Analysis of object interactions in dynamic scenes

Birgit Möller[1] and Stefan Posch[2]

[1] Technical Faculty, Applied Computer Science, University of Bielefeld
[2] Institute of Computer Science, University of Halle
Von-Seckendorff-Platz 1, 06099 Halle/Saale
posch@informatik.uni-halle.de

Abstract. One important source of information in scene understanding is given by actions performed either by human actors or robots. In this paper an approach to recognition and low-level interpretation of actions is presented. Since actions are characterized by specific motion patterns of moving objects, recognition is done by detecting such motion patterns as specific constellations of interactions between moving objects. First of all, motion detection and tracking algorithms are applied to extract correspondences between moving objects in consecutive images of a sequence. Subsequently these are represented with a graph data-structure for further analysis. To detect interactions of moving objects robustly a short history of motion of objects is traced using a finite-state automaton. Finally activities are segmented based on detected interactions. Since robust motion data are required consistency checks and corrections of the acquired motion data are performed in parallel.

1 Introduction

The analysis of dynamic scenes is of growing interest in todays computer vision research. Especially interacting systems often rely on the analysis and interpretation of motions and actions taking place in their work space. Since this analysis is based on image sequences, a large amount of data has to be analyzed to extract relevant information. To simplify this task one might, on the one hand, reduce the data representing image sequences by one single mosaic image [5, 8]. On the other hand, one can focus on preselected, potentially interesting parts of the scene. Often such parts are characterized by actions in the scene projected into the image. In this paper we present an approach to detect events and activities in order to determine such regions of interest.
In the literature events or actions are often defined by physical data of monitored objects such as velocities and motion directions [3] or spatial relations [4]. Mann et al [7] suggest to use dynamic and kinematic models to understand objects' behaviors and interactions. Since the computation of physical features requires highly accurate segmentation data, we in contrast define events as changes in the motion characteristics of moving objects originating from object interactions, e.g. merging, splitting, and start or end of motion. Activities are defined as specific constellations of such events. Similar to [1], we assume that these activities, and thus as well underlying actions, can be detected to a large degree based on interactions of moving objects and without exploitation of high level or scene knowledge [3]. Our analysis is therefore focused on blob constellations

[⋆] This work has been supported by the German Research Foundation (DFG) within SFB 360.

and interactions (e.g. in contrast to [1], where blob shapes or orientations are explored), solely based on motion detection and tracking results presented in [8]. Actions are usually characterized by varying temporal scales. To achieve a robust and flexible recognition, this temporal variance has to be considered. Often this is done using HMM- or parser-based approaches [10,6]. Since the usage of HMMs usually requires large amounts of training data, we use a training-free graph-based approach, similar to the one in [2]. All detected temporal correspondences between moving objects are represented within a graph data-structure. Interactions and activities can then be defined and recognized as specific subgraph constellations. To achieve robust recognition, additionally a finite-state automaton is used to process a short history of detected moving objects and thus verify recognized interactions. According to the states and transitions of the automaton, events are hypothesized, consistency checks on the motion data performed and further analysis steps invoked.
In the following section the basics of motion detection and tracking are outlined as well as the graph data-structure used. The graph-based annotation and correction of motion data using a finite-state automaton is described in section 3, subsequent activity detection in section 4. Results are presented in section 5 and we conclude with some final remarks.

2 Motion data extraction and representation

To detect moving objects[3] within a scene and to track them over time the two-step strategy outlined in [8] is used, which is briefly summarized in the following paragraph. Subsequently the graph data-structure is described in detail.
In a first step intensity residuals between the current image and a reference image[4] are calculated. Since often only small displacements occur between two consecutive images there is the tendency to detect only parts of the moving object, especially in case of objects with homogeneous surfaces. Therefore we use a continuously updated representation of the static scene background as reference image for motion detection. This background representation is generated by integrating static parts of all sequence data into one single (mosaic) image. Given these motion data, subsequent binarization and region segmentation yields moving regions, which are grouped into connected components with regard to spatial neighborhood. These connected components are tracked over time using their intensity histograms as well as the distance of their centroids and difference of size as matching criteria. Tracking connected components instead of single regions in isolation yields more robust tracking data despite variance in segmentation. To handle also cases where connected components could not be matched as a whole, all subsets of their constituting regions are calculated and matched against each other. Thus splitting and merging of connected components can easily be detected and serves as a basis for recognition of object interactions.
Resulting temporal correspondences between moving connected components are

