UNIVERSITÀ
DEGLI STUDI
FIRENZE

PHD PROGRAM IN SMART COMPUTING
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE (DINFO)

Event Detection in Videos

Abdullah Khan

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Event Detection in Videos

Abdullah Khan

Advisor:

Prof. Beatrice Lazzerini

Prof. Luciano Serafini

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Sabrina Senatore, *Università degli Studi di Salerno*
Prof. Alessandra Russo, *Imperial College, London*

XXXII cycle — October 2020

To XXXXX

Acknowledgement

I am grateful to my supervisors, Professor. Beatrice Lazzerini and Professor. Luciano Serafini for taking their time over the past three years to make this dissertation possible. I would really like to express my gratitude to Dr.Loris Bozzato, whose help in discussions and implementation phase holds great value for this thesis. I had the opportunity to learn many useful skills from him expected from a researcher. I am also grateful to Dr.Edward Curry, who hosted me in the Insight Center of Data Analytic in Galway, Ireland. I would also like to thank all my colleagues and my friends for always being there for me especially: Carlo Puliafito, Alessandro Renda, Mattia Mogetta, Umar Ibrahim Minhas, Ali Ahmed Khan, Osama bin Tariq, Arafat Shabbir, Talha khan, Salman Saeed. Finally, I would like to thank my parents and sisters for their countless prayers and encouragement.

Abstract

In this age of the digital era, the rapid growth of video content is a common trend. Automated analysis for video content understanding is one of the most challenging and well researched area in the domain of artificial intelligence. In this thesis, we address this issue by analyzing and detecting events in videos. The automatic recognition of events in videos can be formally defined as: "detecting interactions between human-object, object-object or human-human activity in a certain scene". Such events are often referred to as simple events, whereas, complex event detection is even more demanding as it involves complicated interactions among objects in the scene. Complex event detection often provides rich semantic understanding in videos, and thus has excellent prospective for many practical applications, such as entertainment industry, sports analytic, surveillance video analysis, video indexing and retrieval, and many more.

In the context of computer vision, most of the traditional action recognition techniques assign a single label to a video after analyzing the whole video. They use various low-level features with learning models and achieve promising performance. In the past few years, there has been significant progress in the domain of video understanding. For example, supervised learning and efficient deep learning models have been used to classify several possible actions in videos, representing the whole clip with a single label. However, applying supervised learning to understand each frame in a video is time-consuming and expensive, since it requires per-frame labels in videos of the event of interest. To accomplish this task, annotators apply fine-grained labels to videos by manually adding precise labels to every frame in each video. Only then can the model be trained, and that also mostly on atomic action. Training on new event requires the process to be repeated. Also, such approaches lack the potential of interpreting the semantic content associated with complex video events.

To sum up, we believe that understanding of the visual world is not limited to recognizing a specific action class or individual object instances, but also extends to how those objects interact in the scene, which implies recognizing simple and complex events happening in the scene. In this thesis we present an approach for identifying complex events in videos, starting from detection of objects and simple events using a state-of-the-art object detector (YOLO). It is used both to detect and to track objects in video frames. In this way, it is possible to locate moving objects in a video over time in order to enhance both the definition of events involving the objects considered and the activity of event detection. We provide a logic based representation of events by using a realization of the *Event calculus* that allows us to define complex events in terms of logical rules. Axioms of the calculus are encoded in a logic program under Answer Set semantics in order to reason and formulate queries over the extracted events. The applicability of the framework is demonstrated over scenarios like *"occupancy of the handicap slot"*, recognizing different kinds of *"kick"* events in

soccer videos. The results compare favorably with those achieved by the use of deep neural networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increase in availability of data in both structured and unstructured formats is a common trend now a days: on the other hand, information extraction for a meaningful use from this ocean of data is still a challenging task. The interpretation of these data need to be automated in order to be transformed into operational knowledge (Akbar et al., 2017; Khan et al., 2018). In particular, events are mostly important pieces of such knowledge, as they represent activities of unique significance. More precisely, an event is a time-stamped piece of information that represents an occurrence within a system or domain of interest (Etzion et al., 2011). For instance, an event may be a sensor reading, an online activity (e.g., a tweet), a video frame, a financial transaction etc. On the other hand, it may also be a piece of information resulting from some computational process, e.g., a statement representing the fact that one object moves towards another object at a particular time, resulting by feeding the object's (x; y) coordinates to the system over a period of time.

Events occurring in close proximity may be correlated with other events. For instance, in a traffic management application (Akbar et al., 2015), a congestion event may be related to a slow average vehicle speed event, which results in averaging sensor data over a period of time. Such correlations between events may be expressed in rule-like patterns of the form "when the average vehicle speed and average traffic flow are less than the threshold values, then a traffic congestion event occurs". An event may occur instantaneously, or it may occur during the period of time. For instance, a *GPS signal event* occurs instantaneously, whereas a *traffic congestion event* persists over a period of a time. Additionally, events often involve relations between entities. Consider, for instance, an event that involves two objects e.g., "object1 and object2 are moving towards or away from each other at a particular time". In order to understand the events happening in this context, it is not only essential to understand individual object instances, but also extends to how those objects interact in the scene.

1.1 Complex Event Recognition

Complex event recognition is a sub-field of complex event processing (Luckham, 2002) that seeks to detect interesting event patterns in temporal data, which allows scientists and researchers to analyze and extract insights from the data, providing reactive measures promptly. Examples include recognition of human activity from videos (Brendel et al., 2011), business process management (Janiesch et al., 2011), complex event detection from IoT (Internet of things) data streams (Akbar et al., 2017), and so on.

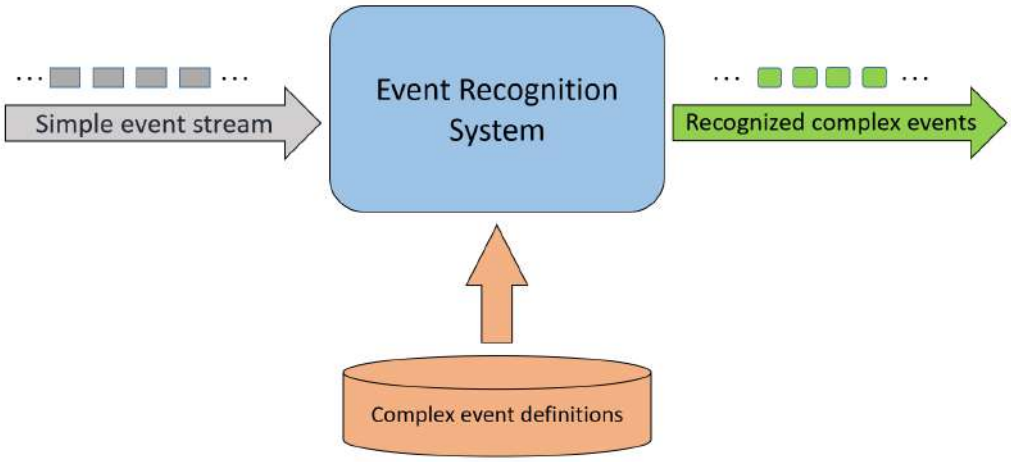


Figure 1.1: Generic pipeline for complex event recognition

Figure 1.1 illustrates a generic pipeline for the complex event recognition process. The input stream to an event recognition system consists of *basic*, or *simple events*. The occurrence of these *simple events* is not dependent on other events. The core of the event recognition system consists of a reasoning engine that matches the input stream against the event patterns defined in terms of logical rules (complex event definitions), which get fired only when several conditions on the event description are met.

The goal of complex event detection from unstructured data formats (e.g., videos and images) consists in identifying and localizing specified spatio-temporal patterns in such data, where each pattern represents a significant event. Understanding of complex events taking place in videos is a challenging problem for the vision community due to factors such as, e.g. background clutter and variations in pose, illumination, and camera point of view. Moreover, complex video sequences (Budvytis et al., 2010) may contain many activities and involve multiple interac-

tions between objects: the recognition of such composite interactions is thus more demanding than the classification of single actions. Humans can infer what happened in a video within a few frames. They can also understand complex situations happening between a pair of frames. But for computers to understand complex interactions between objects and come up with a piece of meaningful information is still a challenging problem.

In fact, event recognition is considered to be a paradigm for all computer vision tasks (Ahmad et al., 2018)(Ahmad and Conci, 2019), also because of its wide applicability to different real-world scenarios. Advances in deep convolutional neural networks (CNN) in recent times have mostly focused on developing end-to-end black box architectures that achieve high accuracy in recognizing events. However, the major drawback of such approaches is the interpretability of the model (Prapas et al., 2018). For complex events, humans can analyze the properties of complex actions and inject some semantic knowledge to extract semantically meaningful events. Whereas, CNN-based black box architectures often rely on high accuracy given the event is happening or not.

1.2 Explainability

An exciting and relatively recent development is the uptake of machine learning in the complex real-world applications, where the major goal is to obtain semantically rich interpretation from observed data. Usually, CNN based black box architectures are trained with regard to high accuracy, but recently there is also a high demand for understanding the way a specific model operates and the underlying reasons for the predicted output. One motivation behind this is that researchers increasingly adopt ML(Machine Learning) for optimizing and predicting specific events, where the model lacks *details* or *explainability* in reaching the predicted output.

The core of the black box architectures is the basic ML chain, in which a model is learned from given input data and with a specific learning paradigm, yielding output results utilizing the learned model. In order to derive a semantically rich interpretation of the outcome, either the output results or the model must be explained. Injecting semantic definition and structural knowledge in these black box architectures is rather difficult. So, this motivates us to start from the basic building blocks and rebuild a system that allows exploiting the semantic knowledge about events using logical frameworks to recognize the intermediate and high-level complex events. The aspects involved in injecting semantic knowledge into the event definition depend upon the type of knowledge and representation of knowledge. Knowledge is often given in terms of mathematical equations, as relations between instances, in the form of rules or constraints. It can also be represented in the form of ontologies.

1.3 Motivation of this Thesis

The event recognition system can be divided into two broad categories, keeping in view different processing mechanisms and event specification languages. The first category consists of a system that descents from classic database theory, operates by executing "queries"on stand-alone or real-time data. A well known open-source event specification language for this category is ESPER[1] for performing algebraic operations on data streams, like aggregates, joins, selection and so on. The second approach for event recognition systems consists of *logical rules* to represent event patterns and rely on logical inferences to perform complex event recognition. Some examples of systems in this category are ETALIS (Anicic et al., 2011), ASP (Brewka et al., 2011), RTEC (Artikis et al., 2014). Logic-based systems have several advantages over the non-logic-based ones. For example, logic-based systems exhibit a formal, declarative semantics, whereas non-logic based event recognition systems typically have informal procedural semantics (Cugola and Margara, 2010; Bry and Eckert, 2007). More importantly, logic-based event recognition systems allow to represent and reason with complex relations between entities and utilize rich background knowledge, contrary to non-logic-based systems.

Logic-based systems require a framework that models the effects of event occurrences on properties of a time-evolving system, as well as the duration of such effects. This is a well-studied problem in the field of artificial intelligence, and several temporal logical formalisms exist, designed precisely for that task, such as the Event Calculus (Kowalski and Sergot, 1986). Such formalisms may be easily incorporated in logic-based event recognition systems, in contrast to non-logic-based systems. The use of Event Calculus for event recognition is performed with success in the past for several challenging real-world applications (Patroumpas et al., 2017, 2015). However, learning Event Calculus theories that allow us to define complex events in terms of logical rules where axioms of the calculus are encoded in a logic program under Answer Set semantics in order to reason and formulate queries over the extracted events still remains a challenging task, while dealing with video data.

1.4 Problem Description

The focus of our work is to recognize complex events from the scene, starting from the simple facts that are detectable from the visual information in the input video frames. In particular, complex events can be characterized by conditions that may involve the validity of certain situational variables and temporal facts: thus, their recognition is not limited to verifying that certain sets of atomic events occur in the

[1] see `https://www.espertech.com/esper/`

scene, but reasoning on different aspects of the represented situation is required in general.

Requirements

To process complex event scenario, the following requirements are tackled in this thesis with in an event-based paradigm.

- A system which supports the detection of meaningful events occurring in a video.
- A system which exploits the state-of-the-art object detector, providing candidate objects for the recognition of events in the video.
- A logical machinery, to encode and satisfy the rules used to represent the defined complex events.

Research questions

The requirements mentioned above can be further presented into following research questions.

- How can we perform object-detection and tracking for the extraction of basic facts from the video?
- How can we exploit the basic facts extracted as simple events from visual data to represent complex events using logical reasoning?
- How can we reason and formulate queries over the defined complex events?

