Video Tracking Improvement Using Context-Based Information

A.M. Sánchez, M.A. Patricio, J. García and J.M. Molina
Universidad Carlos III de Madrid
Computer Science Department
Applied Artificial Intelligence Group
Avda. Universidad Carlos III 22, 28270 Colmenarejo (Madrid)
{amsmonte,mpatrici,jgherrer}@inf.uc3m.com,molina@ia.uc3m.es

Abstract—Video target Tracking is a complex task, specially when the tracking system is expected to work well in different scenarios. For this reason, this paper proposes an architecture based on a two layer image-processing modules: General Tracking Layer (GTL) and Context Layer (CL). GTL describes a generic multipurpose tracking process for video surveillance systems. CL is designed as a symbolic reasoning system that manages the symbolic interface data between GTL modules in order to assess a specific situation and take the appropriate decision. CL intervenes at three different stages of the tracking process, these are initialization, association and update. Our architecture has been tested in two different scenarios to show the advantages in improved performance and output continuity.

Keywords: Visual surveillance, Tracking System, Context-based information

I. INTRODUCTION

Tracking based on video sequences is an important task for computer vision [1], particularly for visual-based surveillance systems [2], [3]. Generically, we can say this process follows the structure of a Multi-Target Tracking application (MTT): extracting detections, gating detections with existing tracks, matching detections to tracks and updating them, and initializing tracks.

However, Target Tracking is not an easy task, the particular conditions for operating this type of systems require quite specialized solutions. It must handle complex situations and interactions such as passing, occlusions and often sudden and extraneous manoeuvres. Each real object may generate multiple observations and collide with other projected objects, so the generation of a continuously reliable and stable output requires a deep effort in design and adjustment. Tracking algorithms are usually the most flexible and parametrical part of vision systems, and practically all of them exploit external information to model the objects and their context. In [4] authors use an evolutive solution to adjust some parameters of their tracking systems. The configuration is done aiming at a trade-off between computational resources, needs of robustness and behavior. For this reason, extensive research has been also carried out on the creation of semi-automatic tools which can help create a large set of ground truth data, necessary for evaluating the performance of the tracking algorithms [5].

In spite of the complexity, an expected feature of these systems is the capability to track and maintain the identity (ID) of all detected objects over time, even if their detected objects split, get occluded or merge with other targets moving with the Field of View (FoV) of the camera [6]. For every mobile object, the analysis trajectory must be improved using context information, in order to achieve a proper behavior analysis process [7].

A lot of research has been carried out in order to improve tracking multiple sports players [8], with the aim of strategy understanding. Some research makes use of observations such as color, texture, shape and motion estimation to improve player tracking [9]. Furthermore, there are researches where contextual information and specific knowledge is used to improve tracking performance. In [10], contextual information is represented by means of 2D polygons on the image, each of them having a list of attributes: IN-OUT zone, NOISY zone, OCCLUSION zone, AREA-OF-INTEREST ZONE, ETC. This type of information helps to take suitable decisions. So, specific rules are typically applied to protect the system against occlusions and initialization of false targets. In [11], the goal is keeping track history consistent before and after the occlusions, even trying to separate the group of people into individuals during the occlusion.

Therefore, it is very usual to develop visual tracking systems which employ contextual information to address specific problems and improve their performance [12]. It can be seen as enrichment of unconstrained output resulting of a general-purpose tracker, in which observations are associated with targets which are updated, created and deleted accordingly. In this contribution we propose a general architecture to define and exploit contextual information of visual trackers to apply them in specific operational conditions.

In our proposed architecture, we aimed to overcome the limitations of a real tracking system, where we have to endow it with generality and flexibility, while at the same time coping with the efficiency requirements mandatory in real-time systems. The architecture is based on a two layer image-processing modules: General Tracking Layer (GTL) and Context Layer (CL). GTL describes a generic multipurpose tracking process for video-surveillance systems. CL is designed as a symbolic reasoning system that manages the symbolic interface data between GTL modules in order to assess a specific scenario and take the appropriate decision.
Next section describes the architecture proposed in this work and gives an overview of the tracking system. Section III shows the design of the Context Layer added to the tracking system, and finally a performance evaluation and some conclusions are presented.