[3] Using 2D image sequences, only projections of moving objects can be detected and tracked, obviously. However, for simplification we use the terms objects and their projections synonymously.
[4] For active cameras images are first aligned using projective transformations, see [8].

represented using a graph data-structure, in the following referred to as *correspondence graph.* For every frame and each connected component a node of this graph is created while matches are represented by edges in between. Matches of region subsets are represented by inserting and connecting the connected components the subsets belong to. The associated match information is stored using a list of the corresponding edge. In this way, we not just link connected components in case they match as a whole, but also if matched only partially. Hence it is possible to detect events like splitting and merging as graph nodes with multiple incoming or outgoing edges.

3 Graph-based data annotation and correction

Based on our assumption that interesting parts of a scene (i.e. parts where actions are expected to occur) can be detected analyzing motion-based activities, we propose a two-step strategy to locate such activities.
In the first step of our approach we detect interactions (events) between moving connected components or region subsets based on the above mentioned correspondence graph. Since matches of connected components as well as region subsets are represented by the graph representation we will in the following refer to both as *components.* Each sequence of linearly linked nodes in the graph where all nodes have in-degree and out-degree equal to one, except the first with in-degree and the last with out-degree larger one, represents a period of continuous[5] motion of a component. Thus to detect events it is sufficient to analyze the first and last node of such a sequence with respect to the number of incoming and outgoing edges. However, due to inaccurate tracking data or variance in segmentation, events like splitting or merging might occur even if no real object interaction takes place. To cope with such situations a short history of motion of components is used as temporal context which is analyzed using a finite-state automaton. This automaton currently consists of ten states and an alphabet of five possible input symbols, listed in the tables below. The transition diagram is shown in figure 1.

State	Description
s_{start}	start of motion
$s_{tracked}$	simple tracked, $T \leq \theta_T$
$s_{tracked\_st}$	simple tracked, $T > \theta_T$
$s_{tracked\_am}$	tracked after merge
$s_{tracked\_as}$	tracked after split
s_{merged}	merge
s_{split}	split
s_{false}	detected, but not in motion
$s_{stopped}$	not detected

Symbol	Edge Constellation	Line Code
t_{match}	1 in - 1 out	solid
t_{split}	1 in - multi out	dashed
t_{merge}	multi in - 1 out	dotted
$t_{mergesplit}$[6]	multi in - multi out	
t_{stop}	not matched	

Each state of the automaton codes either a certain event (gray boxes) or a specific phase of continuous motion (white boxes). If a state coding an event is reached, the corresponding event is hypothesized for further analysis in the activity detection step. In case a state coding a specific phase of motion is reached

[5] In this context we do not consider continuity as continuous velocity or acceleration but periods of motion without any interaction.

[6] This constellation is currently only recognized but not explicitly handled.

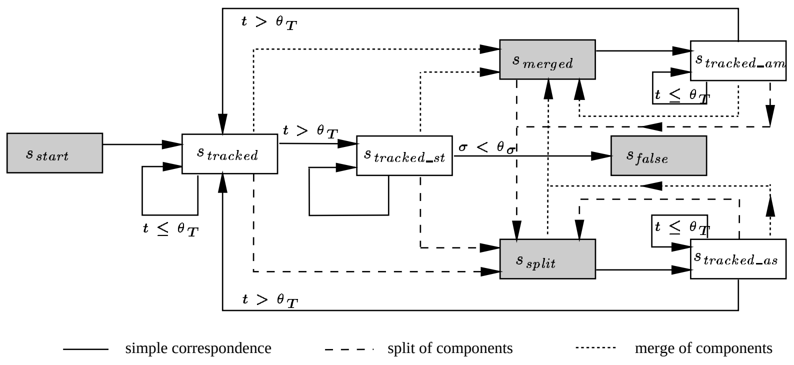

Fig. 1. The finite-state automaton for tracing the short history of motion of objects. For clearness reasons some self-transitions and the state $s_{stopped}$ have been omitted.

special procedures are invoked to perform e.g. consistency checks or to detect components incorrectly classified as moving. The input alphabet of the automaton is defined based on possible constellations of incoming and outgoing edges of graph nodes (see table above). When analyzing an image sequence, the correspondence graph is annotated based on the automaton. In this way each graph node, respectively the corresponding component, is associated with one state of the automaton. Edges between graph nodes are labeled with symbols of the input alphabet where the edge constellation of the source node is evaluated. This analysis of the correspondence graph is now discussed in some detail.

