Real-Time Multi-Resolution Blob Tracking

Alexandre R.J. François
Institute for Robotics and Intelligent Systems
University of Southern California
afrancoi@usc.edu

April 2004

Abstract

This paper introduces a new real-time blob tracking algorithm. Segmentation is the first step in many video analysis approaches. A number of successful segmentation techniques extract regions of interest, or blobs, in successive frames. The problem addressed here is that of establishing temporal relationships between blobs, without the use of domain-specific information. These relationships can then be analysed at a higher semantic level. The proposed algorithm maintains a multi-resolution tracking graph that encodes, at each resolution, the temporal relationships of the blobs detected in successive frames. As blob displacement in image space is reduced in lower resolution levels, a coarse-to-fine correspondence hypotheses generation, propagation and refinement approach allows to track not only large, slow blobs but also small, fast blobs. Tracking performance is illustrated on various simple application scenarios using a real-time implementation of an integrated segmentation and tracking system. Blob tracking results are demonstrated on standard video surveillance datasets, as well as real-time ball (and player) tracking results in professional tennis and racquetball videos.

1 Introduction

A large number of works in the vision community have focused on video analysis, especially for video surveillance (see e.g. [7] for an overview, and [1, 2, 3, 4] for recent developments).

Most approaches, involve, as a first step, moving or non-background object segmentation (see e.g. [12, 13, 18]). The output of this stage is, for each frame, a set of labeled regions, or blobs. Establishing temporally consistent labels for the detected blobs falls under the generic domain of tracking. Different levels of tracking, however, should be distinguished, depending on the semantic level of the tracked entities, and the amount of domain-specific knowledge involved.

Some approaches attempt to design algorithms for tracking specific objects in specific situations, thus collapsing all aspects of tracking. The approach followed here relies on the assumption that a robust system can be achieved by using a collection of modules producing inherently "imperfect" results. Figure 1 outlines a possible classification of video data by semantic level, and the generic tasks associated with their computation or inference.

Tracking is addressed in this paper at the lowest possible semantic level, that is in the sense of establishing temporal relationships between segmented generic features (blobs), without the use of domain-dependent information. The proposed tracking algorithm comes at the second step up the semantic ladder, right after the initial segmentation. In particular, the concept of layer (and thus occlusion/disocclusion) is irrelevant at this stage: blobs can only appear, disappear, split and merge (in 2-D image space). Layers must be embeded in higher dimensional space, and should be handled at the next level of processing. Blob appearance/disppearance and split/merge caused by noise, reflections, and shadows, may be analysed to infer trajectories and layers (see e.g. [15]). Reasoning involving higher level, possibly domain-specific knowledge, can then be applied to infer entities, actors (see e.g. [19]) and events to finally allow activity recognition. These issues are addressed in processes of higher semantic levels, and are out of the scope of the reported work.

The goal here is to build a module capable of *establishing, in real-time, temporal relationships between image features belonging to possibly small and fast targets, without domain-dependent information.*

This paper is organized as follows. Section 2 places this work in the rich context of tracking. Section 3 gives an overview of the proposed tracking algorithm. Section 4.1 describes . Section 4 describes the details

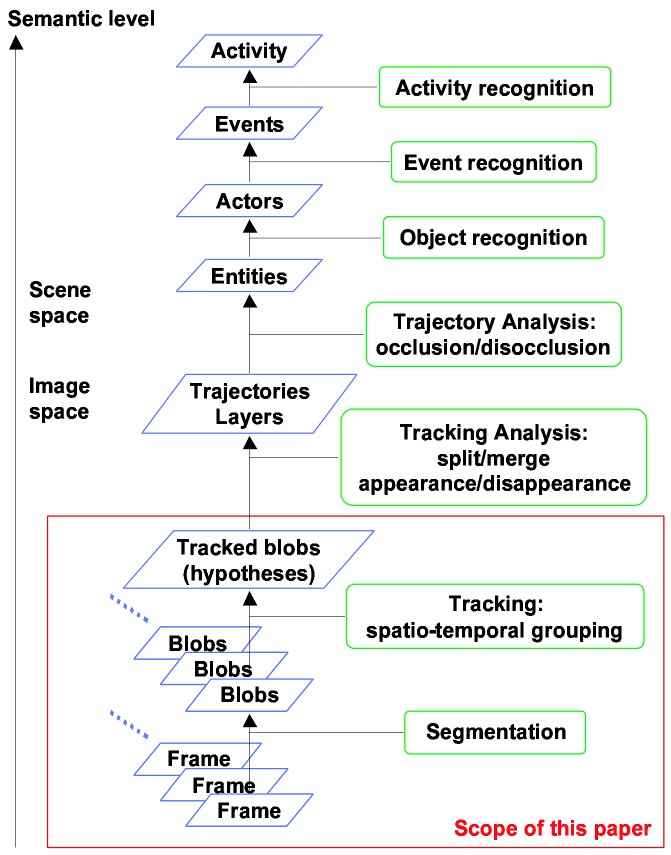

Figure 1: Possible semantic levels in video analysis.

