Recognition of Multi-Agent Activities with Petri Nets

Matej Perše, Matej Kristan, Janez Perš, Stanislav Kovačič

Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, 1001 Ljubljana, Slovenia
{*matej.perse*}*@fe.uni-lj.si*

Abstract

We describe the use of Petri Nets (PNs) for the recognition and evaluation of complex multi-agent activities. PNs are built automatically from the activity templates, which are often used by the experts to encode the domain specific knowledge. The original PN formalism is extended to handle the propagation of evidence using the net tokens. The presented approach was tested on several examples of real basketball activities.

1 Introduction

Understanding of "What is going on in the video" has been one of the paramount challenges of the video analysis domain for more than a decade [1]. The main reason for this is the complexity of the natural scenes. This is especially true in the cases when more then one object of interest is present in the scene (multi-agent and/or multi-object activities), as the analysis system should be able to interpret different temporal, spatial and logical relations among them.

There are different stochastic and deterministic inference engines for addressing the problem of high-level semantics, such as Bayesian Networks [2, 3], Hidden Markov Models [4] and Stochastic Grammars. For simple activities whose structure is known in advance and can be easily learned from training data, stochastic systems can be used. On the other hand, for higher-level-activities which include temporal combinations of events, deterministic inference seems to be preferable.

Petri Nets have already been used for recognition of highly-structured events [1, 5, 6]. They have proven to be particularly suited for:

- modeling sequential and concurrent events and their synchronization,
- handling multiple scenarios using the same PN model,
- modeling hierarchical structure of activities, and
- modeling deterministic and stochastic inference of event occurrences.

Our approach extends the work of Lavee et al. [5] and Ghanem et al. [6] in several ways. First, we propose an automatic procedure for building Petri Nets from activity templates. Such templates, built using a user friendly graphical interface, can be used in several different areas of computer vision (e.g. video surveillance and sports analysis) to encode the experts knowledge. Furthermore, we extend the basic PN concept to handle evaluation of the performed activities through the use of tokens as the carriers of the information about *mistakes* in the already observed activity. Finally, we extensively test the proposed approach on several real examples obtained from the sport domain.

The reminder of the paper is organized as follows: a short introduction to the multi-player activity structure is given in Section 2. In Section 3, we first give a short overview of the PN framework. Next, we present the procedure for building the PNs from an activity template. In Section 4 we present recognition results. Finally, in Section 5 we offer a conclusion and discussion, derived from our experiments.

2 The structure of multi-agent activities

One of the key factors for the correct interpretation of an activity is understanding of its temporal profile. An activity is comprised of several elements/actions (e.g. in basketball these elements are *player motion, dribbling, passing, shooting, screening, rebounding, team starting formation*, etc.), which have to be executed in a precise temporal order [7]. The idea behind our approach is that it is possible to establish the overall score and current stage of the observed activity by evaluating how well have its elements been performed and what where the temporal relations among them.

3 Building the PN Activity model

Petri Nets have proven to be one of the most useful formalisms for modeling systems containing concurrent processes. They were used in several different applications for recognition and interpretation of video data [1, 5, 6].

Our automatic process for building PNs from activity templates consists of two steps:

- First, the *action chains* [5], which represent individual actions are constructed.
- Next, the imposed temporal constraints are integrated into the PN activity model. This way the action chains are linked together by using the knowledge about temporal relations between elements.

To be able to evaluate the overall performance of the observed activity, we also derived a mechanism that allows the propagation of information about the activity through the network.

3.1 The Petri Net formalism

Petri Net model is graphically represented by a directed bipartite graph (see Figure 3) in which the two types of nodes (places P and transitions T) are drawn as circles, and either bars or boxes, respectively [8]. Formally, it can be described as a 6-tuple

$$PN = \{P, T, I, O, H, M_0\}. \quad (1)$$

The arcs of the graph are classified (with respect to transitions) as: input arcs (I) - arrow-headed arcs from places to transitions, output arcs (O) - arrow-headed arcs from transitions to places, inhibitor arcs (I) - circle-headed arcs from places to transitions. Places can contain tokens that are drawn as black dots within them. The state of a PN is called marking (M), and is defined by the number of tokens in each place. The initial PN state is called the initial marking (M_0). The state of the net changes, when one or more transitions fire. In this case the new state ($M_{(t+1)}$) of the net is calculated as:

$$M_{(t+1)} = M_{(t)} + (O - I) \cdot E_{(t)}, \quad (2)$$

where $M_{(t)}$ is the state of the network before the transition is fired and $E_{(t)}$ is the vector of the transitions that fired. For further details readers are referred to [8] and [9].

3.2 Building the PN from activity template

Lets take a look at an activity template called "Double Screen" which is presented in [3]. Event hough this is one of the simplest basketball activities one could imagine, it contains several concurrent elements that should be performed in a specific temporal order. To be able to define the temporal relations between elements, we generate synthetic trajectories using the spatio-temporal information from the template and apply the element detectors to those trajectories. This way we obtain the *activity timeline* (Figure 1) which defines the actual time intervals in which the elements occur.