1.5 Thesis Contribution

In this thesis, we approach the event recognition problem by aiming at bridging the gap between the methods based on deep learning, used in the extraction of events from the visual data, and logical reasoning, used for complex inferences on event conditions. Intuitively, the goal of such hybrid solution is the possibility to exploit the accuracy of learning and the rich semantic characterization of a logic based representation for complex events. To achieve this objective, in this work we propose a framework which combines both aspects: first, we make use of the state-of-the-art object detector YOLO (You only look once)(Redmon et al., 2016) for extracting basic information (appearance and movement) about objects from video streams. Events

are then represented inside the logical framework of the Event Calculus (Kowalski and Sergot, 1986), which allows for the definition of complex events: the calculus and events representation are implemented as a logic program interpreted under Answer Set semantics in order to reason and formulate queries about the represented scenario. In this thesis, we demonstrate and evaluate our approach over videos from two real-world use-cases, "occupancy of the parking slots" and "soccer kicks": in particular, we consider clips extracted from soccer matches and parking lots and we apply our framework to classify them on the basis of a set of complex events (namely "corner kick", "free kick", "goal kicks" and "occupancy of the handicap slots" events) that can occur in these videos. We note that understanding of such complex events from videos is a very challenging task due to the dynamics and variation of video sequences (e.g., different camera angles and cuts, no fixed composition of video sequences).

1.6 Structure of the Thesis

The remainder of the thesis is structured as follows:

- **Chapter 2.** This chapter provides the state-of-the-art on event recognition in videos with focus towards the hybrid approaches utilizing deep learning and logical reasoning.
- **Chapter 3.** This chapter provides the necessary background material required for this thesis, starting from objection and tracking and finally brief description of logical reasoning frameworks for event representation.
- **Chapter 4.** This chapter explains two real-world applications considered to show the applicability of our approach and the formal definition of events.
- **Chapter 5.** This chapter explains the extraction of simple events from videos.
- **Chapter 6.** This chapter explains the encoding of the rules under Answer Set semantics in order to reason and formulate queries over the extracted events.
- **Chapter 7.** This Chapter discusses the existing datasets towards event detection in videos and the dataset created to perform experimental evaluation.
- **Chapter 8.** This chapter provides the performance evaluation of the the proposed event recognition system. It also provides the comparison of the results with the state-of-the-art.
- **Chapter 9.** This chapter provides a discussion and conclusion of the presented approach. Here, we also discuss limitations and possible research directions.

Chapter 2

State of the Art

In this thesis, we are interested in recognizing events in the video, with particular attention to the approaches that do event recognition by combining a data driven approach (machine learning and image/video processing) with high level semantic approach (logic, ontologies, and reasoning). For this reason, initially we discuss the approach mostly followed by the computer vision community towards event recognition in videos and the drawbacks attached to them. We then review a number of approaches integrating sub-symbolic with symbolic knowledge, and the usage of logical knowledge to recognize events in the video.

2.1 Video Event Detection

Event recognition in videos mostly focuses on understanding videos by classifying them according to the predefined set of action classes. Many event detection approaches rely on shallow low-level features such as SIFT (Lowe, 2004) for static key frames and MOSIFT (Xu et al., 2014) for videos. A generic pipeline of a video event detection system usually consists of the following procedure: feature extraction, pooling, and classifier training. State-of-the-art shallow video representation makes use of this pipeline, e.g., dense trajectories (Van Gemert et al., 2015), where feature vectors are obtained by tracking densely sampled points and describing the volume around tracklets by histograms of optical flow (HOF) (Van Gemert et al., 2015), histograms of oriented gradients (HOG) (Dalal and Triggs, 2005). To aggregate video-level features, it then applies *Fisher vector* coding (Dalal et al., 2006) on the low-level features to look for a structure from the extracted features using clustering and then pool them in a way to improve classification.

On the other side, Deep Neural Network(DNN) architectures try to model high-level abstraction of data by using model architectures composed of multiple non-linear transformations. Considering the dynamic nature of videos, the encoding of temporal information as input is a common trend: the concept involves extending

the two-dimensional convolution to three dimensions, leading to 3D CNNs, which includes temporal information as a distinct input (Tran et al., 2015; Ma et al., 2016). Another approach to encoding temporal information is through the use of Long-Short Term Memory (LSTM) networks (Ma et al., 2016). One of the most successful frameworks for encoding both spatial and temporal information is the two-stream CNN (Simonyan and Zisserman, 2014). The combination of 3D convolutions and the two-stream approach recently reported for video classification received appreciation from the scientific community. Early attempts adapt 2D convolutional networks to videos through temporal pooling and 3D convolutions (Ji et al., 2012). 3D convolutional networks are now widely adopted for action recognition with the introduction of feature transfer by inflating pre-trained 2D convolutional kernels from image classification models trained on ImageNet through 3D kernels. LSTMs have also shown signs of improved performance over the two-stream network for action recognition (Donahue et al., 2015). In (Ma et al., 2016) authors have fed the CNN features of the neighbouring frames to the LSTM architecture to detect actions happening at every frame. Few DNN-based approaches, NetVLAD (Arandjelovic et al., 2016) and ActionVLAD (Girdhar et al., 2017) caught the eye of the research community, which look for co-relations between a set of simple actions representations.

The major drawback of these approaches lies in the fact that, due to the high capacity of the hidden layers, training them requires large amount of labelled data. Whereas, sometimes it is still difficult for LSTMs to memorise the information of the entire sequence, and the downside of 3D kernels is also their computational complexity. Despite the mentioned concerns, deep learning-based action recognition frameworks have been applied with some success in the domain sports activities, e.g., event detection in soccer videos recently proposed framework (Liu et al., 2017) combines temporal action localization using 3D convolutional networks and play-break (PB) rules for soccer video event detection achieving satisfactory results. In (Jiang et al., 2016) authors construct a deep neural network for soccer video event detection combining CNN and RNN, taking advantage of Convolution Neural Network (CNN) in fully exploiting features and the ability of Recurrent Neural Network (RNN) in dealing with the temporal relation. Most of the end-to-end black box architectures discussed above try to capture the pose, movements that are the part of actions known as atomic actions (e.g., walking, running, surfing, riding etc.), but such methods are not very successful in capturing semantically meaningful representation of actions. Injecting semantic definition and structural knowledge in these approaches is rather difficult: it is of great importance for the model to be interpretable, and this is a part where neural networks fall short.

2.2 Integration of Logical Reasoning and Deep Learning

In AI applications, computers process symbols rather than numbers or letters. In the Symbolic approach, AI applications process strings of characters that represent real-world entities or concepts. The Symbolic approach possess few advantages e.g., it offers good performances in reasoning, and is also able to give explanations and can manipulate complex data structures. The key advantage of symbolic logic-based approaches is that the program easily explains how a certain conclusion is reached and what reasoning steps had been performed. On the other side, the origins of non-symbolic AI come from the attempt to mimic a human brain and its complex network of interconnected neurons. Non-symbolic Deep neural networks can learn high-level abstraction from given input data, they also possess certain advantages e.g., features are automatically deduced and optimally tuned for the desired outcome. Massive parallel computations can be performed using GPUs and are scalable for large volumes of data. However, despite that optimisation process, during learning it is difficult for humans to comprehend how a decision has been made during inference time. Therefore, placing a logic network on top of a deep neural network to learn the relations of those abstractions, can help the system to be able to explain itself. The main challenge attached with these non-symbolic networks is the requirement of large amount of data, whereas small datasets are more prone to over-fitting. Most of the recent work in the domain of AI have either learning capabilities or reasoning capabilities rarely they combine both. One of the major challenges for computer scientists is to develop an effective AI system with a layer of reasoning, logic and learning capabilities.

The integration of neural-symbolic computing aims at building a system where it learns from the environment, and its ability to reason from what has been learned(Garcez et al., 2019). It is evident from many recent studies (Serafini and Garcez, 2016; Donadello et al., 2017; Tran, 2017) integration of logical reasoning with deep learning can be helpful in not only improving the overall efficiency of the neural network, also providing a rich semantic interpretation of the context. This can be obtained by encoding logical knowledge during the training phase of the model. The idea behind this is to extract symbolic knowledge from a specific domain and transfer it to another domain in order to improve the overall learning process. Another way is to place a logic network on top of deep neural network in order to define learn relation of the facts extracted from neural network in making the overall output more explainable. A very recent survey (Besold et al., 2017) on neural symbolic learning and reasoning highlights the main characteristics and few challenges towards the integration of neural-symbolic approach. Research towards the integration of the two have shown promising results in the domain of comput-

ing and cognitive neuroscience. The challenges for neural-symbolic integration lies maintaining the right balance between expressive reasoning and robust learning paradigm.

paper	Keyword
(Wali and Alimi, 2010)	visual descriptors, SVMs, video surveillance
(Parameswaran et al., 2013)	Object recognition, object tracking, feature extraction, video event detection, video surveillance.
(Jiang et al., 2018)	Video Classification, Deep Learning, Framework, CNN LSTM, Fusion
(Bhaskar et al., 2015)	Emotion Classification, Emotions Analysis, Emotion Detection, SVM, Speech Emotion Recognition
(Zhang et al., 2015)	Multimedia event recognition, deep learning, fusion
(Gan et al., 2015)	CNNs , Deep Event Network, spatial-temporal evidences
(Suchan et al., 2018)	Reasoning with video data, answer set programming, object tracking and motion analysis
(Prapas et al., 2018)	Activity Recognition, Activity Reasoning, C3D features,Event Calculus

Table 2.1: Few hybrid approaches towards event detection in videos

2.3 Hybrid Approaches Towards Video Event Detection

In this section, we present an overview of the pertinent research work carried out in recent years following hybrid approaches towards event detection in videos in table 2.1. Research in this area generally falls into two major directions. Mostly the computer vision community follows an approach in which model is learned from given input data and with a specific learning paradigm, yielding output results utilizing the learned model. Recently, some approaches use the information extracted from the computer vision techniques and apply rule-based reasoning to reason on extracted information and come up with semantically rich interpretations about the context. We will briefly review few hybrid approaches mentioned in table 2.1 towards event detection in videos.

In (Wali and Alimi, 2010) authors propose a strategy of combined SVMs learning system for the detection of predefined events in the video. The extraction and synthesis of a suitable visual descriptor based on the movement of objects are used for this purpose. They demonstrate the usefulness of the toolbox in the context

of feature extraction, events learning and detection in a large collection of video surveillance dataset. This paper (Parameswaran et al., 2013) presents an approach for detecting and identifying moving objects by their color and spatial information. The system makes use of the motion of changed regions in tracking multiple moving objects. The proposed model shows promising results for indoor environments and different types of background scenes. Another hybrid approach (Bhaskar et al., 2015) of audio and video has been applied for emotion recognition. The novelty of this approach is the selection of audio and video features as a unique specification for classification. Few more recent hybrid approaches e.g., (Jiang et al., 2018) fuse spatial information, motion information and temporal clues to classify videos of activities that have a general label. (Zhang et al., 2015) recognize complex events in video data by fusing multiple semantic cues, including human actions, objects, and scenes. They focus on complex event detection in videos while also providing the key evidences of the detection result. They also generate a spatial-temporal saliency map and find the key frames in the video which are most indicative of the event. In this work (Gan et al., 2015) propose a deep CNN infrastructure, Deep Event Network(DevNet), that simultaneously detects pre-defined events and provides key spatial-temporal evidences. Taking key frames of videos as input, it detects the event of interest at the video level by aggregating the CNN features of the key frames. The pieces of evidences which recount the detection results, are also automatically localized, both temporally and spatially.

Although the above-mentioned works have shown promising results in their respective domains, they do not have a clear definition of complex events. Generally, they use the term to describe events that contain interactions between different elements. Although multimodal data fusion has been explored, they fail to fuse the information at a semantic level to provide a clear explanation of the result. Finally, in order to have good performance without the integration of human logic, learning-based methods necessitate the consumption of large amounts of data with an expensive annotation process. For clarity, some works provide a formal definition of complex events benefiting from the rule-based reasoning.

Some recent frameworks (Prapas et al., 2018; Suchan et al., 2018), leverages the power of deep learning models to process raw data from a video. The instantaneous inferences made by the deep learning framework are then fed into rule-based engines. Such rule-based engines allows for the definition of events that represent a candidate complex event. For example, in this work (Prapas et al., 2018) authors extract deep learned features from a state-of-the-art 3D convolutional neural network, with which they train an SVM to classify simple actions. For the complex ones, which involve interaction between more than one individual, they use the recognized simple events to generate Event Calculus theories. This process is not very accurate, as transformations that add noise are needed to prepare the input data to

the pre-trained neural network. Another work (Suchan et al., 2018) implements a hybrid architecture for computing visual explanations with video data. They implement the theory for visual explanations combining visual processing for object detection and tracking, and estimating movements in the scene, with ASP (Answer Set Programming) based reasoning about events, objects, and spatial-dynamics over a Movie Dataset. We follow a similar visual pipeline as discussed in (Suchan et al., 2018), besides we provide a logic-based representation of events by using a realization of the Event calculus that allows us to define complex events in terms of logical rules. Axioms of the calculus are encoded in a logic program under Answer Set semantics.