of the multi-resolution blob tracking algorithm, specifically, multi-level tracking graph construction, blob matching hypothesis gereration, and hypothesis propagation and refinement down hierarchy levels. Section 5 presents experimental results. Section 6 offers a summary and perspectives.

2 Related work

A number of techniques have been used successfully for blob tracking, using various properties such as blob size, shape or color, possibly involving higher level or domain-specific knowledge, with open or closed world assumptions.

Kalman filters work well and efficiently when an accurate model of the problem is available. Unfortunately, 2D trajectories as observed in arbitrary video sequences are projections of the 3D trajectories of possibly autonomous and independent agents. When used for tracking (see e.g. [11]), interesting (in the sense as non trivial and thus difficult to track) trajectory changes are modeled as noise. Unless their sensitivity is artificially dampened, Kalman filters, while providing a solid mathematical foundation, fail when the problem becomes interesting. Another limitation of Kalman filters is their inability to represent multiple simultaneous hypotheses. Particle filtering and related algorithms (see e.g. [14]) require no model and support multi-modal tracking, at the cost of higher computational load. More importantly, because of their stochastic nature, they do not allow to track small fast objects, whose corresponding blobs might be far apart, and are more likely to be lost in noise.

Since segmentation (and grouping) is performed separately, a deterministic approach, consisting of establishing temporal relation hypotheses between blobs in successive frames, resulting in a tracking graph, seems a more efficient approach. Previous graph-based algorithms (see e.g. [6]) give satisfactory results on large, slow targets. The difficulty here is to generate *all relevant* hypotheses without generating all *possible* hypotheses. The method presented in this paper makes use of multi-resolution to address this challenge.

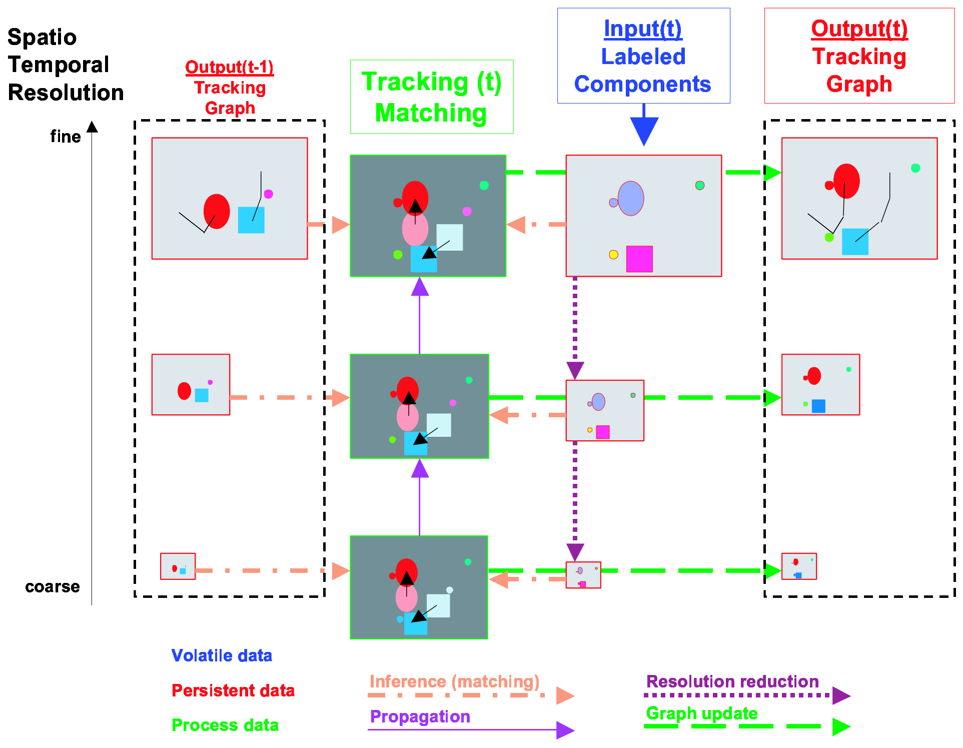

Figure 2: Overview of multi-resolution blob tracking.