With each frame processed the correspondence graph is updated by inserting the detected matches. Nodes associated with components detected for the first time are labeled with the state s_{start}. However, if a component at a nearby position has been detected a few frames previously, an undetected correspondence is assumed. Hence the graph is corrected accordingly by inserting an additional edge. Each newly detected component is subsequently associated with the state $s_{tracked}$ for a total of θ_T (typically five to eight) frames. If it is further tracked, state $s_{tracked\_st}$ is entered. In this state the component is assumed to have been stably tracked and additional analysis steps are invoked. As an example, the variance σ of its centroid positions during the last frames yields the possibility to identify incorrectly classified moving regions, which have ceased moving and thus show little variance ($\sigma < \theta_\sigma$). If such an incorrectly classified component is detected, the state s_{false} is assigned and it is removed from the set of moving components. Furthermore this observation provides the opportunity for top-down verification and update of the background representation, since these classification errors are due to an inconsistency between the static mosaic image and the current state of the scene: If e.g. an initially static objects starts to move, intensity residuals result at its former position and a seemingly moving region is detected. However, integrating data of the current image to this area the representation can be corrected.

If an interaction is detected during continuous tracking the state of the involved components switches to s_{merged} or s_{split}. Thus, merge or split events are hypothesized, however not accepted straight away. Rather checks for spurious splitting and merging due to segmentation errors are performed first. To this end for subsequent frames the components originating from a split or merge enter the state $s_{tracked\_as}$ ('tracked after split') or $s_{tracked\_am}$ ('tracked after merge'). Only if no further events occur until they are stably tracked again the hypothesized event is accepted and the state $s_{tracked}$ is assigned. If further events are recognized the previously hypothesized event is assumed to be incorrect, probably due to very large variance in segmentation. Therefore these events are rejected and the graph representation is corrected accordingly: In case of a spurious split we directly link the two nodes where the split respectively the merge event had been detected. The subgraph in between is not considered any longer. However, in the future one could merge components of this subgraph which share the same time stamp and thus interpolate the trajectory of the moving object. Similarly spurious merges of components can be amended. As outlined above, the annotation and analysis of the correspondence graph is performed simultaneously to motion detection and tracking. Thus consistency checks and if necessary corrections are continuously performed on the acquired motion data.

4 Activity detection

In the construction scenario of the SFB 360 [9] under consideration a human and a robot in cooperation are supposed to perform an assembly task. Activities are given by more or less complex steps of such tasks, e.g. to add objects to the scene, to remove objects, to move objects from one position to another and to manipulate, assemble or disassemble them. According to our basic assumption that each single activity can be coded as a constellation of previously defined interaction events, the final step in our approach consists of evaluating the formerly generated event constellations as described below.
In the following diagram two activity definitions are shown. Since adding objects to the scene and removing them share the same event constellation, they share the same definition as well and therefore can only be distinguished analyzing the image data directly.

Add / Remove

$$\begin{tikzcd}
\text{START} \arrow[r] & \text{SPLIT} \arrow[r] \arrow[dr] & \text{FALSE} \\
 & & \cdots
\end{tikzcd}$$

Object-Displacement

$$\begin{tikzcd}
\text{START} \arrow[r] & \text{SPLIT} \arrow[r] \arrow[dr] & \text{FALSE} & \\
 & & \text{SPLIT} \arrow[r] \arrow[dr] & \text{FALSE} \\
 & & & \cdots
\end{tikzcd}$$

Obviously, to recognize such an activity, the start of a motion has to be detected first. In case of a 'Remove'-activity, which is explained in detail now, an actor, e.g. the arm of a human, starts moving, grasps an object and leaves the workspace (see figure 3). During monitoring this activity, initially only one single moving region is detected since the object to be removed is still part of the static scene background. When contact has been established between actor and object this moving region grows since actor and object are now moving simultaneously. With growing distance between the moving actor and the former object position,