We motivate the significance of semantically-driven methods rooted in knowledge representation and reasoning in addressing research questions pertaining to explainability and interpretability particularly from the viewpoint of high level representation of dynamic visual imagery. Although deep learning solutions have been successful for many applications. We believe that there is a clear need and tremendous potential for hybrid visual sense-making solutions (integrating vision and semantics) towards fulfilling essential responsibilities involving explainability in the domain of Artificial Intelligence.

Chapter 3

Background

This chapter provides some necessary background material for the thesis. We begin with the state-of-the-art in the field of object detection and tracking. We then give a brief overview of logic-based action recognition frameworks and rationale behind our choice of Event Calculus formalism and the fundamentals of Answer Set Programming.

3.1 Object Detection and Tracking

Object Detection

The literature on object detection is enormous and to have a detailed review is out of the scope of this thesis. Hence, we list the most relevant recent works in the domain of object detection. For every object detector the initial pipeline consists of extracting region proposals. These are bounding boxes which potentially contain an object. Work in the domain of region proposals can be divided into two broad categories.

- Selective Search: It is based on grouping of pixels (Uijlings et al., 2013).
- Sliding Window: It is based on objectness percentage inside the window (Alexe et al., 2012).

The *Selective search* algorithm works by computing hierarchical grouping, where similar regions are grouped together based on their: *shape, size, color and texture*. Selective search starts by over-segmenting the image based on the intensity of the pixels. The most similar regions are then merged to generate a single region for the whole image. Finally, similarity is calculated based on the likeness of color, texture, and shape.

The *Sliding Window* algorithm works on the idea of generating grids or windows of multiple images taking into account various aspect ratios (sizes), angles, shapes,

and then inputting them into a neural network for the classification. This whole pipeline is repeated several times, which makes selective search algorithm computationally expensive.

Now we will discuss some of the most recent works in the domain of object detection. Object detection in videos/images aims to detect objects belonging to a predefined class and localize them with bounding boxes in a given video stream. Object detectors based on bounding boxes have seen a steady improvement over the years. The real improvement in the performance of object detection was seen after the use of deep learning techniques. Some of the state-of-the-art approaches for object detection are discussed in the following paragraph.

OverFeat(Alexe et al., 2012) gives one of the first improvements in object detection with deep learning. It is based on a multi-scale sliding window implemented with a Convolutional Neural Network (CNN). More recently, CNN-based object detector was R-CNN (Girshick et al., 2014), which involved a two-stage pipeline: one part of the system provides region proposals, then for each proposal CNN is used for classification. To reduce the computational cost, Region of Interest Pooling is used in FAST R-CNN (Girshick, 2015), leading to efficient results.Furthermore, the most recent object detectors (Redmon et al., 2016; Liu et al., 2016) combine the two tasks of region proposal and classification in one system. Single-shot object detectors, YOLO (You Only Look Once)(Redmon et al., 2016), SSD (single shot multi-box detector)(Liu et al., 2016) significantly improved the detection speed compared to prior object detection systems.

Tracking

Object tracking is one of the important problems of computer vision. The process can be defined as locking on to a moving object and being able to determine if object in the current frame is same as the one in the previous frame. Object tracking mostly works on the following assumption, it starts assigning a unique ID to all possible detections in a frame and the subsequent frames try to carry forward those detections with the same ID. If a certain detected object has moved away from the frame then that ID is dropped. If a new object appears then they start off with a fresh ID. In real world applications one needs to do bounding box detections, so a tracker needs to be combined with a detector. Once we have the bounding box information for an object with a ID in frame 1, there are several algorithms which can be used to assign the IDs in the subsequent frames, we will discuss some of them.

Centroid based ID assignment (Nascimento et al., 1999). Using this technique we can assign IDs by looking at the bounding box centroids. This can be done by calculating centroids for each bounding box in frame 1. In frame 2, it looks at the new centroids and based on the distance from previous centroids it can assign IDs by looking at relative distance. The basic assumption is that frame to frame cen-

troids would only move a little bit. This simple approach works quite well as long as centroids are spaced apart from each other.

Kalman Filter (Li et al., 2010). Kalman Filter allows us to model tracking based on the position and velocity of an object and predict where it is likely to be. It models future position and velocity using gaussian distibution. When it receives a new reading it can use probability to assign the measurement to its prediction and update itself.

Deep Sort Algorithm (Wojke et al., 2017). This algorithm performs tracking based on not just distance, velocity but also what that object being tracked looks like. Deep sort allows us to add this feature by computing deep features for every bounding box and using the similarity between deep features to also factor into the tracking logic.

3.2 Recognising and Reasoning About Events

The formalization of logical reasoning and knowledge representation has evolved over the years, keeping in view the dynamics of changing world. Attempts have been made to automate the process of reasoning about common-sense knowledge, i.e., knowledge about how the world works. Action recognition theories have made a significant progress in defining theoretical foundations to model complex domains. The objective here is to briefly review some of the knowledge representation and reasoning techniques from the action recognition point of view that have been developed and used with success by the Artificial Intelligence community. Some of the famous knowledge-based action recognition formalisms proposed include: Event Calculus (Kowalski and Sergot, 1986), Situation Calculus(McCarthy, 1963), Fluent Calculus, action language C+(Akman et al., 2004), Temporal Action Logics (Doherty et al., 1998). The elements of all these formalisms consists of two entities: events and fluents. Events are instantaneous occurrences, whereas fluents are used to describe the state of the world and they persist in time until affected by an event. Situation and Fluent Calculus use multiple time-line model, where each point in time has a single predecessor point, but it may have multiple successor points. Whereas, Event Calculus consists of a single time-line for the occurrence of events. In the next sections, alongside the short overview of the Situation and Fluent Calculus, we also justify our preferred choice of *Event Calculus* as a logical framework.

Situation Calculus

Situation calculus was first proposed by McCarthy (McCarthy, 1963), later formalized by Lévesque and Reiter (Levesque et al., 1998; Reiter, 2001). It is designed for knowledge representation to cater dynamic applications of the real world. In situa-

tion calculus, the sequence of actions are represented by a term called *situation*. Relation and functions whose truth value varies from situation to situation are termed as (functional and relational) fluents. Main ingredients of the situation calculus include: *actions, situations* and *fluents* to reason about real-world complex scenarios. Two axioms that define every action in situation calculus are called *Successor State* and *Action Precondition Sate*. State axioms define the conditions under which an action is executable and how fluents change after the action is performed respectively. To utilize situation calculus theory in practice, a high-level programming language *Golog* was introduced by Levesque (Lévy and Quantz, 1997). One of the limitations of the original situation calculus lies in the fact that it does not specify the explicit duration of performing an action (Reiter, 2001). Hence, in (De Giacomo et al., 2000), the authors proposed an extension of Situation calculus which explicitly represents time by adding a temporal argument to all instantaneous and concurrent actions.

Fluent Calculus

Fluent calculus was first proposed by Thielscher as a logical framework that defines the *sort of states* in addition to actions, situations and fluents. Each situation in fluent calculus represents a unique state. Unlike situation calculus, fluent calculus maintains a history of actions that have been performed, whereas, the former refers to the actual fluents that hold. To utilize fluent calculus theory in practice, a high-level programming language Flux (Thielscher, 2005) was introduced, which uses the structure of constraint logic programming. It successfully encodes the incomplete states and update the states according to the declared primitive events. The main advantage Flux language has over Golog is that it adopts the progression principle (update axioms, to scale up well during long action sequences performed by agents) in contrast to regression applied in Golog. Unlike situation calculus in fluent calculus, time is parameterized as an argument of fluents and actions and new sorts for concurrent actions.

Discussion and Comparative Study

For many years, situation calculus remained one of the most widely adopted formalism for action recognition; still, many researchers work on it(Arenas et al., 2018; Batusov et al., 2019). Extensions of situation calculus support noisy input and stochastic events (Bacchus et al., 1995). However, a major drawback of this approach lies in the effort required to put theory into practice that limits its applicability in the real-world complex domains. To their credit, both Situation Calculus and Fluent Calculus use single time-line on which the events occur, which is suitable for action recognition where the task is to recognise complex actions in time spaced sequence of simple actions. Fluent Calculus despite its high-level expressiveness and signif-

icant results achieved by Flux programming language, could not provide a unified ontology for most of the problem encountered for complex domains.

In this thesis we employ *Event Calculus* another famous comprehensive action recognition framework, which handles reasoning about action and change in a flexible way. It also expresses the multitude of domains in its ontology. Furthermore, active research in implementing reasoners for this calculus is combined with recent success in other fields of logic programming, resulting in highly effective new reasoners that exploit Answer Set Programming planners.

Event Calculus

Event Calculus (EC) was first introduced by Kowalski and Sergot in (Kowalski and Sergot, 1986) as a logic framework for representing and reasoning about events and their effects. Since then, it has been successfully used in numerous event recognition applications (Artikis et al., 2010),(Artikis et al., 2014), (Khan et al., 2019),(Artikis et al., 2019). Several alternative formalizations and dialects of the Event Calculus have been proposed over the years (Miller and Shanahan, 2002; Lifschitz et al., 2008). Most of these dialects share an ontology consisting of (ordered) time points, events and fluents. *Time points* are simply integers or real numbers. A *fluent* is a property whose truth value may change over time, such as the location of a physical object. The expressions referring to temporal entities that occur over some time interval are called *events*.

After an event occurs, it may change the truth value of a fluent. It is assumed that the value of a fluent is preserved in successive time points, if no event changes its state. In particular, an event can *initiate* a fluent, meaning that the fluent is true after the happening of the event, or *terminate* a fluent, meaning that the occurrence of the event makes the fluent false. The calculus makes use of the predicates listed in Table 3.1. The language provides predicates expressing various states of an event occurrence: *happens* defines the occurrence of an event at a given time point, while *holdsAt* states that a fluent holds in a point in time. The predicates *initiates* and *terminates* specify under which circumstances a fluent is initiated or terminated by an event at a specific time point. We also note some similarities of the chosen approach with other rule based approaches for reasoning on events. For example, Cached Event Calculus reasoner (jREC) presented in (Falcionelli et al., 2017) has been successfully used with the aim to monitor the health status of patients. Etalis (Anicic et al., 2011) provides a rich set of operators for specifying how composite events are derived from primitive ones. A reactive and logic-based version of Event Calculus REC (Reactive Event Calculus) is presented in (Bragaglia et al., 2012) for providing a solid formal background for monitoring declarative properties of events.

Table 3.1: Event Calculus predicates

Basic Predicates	Description
$holdsAt(f,t)$	fluent f is true at time-point t
$happens(e,t)$	event e occurs at time-point t
$initiates(e,f,t)$	if event e occurs at time-point t, then fluent f will be true after t.
$terminates(e,f,t)$	if event e occurs at time-point t, then fluent f will be false after t

Reasoning About EC Theories

In our work, we are interested in exploiting the EC as a complex action recognition framework, which requires a reasoning paradigm able to account for a dynamic, evolving domain. Indeed, complex action recognition focuses on a running execution, which cannot be described by a single action but by a stream of event occurrences. The main objective is to infer how the occurring events impact on the evolution of fluents. Researchers in (Ma et al., 2013; Kim et al., 2009) studied this issue in the context of active temporal datasets, where the dynamic acquisition of new facts changes the validity of timed data. In particular, they developed a Prolog-style reasoning engine based on Answer Set Programming (Eiter et al., 2009). There exist few examples of the recent works which successfully employed ASP for the recognition of action or events in the domain of multimedia (Al Machot et al., 2018; Suchan et al., 2018).

3.3 Answer Set Programming

Answer Set Programming (ASP) is a declarative problem solving approach with strong roots in non-monotonic reasoning and logic programming. It is mostly used for automatic problem solving related to: common-sense reasoning, non-monotonic inferences, modeling reasoning agents and much more. The basic idea behind the use of ASP is to describe the problem by means of a logic program and provide solutions to the problem, represented in the form of models of the program known as (answer sets, or stable models). The description of the problem in ASP is provided in the form of rules, facts and constraints. The encoding of such problem description is fed to the Answer Set Solvers which provide solutions by presenting the alternative stable models of the input program.

The success of ASP lies on its ease of usage as a modeling language, and on the variety of sophisticated algorithms and techniques for evaluating Answer Set programs, which originated from research on computational complexity of reason-

ing. Commonly used Answer Set solvers include: DLV[1], Cmodels[2] and Clingo[3] etc., which are able to deal with large problem instances.