3 Approach overview

For each frame, randomly labeled connected components (blobs) are assumed to have been produced by some segmentation technique. In the case of video captured with a fixed camera, adaptive background color model-based temporal segmentation gives foreground blobs that are often satisfactory, although imperfect [17].

The proposed tracking algorithm maintains a multi-resolution tracking graph that encodes, at each resolution, the temporal relationships of the blobs detected in successive frames. Multiple hypotheses are preserved with a measure of confidence, derived from a blob dissimilarity measure. The tracking graph is composed of time slices that encode the information corresponding to a given input frame. Each time slice of the graph is a pyramid, of which the highest resolution level is directly built from the input labeled components image. Lower resolution levels encode the same data with a spatial resolution reduction. As a result, apparent blob motion is reduced in these levels (when interpreted as images and related to the original image space). Links between graph time slices encode temporal relationships between blobs in successive slices. A coarse-to-fine correspondence hypotheses generation, propagation and refinement approach allows to track large, slow blobs as well as small, fast blobs.

Figure 2 depicts the main steps of the algorithm. For each new set of blobs (extracted from a new frame):

- Construct the corresponding time slice of the tracking graph
- Hypothesize blob correspondences at lowest resolution level
- For each level of the graph, by order of increasing resolution:
  - Propagated hypotheses from previous level
  - Refined and/or complete hypotheses

The process is repeated for each level until hypotheses are formed for the original resolution. These steps are explained in detail in the next section.

The use of multi resolution allows to achieve robust, real-time tracking. At the low semantic level considered here, the maximum confidence path combined with unicity constraints can be used to infer consistent labeling of the blobs. Most applications however will require higher level analysis of the graph, as mentioned above.

4 A multi-resolution blob tracking algorithm

4.1 Graph time-slice construction

The tracking graph is composed of time slices that encode the information corresponding to a given input frame. Each time slice of the graph is composed of a multi resolution pyramid (image space) and a set of potential targets (target instances space) for each resolution. A potential target instance is instantiated for each blob in each pyramid level. The temporal relations between blobs, or targets, are encoded as graph edges between the potential targets in consecutive slices for a same level. Each edge is labeled with a measure of confidence computed using a blob dissimilarity measure. Each target also points to the corresponding targets in the levels just above and just below, in the same time slice.

Each element in a pyramid level–called pixels hereafter, although they are not strictly speaking picture elements–refers to a potential target. The highest resolution level of the pyramid (by convention level $l = 0$) is directly built from the input labeled components image. Thus, lower resolution levels (by convention levels $l = 1 \dots n - 1$, where n is the number of pyramid levels) encode the same data with a spatial resolution reduction. Apparent blob motion is therefore slowed down in these levels (when interpreted as images and related to the original image space).

Lower levels are built by reducing the resolution by a factor 2. Four pixel blocs in one level are mapped to one pixel in the next lower resolution level. Depending on the method used for downsampling, the level of detail and noise can also be reduced, although it is not necessarily desirable. It is clear that large blobs will be preserved through the pyramid. Furthermore, two corresponding large slow blobs at the lowest resolution of two consecutive time slices will have most of their pixels overlapping (if only because the overall number of pixels is dropped dramatically compared to the original resolution). For these blobs, matching the new target with the one corresponding to the largest overlap area yields the right solution in most cases. Consequently, large slow blobs do not require a large number of pyramid levels for accurate robust tracking.

Small, potentially fast moving blobs are more challenging. Small blobs might be noise, but might also be important features of the scene (e.g. in a tennis match, it is imperative not to discard the ball as noise). In accordance with the conclusions of [16], discerning between small objects and noise (and shadows) is not addressed at the segmentation stage. Rather, segmentation, being a very low level process, will give as good an output as possible, but will never be "perfect" in the sense of the final desired output of a whole system. The tracking stage must be designed to perform under "imperfect" input conditions.

If the downsampling method used erodes the blobs, small blobs will not be present in lower pyramid levels, and matching hypotheses taking advantage of the apparent motion reduction will not be possible. Instead, the following downsampling rule is used:

A graph pixel is set to point to the potential target corresponding to the the majority target (non-background) among the four mapping pixels of the higher level.

Consequently, even an extremely isolated one pixel blob in the input resolution will be preserved in all the pyramid levels. In general however, small blobs will frequently be merged with other close blobs because of the reduction of resolution. A more elaborate scheme is then needed for establishing hypotheses (blob matching) and to propagate and refine these hypotheses in the coarse-to-fine process. These aspects are described in the next two sections.