II. ARCHITECTURE

In this section we describe our approach to design video-tracking systems. Our architecture is depicted in Figure 1, it allows the control the tracking system, at three different levels of the tracking, by adding a Symbolic Reasoning Layer over a General Tracking Layer (GTL). GTL is arranged in a pipe-line structure of several modules, shown in Figure 1; it directly interfaces with the image stream coming from a camera and extracts the track information of the mobile objects in the current frame. The interface between adjacent modules in GTL is symbolic data and it is set up so that for each module different algorithms are interchangeable.

A. General Tracking Layer

The main modules of the GTL are the following:

1) A detector process of moving targets.
2) An association process.
3) A prediction process.
4) Blob deleter.
5) Track Updater.

The detector process (1) of moving targets must give a list of blobs found in a frame, this list must contain information about each blob’s size and position. Within the tracking process and continuing with the list of blobs obtained by the previous module, the association process (2) will solve a problem of blob-to-track multi-assignment, where several (or none) blobs may be assigned to the same track and simultaneously several tracks could overlap and share common blobs. So the association problem to solve is the decision of the most proper grouping of blobs and assignment to each track for each frame processed. The prediction process (3) uses the association made by the tracking process and predicts where each track will move to during the next frame, this prediction will be used by the tracking process in order to make the association. The blob deleter (4) module eliminates those blobs that have not been assigned to any track, thus they are considered to be noise. The last main module, the track updater (5), updates the tracks obtained in the last frame, with the information obtained from the previous modules for this frame.

B. Context Layer

General Tracking Layer modules depend on many parameters that should be adjusted for a specific implementation, but that’s not enough specifics scenarios give special conflicts that the tracking system does not know how to solve properly. Context Layer performs symbolic reasoning based on the context data of a specific tracking scenario. CL manages the symbolic interface data between GTL modules with the goal of assessing a specific tracking scenario. CL is designed as a symbolic reasoning system. One of the most employed reasoning systems in industry and services are knowledge-based system (universally known as expert systems). Knowledge-based systems embed a large component of domain-specific knowledge but, differently from other heuristic-based systems, knowledge is represented in an identifiable separate part of the system rather than being dispersed throughout the whole program.

The CL intervenes after three different modules of the tracking system:

- Track Initializer
- Association Process
- Track Updater

After each frame the GTL initializes new tracks, associates blobs (grouping moving pixels) to tracks (real targets) and updates the tracks, the expert system analyzes the solution proposed by GTL modules and modifies, if necessary, the solution. The CL design will be explain in next the section.

In the Track Initializer stage the tracking system visualizes a new group of moving pixels (blobs) which has been analyzing for some iterations and the system decides to create a new track for that group of moving pixels. This is where the knowledge-based system should intervene, allowing that initialization to be done or not allowing it. The decision of the knowledge-based depends on the scenario that is been watched by the tracking system.

During the association process, GTL has a group of moving pixels and it assigns them to one or several tracks that have already been initialized before. Once again the knowledge-based system merging the contextual information and the information obtained from the tracking system, decides whether the tracking system is making the right association or if the tracking system association solution has to be modified, and in that case the knowledge-based system modifies the solution.

Sometimes GTL after having done the association, updates the tracks properties in a way that is not recommended, because it can cause later on a tracking not desired. CL must make some decisions on whether the update has to been done another way.

The algorithms used by the tracking system are not explained in this work, because each step of the tracking system can be implemented in a lot of different ways, and it is not important for the CL how the output of the different stages are obtained. CL only manages the symbolic data between adjacent stages of the GTL layer. Therefore the architecture that we propose can be used with any implementation of the modules involved in the GTL layer.

III. CONTEXT LAYER DESIGN PROCESS

As mentioned before the CL intervenes at three different stages of the GTL layer. These stages are track initializer, association process and track updater. CL system keeps statistics to be able to make decisions. In this section we will give a description of the CL and the improvements it makes on the tracking system.
In order to design the CL a study of the mistakes made by the GTL on several scenarios has been made. During this study we noticed that the problems were made on the three different stages mentioned before. These mistakes were caused when collisions occurred between tracks and when the collision disappeared.