Syntax of Answer Set Programming

The basic component of any ASP program is an atom. An atom is an expression of the form $p(o_1, \ldots, o_n)$, where p is a predicate symbol of arity n, and $o_1 ... o_n$ are n terms belonging to the predicate p, and $n \geq 0$. An ASP Program is composed of a collection of rules of the form given in (Schindlauer, 2006)

$$a_1 \vee ... \vee a_n \leftarrow b_1, ..., b_k \; not \; b_{k+1}, ..., not \; b_m, \qquad n \geq 0, m \geq k \geq 0 \tag{3.1}$$

The equation (2.1) shows the the syntax for disjunctive datalog. In the rule above $a_1, ..., a_n$, $b_1, ..., b_m$ are known as literals. A literal is basically an atom or a negated atom. An ASP rule is divided into two parts, *head* and a *body*. A head is a literal on the left side of the rule and a body is a set of literals on the right side of the rule. In the rule (2.1) we can say that $a_1, ..., a_n$ is the head of rule, while the conjunction $b_1, ..., b_k, not \; b_{k+1}, ..., not \; b_m$ is the body of rule. The head or the body in a rule can be empty. A rule with an empty head literal is known as a *integrity constraint*, whereas a rule with an empty body is called a *fact*.

Applications of Answer Set Programming

ASP applications can be divided into two main categories discussed by the authors in (Falkner et al., 2018; Erdem et al., 2016). Firstly, from the perspective of academic research where real-world data is employed by the research community to test the feasibility of applying ASP-based problem solving paradigm. Then, from the commercial aspect, many companies are investing their resources in ASP to solve business cases with the help of academic people. We will structure the applications of ASP starting from the application areas in the domain of artificial intelligence, i.e. planning, classification, configuration. Afterwards, we will report on the recent and emerging industrial applications of ASP, i.e. healthcare, robotics.

Planning: ASP is applied in many areas of planning such as: planning in robotics, combination of monitoring, diagnosis and replanning (Nouman et al., 2016; Erdem et al., 2015). In (Nguyen et al., 2018) a framework is proposed which comprises a planning module, configuration module and a monitoring module for the extraction of re-usage of phylogenetic trees. ASP-based system was introduced to generate plans for setting up the space shuttle for manoeuvres (Nogueira et al.,

[1]http://www.dlvsystem.com/
[2]http://www.cs.utexas.edu/users/tag/cmodels/
[3]https://potassco.org/

2001). Typically, such plans are prepared for standard situations without taking into account the aspect of failure as it is not possible to take into account all possible eventualities in case of failures. Hence, an ASP-based system was implemented, which generates such plans to help the human controllers. ASP also serves as an excellent framework for realizing knowledge-based planning e.g. conditional planners (Yalciner et al., 2017).

Classification: Based on the need of customers, many systems provide a classification of items or services which best fits the needs of the customer. Following this approach, the application of ASP exist in *e-tourism* and *intelligent call routing*. The use of ASP in the area of classification is successfully applied by Telecom Italia, where the classification system for incoming calls to contact centers of Telecom Italia is implemented by a company called Exeura[4]. The company developed a platform for customer profiling for phone-call routing based on ASP that is known as zLog. The idea behind zLog is to classify customer profiles and try to anticipate their actual needs for creating a personalized experience of customer care service. Such a system makes use of the background data of each customer to anticipate the calls. Operators are enabled to define categories by a decision graph, which is eventually translated to ASP rules. Following this approach, a tourist advisor implemented in the ASP was introduced in the e-tourism portal (Ielpa et al., 2009). Keeping in view the priorities and preferences of the customers, the system automatically selects the vacation packages which best fit the needs of the customer. The objective here is to help the staff of the travel agency in analyzing the best possible solutions based on the need of their clients in short time.

Configuration: The configuration of systems is regarded as one of the most successful applications of the ASP. ASP was successfully employed for the configuration of Linux packages (Gebser et al., 2011). Siemens evaluated ASP for its RECONCILE [5] project. Afterward, Siemens successfully applied to configure parts of a railway safety where specialised configurators failed to provide solutions for some complex real-world instances. Another interesting application of ASP in the area of configuration is the management of the workforce by the companies. If we regard a configuration task as the selection of entities such that certain constraints are satisfied, e.g., the total number of employees, the service provided by each employee, the skill set of employees, the role each and every employee in the company. To accomplish such tasks, ASP is successfully employed by transshipment port company in the city of Gioia Tauro(Ricca et al., 2012).

[4]www.exeura.eu

[5]http://isbi.aau.at/reconcile/

Bioinformatics: In the domain of bioinformatics, for large scale biological networks and datasets, ASP has been successfully employed to detect, explain, and repair errors (Gebser et al., 2010). An abstraction of the protein structure prediction problem is discussed in (Dovier et al., 2009). The use of ASP in genomics studies, such as haplotype inference and phylogenetic inference, is investigated in (Erdem and Türe, 2008). In (Dal Palù et al., 2016) authors employ ASP for the analysis of cancer after searching time-dependent relationships in the genomic and epigenomic. In (Erdem and Oztok, 2015) they show how ASP is used for generating explanations for complex biomedical queries related to drug discovery.

Robotics : ASP has been successfully employed for many robotic applications, such as multi robot coordination, multi robot path finding, human-robot interaction. The manufacturing industry makes use of the ASP to implement an optimal plan for heterogeneous robots to manufacture a specific number of orders in a given time frame (Erdem et al., 2013). Another interesting approach is to use ASP for investigating plan failures while monitoring the execution of a specific plan during monitoring over the Robot Operating System (ROS)(Erdem et al., 2015). In (Havur et al., 2014), for the geometric re-arrangement of movable objects on a cluttered surface, ASP is employed to control the movements of robots on such a surface.

Software engineering and Embedded systems: In software engineering, ASP is successfully applied for testing, particularly for constrained combinatorial testing (Banbara et al., 2017). It turned out that the high-level implementation using ASP showed excellent performance compared to specialised tools for testing. The design of embedded systems is supported by systems synthesis during which a structural representation is given to behavioural description for a specific task. To accomplish this objective, ASP is integrated with background theories developed in the area of embedded systems, allowing more sophisticated description for the complex systems(Neubauer et al., 2017) .

Comparison of ASP to traditional approaches

In this section, we address some of the useful features of the ASP and also highlight some of the advantages of ASP when compared with other existing paradigms like Prolog and Constraint Programming.

Useful Features

ASP comprises of several useful features, few of these features are mentioned here. ASP supports a number of arithmetic symbols such as: addition, subtraction, multiplication, integer division, exponentiation, absolute value, bit-wise AND,

bit-wise OR, bit-wise exclusive complement. It also braces a number of built-in predicates , e.g. equal, not equal, less than, less than or equal to , greater than, greater than or equal to. There also exists an aggregate operation that works on a multi-set weighted literals which evaluates some value. In combination with comparisons, one can extract a truth value from an aggregate's evaluation; thus, obtaining an aggregate atom. It also supports the constraints to a logic program that affects the occurrence of stable models in such a way that it eliminates the stable models that violate the constraint.

ASP and ProLog: The semantics of ASP is more flexible towards multiple views of the world. The same ideas cannot be implemented flexibly in Prolog and can lead to confusion as the numerous possible views may manifest themselves differently, depending on the respective query(De Vos et al., 2005). For example, in ASP terms, Prolog would answer a query on x as true if there is at least one answer set in which x is true. But, there is no mention of which of the answer set is true. Another query on y might also be true, but an additional query would be required to infer if x and y are true simultaneously. The order of rules is flexible in ASP; on the other hand strict in Prolog, change in the rules might cause the working of the program to be less effective or useless. In this sense, ASP is more declarative and its semantics are more robust to the changes in the order of literals in the rules (Brewka et al., 2011).

ASP and Constraint Programming: Modeling of problems into constraints requires a strong mathematical background, and certain level of expertise in constraint modeling. Whereas, ASP is simple in a sense, it was developed with knowledge representation applications in mind and their constructs were designed to capture patterns of natural language statements, definitions, and default negation(Brewka et al., 2011).

Event Calculus in Answer Set Programs

In literature another popular choice closely related ASP for encoding the EC problem is satisfiability (SAT)-based approach from (Mueller, 2008) . Discrete event calculus (DEC) reasoner which uses SAT to solve EC problems can be used to perform automated reasoning and model finding. The limitation of these SAT based approach lies in the fact it can not handle certain recursive axioms of the event calculus, whereas, ASP based approach can compute the full version of the event calculus assuming that the domain is given and finite. Compared to SAT solving, ASP offers many constructs besides negation as failure, and, importantly, allows also problem descriptions with predicates and variables. This can be utilized for generic problem solving. An implementation of the Event Calculus in answer set programs is provided in (Mueller, 2014).The EC axioms determining the relation across fluents and events are defined by the rules that follow.[6]

$$initiated(F,T) \leftarrow happens(E,T), initiates(E,F,T). \tag{3.2}$$

$$terminated(F,T) \leftarrow happens(E,T), terminates(E,F,T). \tag{3.3}$$

$$holdsAt(F,T_1) \leftarrow holdsAt(F,T), \neg terminated(F,T), time(T), T_1 = T+1. \tag{3.4}$$

$$\leftarrow holdsAt(F,T_1), \neg holdsAt(F,T), \neg initiated(F,T), time(T), T_1 = T+1. \tag{3.5}$$

$$holdsAt(F,T_1) \leftarrow happens(E,T), initiates(F,T), time(T), T_1 = T+1. \tag{3.6}$$

$$\leftarrow holdsAt(F,T_1), happens(E,T), terminates(F,T), time(T), T_1 = T+1. \tag{3.7}$$

Axioms (2.2) and (2.3) state that a fluent is *initiated* with the occurrence of an event that *initiates* it, and that fluent will be *terminated* when another event occurs and *terminates* it. Axiom (2.4) states that if a fluent holds at time-point T and is not terminated in T, then the fluent is true at the next time-point T_1. Axiom (2.6) states that if a fluent is initiated by some event that occurs at time-point T, then the fluent is true at T_1. Constraint in Axiom (2.5) state that it can not be that a fluent F that is not initiated nor true at time T becomes true at time $T+1$. Similarly, constraint in axiom (2.7) states that it can not be true that fluent F holds at time $T+1$ if an event happened at time T that terminated F.

[6]We use the DLV syntax of rules (in particular, for the use of functors and number operations in rules).

Chapter 4

Complex Event Detection Using Event Calculus

4.1 Introduction

We propose a generalized approach CEDEC (Complex event detection using event calculus) for detecting and predicting events in videos. Events can be broadly classified into two categories (Simple and Complex) according to their abstraction levels.

(i) The occurrence of simple events are not dependent on other events. For example, in a video scene, the appearance or disappearance of objects from the scene can be termed as "simple events". Whereas, complex event detection often involves complicated interactions among objects in the scene. Complex events often provide rich semantic understanding of the context in videos.

(ii) The proposed architecture uses deep convolutional neural networks based methodology for the extraction of simple events from videos. Rule-based logical reasoning on the extracted simple events for the definition and extraction of complex events.

CEDEC enables the analysis of unstructured video data for the extraction of simple events which leads to the recognition of complex events after satisfying the validity of certain logical conditions and temporal facts. The proposed framework can support different kinds of operations on multiple applications e.g., Transportation, Entertainment, Security, Energy Consumption, Temperature readings, etc. In autonomous vehicle application, our proposed approach can determine the simple events of the types, appearance and disappearance of vehicles and pedestrians, and this could be useful to predict the complex event of the type "collision". The entertainment industry in recent years has attracted an increasingly large and diverse group of people. Most of the scenes involve multiple moving objects and moving cameras. Object detection and tracking with the movie dataset will result in simple events explaining the occurrence of missed detection, occlusion, and re-appearance,

as well as objects entering, and leaving the scene. Based on this information and chosen high-level commonsense knowledge representation and reasoning method complex events can be detected. Similarly in the surveillance application, detection of simple events e.g, appearance and disappearance of persons and other objects entering and leaving certain premises of sensitive buildings can be helpful to detect complex event of the type "theft". The proposed framework can utilize existing deep learning based feature extraction models, and semantically rich event recognition logical frameworks irrespective of their domain and nature of events. Figure 4.1 demonstrates the interaction of the proposed system with different applications from simple to complex events defined in more detailed in Figure 4.2. Our method follows two phases: (1) objects are detected and tracked from every single frame using YOLO, providing simple events such as appearance and disappearance of an object; (2) based on those candidate objects and simple events, complex events are represented in the logical framework of the *Event Calculus*. Reasoning on complex events is obtained by encoding in logic programs under *Answer Set* semantics (in particular, programs are run using DLV (Leone et al., 2002)). More details about the simple event detection pipeline are provided in Chapter 5, e.g, type of object detector and tracker used to determine appearance, disappearance and movement of objects. Whereas, Complex event detection pipelines are provided in Chapters 6, e.g, implementing the theory of Event Calculus to represent complex events related to the detected objects, and a declarative programming language, Answer set programming to implement the logical rules respectively. In the following sections, we detail the description of the use-cases, and the formal definition of events.