Note that the minimum size and maximum apparent velocity of potential targets dictate the number of resolution levels to use. In the most general case, complete (or close to complete) pyramids will be used without significantly increasing the computational load, and without affecting the outcome of the tracking when they are not strictly needed.

4.2 Blob matching

This section elaborates on the generation of target instance correspondence hypotheses across time, i.e. blob matching. Once each level is initialized for the new graph time slice, starting at the coarsest level, matching hypotheses are generated, resulting in the instantiation of links between known tracked targets in the previous time slice, and potential targets in the current time slice. Hypotheses are given a measure of confidence derived from a blob dissimilarity measure. Only plausible hypotheses are formed (by specifying a threshold on the measure of confidence).

As argued above, with the multi-resolution approach, a simple similarity based on overlap area yields correct hypotheses in most cases for large slow blobs. The confidence measure used here is an approximation, based on normalized area difference and centroid distance.

The description for a target instance (blob) a_t^l at level l and time t includes its area $A(a_t^l)$ (pixel count) and a centroid $c(a_t^l)$, computed as:

$$c(a_t^l) = \frac{1}{A(a_t^l)} \sum_{i=0}^{A(a_t^l)-1} P_i$$

where the P_i are the blob pixels.

Area difference, normalized by the size of the (new) target is the core of the measure of confidence. It is modulated by the distance between the centroids. The size-based dissimilarity $D(a_t^l, b_{t-\Delta t}^l)$ between two target instances (blobs) a_t and $b_{t-\Delta t}$, from the same resolution level l, in two consecutive time slices t and $t-\Delta t$, is defined as:

$$S(a_t^l, b_{t-\Delta t}^l) = \left| \frac{A(a_t^l) - A(b_{t-\Delta t}^l)}{A(a_t^l) + A(b_{t-\Delta t}^l)} \right|$$

Introducing the centroid distance

$$D(a_t^l, b_{t-\Delta t}^l) = \sqrt{(c(a_t^l) - c(b_{t-\Delta t}^l))^2}$$

and a maximum distance threshold D_m proportional to the level's image dimensions

$$D_m = \frac{width(l) + height(l)}{2}$$

yields the following confidence measure for the match:

$$K_1(a_t^l, b_{t-\Delta t}^l) = 1 - \frac{D(a_t^l, b_{t-\Delta t}^l)}{D_m} S(a_t^l, b_{t-\Delta t}^l)$$

when $D(a_t^l, b_{t-\Delta t}^l) \leq D_m$, and $K_1(a_t^l, b_{t-\Delta t}^l) = 0$ otherwise.

Intuitively, the measure K_1 favorizes matching of blobs, in consecutive frames, that are of similar size and spatially close. This distance- and area-based confidence measure handles the relatively easy cases, but does not perform satisfactorily when the blobs are far apart. It is therefore used only at the coarsest resolution level, to form intial hypotheses. In all other levels, the description for a target a_t^l includes a velocity estimate $v(a_t^l)$ computed from the highest confidence hypothesis generated at the previous (coarser) level $l+1$:

$$v(a_t^l) = 2\left[c(a_t^{l+1}) - c(\hat{a}_{t-\Delta t}^{l+1})\right]$$

where $\hat{a}_{t-\Delta t}^{l+1}$ is the target instance corresponding to the highest confidence hypothesis for a_t^{l+1}.

The confidence measure computation at these levels is based on the relative size difference and distance between the new blob's centroid and the *predicted* target centroid position. The distance term used is now:

$$\hat{D}(a_t^l, b_{t-\Delta t}^l) = \sqrt{(c(a_t^l) - (c(b_{t-\Delta t}^l) + \Delta t\, v(b_{t-\Delta t}^l)))^2}$$

and the corresponding confidence measure is:

$$K_2(a_t^l, b_{t-\Delta t}^l) = 1 - \frac{\hat{D}(a_t^l, b_{t-\Delta t}^l)}{D_m} S(a_t^l, b_{t-\Delta t}^l)$$

when $D(a_t^l, b_{t-\Delta t}^l) \leq D_m$, and $K_2(a_t^l, b_{t-\Delta t}^l) = 0$ otherwise.

