body. In the spatio-spectral tracking module a list of known objects of interest is maintained. This list represents each individual object and its corresponding spatial and color information along with its unique ID. During the tracking process the system uses the raw spatial connected component list as the list of observed objects and uses a statistical correspondence matching to maintain the ordered objects list and track each object individually. The tracking module is composed of three components:

- **Appearance modeling.** For each object in the raw connected component list a model is generated which contains the object center of mass, its hight and width, the upper and lower section foreground masks, and the multivariate Gaussian distribution models of its upper and lower section pixels.
- **Correspondence matching.** The pixels in the upper and lower sections of each object in the raw list are used against each model in the ordered list of tracked objects. The winner model's ID then is used to represent the object.
- **Model update.** Once the tracking is performed the models will be updated. Any unseen object in the raw list is then assigned a new ID and their models are updated accordingly.

Collision Resolution In order for the system to be robust to collisions – when individuals get too close so that one occludes the other– the models for the occluded individual may not reliable for tracking purposes. Our method uses the distance of detected objects and uses that as a means of detecting a collision. After a collision is detected we match each of the individual models

with their corresponding representatives. The one with the smallest matching score is considered to be occluded. The occluded object's model will not be updated but its new position is predicted by a Kalman filter. The position of the occluding agent is updated and tracked by a well known mean-shift algorithm. After the collision is over the spatio-spectral tracker resumes its normal process for these objects.



(a) (b) (c) (d)

Fig. 4. Detection results in four frames of a video sequence: Top row: detected foreground masks. Bottom row: Tracked objects.

Tracking Results The proposed tracking algorithm is tested on several video sequences in which people would appear in the scene, pass each other and reappear in the scene after leaving it. Fig. 4 shows the detection and tracking results of the algorithm. As it can be seen from Fig. 4(b) the system resolves the occlusion and recovers very quickly once the occluded object reappears, Fig. 4(c).

3 Context-Based Intent Recognition

The information from the vision system is used by the intent recognition module. This module has two parts: a set of activity models and a set of context models. The overall architecture is inspired by Theory of Mind [10].

3.1 Hidden Markov Models for Representing Activities

Our activity modeling approach uses hidden Markov models (HMMs). In our system, a single HMM models a single basic activity ("two people approach one another"). The hidden states of such a model correspond to "small-scale" intentions or components of a complex activity. The visible symbols of our models correspond to changes in the parameters produced as the output of the tracking module.

The system is trained using the approach described above during the activity modeling stage and the Baum-Welch algorithm.

3.2 Context Models for Inferring Intent

Recent neuroimaging work [11] suggests that humans use the context in which an activity is performed to infer the intent of an agent. Our system mimics this capability to improve intent recognition.

In our system, an *intention* consists of a name and an activity model, and a *context* consists of a name and a probability distribution over the possible intentions that may occur in the current context. Our goal is to determine the most likely intention of an agent, given a sequence of observations of the agent's trajectory and the current context. Letting s be an intention, v be a sequence of visible symbols, and c be a context, we want to find

$$\arg\max_s p(s|v, c).$$

Applying Bayes' rule and simplifying, this is equivalent to finding s that maximizes

$$p(v|s, c)p(s|c).$$

Moreover, the above product can be further simplified to

$$p(v|s)p(s|c).$$

This follows from our definition of a context. Since we assume that a context only provides a distribution over intention names, knowing the current context tells us only about the relative likelihood of the possible intentions. A context says nothing about the hidden or observable states of any activity model, and so does not affect the independence assumptions of the underlying HMM.

Thus to find the most likely intention given a context c and observation sequence v, compute $p(v|s)p(s|c)$ for each intention s, and select the one whose probability is largest. The value of $p(v|s)$ can be calculated using the HMM that belongs to s, and $p(s|c)$ is assumed available in the definition of the system's context.

3.3 Intention-Based Control

Our system is also able to dispatch behaviors on the basis of an observed agent's intentions. To do this, the system performs inference as above, and examines both the current context and decoded internal state of the activity model. On the basis of this state, the robot performs a predefined action or set of actions.

4 Experimental Results

To test our system, we had a robot observe interactions among multiple human agents and multiple static objects in three sets of experiments. To provide a quantitative evaluation of intent recognition performance, we use two measures:

- *Accuracy rate* = the ratio of the number of observation sequences, of which the winning intentional state matches the ground truth, to the total number of test sequences.
- *Correct Duration* = C/T, where C is the total time during which the intentional state with the highest probability matches the ground truth and T is the number of observations.