4.2 Example Use-Cases

Use-case 1: handicap parking occupancy The deployment of sensors in parking lots to address the issue of automatic parking-lot detection is performed with great success, but results in high deployment cost. Recently, smart cameras have been used to detect the occupancy of the slots in real-time relying on CNN-based systems (Amato et al., 2016; Ma et al., 2016). But, most of these camera-based solutions cannot be generalized for different parking lots. Visual occupancy detection in parking lots essentially involves the detection of vehicles (car, bike, bus, etc.) and parking spaces. However, to the best of our knowledge, the detection of vacant parking space for people with special needs by considering visual information and logical reasoning is still an open problem. We recognize the occupancy of the handicap slot as a complex event, with visual conditions defined by the time line in Figure 4.3: we recognize the beginning of the event at time T_0 when the car and the handicap slot appear in the scene; then at time T_1 the handicap slot disappears from the scene; and reappears at time T_3, whereas from T_1 to T_2, the car remains parked on the handicap

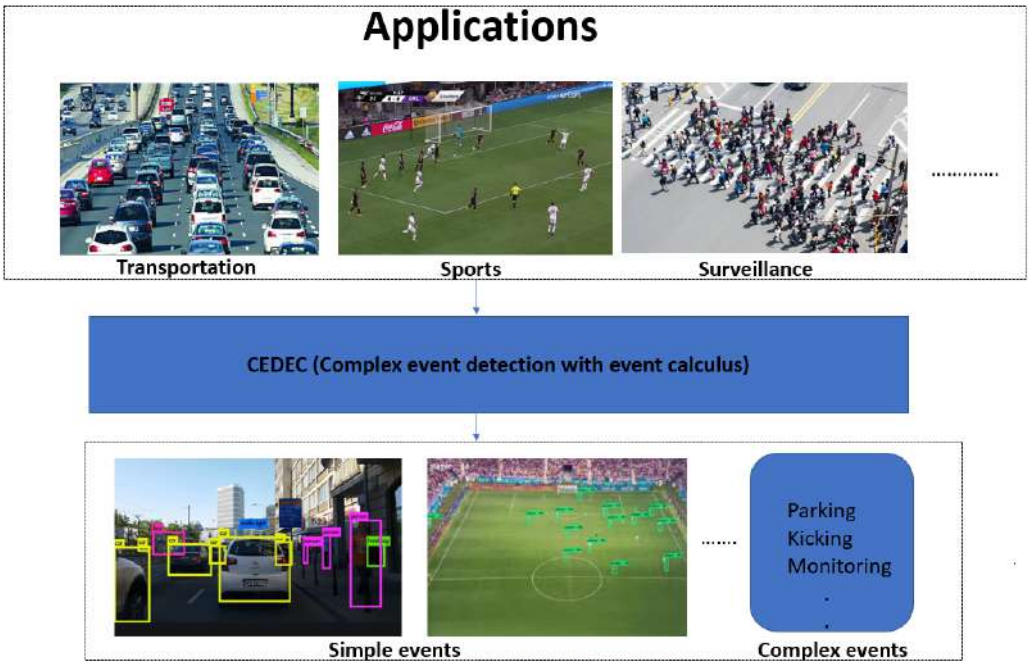


Figure 4.1: CEDEC (Complex event detection using event calculus)

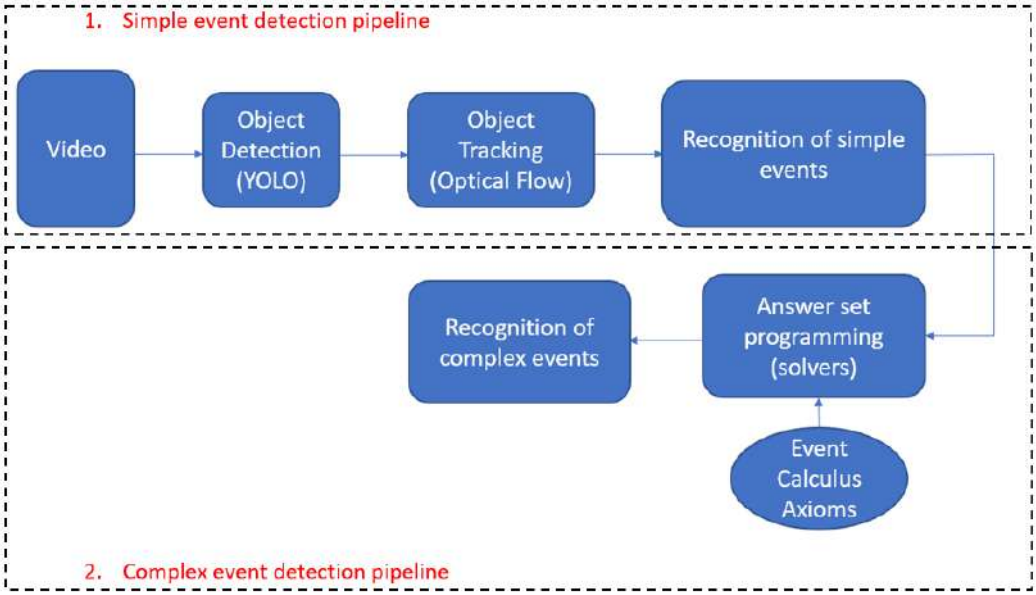


Figure 4.2: Block diagram of the proposed architecture.

slot.

Use-case 2: soccer kick events. To show the applicability of our approach, we also consider the case of videos from soccer matches, in particular videos edited in the style of television or streaming broadcasting. Several atomic and complex events related to the soccer match happenings can be recognized, e.g., goals, substitutions,

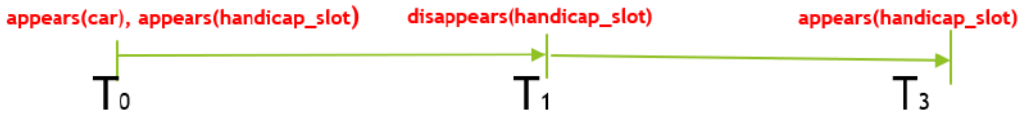


Figure 4.3: Time line definition for the "parking occupancy" complex event

fouls, off sides, etc.In this work we concentrate on recognizing different types of "kick" events, namely different situations in which the match has been stopped (e.g., after a foul) and the game is resumed by some player kicking the ball. In particular, we consider three common cases: *corner kicks*, *goal kicks* and *free kicks*.

A corner kick is an action occurring when the ball passes over the goal line by being hit by a player of the defending team: the ball is put into play by a player of the attacking team by taking the kick from the flag on a corner of the playing field. We recognize a corner kick event as a complex event, with visual conditions defined by the time line in Figure 4.4: we recognize the beginning of the event at time T_0 whenever the ball is seen close to a flag; then at a successive time T_1 a player should come near the position of the ball and kick the ball (i.e. cause the movement of the ball) at time T_2; after the ball has been kicked (usually towards the goal), we require that at T_3 the goal post is visible in the scene.

A free kick occurs when the game has been stopped due to a foul (or other soccer rules infringement): the game is resumed by a player kicking the ball from a position inside the field of play. The definition of the complex event for free kick is thus more relaxed than the one for corner kicks, as shown in Figure 4.5: at the beginning of the event at time T_0, a player comes in possession of the ball; then at time T_1 the ball is kicked (and thus the ball moves at the successive time point T_2). No further conditions on the visibility and nearness of other objects is required: clearly, we have also to constrain that event types are disjoint and no multiple kick events can happen simultaneously.

A goal kick is an action occurring when the ball passes over the goal line being hit by the player of the attacking team. Goal kicks are mostly taken by goalkeepers within the defending goal area (referred to as the six-yard box), where often there is the visibility of the goalpost. We recognize goal kick, with visual conditions defined in the time line in Figure 4.6: we recognize the beginning of the event at time T_0 whenever the goalpost is visible in the scene; then at a successive time T_1, a player should come near the position of the ball and kick the ball (i.e. cause the movement of the ball) at time T_2 and T_3, respectively.

Figure 4.4: Time line definition for the "corner kick" complex event

Figure 4.5: Time line definition for the "free kick" complex event

Figure 4.6: Time line definition for the "goal kick" complex event

4.3 Video-events

A precise ontological definition of events discussed above is still an open point. To the purpose of this thesis we take the approach recently proposed in (Skarlatidis et al., 2015) to define the events discussed in the *soccer kicks* use-case following the same structure from our previous work (Khan et al., 2018).

Simple Events

A *simple event type* is defined as follows:

$$SE = \langle ID, seType, t, \langle role_1, oType_1 \rangle, \ldots, \langle role_n, oType_n \rangle \rangle \tag{4.1}$$

where ID is the identifier, $seType$ is the event type, e.g. *"kicking the ball"*, and t is the time instant in which the event occurs, $role_1 \ldots, role_n$ $(n = 1, .., n_{max})$ are the roles that different objects play in an event of this type, e.g. one role of simple event *"kicking the ball"* is the subject who kicks and a second role is the kicked object; finally $oType_i$ is the legal type of object that can play the role $role_i$, e.g., it is only players who can kick, and only balls can be kicked. Summing up, the complete definition of the event type *"kicking the ball"* is

$$\langle ID, Kicking\_the\_ball, t, \langle Kicking\_Player, player\rangle, \langle Kicked\_Object, ball\rangle\rangle$$

A specific instance of an event of simple type defined in (4.1) is the following tuple:

$$\langle ID, seType, t, \langle role_1, O_1\rangle, \ldots, \langle role_n, O_n\rangle\rangle$$

where *ID* is the event identifier, O_1 and O_n are identifiers of objects detected in the frame associated to the time t, respectively. The instance of "Kicking_the_ball"

$$\langle 12, Kicking\_the\_ball, t, \langle Kicking\_Player, obj02\rangle, \langle Kicked\_Object, obj01\rangle\rangle$$

describes a simple event of type "kicking_the_ball" that happened at time t, where the obj02 throws the obj01. Furthermore obj01 and obj02 are two objects detected in the frame corresponding to time t, of type ball and player respectively.

Complex Events

Complex events are built by appropriately aggregating events, previously defined. More precisely, starting from simple events, we can apply logical operators or temporal operators to build higher-level complex events. We can thus define the hierarchy of events, from the lowest level including the simple events to the higher and higher levels corresponding to more and more complex events.

- *Complex events* stems from the application of logical operators like AND, OR, NOT to a set of events which may be simple or complex.

  $$CE = \langle ID, ceType, t, L =< e_1\ op\ e_2\ op...op\ e_n >\rangle$$

  where *ID* is the event identity, *ceType* is the complex event type (such as "The goal is valid only if there is no foul"), t is the time instance in which the complex event occurs, L is the set of lower-level simple or complex events $e_{1.....}e_n$ joined by logical operators *op* (*i.e. AND, OR, NOT*).

4.4 Examples of Complex Event Definitions

One of the most interesting things about soccer analysis is the ability to recognize events, such as free-kicks, corner-kicks, goal-kicks, ball possession, etc. from a common video. Most of the videos previously used in the event recognition use multiple fixed cameras to observe the position of all the players and the ball on the soccer field (Girshick et al., 2014). The use of such cameras improves the overall

accuracy of the system for object tracking but they are computationally expensive. The fragment of video clips we have used can be easily accessible from the internet.

Player ball possession Event: Player ball possession starts when a player begins to interact with the ball and ends when the player is no more able to perform any action with the ball or there is game interruption. We can also state this in a formal way: the event occurs when the distance between a player and the ball is below a threshold value and that player is the nearest to the ball.

$$\begin{aligned}&\left\langle ID, Player\_Ball\_Possession, t+\tilde{k}, \left\langle Poss\_Player, p_i\right\rangle, \left\langle Poss\_Object, b\right\rangle\right\rangle \leftarrow \\ &player(p_i), ball(b), D(p_i, b, t) < T_h, \\ &\forall j \neq i, player(p_j), D(p_j, b, t) > D(p_i, b, t) \wedge \\ &\forall k = 1 \ldots \tilde{k}, D(p_i, b, t+k) \approx 0\end{aligned}$$

The event "Ball possession" occurs when the distance $D\ (p_i, b, t)$ between the player p_i and the ball b at time t is less than the threshold T_h, and the distance $D\ (p_j, b, t)$ between the ball and any other player p_j, $j \neq i$, is greater than $D(p_i, b, t)$. Also, after interaction, the distance between the player and the ball is very low for an appropriate number of consecutive frames. The value T_h determines the threshold value for a player being able to physically interact with the ball and must be calculated experimentally.