After all hypotheses corresponding to a non null confidence between the blobs have been formed, some blobs are left unexplained, i.e. no tracking hypothesis could be generated for them. They might correspond to (1) first occurence in a new trajectory, (2) noise, or (3) blobs that the tracking algorithm failed to match (lost trajectory). Case (1) is a natural occurence. Noise is inconsistent with trajectories of tracked objects, and should be rejected. Case (3) should be addressed in the trajectory analysis stage (out of the scope of this paper): if the object is tracked most of the time, an occasional lost match will result in a fragmented trajectory. Further analysis should allow to reconnect trajectory fragments which have consistent properties.

The blob matching phase described above allows to generate only relevant hypotheses for a given level. However, it is the multi-resolution aspect of the algorithm that allows to generate such hypotheses even when the blobs to be matched are similar in appearance (in this case, in size) but far apart in the original images.

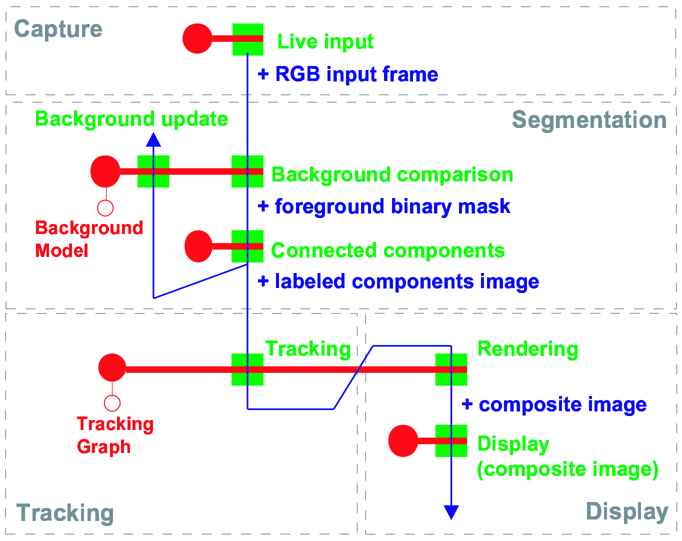

Figure 3: conceptual graph for real-time color background model-based segmentation and multi-resolution graph tracking application.

4.3 Hypothesis propagation and refinement

The fundamental principle behind the use of multi-resolution is to bring those blobs that must be matched, but are far apart, closer together in image space, to the point where the corresponding match confidence K_2 will pass the threshold. The success of the multi resolution approach is then contingent on the design and application of a scheme for propagating and refining temporal relationship hypotheses down the pyramid levels (coarse-to-fine process). Once hypotheses are generated at a given level, they are propagated to the next finer level as follows. For each potential target at the new level:

- Propagate the best estimate hypothesis from the previous level.
- Form additional hypotheses at the current level, as described in the previous section. The new hypotheses might have higher confidence than the hypothesis inherited from the coarser level.
- Update velocity estimate using the highest confidence hypothesis at the current level.

Since each target in a level might correspond to several targets in the finer level (because of blob merging during downsampling), new hypotheses might be generated. The process is repeated in each level until hypotheses have been generated for the highest (original) resolution level.

5 Experimental results

5.1 Real-time implementation

The tracking algorithm was implemented using the MFSM open source architectural middleware [8]. MFSM implements and supports an architectural style for distributed asynchronous parallel processing of generic data streams, called SAI [9]. SAI and MFSM have been used for the design and implementation of various experimental systems. A number of projects ranging from single stream automatic real-time video processing to fully integrated distributed interactive systems mixing live video and graphics are presented in [10].

A real-time testbed system was designed in the SAI architectural style. Figure 3 shows the application graph, color coded following SAI notation conventions: Volatile data, such as video frames, flows down streams (in blue), and is created in processing centers called cells (green squares) from incoming volatile data and relevant persistent data (process parameters). Persistent data, such as the background model for segmentation, and the tracking graph), is held in shared repositories called sources (red disks).

The system was implemented using existing open source components for video input (either from a live color camera or from a video file) and image display. Focusing on sequences taken from a fixed camera, a standard adaptive color (RGB) model-based segmentation module (as described in [5]) was used to segment