Collisions between players are troublesome as they became one big blob and this can cause errors in:

1) track initializer, with the consequent formation of false tracks;
2) association process, with the consequent loss of target tracks;
3) track updater, with the consequent union of several tracks;

To prevent such problems, the idea pursued in this work is to use context information of the different situations in this scenario, in order to improve the performance of the tracking system at a specific scenario. As we mentioned earlier, Context Layer adopts the general-purpose model for knowledge representation based on a production-system model with forward chaining reasoning. The major two components of the expert system are the knowledge base and the inference engine. The knowledge base contains facts and rules about the subject at hand i.e. the domain the expert system is functioning in. The rules consist of heuristics that enable the human expert to make educated guesses when necessary, to recognize promising approaches to problems, and to deal effectively with erroneous or incomplete data. Expert systems are data-driven reasoning systems which use ‘IF THEN’ rules to deduce a problem solution from initial data. This set of rules and facts is specific of the scenario and of the stage were it is operating.

In order to clarify the design of the Context Layer, we present the design process employing one dataset. The dataset used is from the publicly available CVBASE [13] called ‘Squash tournament’. This scene is set up of videos which were taken on a tournament of recreative players. The videos were recorded in S-VHS video recorder, using a birds-eye view with wide angle lens. The videos were digitized to digital video format with 25 fps, resolution 384x576 and M-JPEG compression. The selected videos are zenithal records of two players playing squash. The players are with close proximity to each other, similar modality of dress, slightly faster movements and constant crossings between them, which makes it a challenging sequence to design the Context Layer.

A. Track Initializer using Context Layer

Before the track initializer takes place, a blob detector algorithm finds new blobs in the image. If these blobs have a minimum size they can be initialize by introducing them to the tracker. It is possible that a blob that was being tracked before, becomes initialize again and considered as a new track. To avoid this before initializing any blob the CL must allow the initialization.

Figure 3 shows one of the problems that the CL can solve.
during initialization. In this example, a collision between the two tracks has just occurred and when the collision ended, thus separating back into two tracks, the two initial tracks are being associated to the same player. And the player left with no track becomes initialize and a new track is given to it.

At this level the CL modifies the tracking algorithm by preventing the initiation of new tracks when the number of tracks equals or is greater than the number of players and it is known that there is at least one collision between tracks. This situation is shown in Figure 3.

![Figure 3](image.jpg)

**Figure 3.** An example where a track has been initialized improperly.

This situation arises after a collision between players, when two players become one blob. After colliding when the players come apart one of the players may take with him the two tracks, leaving the other player as a new blob. GTL will try to initialize a new track, but the CL must notice that after the collision the track was transferred to the other player thus it must prohibited the initialization of a new track until the collision is fixed.

In this scenario and by way of illustration, we are able to establish some of the context information and tracking information as the following facts:

\[
\begin{align*}
&\text{(Number-of-tracks } N) \\
&\text{(Number-of-players } 2) \\
&\text{(Number-of-tries } T) \\
&\text{(Association } A)
\end{align*}
\]

where in \(N\) represents the number of tracks existing before the initialization; \(T\) represents the number of consecutive tries that the tracking system tried to initiate a new track; and \(A\) is the assignment matrix, in which it is possible to see if there is conflict between several tracks.

The following rules are incorporated to the knowledge-based system:

\[
\begin{align*}
\{ & \text{p match-} A \\
& \text{p match-} N \\
& \text{p match-} T \\
\text{if } & \{ \text{All-players } N\} \\
& \{ \text{Conflict } A\} \\
& \rightarrow \text{modify( } N \} \\
\text{forbid-initialization( } T \})
\end{align*}
\]

where predicate \(p\) looks for facts that match assignment matrix \(A\), the number of tracks \(N\) and the number of initialization tries \(T\). All-players function returns true whether there exists the same or more number of tracks than the number of players. Function Conflict indicates if there exists a conflict at least between two tracks, this means if there are two players close enough for the tracking system to consider it as one track. Finally function forbid-initialization prohibit the initialization of a new track and increments the number of consecutive tries.

The usage of these rules will prevent the tracking system from making a mistake by initializing a track that is not necessary. For example in Figure 3, the tracking system has initialized a track for the player on the left, but instead of initializing that track it should have distributed the two tracks that on figure 3 are assigned to the player on the right. So the rules at the initialization level forbids the initialization of new tracks when the number of tracks is equals or greater than the number of players, if and only if the are conflicts between tracks. The conflicts can be described as when two or more tracks share blobs.

These rules by themselves are far from being good enough to improve the tracking system, they need to be complemented with rules at other stages of the tracking system, as association and track update. It is important to notice that the rules described in this paper are specific for a Squash scenario.