Free-kick Event: In soccer videos, with reference to the consecutive sequence of frames, the event corresponding to "Free kick" is identified, initially if the distance between a player and the ball is very low for a few frames. Then, if the distance between a player and the ball increases in an appropriate number of the subsequent frames and the player is no longer able to interact with the ball. We can formally define the event "Free kick" as follows:

$$\begin{aligned}&\left\langle ID, Free\_Kick, t+\tilde{k}, \left\langle Kicking\_Player, p_i\right\rangle, \left\langle Kicked\_Object, b\right\rangle\right\rangle \leftarrow \\ &player(p_i), ball(b), D(p_i, b, t) < T_h \quad \wedge \\ &\forall k = 0 \ldots \tilde{k} - 1, D(p_i, b, t+k) < D(p_i, b, t+k+1)\end{aligned}$$

The expression above holds true as long as the distance between the player and the ball increases after their interaction.

Corner-kick Event: Keeping in view the conventional definition of the corner-kick event in *use-case 2*, objects involved in defining a *Corner Kick* consist of a Kicking Player (KP) p_i, Kicked Object (KO) *ball*, *Flag* f, and Goalpost g. Initially, the distances between the ball and the flag, and the player and the ball are less then

respective thresholds T_{bf} and T_h. With reference to the consecutive sequence of frames, after the player ball interaction, the distance between player and ball increases and goalpost remains visible in the scene.

$$\begin{aligned}&\left\langle ID, Corner\_Kick, t+\tilde{k}, \langle KP, p_i\rangle, \langle KO, b\rangle \langle Flag, f\rangle, \langle GP, g\rangle\right\rangle \leftarrow \\ &player(p_i), ball(b), flag(f), goalpost(g), (D(b,f,t) < T_h \\ &\wedge D(p_i,b,t) < T_{pb}) \ \wedge \ \forall k = 0 \ldots \tilde{k}-1, (D(p_i,b,t+k) < D(p_i,b,t+k+1))\end{aligned}$$

Goal-kick: Goal kicks can be defined in the similar manner as free-kicks. The prominent difference between goal kicks and free-kicks is the visibility of the goal-post when a goal kick takes place.

$$\begin{aligned}&\left\langle ID, Goal\_Kick, t+\tilde{k}, \langle KP, p_i\rangle, \langle KO, b\rangle, \langle GP, g\rangle\right\rangle \leftarrow \\ &player(p_i), ball(b), goalpost(g), (D(p_i,b,t) < T_h \quad \wedge \\ &\forall k = 0 \ldots \tilde{k}-1, D(p_i,b,t+k) < D(p_i,b,t+k+1)\end{aligned}$$

We remark that while defining the events, we are not considering all special cases that might occur during the game. In some cases, the player does not interact with the ball and runs beside the ball without touching it. Player ball possession only starts with the first touch. Also, considering ball possession for the player nearest to the ball is wrong, e.g. when that player is standing with back to the ball. Furthermore, for corner kicks, it is not mandatory that after the kick happens, there must be the visibility of the goalpost as in some cases, the player can kick the ball in the other direction. Similarly, for goal kicks, visibility of the goal post is strongly dependent upon the position of the camera.

Chapter 5

Simple Event Detection in Video

Object detection and tracking are arguably the most important problems in the field of computer vision. The extraction of simple events events takes place using the object detector and tracker. In this Chapter, we review the state-of-art object detector Yolo, and also discuss how it works as a tracker on the two use-cases discussed in the previous chapter. We also throw some light on the detected objects leads to simple events.

5.1 Yolo

Yolo is a recent approach for object detection from images and videos, where detection is treated as a regression problem, predicting spatially separated bounding boxes with a certain probability. The key idea behind yolo is the use of small convolutional filters applied to feature maps of bounding boxes to predict the category scores, using separate predictors for different aspect ratios to perform detection on multiple scales. It uses a single network for the prediction of bounding boxes and class probabilities, whereas most of the regional proposal-based object detectors have one part of their system dedicated to providing region proposals which include re-sampling of pixels and features for each bounding box, followed by a classifier to classify those proposals. These methods are useful and accurate but at the same time, computationally expensive and results in a low frame rate. The architecture of yolo is shown in Figure 5.2, which consists of 24 convolutional layers and two fully connected layers. yolo takes an input image and resizes it to 448×448 pixels. The image further goes through the convolutional network and gives output in the form of 7×7×30 tensor. Tensor gives the information about 1) coordinates of bounding boxes 2) confidence score of all classes. Using a specific threshold eliminates class labels below a specific threshold.
A more detailed explanation on how yolo operates (Redmon et al., 2016) is the following: it divides the entire input image into a grid of size S × S, where each grid

cell predicts fixed number of bounding boxes. Such bounding boxes allow one grid cell to detect multiple objects. In Figure 5.1, we see that the input image on the left consists of a dog, a bicycle and a car overlapping in the image. Since the output vector of each grid cell can only have one class, then it will be forced to pick either the dog, bicycle or car. But by defining bounding boxes, it can create a longer grid cell vector and associate multiple classes with each grid cell. Bounding boxes have a defined aspect ratio, and they try to detect objects that particularly fit into a box with that ratio. For example, since in the mentioned example, we are detecting three objects we should define one bounding box that is roughly the shape of a dog, another bounding box slightly wider that can fit a bicycle inside of it. The test image is first broken up into a grid and the network then produces output vectors, one for each grid cell. These vectors tell, if a grid cell has an object inside it or not, and what class the object belongs to and finally the bounding boxes coordinates for the object. Summarizing for each grid cell(Jonathan Hui, 2018):

- it predicts **B** bounding boxes and each box has a specific confidence score.
- it predicts **C** conditional class probabilities (one per class for the likeliness of the object class).

Most of the predicted bounding boxes will have a very low probability of object being present inside it. After producing these output vectors, yolo uses non-maximal suppression to get rid of unlikely bounding boxes. For each class, non-maximal suppression gets rid of the bounding boxes that have a confidence value lower than some given threshold. The first step in non-maximal suppression is to remove all the predicted bounding boxes that have a detection probability less than a specific threshold. After removing all the predicted bounding boxes that have a low detection probability, the second step is to select the bounding boxes with the highest detection probability and eliminate all the bounding boxes whose Intersection Over Union (IOU) value is higher than a given IOU threshold. It then selects the bounding boxes with the highest class probability and removes bounding boxes that are too similar to this. It will repeat this until all of the non-maximal bounding boxes had been removed for every class. The end result will look like the image in Figure 5.1 on the right.

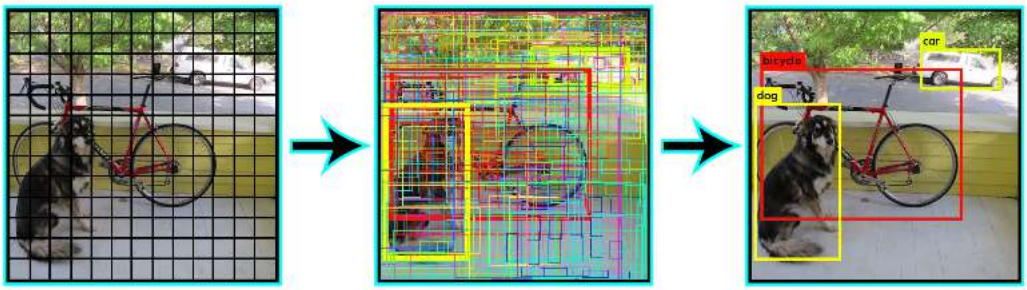


Figure 5.1: yolo grid, (Redmon et al., 2016)

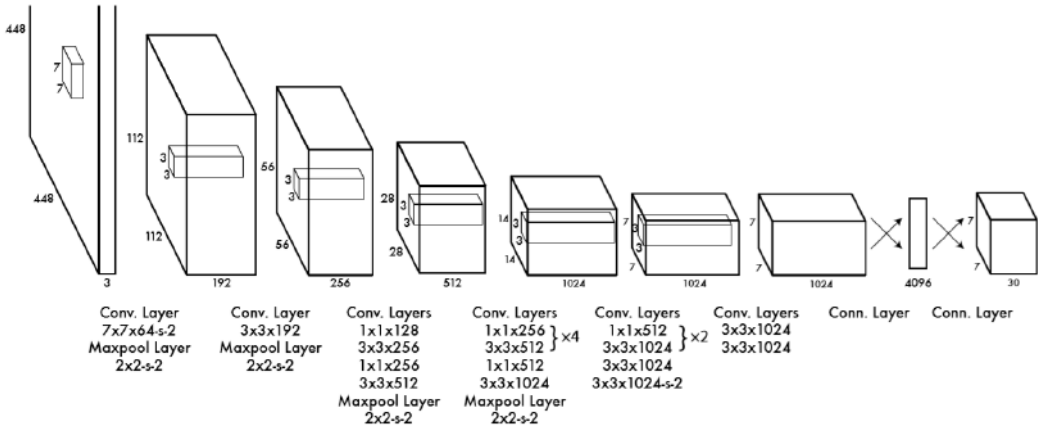


Figure 5.2: yolo architecture, (Redmon et al., 2016)

5.2 Training Yolo

Figure 5.3 shows the steps in the form of a flow chart for training yolo. For training we use convolutional weights that are pre-trained on Imagenet. yolo needs some specific files, namely (.data, .names and .cfg) to know how and what to train. The .data and .names files address the number of classes for which we are training the system, paths to training and validation set files, and the backup path where we want to store the *yolo* weights files after every iteration. In the configuration file, the batch size (images for every training step) and filter size are set. As mentioned here (Shinde et al., 2018), *yolo* requires the following files to start training :

- Pre-trained convolutional weights.
- Total number of action and object classes.
- Text file with the path to all frames for training and validation.
- The path to save trained weight files.
- Configuration file with all convolutional and fully connected layers of yolo architecture, along with batch size and filter size.

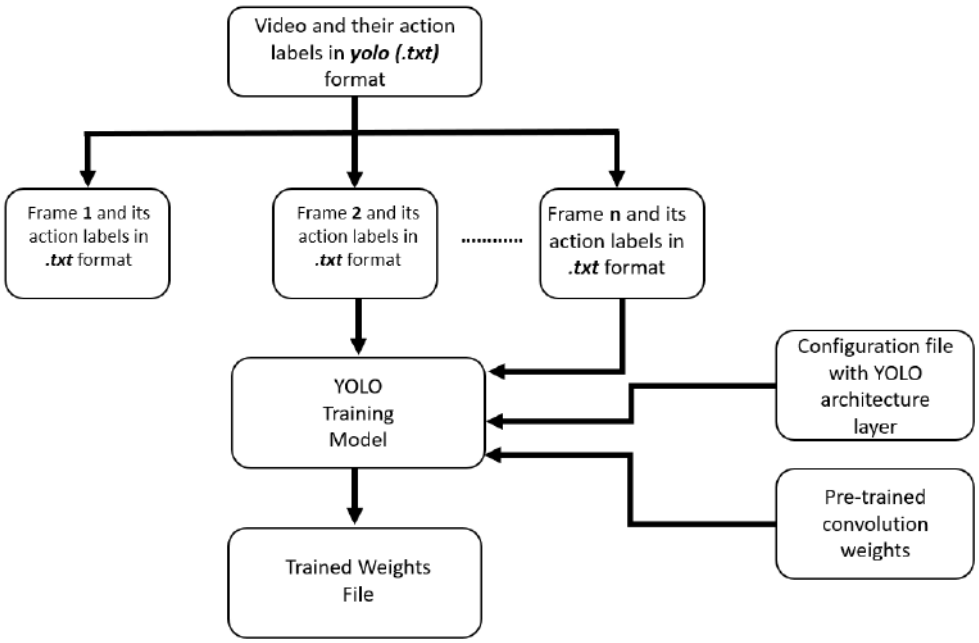


Figure 5.3: yolo training, (Shinde et al., 2018)

The value of the filter size is not randomly assigned in the configuration file, it depends upon the number of classes. For example, if we are training the system to detect 4 classes, filter size can be determined as *filters= 5 * (2+ number of classes)*. Hence, for 4 classes it will be 30.

Evaluation of Yolo: In order to evaluate the performance of yolo the dataset for parking occupancy use-case was collected from parking lots of *National University Ireland Galway*(NUIG) after the security clearance, where fixed cameras were used to record video clips. The total duration of the video clips is approximately 4 minutes, composed of multiple sequences, where each sequence is approximately 12 to 15 seconds, depicting the event of interest (parking of cars on the handicap slots) under different viewing conditions including camera movement, ego-motion, change in illumination, clutter, motion artifacts. The dataset for the soccer use-case at our disposal consists of approximately 5 minutes long video, consisting of approximately 7.5k manually annotated frames with four objects classes (player, ball, flag, goalpost), each clip depicts a specific action under different viewing conditions. Both data sets are manually created using a video annotation tool Vatic. It allows annotating objects inside each frame drawing a bounding box around them. The output of this process is a set of images with relative bounding boxes coordinates saved in yolo format. Such data set is later organised in manner to be used as a training set to produce automatic annotations of objects using yolo.