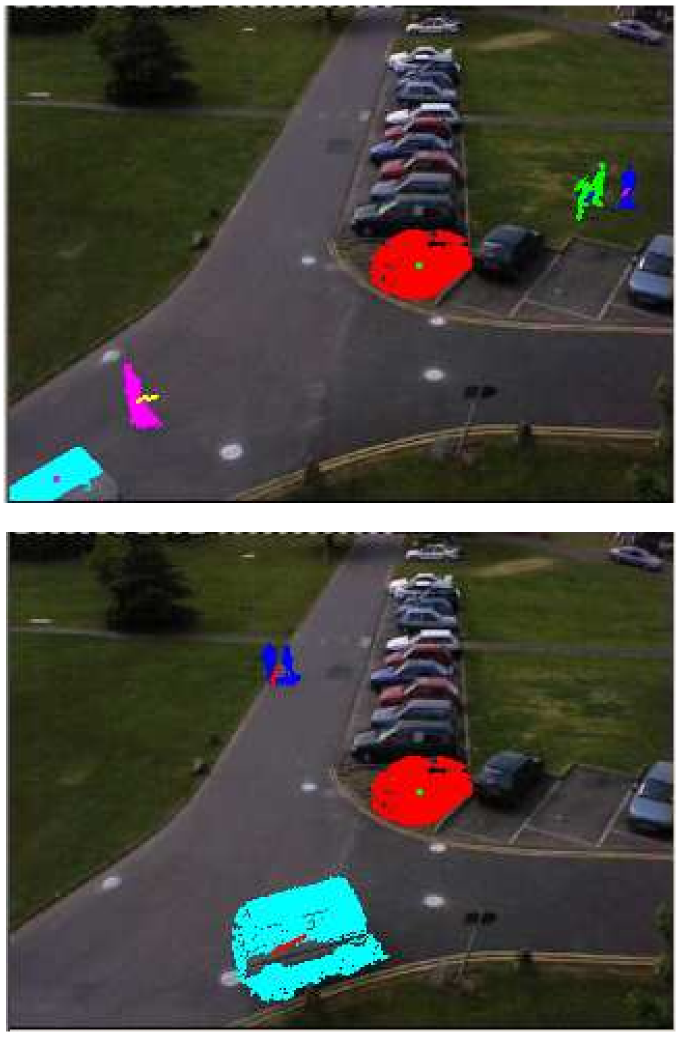

Figure 4: Object segmentation and tracking results in a PETS02 test sequence.

non-background objects in the incoming frames. The multi-resolution blob tracking algorithm described above is at the heart of a tracking cell, that processes each incoming frame and associated segmentation data (volatile data flowing down a stream) to update a tracking graph (persistent structure held in a source). For testing and visualization purposes, a specific renderering cell was developed that computes, from each incoming processed frame and the tracking graph, a composite image showing the blobs and their trajectory. The resulting composite frames are displayed on screen. They may also be saved to disk using for example an existing open source module.

The system performing capture (either from a live color camera or from a video file), segmentation, blob tracking as described above, compositing and on-screen display, typically maintains a 30 fps (320x240) throughput (or better) on state of the art dual-processor machines. Frame capture or loading from a file and image display are source of significant variations in average (main) processor overall load.

5.2 Blob tracking: people, vehicles and balls

In all the tests presented here, the sequences were processed at original resolution of 320x240 pixels, with the same parameters for segmentation and tracking (7 pyramid levels).

As can be expected, the algorithm gives state-of-the-art real-time performance for low-level blob tracking in scenarios involving large, slow blobs (subject of course to the quality of the segmentation results). Figure 4 shows two frames from one of the PETS 2002 test sequences [2], with tracked blobs and their trajectories (maximum confidence path on the tracking graph) over a few frames.

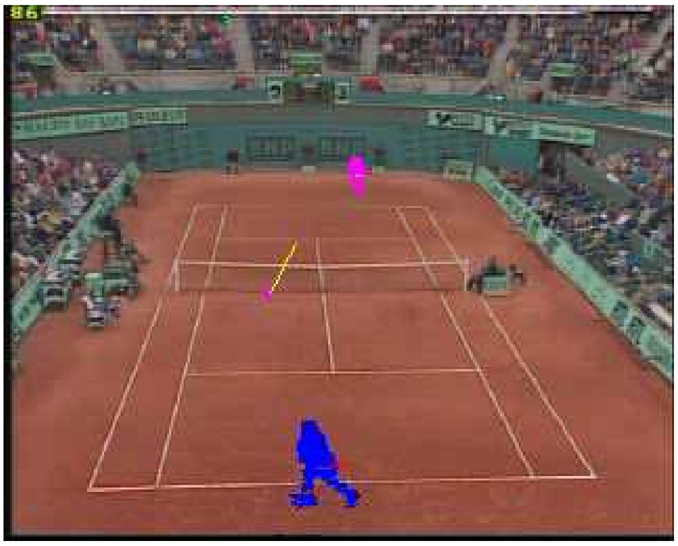

Figure 5: Segmentation and tracking of the players and the ball in professional tennis broadcast video.