B. Association Process using Context Information

This is an essential block for any multi-target tracking system. GTL will assign blobs to tracks, this can be done in different ways depending on the association algorithm that is being used. After a collision between players, when the two players are separating the association process should give a track to each player. However, this does not occur always as it should. Figure 4 shows an example, where after colliding the two players the tracking system assigns both tracks to one player.

The CL will intervene when a blob does not have any track associated to it. The CL will analyze the situation and associate to that blob the track that best fits.

In this scenario, we are able to establish some of the context information and the blob-to-track association as the following facts:

\[
\begin{align*}
&\text{(Number-of-tracks } 2) \\
&\text{(Player-Size } 20) \\
&\text{(Player-Aspect-Ratio } 0.5) \\
&\text{(Track Player1)} \\
&\text{(Track Player2)} \\
&\text{(Association } A)
\end{align*}
\]

where in \(\text{Player1}\) and \(\text{Player2}\) is represented the position, size and kinematics of the players; and \(A\) is the assignment matrix.
Collisions between players are troublesome as they became one big blob (Figure 2.3) creating an error in data association with the consequent loss of target tracks. To prevent such problems, we need some rules based on the actual facts, that try to improve the assignment established by matrix $A$. So, the following rules are incorporated to the knowledge-based system:

\[
\begin{align*}
(p \text{ match-player1} \\
(p \text{ match-player2} \\
(p \text{ match-}A \\
\text{if } \{\text{Conflict player1 player2} \} \\
\{\text{All-Assignment A player1 player2} \} \\
\{\text{Empty-Assignment A} \} \\
\rightarrow \text{modify}( A \leftarrow \text{distribute-players(player1, player2)}) \\
\end{align*}
\]

where predicate $p$ looks for facts that match player1, player2 and assignment matrix $A$; Conflict is a function that returns true if there is a conflict between player1 and player2 (see Figure 6); All-Assignment indicates if two players are assigned to the same one blob; Empty-Assignment is true if there is one blob that has not been assigned to any player (see Figure 4); and distribute-players reassign matrix $A$ based on the players’ information (position, size and kinematics).

The reassignment of tracks to blobs, is done comparing information about the tracks and about the blobs. In this scenario, the sizes and the positions of the tracks and of the blobs are studied in order to make the reassignment. At this stage of the tracking system tracks’ sizes have not been updated yet. The distribute-players function compares the tracks and the blobs’ sizes and positions in order to make the reassignment. Other information like color could be used, however in our scenario, the clothes were similar for both players.

### C. Track Update using Context Information

Once again when two or more players collide becoming one big blob, GTL may not act properly. In this case as the players collide, the tracking system may decide to update each track to a larger track that groups all the players. To avoid this from happening, the CL will have into consideration this information about the targets:

- Tracks’ sizes.
- Number of tracks vs number of players.
- Track’s Life-time.

The tracks’ size is taking into consideration in order to avoid the tracker system to consider two players as one track, like occurs in Figure 2.4. This can be done by not allowing to update the sizes of the tracks involve. This means, that the size and shape of the tracks involved in the collision will freeze until the players separate.

Our system uses the number of tracks and compares it to the number of players to avoid deleting tracks that can be important for the system. For the same reason for each track its life time is checked before letting the GTL delete that track.

In this scenario, we are able to establish some of the context information and tracking information as the following facts:

\[
\begin{align*}
(\text{Track-old-size O}) \\
(\text{Track-new-size N}) \\
(\text{Track T}) \\
(\text{Association A})
\end{align*}
\]

where in $O$ represents the actual track size; $N$ represents the size that the tracking system wants to assign to this track; $T$ represents the track that is going to be updated; and $A$ is the assignment matrix, in which it is possible to see if there is conflict between several tracks.

The following rules are incorporated to the knowledge-based system:

\[
\begin{align*}
(p \text{ match-T} \\
(p \text{ match-A} \\
(p \text{ match-O} \\
(p \text{ match-N} \\
\text{if } \{\text{Is-Conflict A track} \} \\
\{\text{Rare N O} \} \\
\rightarrow \text{modify}( T \leftarrow \text{maintain-size(T O )}) \\
\end{align*}
\]

where predicate $p$ looks for facts that match, track $T$ assignment matrix $A$, the tracks previous size $O$ and the track’s size proposed by the tracking system $N$. Is-Conflict function returns true whether there exists a conflict between the track being updated and any other track. Function Rare returns true if the difference between the two sizes is not a normal behaviour (this function is similarly implemented like in [14] using a fuzzy system). Finally function maintain-size does not allow the tracker to update the player’s size.
IV. PERFORMANCE EVALUATION

In this section simulation experiments, carried out in two different realistic scenarios, are discussed in order to demonstrate the superior performance of a knowledge-based tracking system with respect to a standard tracking system that does not exploit the knowledge-based.