By observing the starting and ending times of the elements, we can define whether the element has to be executed *before*, *within* or *simultaneously* according to other elements from the activity.

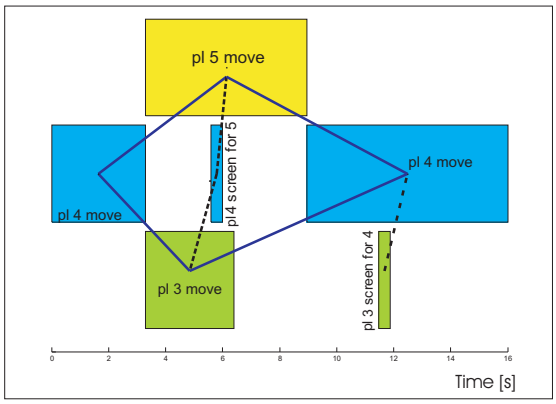

Figure 1: The timeline for the *Double screen* activity described in [3]. The lines represent the learned temporal relations. Full lines represent relation *before* and dashed lines represent relation *within*.

3.2.1 Defining the action chains

Following the previously developed ontologies proposed by Ghanem et.al [6] and Lavee et al. [5], we model the actions as instantaneous time fragment represented as a three-node chains (Figure 2):

- The first node (*precondition place*) represents the preconditions that have to be meet in order for the observed action to begin.
- The second node (*end of action place*) represents the state of a finished action.
- The transition represents the logical state which denotes that an action has or should be observed. When it fires, the token, which represents the state of the observed action, moves from the *precondition place* to the *end of action place*. The firing of the transition occurs after some logical condition was fulfilled (i.e. execution of some action) or due to the fact that the time period allocated for the execution of action has expired. To test the state of the condition we use trajectory-based action detector. If the transition is fired due to the expiration of the allocated time, this means that the action has not been observed and has to be adequately penalized in the final activity score.

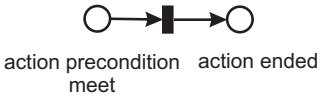

Figure 2: A three-node action chain.

3.2.2 Connecting the action chains into a PN

Once the action chains are created, they are connected into a network using the knowledge about temporal relations between elements that were obtained from the timeline (Figure 1). For this purpose purely logical, null-delay split and join transitions (Figure 3) are added to the network. Additionally, dummy nodes that represent the start and the end of an activity are added.

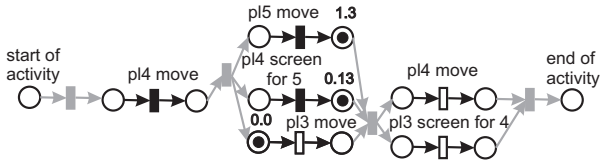

Figure 3: Petri Net model automatically built from *Double Screen* template. Black dots inside places represent tokens which define the current state/marking of the net. The numbers above tokens represent the accumulated penalty. Filled rectangles denote an observed activity, and empty rectangles denote the activity that was not observed yet. Gray elements denote purely logical elements.

3.3 The evaluation scheme

The original PN framework has evolved into several different high-level formalisms among which the most widely used are Coloured Petri Nets and Generalized Stochastic Petri Nets. To obtain the knowledge about how well the observed activity was performed, we follow the idea of the Coloured PNs framework [10], where the tokens are used as carriers of information. Usually, the tokens represent the information about the properties of objects which are part of the modeled system. In our implementation, the tokens are used to carry the information about penalty, which denotes the errors detected in the execution of the already observed activity. The tokens accumulate the information about the overall activity penalty. This penalty is updated every time when a transition is fired. It can be calculated in three different ways depending on the circumstances of a firing:

- If the transition fires when the action preconditions are meet, the penalty is calculated as $X^i_{new} = X^i_{old} + \{1 - S_i\}$, where S_i is the detector response for action i and X^i_{old} is the previous penalty of the token i.
- If the transition fires too early, the penalty is calculated as $X^i_{new} = X^i_{old} + min\{1 - S_i + X_{time}, 0\}$, where X_{time} is the temporal penalty because the element was performed too early. In our implementation the value of the temporal penalty is set to 0.2.
- If the action is not observed at all or it is observed after the predefined temporal interval, the penalty is calculated as $X^i_{new} = X^i_{old} + X_{not\_observed}$, where $X_{not\_observed}$ is the penalty for the action not being observed. In our implementation the value for this penalty is set to 1.

The *logical split* and *logical join* transitions are used to propagate and collect the token information. When n concurrent tokens (X^i_{old}) are joined into a single token (X^i_{new}) the new token penalty is calculated as a sum of penalties of all the incoming tokens $X^i_{new} = \sum_{i=1}^{n} \{X^i_{old}\}$. In order to be able to compare the results from different templates, the total accumulated penalty is normalized with the number of actions in the PN.

4 Experimental results

To test the performance of our activity evaluation scheme, we acquired trajectory segments of 40 basketball activities. All the activities belonged to the same type. However, they were performed in such manner that they ended in different stages of execution. The first 10 examples ended shortly after the start of activity and the last 10 were performed completely. To obtain the trajectory segments, we performed supervised tracking using a probabilistic color-based tracking algorithm [11]. Since the evaluation procedure requires that the roles of players are known (i.e. we have to know which trajectory represents which player role in the template), we cast players into their respective roles using the method described in [12]. The experiments were carried out using a modified version of framework presented in [13].