Since the strength of the algorithm is its ability to reliably track not only large slow blobs, but also small fast blobs, it was also tested on more challenging sports scenarios: professional tennis and racquetball. The blobs corresponding not only to the players, but also to the ball, are automatically segmented and tracked, in real-time. Figure 5 is a frame from a tennis video with tracked blobs and their trajectories (maximum confidence path on the tracking graph) over a few frames. The players are relatively slow, so the corresponding displayed trajectories are often very short. The history buffer length (5 frames) for the trajectory was set to make the ball most visible without cluttering the scene unnecessarily. The players and the ball are correctly tracked. Occasionally, the ball can be lost. Two cases can be distinguished. First, the ball might not be detected (no blob is produced by the segmentation) or the ball blob is merged with another blob. This is a segmentation limitation. Second, the relationship between two ball blobs in consecutive frames might not be recognized as the most likely at this low semantic level. In both cases, the analysis of the tracking graph (next analysis level in our approach, and out of the scope of this paper), should resolve these issues.

Figure 6 shows three frames extracted from a racquetball video. Note that racquetball being an indoor court game, where players share a common space, there are more problems caused by shadows, occlusions between the players and the ball, and light reflections on the glass wall. This particular court also has a framed door, creating split or incorrectly connected player blobs in many instances. These issues are specifically not addressed at the semantic level at which tracking is performed here. The most challenging problem that can be addressed at this level is detection and tracking of the ball. The racquetball ball is significantly smaller than the tennis ball, and it is so fast and produces so faint a signal in the images that it is often impossible for a human not familiar with the game, or even with a particular video sequence, to locate it in isolated individual frames. Yet the segmentation and blob tracking system performs as well as can be expected given the quality of the input and the nature of the data.

Apart from being an enabling step for applications requiring higher level processing (e.g. surveillance), tracking performed at such a low semantic level can already be useful in some particular application domains. In videos captured by a static camera, simple background/foregound segmentation is already sufficient information for efficient compression and for background modification or replacement. In tennis and racquetball videos (taken by a static camera), tracking allows to singularize objects of interest, with evident benefits for compression, but also for enhanced visualization. For example, tracing the ball trajectory in real-time is a major help in following plays. Furthermore, intelligent downsampling allows to produce smaller videos (e.g. for web publishing) that keep visible the important details (e.g. the ball). These downdsampled videos also benefit from visualization data such as traced trajectories. The fact that, in the frames illustrating the above results, the ball and its trajectory are still visible on a printout of this paper, supports help convey the relevance of such applications.

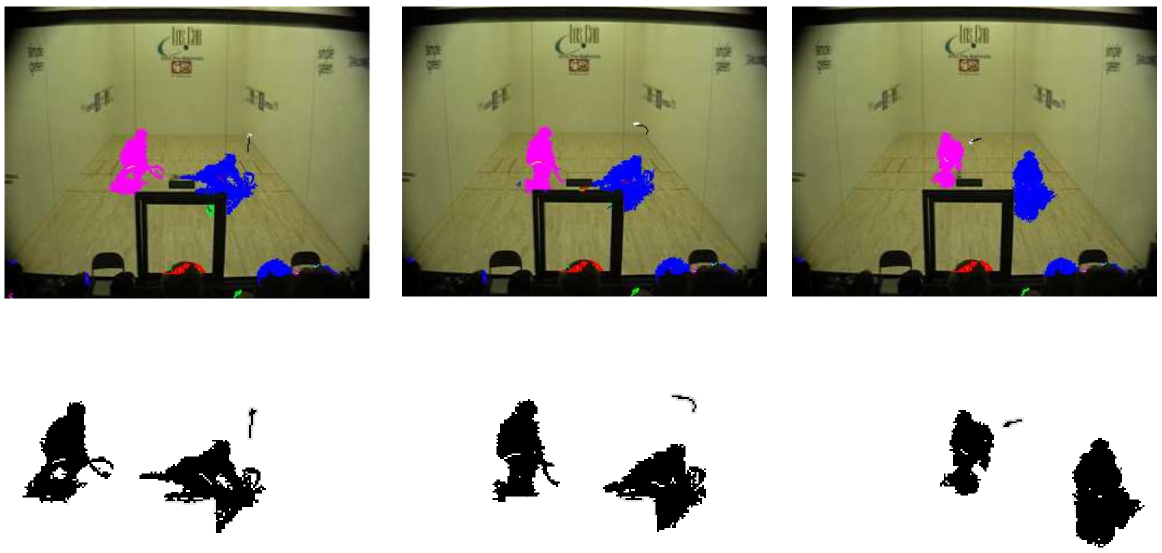

Figure 6: Segmentation and tracking of the players and the ball in professional racquetball video. The frames on the left are the original composite output frames. At the right of each frame, a binary mask corresponding to the players and the ball blobs and trajectories is shown, as manually extracted *from the output frames*, in order to highlight the result data lost to publication artifacts (color to grey scale conversion, compression and printing).