The two scenarios that have been used are the ‘Squash Tournament’ (Figure 5) mentioned earlier, and video sequences of handball matches. Handball match videos (Figure 6) are also from the CVBASE dataset [13] and have the same characteristics than the ‘Squash Tournament’ sequences. Players do not leave the court during the match, there are constant crossings among players with occlusions and disocclusions and the number of objects (players) to track is quite high. These conditions make for a even more challenging sequence to design its particular Context Layer. To evaluate tracking performance the following metrics are adopted:

1) standard deviation of number of tracks;
2) standard deviation of tracks’ size;

The evaluation of our symbolic reasoning system was done by capturing information about the tracks during the tracking process and analyzing it, for a total of 5000 frames. Ground truth was not available and it is inconvenient and labored to extract the ground truth in the used scenarios.

One of the problems we aim to solve is grouping several players in one unique track, as well as not initializing tracks when not necessary. Therefore one metric that can be use to analyze these problems is the number of tracks, so the mean and standard deviation of the number of tracks has been chosen. Table I, shows how the standard deviation for both scenarios has been reduced and how the mean number of players is more approximated to the real number of players (2 for the squash match and 14 for the handball match). It is important to notice that even though the number of players for a handball match is fourteen, the truth is that those fourteen players are not always in the field of view of the camera.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Data</th>
<th>Without CL</th>
<th>With CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squash</td>
<td>Mean</td>
<td>1.49</td>
<td>1.81</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>0.51</td>
<td>0.47</td>
</tr>
<tr>
<td>Handball</td>
<td>Mean</td>
<td>3.82</td>
<td>8.48</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation</td>
<td>2.67</td>
<td>1.74</td>
</tr>
</tbody>
</table>

Table I

NUMBER OF TARGETS

![Figure 7. Number of tracks for a handball game](image)

These results are also shown in Figure 7 and Figure 8. Where it is graphically illustrated the tracking improvement using our proposed architecture.

![Figure 8. Number of tracks for squash game](image)

Another metric used in order to control how the system did improve by not grouping several players in one only track is the track’s size deviation. This metric is used because when grouping many players in one track, the size gets modified.
a lot and consequently the standard deviation of target’s size increases. The values obtained for this situation are shown in Table II, where it is possible to see how the standard deviation of target’s sizes has been reduced by the use of the CL.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Without ACL</th>
<th>With ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Squash</td>
<td>40.60%</td>
<td>29.62%</td>
</tr>
<tr>
<td>Handball</td>
<td>45.59%</td>
<td>34.49%</td>
</tr>
</tbody>
</table>

Table II
STANDARD DEVIATION OF TARGET’S SIZE

A mandatory feature of a tracking system is for the system to be real-time, so the time overload introduced by the CL, is been taken in to consideration and it has been noticed to be barely noticeable, in the case of the squash scenario is 5.50%.

V. CONCLUSIONS

Target Tracking becomes very specific for different scenarios, for example a tracking system developed to operate on a street may not be as good tracking a sport match. For this reason the authors of this work have designed a two layer architecture, in which the first layer is a General Tracking System and second layer (Context Layer (CL)) is added in order to provide the final tracking system with context information and symbolic reasoning about the specific scenario.

The CL intervenes, when a problematic situation occurs, at different stages of the tracking system. So far it intervenes at the initialization, association and update stages.

The performance evaluation shows that the architecture proposed improves the general tracking system. This evaluation was done on two different sports scenarios, a squash match and a handball match. A CL was created for each of the scenarios, in order to provide the tracking system with specific information about the scenario that was being evaluated.

The design and deployment of the CL has proved to soften the errors that the general tracking system made on a specific scenario. A very important observed feature of the CL was the barely noticeable time overload introduced to the tracking system. The tracking improvement and the unnoticeable time overload proves the architecture as a solution to some of the tracking problems.

ACKNOWLEDGMENT

The authors are supported by projects CICYT TEC2005-07186, CAM 15 MADRINET S-0505/TIC/0255 and IM-SERSO AUTOPIA

REFERENCES