Our goal was to determine if it is possible to establish the type of the activity that team performed, solely from the final activity penalty. Additionally, we wanted to establish if it is possible to correctly determine the stage at which the activity ended. To do that, we have built four templates representing all activities up to a certain stage of the appropriate basketball play (Slovan1-Slovan4). Additionally, three templates obtained from the basketball literature (52, flex and motion) were used in the evaluation process. Table 1 shows the average penalty of the ten activities belonging to the same length, when they are evaluated using the PNs built from different templates. Additionally, the videos demonstrating the results of the evaluation procedure are available online at http://vision.fe.uni-lj.si/research/SportA/PN.html.

In Table 1, it can seen that the smallest penalties were obtained in cases when both the type and stage of template and observed activity matched. Furthermore, we can observe that even when the

	Slovan1	Slovan2	Slovan3	Slovan4
Slovan1	**0.1400**	0.3184	0.3997	0.4450
Slovan2	0.3523	**0.2043**	0.2766	0.3964
Slovan3	0.6155	0.5291	**0.2354**	0.3509
Slovan4	0.7317	0.6495	0.3968	**0.3368**
52	0.8757	0.8485	0.8448	0.7876
flex	0.9394	0.9120	0.7996	0.8068
motion	0.9507	0.9426	0.9522	0.9333

Table 1: Average activity penalty when evaluating activities using PNs built from activity templates of different types. The smallest penalty is displayed in bold.

stages of the template and activity mismatched, the obtained penalty is smaller than in cases when the types mismatched. This would suggest that the proposed method can be used for activity recognition regardless of the stage at which the activity was concluded. Additionally, the results suggest that by using several templates of different lengths it is possible to determine the correct stage of the observed activity since the penalty is higher in cases when the difference in stages is higher.

5 Conclusion and future work

An approach for automatic evaluation of complex multi-agent activities with Petri Nets was presented. The PNs were built automatically from the activity templates. The building process is comprised from two stages. In the first, three-node chains (two places and one transition) are built for each action. In the second stage, the action chains are connected together so that they encode the complex temporal relations between the actions. In order to evaluate how well were individual actions performed, we applied the trajectory-based action detectors to each transition which represents an action. To obtain the knowledge about the overall activity, a method, which allows the propagation of information about activity performance, was developed.

The obtained experimental results suggest that the presented method can be used to recognize the type and the length of the observed activity and to evaluate how well it was performed according to the activity template.

In our current implementation the maximum durations of actions were set manually and were equal for all actions (60 frames). This is a potential weakness, since obviously actions have different durations. Therefore, the future work will focus on extending the current method to handle different durations of elements. Furthermore, the automatic learning of temporal intervals and the stochastic modeling of these intervals should be introduced to further improve the robustness of the method.

References

[1] C. Castel, L. Chaudron, and C. Tessier. What is going on? A high level interpretation of sequences of images. In *Proc. of the Workshop on Conceptual Descriptions from Images (ECCV, 96)*, pages 13–27, 1996.

[2] S. S. Intille and A. F. Bobick. Recognizing planned, multiperson action. *Computer Vision and Image Understanding: CVIU*, 81(3):414–445, 2001.

[3] M. Perše, J. Perš, M. Kristan, and S. Kovačič. Automatic evaluation of organized basketball activity. In *CVWW' 07, St. Lambrecht, Austria*, pages 11–18, February 2007.

[4] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh. Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model. In *CVPR '05: Proceedings of*, pages 838–845, 2005.

[5] G. Lavee, A. Borzin, E. Rivlin, and M. Rudzsky. Building Petri Nets from Video Event Ontologies. In *ISVC07*, pages 442–451, 2007.

[6] N.M. Ghanem, D.F. DeMenthon, D. Doermann, and L.S. Davis. Representation and Recognition of Events in Surveillance Video Using Petri Nets. In *EventVideo04*, pages 112–120, 2004.

[7] J. Kresse and R. Jablonski. *The Complete Book of Man-To-Man Offense*. Coaches Choice, 2nd edition, 2004.

[8] M. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1998.

[9] Peter J. Haas. *Stochastic Petri Nets - Modelling, Stability, Simulation*. Springer Series in Operations Research and Financial Engineering, 2002.

[10] Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer-Verlang, Berlin, 1997.

[11] M. Kristan, J. Perš, M. Perše, M. Bon, and S. Kovačič. Multiple interacting targets tracking with application to team sports. In *ISPA 05*, pages 322–327, September 2005.

[12] M. Perše, M. Kristan, J. Perš, G. Vučkovič, and S. Kovačič. A trajectory-based analysis of coordinated team activity in a basketball game. *CVIU,In Press*, March 2008.

[13] G. Mušič, T. Löscher, and D. Gradišar. An open Petri net modelling and analysis environment in Matlab. In *I3M International Mediterranean Modelling Multiconference 2006*, pages 123–128, 2006.