6 Summary and perspectives

This paper introduced a new real-time blob tracking algorithm. Use of multi resolution, following a coarse-to-fine hypothesis generation, propagation and refinement process, allows to reliably track not only large, slow blobs, but also small fast blobs as well. The algorithm has been implemented and integrated in an extensible, real-time testbed system. It has been tested in a variety of surveillance scenarii (indoors and outdoors, live and recorded) and has produced state-of-the art (or better) results on standard test datasets. It has also been tested on professional tennis and racquetball videos, producing unprecedented real-time tracking results on the ball.

The specifics of the algorithm as presented are characterized by their simplicity, making it especially suitable for integration in real-time systems. More elaborate similarity measures could be used, involving for example blob shape, color distribution, etc. Hypothesis propagation and refinement could certainly be made significantly more complex. Such improvements would probably improve the quality (e.g. by reducing further the number of hypotheses generated) and confidence of the data produced at this low level tracking stage, and are certainly worth investigating.

Trying to address higher level issues (such as trajectory fragmentation and blob split/merge) at this stage would then be very tempting. This work however is motivated by the belief that a robust system can be achieved by combining a collection of "imperfect" modules, rather than in a one stage approach (with one do-it-all algorithm). The presented algorithm is viewed as the second earliest stage in a robust video analysis system, coming right after segmentation. The real-time testbed must be pushed up the semantic ladder, by designing relevant algorithms and developing the corresponding modules.

Acknowledgments

This work was supported in part by the Advanced Research and Development Activity of the U.S. Government under contract No. MDA-908-00-C-0036, and by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

References

[1] PETS01: Second IEEE International Workshop on Performance Evaluations of Tracking and Surveillance, December 2001. URL `http://pets2001.visualsurveillance.org`.

[2] PETS02: Third IEEE International Workshop on Performance Evaluations of Tracking and Surveillance, June 2002. URL `http://pets2002.visualsurveillance.org`.

[3] PETS-ICVS: Fourth IEEE International Workshop on Performance Evaluations of Tracking and Surveillance, March 2003. URL `http://petsicvs.visualsurveillance.org`.

[4] PETS-VS: Joint IEEE International Workshop on Visual Surveillance and Performance Evaluations of Tracking and Surveillance, October 2003. URL `http://vspets.visualsurveillance.org`.

[5] Author. Reference widthheld.

[6] I. Cohen and G. Medioni. Detecting and tracking moving objects in video surveillance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages II: 319–325. Fort Collins, CO, 1999.

[7] R.T. Collins, A.J. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):745–746, August 2000.

[8] A. R.J. François. Modular Flow Scheduling Middleware. URL `http://mfsm.sourceForge.net`.

[9] A. R.J. François. A hybrid architectural style for distributed parallel processing of generic data streams. In *Proc. Int. Conf. Software Engineering*. Edinburgh, Scotland, UK, May 2004.

[10] A. R.J. François. Software Architecture for Computer Vision. In G. Medioni and S.B. Kang, editors, *Emerging Topics in Computer Vision*. Prentice Hall, 2004.

[11] T. Frank, M. Haag, H. Kollnig, and H.H. Nagel. Tracking occluded vehicles in traffic scenes. In *Proc. European Conference on Computer Vision*, pages II:485–494. Cambridge, UK, 1996.

[12] I. Haritaoglu, D. Harwood, and L.S. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[13] S. Hongeng and R. Nevatia. Multi-agent event recognition. In *Proc. International Conference on Computer Vision*, pages II: 84–91. Vancouver, BC, Canada, 2001.

[14] M. Isard and A. Blake. C-conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

[15] P. Kornprobst and G. Medioni. Tracking segmented objects using tensor voting. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages II: 118–125. Hilton Head, SC, 2000.

[16] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Proc. European Conference on Computer Vision*, pages I: 661–675. Copenhagen, Denmark, 2002.

[17] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proceedingsz of the International Conference on Computer Vision*, pages 255–261, 1999.

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

[19] T. Zhao, R. Nevatia, and F. Lv. Segmentation and tracking of multiple humans in complex situations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages II:194–201. Kauai, HI, 2001.