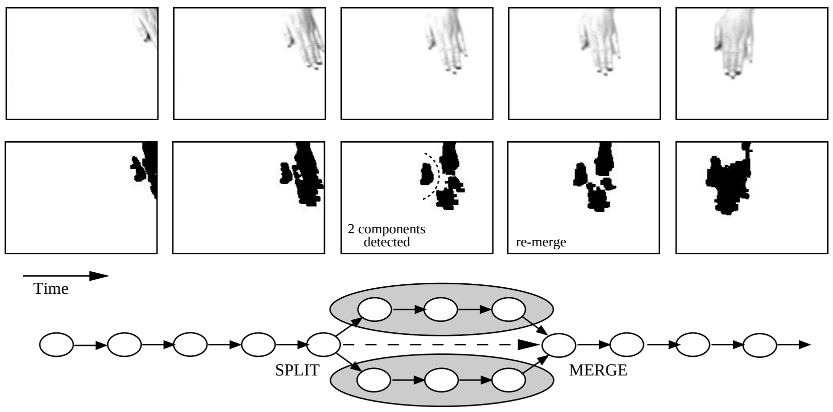

Fig. 2. Robust tracking: Due to a leak of contrast the moving hand is only partially detectable, but graph correction enables tracking of the hand as one object.

the moving component splits up: One resulting component corresponds to the moving actor including the object, the second one results from intensity differences at the initial object position (see section 3). After this split the actor is tracked further while the second component is detected as incorrectly classified and eliminated. Summarizing resulting event constellations one can conclude that initially a new motion is recognized, then the component splits up. One of the components resulting from the split keeps moving, while the second one is classified as non-moving a short period of time later. As a result a 'Remove'-activity is hypothesized to have taken place.
These explanations show that the proposed activity detection is based on interactions of tracked components exclusively. World or context knowledge is used only to interpret the different activities with regard to the scenario under consideration. Thus it is easy to transfer this approach to other scenarios as well, where activities can be identified by moving entities and their interactions.

5 Results

To evaluate the presented algorithms we have chosen image sequences from the the above mentioned scenario where assembly tasks are performed. In this section we present three examples. The first one illustrates results of the graph-based consistency checks, the two others show activity detection results. In each figure the first row contains several images of the underlying image sequence, the second row segmented moving regions/components and in the bottom row the development of the correspondence graph over time is visualized.
Figure 2 shows several images of a hand passing through the scene. Due to low contrast between the bright background and the moving hand it is not possible to completely detect it. Instead multiple moving regions result from the segmentation process. At the beginning of tracking, these regions are correctly grouped into one component which is easily tracked. However, after some frames, more than one component results, caused by segmentation errors, and two independent components are tracked in the following. This error can not be detected until both merge again into one component only a few frames ($t < \theta_T$) later.

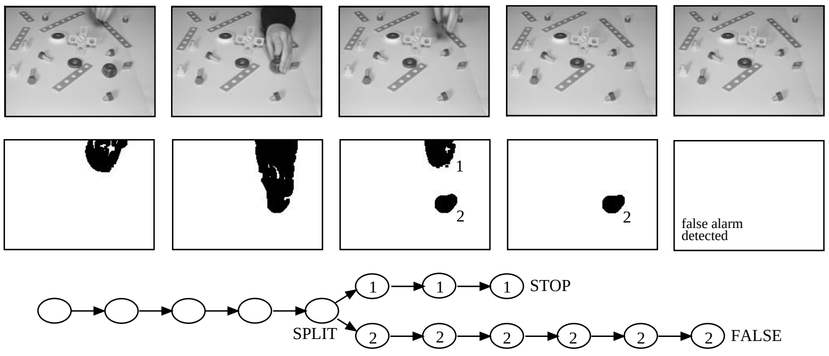

Fig. 3. 'Remove'-activity: The hand grasps the cube and removes it.

Therefore, the events detected are assumed to be due to variance in segmentation and the graph is corrected as shown: The split and merge events are removed (marked gray in figure 2) and the associated graph nodes are connected directly (dashed line).