The accuracy rate of our system is 100%: the system ultimately chose the correct intention in all of the scenarios in which it was tested.

In the first set of experiments, the same footage was given to the system several times, each with different a context, to determine whether the system could use the context alone to disambiguate agents' intentions. We considered three pairs of scenarios: leaving the building on a normal day/evacuating the building, getting a drink from a vending machine/repairing a vending machine, and going to a movie during the day/going to clean the theater at night. Quantitative results can be seen in Table 1.

The second set of experiments was performed in a lobby, and had agents meeting each other and passing each other both with and without contextual information about which of these two activities is more likely. Results for the two agents involved are given in Table 1 for both the context-free meet scenario and the context-assisted meet scenario.

Table 1. Quantitative Evaluation

Scenario (with Context)	Correct Duration [%]
Leave building (Normal)	96.2
Leave building (Evacuation)	96.4
Theater (Cleanup)	87.9
Theater (Movie)	90.9
Vending (Getting Drink)	91.1
Vending (Repair)	91.4
Meet (Unbiased) - Agent 1	65.8
Meet (Unbiased) - Agent 2	72.4
Meet (Biased) - Agent 1	97.8
Meet (Biased) - Agent 2	100.0

We tested intention-based control in two scenarios. In the first, the robot observes a human dropping a bag in hallway, recognizes this as a suspicious activity, and follows the human. In the second scenario, an agent enters an office and sets his bag on the table. Another agent enters the room and steals the bag. The observer robot recognizes the theft and signals a robot in the hallway. The second robot then finds the thief, and follows him. In both cases, the observer robot was able to correctly use intentional information to dispatch the context-appropriate behavior.

Additionally, by using context to cut down on the number of intentions the robot has to consider in its maximum likelihood calculation, we improve the efficiency of our system.

5 Conclusion

In this paper, we proposed a vision-based framework for detecting and tracking humans in video. The trajectories of these entities are passed to an intent recognition module that allows the system to infer the intentions of all of the tracked agents based on the system's spatio-temporal context. We validated our system through several experiments, and used agents' trajectories to correctly infer their intentions in several scenarios. Lastly, we showed that inferred intentions can be used by a robot for decision-making and control.

6 Acknowledgments

This work was supported in part by the Office of Naval Research under grant number N00014-06-1-0611 and by the National Science Foundation Award IIS-0546876.

References

1. Dalal, N., Triggs, B.: Histogram of Oriented Gradients for Human Detection. International Conference on Pattern Recognition. (2005) pp. 886–893
2. Ramanan, D., Forsyth, D., Zisserman, A.: Tracking People by Learning Their Appearances. IEEE PAMI **29** (2007) pp. 65–81
3. Efors, J., Berg, A., Morri, G., Malik, J.: Recognizing action at a distance. In: Intl. Conference on Computer Vision. (2003)
4. Stauffer, C., Grimson, W.: Learning Patterns of Activity using Real-Time Tracking. IEEE Transactions on PAMI **22** (2000) 747–757
5. Tavakkoli, A., Nicolescu, M., Bebis, G.: Automatic Statistical Object Detection for Visual Surveillance. In proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation (2006) 144–148
6. Tax, D., Duin, R.: Support Vector Data Description. Machine Learning **54** (2004) 45–66.
7. J. Platt: Fast Training of Support Vector Machines using Sequential Minimal Optimization. Advances in Kernel Methods - Support Vector Learning. **MIT Press** (1998) 185–208.
8. Osuna, E., Freund, R., Girosi, F.: Improved Training Algorithm for Support Vector Machines. In Proc. Neural Networks in Signal Processing (1997)
9. Tax, D., Laskov, P.: Online SVM Learning: from Classification and Data Description and Back. Neural Networks and Signal Processing (2003) 499–508.
10. Premack, D., Woodruff, G.: Does the chimpanzee have a theory of mind? In: Behavioral and Brain Sciences. Volume 1. (1978) 515–526
11. Iacobini, M., Molnar-Szakacs, I., Gallese, V., Buccino, G., Mazziotta, J., Rizzolatti, G.: Grasping the Intentions of Others with One's Own Mirror Neuron System PLoS Biol (2005) 3(3):e79.