A common practice to evaluate the performance for object detection task is by assessing the *Average Precision* (*AP*) of each class. AP score is defined as the mean

precision on the set of 11 equally spaced recall values, $Recall_i = [0, 0.1, 0.2, \ldots, 1.0]$. Thus,

$$AP = 1/11 * (AP_r(0) + AP_r(0.1) + ...AP_r(1)) \tag{5.1}$$

Like any other object detector, yolo is trained on a fixed set of classes, so it would locate and classify only those classes in the image. The location of objects is generally in the form of a rectangle box known as a bounding box. Therefore, it involves both the localisation of the object in the image and classifying that object. The standard approach to evaluate the performance for image classification problems is the metric of precision, but it cannot be directly applied here, as here both the classification and localisation of a model need to be evaluated. This is where average precision comes into the picture. The metric that tells us the correctness of a given bounding box is the Intersection Over Union (IOU). IOU measures how much overlap exists between the ground truth and actual prediction: this measures how good is our prediction in the object detector with the ground truth (the real object boundary). For every correct detection that model reports, IOU with ground truth is calculated, using this value and a fixed IOU threshold, it then calculates the number of correct detections (*True Positives*) for each class in an image. Once the system has calculated the number of correct predictions and the missed detections (*False Negatives*), we can calculate the Recall of the model for that class using this formula $Recall = TP/(TP + FN)$.

Object	**AP**
player	80.46
flag	88.63
goalpost	90.33
ball	51.2
handicap slot	90.55
car	90.76

Table 5.1: Average Precision of the objects.

Table 5.1, shows the average precision of the objects at *Intersection over union* (*IOU*) threshold of (0.5). As it is evident from the table, yolo has problems to detect small objects, the reason stands in the fact, that the feature map it uses consists of very low resolution, and the small object features get too small to be detectable.

5.3 Pros and Cons of Yolo

Pros

When compared with conventional methods for object detection, yolo possesses certain advantages.

- It is fast, and a streaming video can be processed in real-time within few milliseconds.
- It consists of a single pipeline for both classification and localization.
- It works better than other detectors when generalizing from real world images to other domains like artwork.
- It maintains spatial diversity while making predictions.
- It consists of active online community.

Cons

Despite being extremely fast, yolo imposes spatial constraints on bounding box predictions in situations when the number of nearby objects appears in close proximity, such as, crowd of people or flock of birds. Another disadvantage remains in difficulty in generalizing in some cases for new objects with an unusual aspect ratio. It also struggles with small objects.

5.4 Examples of Yolo Output

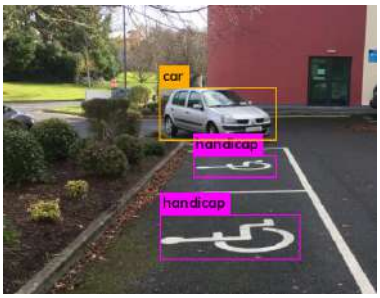
Figures 5.4 and 5.5, shows the results for detection and tracking produced by yolo over the sequence of frames, where a car moves towards the handicap slot and occupies the handicap slot for the first use-case. Whereas, for the second use-case, Figures 5.6, 5.7 and 5.8, shows the results of yolo for the detection of objects, while the inputs are free kick, corner kick and goal kick respectively. The sequence of frames over the period provides the co-ordinates of the bounding for the objects involved during the occurrence of free kick, corner kick and goal kicks respectively. The extracted information about objects and simple events derived from the visual data can be considered as the output of the first phase of our workflow: this knowledge will then be used as the input instance data of the logic based representation of events in the second step of the workflow, detailed in the following Chapter.

5.5 Simple Event Detection from Yolo Output

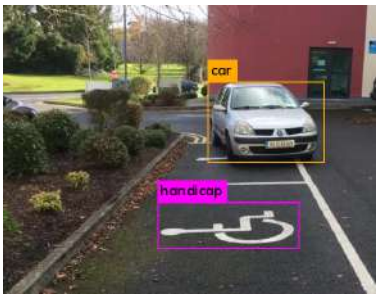
In order to extract simple events, we use yolo as an object detector and tracker, It uses optical flow method from OpenCV [1] to track objects by determining the pattern of motion of objects for two consecutive frames, which occurs due to the movement of the objects, helping in image segmentation and tracking. It works on the following assumptions. (1) Pixels grouped with a similar motion, result in a blob of pixels for all objects having different motion. (2) Intensities of pixels do not change between consecutive frames. (3) Neighbouring pixels have similar motion. We trained the system to detect four objects: player, ball, flag, goalpost.

The objects extracted from yolo consist of a unique frame identifier, class identifier, track identifier and bounding box co-ordinates for each object in the video. This information is post-processed using Python for the extraction of simple events, which include: appears, disappears, moves, close. The appearance and disappearance of the objects are simply identified from the track identifier of the objects. Whereas, the closeness and movement of objects are determined based on the centroid distance between the objects. For example, close(a,b) could be more precisely defined by specifying the threshold on the distance between a and b.

[1] see `https://opencv.org/` and `https://github.com/AlexeyAB/darknet`

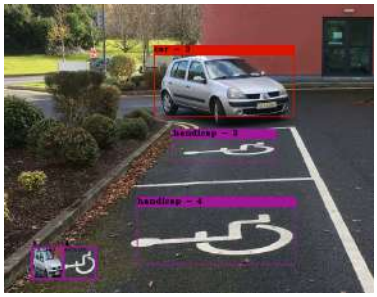


car moving to the handicap parking slot



car parked at the handicap slot

Figure 5.4: Object detection using yolo



car3 moving to the handicap parking slot

car3 parked at the handicap slot

Figure 5.5: Object tracking using yolo

Figure 5.6: Example of free kick defined in the context of soccer. The sequence of frames represents the occurrence of a free kick.

Figure 5.7: Example of a corner kick defined in the context of soccer. The sequence of frames represents the occurrence of a corner kick.

Figure 5.8: Example of a goal kick defined in the context of soccer.The sequence of frames represents the occurrence of a goal kick.

Chapter 6

Inferring Complex Events from Simple Events

6.1 Logical Reasoning

Logical reasoning is a process that involves taking information about a particular aspect of the real-world scenario and making inferences based on our knowledge of the situation or how the world works. Logical reasoning allows us to reconstruct the given situation so as to make predictions about what will happen in the future or to draw conclusions about what happened.

Logical reasoning comes naturally to us (humans) and appears to be simple; on the contrary, despite all the advances in recent times in the domain of artificial intelligence, it is still a challenging problem for computers. As an example scenario, if we consider the case of *Handicap parking occupancy* discussed in Chapter 3. To automate common-sense reasoning about such a scenario, the first step is to build the representation of the situation, which defines the structure of the scenario in terms of logical rules and facilitates the automated reasoning process.

The formal representation to any given scenario requires to represent objects in the given problem. For example, in use-case 1, objects which represent the discussed situation include cars and handicap slots. Second, we must describe the properties of the world that change over time, e.g., how the variation of time changes the location of objects in the scene. Lastly, we should also observe how the occurrence of certain events gives origin to other activities. For instance, the *disappearance* of the handicap slot from the scene provides a license to a *parking* event.

After forming a formal representation of the scenario, we should be able to perform reasoning. Our goal should be to achieve automation; hence, the method of logical reasoning should be expressed in the form of rules or an algorithm taking as basic input facts from any given context to produce a precise output. We should not only reason about the specific event and its effects, but we must also be able

to represent and reason about concurrent events and their implications. We should also be able to express if certain simultaneous events are possible or not in any given scenario by expressing constraints in the rules.

6.2 Temporal reasoning

Temporal reasoning is an essential requirement to reason for video data. It can help to detect complex events. The relations between simple events define complex events. For example, in the given scenario of soccer kicks, simple facts such as, the appearance and disappearance of players, ball, flag, and goalpost must be identified before the extraction of the complex event of the type free kick, corner kick, goalpost. Temporal reasoning is used to detect the sequence of simple activities that occurs over time. Additionally, it can also be used to identify abnormal behavior in the context; for example, some objects in the scene keep appearing and disappearing for very short intervals. Hence, keeping in view, the factor of time while reasoning on simple and complex events is of primary importance along with the availability of accurate and complete data.

Logical Reasoning in Temporal Domains

Several approaches exist that attempted to learn temporal action theories (Lorenzo and Otero, 2000; Lorenzo, 2002; Rodrigues et al., 2010). These approaches replace inertial axioms involving non-monotonic operators by rules which specify the *non-effects* of events and make use of the monotonic logic programming approaches to learn from narratives (Inoue et al., 2005). In the domain of process mining, where the input consists of logs, e.g., a set of sequence time-stamped traces of events, temporally ordered that capture some specific business logic. The standard approach to deal with process mining is Event-driven Process Chains(Van Dongen and Van der Aalst, 2004), but logic-based methods have been attracting attention because of their ability to induce processes in the form of rules that describe dependencies between the events. A logic-based framework *SCIFF* (Alberti et al., 2008) used in (Chesani et al., 2009) to represent actions. It comprises of a logic-based system to learn patterns of the traces of action in the form of *SCIFF* rules. Such rules contain events in the body and conjunctions or disjunctions in the head.

The Chronicle Recognition System (CRS)(Ghallab, 1996) is another temporal reasoning system. It is a temporal constraint network, where the definition of high-level events is linked together by a set of low-level events via a set of temporal constraints. CRS has been successfully used in many real-world applications like medical applications and computer networks (Quiniou et al., 2010; Dousson and Le Maigat, 2007) for event recognition. Another interesting application of CRS is

in the stock market(Badea, 2000), where trading rules are represented in the form of "buy/sell" policies from historical market data organised in temporal manner. Every trading rule defines a pattern trading technical indicators.

There exist many works for the incorporation of temporal reasoning into ASP solving technology, one recently proposed in (Cabalar et al., 2018) works by introducing a variant of Temporal Equilibrium Logic. It offers expressive and semantically rich language for modeling dynamic systems in ASP. It also enables a logical analysis of temporal properties from simple ASP specifications or other event recognition languages. In the next sections, we will see how reasoning can be performed under the semantics of Answer Set Programming.

6.3 Reasoning Based on Answer Set Programming

Answer Set Programming (ASP) has been successfully applied to the wide range of applications both in academia and industry. A few examples include: planning, classification, configuration, bio-informatics, software engineering, robotics, data integration and answering queries, etc. These applications have been discussed in detail in Chapter 2. The use of ASP can significantly reduce the effort needed to recognize complex events while obtaining the same level of quality in the detected events. The formal declarative syntax of ASP is convenient and expressive. Hence, ASP can be used to detect a large number of simple and compound events within a reasonable time, which allows real-time operations despite limited hardware resources.

We implement the theory of *Event Calculus* to represent simple and complex events related to the detected objects, and a declarative programming language, ASP to implement the logical rules as expressed in (Mueller, 2014). The general notion behind the use ASP is depicted in Figure 6.1: given an input instance of the problem, encoding of the problem in terms of a logic program such that it provides solutions to the problem by using answer set solver of our choice. There can also more than one solution to the problem. To summarize, our approach is the combination of the visual processing pipeline for object detection and tracking, and estimating appearance, disappearance, and movements of objects in the scene, with ASP-based reasoning about events, objects, and spatial-dynamics. The main component of the overall integrated approach is to build complex events from the underlying facts and simple events extracted from the visual pipeline. In the next section, we see in some detail how we express our example scenarios of *Soccer kicks* and*Occupancy of the handicap slots* in terms logic based representation of events by using a realization of the Event Calculus that allows us to define events in terms of logical rules. Axioms of the calculus are encoded in a logic program under Answer Set semantics.

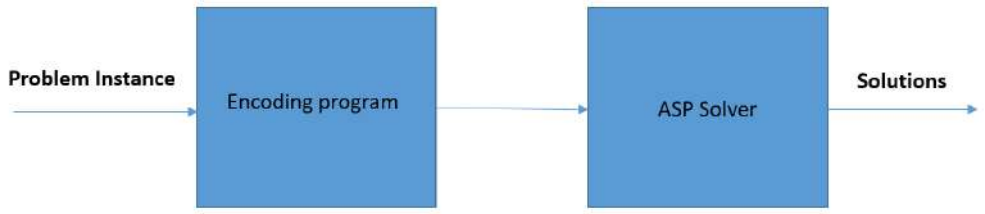


Figure 6.1: Encoding of problems in ASP (Eiter et al., 2009)

Event Reasoning on Use-Case 1

Let us consider the use-case1 explained in chapter 3. For explaining the perceived dynamics of objects in the scene described in more detail in Chapter 4, we define the simple and complex events taking into account the information extracted from the visual detection pipeline, listed in Table 6.1 below.