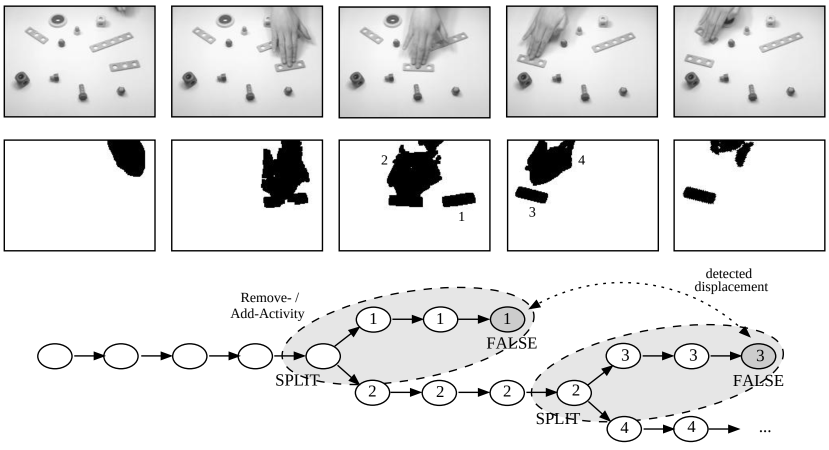

Fig. 4. 'Object Displacement'-activity: The 3-hole-bar is moved from right to left.

In the second sequence (see figure 3) a hand enters the scene removing a cube. As shown, initially one single moving component is detected correctly. After grasping took place, the hand starts to leave the scene and still one component consisting of two regions is detected. This component finally splits into two components, where the second one (id 2) results from intensity differences between the background representation and the current image since the removed cube is still assumed to be part of the background. Nevertheless, after θ_T frames it is identified as incorrectly classified moving, hence eliminated and the background representation is updated accordingly.
The last example in figure 4 illustrates results from a more complex activity. The 3-hole-bar is moved from right to left. Due to interactions of moving components one can deduce that the moving region appearing initially belongs to an acting entity. The first split and subsequent false alarm indicates adding or removing of

an object. Since the same sequence of events is observed some frames later again, either a second object has been removed or the first object has been moved to this new position. To decide for one of the two cases is currently not possible since to this end image data has to be scrutinized directly with respect to e.g. significant gradient norms or object contours in the region of interest. Nevertheless one might hypothesize that an 'Object-displacement'-activity has taken place which could be verified by high-level analysis.

6 Conclusion and future work

In this paper we presented an approach to detect activities within arbitrary scenes. Based on motion detection and tracking, resulting interactions between moving objects in the scene, called events, are analyzed. Since activities are assumed to consist of specific constellations of events, typical activities of the construction scenario under consideration can be segmented. In conclusion the proposed strategy to scene analysis and event detection provides an opportunity for robust pre-selection of scene parts where more detailed and elaborate analysis steps can be performed. Thus scene understanding and high-level interpretation is possible without the need for analyzing large amounts of data by focussing on interesting regions where activities have been recognized.
This approach might in the future be extended to recognize more complex activities, e.g. complete assemblies, where multiple objects are involved. Further on more work is necessary to extract additional details from analysis steps performed, e.g. to decide whether an object has been added or removed, since both tasks currently share the same event constellation. At last the graph consistency checks should be developed further.

References

1. Matthew Brand. Understanding Manipulation in Video. Proc. of 2nd International Conference on Face and Gesture Recognition (1996)
2. Isaac Cohen and Gérard Medioni. Detection and Tracking of Objects in Airborne Video Imagery. CVPR Workshop on Interpretation of Visual Motion (1998)
3. Gérard Medioni, Ram Nevatia and Isaac Cohen. Event Detection and Analysis from Video Streams. DARPA Image Understanding Workshop (1998), Monterey
4. J. Fernyhough, A.G. Cohn and D.C. Hogg. Building Qualitative Event Models Automatically from Visual Input. ICCV (1998) 350-355
5. M. Irani, P. Anandan. Robust multi-sensor image alignment. PIEEE (1998) 905-921
6. Y. Ivanov, C. Stauffer, A. Bobick and E. Grimson. Video Surveillance of Interactions. Proc. of the CVPR Workshop on Visual Surveillance (1998), Fort Collins, Colorado
7. Richard Mann, Allan Jepson and Jeffrey Mark Siskind. Computational Perception of Scene Dynamics. Computer Vision and Image Understanding (1997) 65(2):113-128
8. Birgit Möller, Stefan Posch. Detection and Tracking of Moving Objects for Mosaic Image Generation. LNCS 2191, Proc. of 23rd DAGM Symposium (2001) 208-215
9. Gert Rickheit and Ipke Wachsmuth. Collaborative Research Centre "Situated Artificial Communicators" at the University of Bielefeld, Germany, Integration of Natural Language and Vision Processing (1996) IV:11-16
10. Junji Yamato, Jun Ohya and Kenichiro Ishii. Recognizing Human Action in Time-Sequential Images using Hidden Markov Models. Proc. of CVPR (1992) 379-385