Table 6.1: Description of simple and complex events

Simple event	**Description**
$appearsCar(A,T)$	The object corresponding to car A enters the scene at time T
$disappearsCar(A,T)$	The object corresponding to car A leaves the scene at time T
$appearsSlot(L,T)$	The object corresponding to parking slot L appears in the scene at time T
$disappearsSlot(L,T)$	The object corresponding to parking slot L disappears from the scene at time T
Complex event	**Description**
$covers(A,L,T)$	The object car A covers the slot L at time T
$uncovers(A,L,T)$	The object car A uncovers the slot L at time T

The focus is on explaining what is visible in the frames by identifying the appearance and disappearance of objects in the scene: thus, the fluents of our scenario are *visibleCar* and *visibleSlot*, that are true respectively if a car or a slot is currently visible in the scene.[1] The occurrences of these events are directly extracted from the output of the tracker: in other words, they will be compiled as facts in the final program. Given this information, complex events are then defined by combining simple events and conditions on fluents: in our example, we can detect when a car *covers* and *uncovers* a parking slot using the information about what is visible at a given time-point. Assuming the previous rules defining the EC axioms in section 2.6, we encode our scenario in a logic program with the rules that follow. We first declare objects, events and fluents of the scenario:

[1]We are currently assuming a simple scenario with one car and one slot in the scene.

$$
\begin{aligned}
event(appearsCar(A)) &\leftarrow agent(A).\\
event(disappearsCar(A)) &\leftarrow agent(A).\\
event(\ appearsSlot(L)) &\leftarrow location(L).\\
event(\ disappearsSlot(L)) &\leftarrow location(L).\\
fluent(\ visibleCar(A)) &\leftarrow agent(A).\\
fluent(\ visibleSlot(L)) &\leftarrow location(L).
\end{aligned}
$$

We can then specify the effects of events on fluents:

$$
\begin{aligned}
initiates(appearsCar(A), visibleCar(A), T) &\leftarrow agent(A), time(T).\\
terminates(disappearsCar(A), visibleCar(A), T) &\leftarrow agent(A), time(T).\\
initiates(appearsSlot(L), visibleSlot(L), T) &\leftarrow location(L), time(T).\\
terminates(disappearsSlot(L), visibleSlot(L), T) &\leftarrow location(L), time(T).
\end{aligned}
$$

Basically, the rules define that the appearance of an object (car or slot) initiates its visibility, while its disappearance from the scene terminates the validity of the visibility fluent. Occurrences of complex events are derived from event calculus reasoning:

$$
\begin{aligned}
happens(covers(A, L), T) \leftarrow\ & agent(A), location(L), time(T),\\
& happens(disappearsSlot(L), T),\\
& holdsAt(visibleCar(A), T).\\
happens(uncovers(A, L), T) \leftarrow\ & agent(A), location(L), time(T),\\
& happens(appearsSlot(L), T),\\
& holdsAt(visibleCar(A), T).
\end{aligned}
$$

By these rules, we recognize that a car covers a slot if the car is visible at the time that the slot disappears. Similarly, the uncovers event occurs when a slot appears and the car is still visible. By combining the information on complex events, we can define that a *parking* from time T_1 to time T_2 is detected whenever a car covers a slot at time T_1, *uncovers* the slot at time T_2 and it stands on the slot for at least a number of frames defined by *parkingframes*:

$$
\begin{aligned}
parking(A, L, T1, T2) \leftarrow\ & happens(\ covers(A, L), T1),\ happens(uncovers(A, L), T2),\\
& parkingframes(N), T3 = T1 + N, T2 >= T3.
\end{aligned}
$$

In our scenario, a query on *parking* can be used to obtain the parking events detected in the scenes and their information.

The final program, enconding the scenario, is obtained by combining these rules (together with the EC axiom rules) with the facts obtained from the tracker output. Let us consider an example instantiation:

$$holdsAt(visibleSlot(hp\_slot), 0).$$
$$happens(appearsCar(car), 1).$$
$$happens(disappearsSlot(hp\_slot), 2).$$
$$happens(appearsSlot(hp\_slot), 4).$$
$$happens(disappearsCar(car), 5).$$

According to the input evidence, initially only one slot hp_slot is visible. Then, object *car* appears and object hp_slot disappears from the scene at time-points 1 and 2, respectively. Whereas, at time-points 4 and 5, appearance and disappearance of the hp_slot and *car* occur. Using the rules, we can thus derive the occurrence of complex events $happens(covers(car, hp\_slot), 2)$ and $happens(uncovers(car, hp\_slot), 4)$. Say that we define $parkingframes(1)$, then we can detect the $parking(car, hp\_slot, 2, 4)$, meaning that *car* parks on hp_slot at time-point 2 and leaves the slot at time-point 4. Over our sample video data, we run the program on DLV using the output of the tracker from previous step. We were able to detect complex events for some of the video sequences, for example considering Figure 5.4 and Figure 5.5 (*car 3* covers the *handicap slot 3* at time-point 87 and uncovers the slot at time-point 107). Unfortunately, we could not apply the method to the whole video: the reason stands in the ambiguities of tracker output (e.g. multiple labelling of the same object, incorrect disappearance of objects) which produce unclean data. A solution to this problem would be to include a pre-processing step for data cleaning (possibly encoded as logical constraints) which is able to resolve such ambiguities.

Event reasoning on Use-case2

We approach the problem of classifying different kicks in the soccer domain based on the objects detected over time. It can be observed that the complexity of the kicks increases with the number of objects detected in the scene and their defined interaction in the rules. We assume here for the recognition of a free kick only two objects player and ball are sufficient; on the other hand for the recognition for the goal kick player, ball, goalpost must be present in the scene. Whereas, for the detection of a corner kick, four objects player, ball, flag, and goalpost must be detected. We declare events and fluents of the scenario in Table 6.2:

Simple event	**Description**
$appears(G)$	The goalpost G enters the scene
$disappears(G)$	The goalpost G leaves the scene
$close(A, B)$	Object A and B are close to each other
$movesBall(B)$	The ball B moves in the scene
Complex event	**Description**
$kick(P, B)$	Player P kicks the ball B

Table 6.2: Description of simple and complex EC events

As discussed in the use-case1, here also the focus is on explaining what is visible in the frames by identifying the appearance and disappearance of objects and also the movement of objects in the scene: thus, the fluents defined in the current scenario are *visible* and *possession*, that are true respectively if a goalpost is visible in the scene, or when the player is in possession of the ball.

$$
\begin{aligned}
event(appears(G)) &\leftarrow goalpost(G).\\
event(disappears(G)) &\leftarrow goalpost(G).\\
event(close(A,B)) &\leftarrow player(A), ball(B).\\
event(close(A,B)) &\leftarrow ball(A), flag(B).\\
event(movesBall(B)) &\leftarrow ball(B).\\
fluent(visible(G)) &\leftarrow goalpost(G).\\
fluent(possession(P,B)) &\leftarrow player(P), ball(B).
\end{aligned}
$$

We can then specify the effects of events on fluents:

$$
\begin{aligned}
initiates(appears(G), visible(G), T) &\leftarrow goalpost(G), time(T).\\
terminates(disappears(G), visible(G), T) &\leftarrow goalpost(G), time(T).\\
initiates(close(P,B), possession(P,B), T) &\leftarrow player(P), ball(B), time(T).\\
terminates(movesBall(B), possession(P,B), T) &\leftarrow player(P), ball(B), time(T).
\end{aligned}
$$

Basically, the rules define that the appearance of an object initiates its visibility and its disappearance from the scene terminates the validity of the visibility fluent. For ball possession, the fluent starts its validity once a player is close to the ball and terminates once the ball starts moving (away from the player). Occurrences of complex events are derived from event calculus reasoning:

$$\begin{aligned} happens(kick(P,B),T) \quad \leftarrow \quad & player(P), ball(B), time(T), \\ & holdsAt(possession(P,B),T), \\ & happens(movesBall(B),T). \end{aligned}$$

By this rule, we recognize that a *kick* event occurs at a certain time slot if a player is in possession of the ball and the ball starts moving. By combining the information derived on fluents, simple and complex EC events, we can define the conditions to recognize the different types of kick events. For example, for corner kicks:

$$\begin{aligned} cornerkick(T_1) \quad \leftarrow \quad & player(P), ball(B), flag(F), goalpost(G), time(T_1), time(T_2), time(T_3), time(T_4), \\ & happens(close(B,F),T_1), holdsAt(possession(P,B),T_2), \\ & happens(kick(P,B),T_3), holdsAt(visible(G),T_4), \\ & T_1 \leq T_2, T_2 \leq T_3, T_3 \leq T_4, \\ & exists\_prev\_event(T_1). \end{aligned}$$

Namely, a corner kick is recognized if: at time T_1 the ball starts being close to a flag; then a player is in possession of the ball at T_2 and kicks the ball at T_3; after the ball has been kicked, a goal post has to be visible at time $T4$. Similar rules can be defined for the recognition of goal kicks and free kicks:

$$\begin{aligned} goalkick(T_1) \quad \leftarrow \quad & player(P), ball(B), flag(F), goalpost(G), time(T_1), time(T_2), \\ & happens(close(P,B),T_1), holdsAt(visible(G),T_1), \\ & happens(kick(P,B),T_2), T_1 \leq T_2, not\ cornerkick(T_1), \\ & not\ exists\_prev\_event(T_1). \\ freekick(T_1) \quad \leftarrow \quad & player(P), ball(B), time(T_1), time(T_2), \\ & happens(close(P,B),T_1), happens(kick(P,B),T_2), \\ & T_1 \leq T_2, not\ cornerkick(T_1), not\ goalkick(T_1) \\ & exists\_prev\_event(T_1). \end{aligned}$$

The rules which define the goal kick and the free kick are relaxations of corner kicks. During a goal kick, we assume that at time T_1, player having the possession of the ball, and the visibility of the goalpost must co-occur. The last condition in all three definitions of soccer kicks is a constraint that means only to consider the first kick event in the video, supported by the following rules:

$$
\begin{aligned}
kickevent(T_1) &\leftarrow cornerkick(T_1).\\
kickevent(T_1) &\leftarrow freekick(T_1).\\
kickevent(T_1) &\leftarrow goalkick(T_1).\\
exists\_prev\_event(T_2) &\leftarrow kickevent(T_1), time(T_2), T_1 < T_2
\end{aligned}
$$

The final program, encoding the scenario of a clip, is obtained by combining these rules (together with the EC axiom rules) with the facts obtained from the tracker output. Let us consider an example instantiation:

$$
\begin{aligned}
&happens(close(player1, ball1), 1).\\
&happens(close(ball1, flag1), 1).\\
&happens(movesBall(ball1), 3).\\
&happens(appears(goalpost1), 4).
\end{aligned}
$$

According to the input evidence, first the player is recognized to be near the ball in frame 1. In the same frame, the ball is recognized to be near a flag. In the successive frame 3, the ball starts moving and in frame 4 a goalpost appears in the scene. Using the rules, we can thus derive the occurrence of complex event $happens(kick(player1, ball1), 3)$ and the validity of fluents, $holdsAt(possession(player1, ball1), 1)$ and $holdsAt(visible(goalpost1), 4)$. This allow us to obtain the conditions to derive $cornerkick(1)$. Similarly, the occurrence of goal kick and free kick takes place, if the input evidence endorses the conditions defined for these two events. In the next Chapter we provide a detail evaluation for the detection of kicks.

6.4 Discussion

The key advantage of the proposed architecture is the interpretability of the model. A human or a domain expert can analyze the properties of the complex event and decompose into individual events that constitute it. Whereas, deep learning based (black box) architecture can only reply with relatively high accuracy, given complex event is happening or not. The contribution of our work mainly relies on bridging the gap between the effectiveness of opaque black-box deep learning based architectures to make sense of raw data and transparent reasoning of Event Calculus which allows us to embed human knowledge.

As discussed before in the soccer use-case in section 5.3, we have more than one object of type player, ball, etc. and for each of these, we can instantiate the rules. The information extracted from the output of the YOLO tracker consists of a unique frame identifier, class identifier, track identifier and bounding box co-ordinates for each object in the video. This information is post processed for the extraction of simple events which includes: appears, disappears, moves, close. The limitation of proposed architecture occurs during the visual pipeline, where the ambiguities of

tracker output (e.g. multiple labelling of the same object, incorrect disappearance of objects) produces unclean data. Basically, the current encoding is an example of encoding with some limitations/assumptions on the scene. The aim of our work is to inspire that ASP based reasoning can be performed on dynamic visual data of the observed scenarios not necessarily covering all possible cases. In our case a "complex event" is simply a combination of conditions on the domain and the information from "simple events" extracted from the video. A possible generalized solution can be of type "graded" version of the rules e.g., for the extraction of a corner kick, if we recognize that we see a flag, then we are looking at a corner kick with a probability of 30%, then if we also see the goalpost, this adds another 30% of probability of the occurrence of the complex event, corner kick.

The existing literature on modeling complex events based on logic programming and particularly using Event Calculus for the video data is not much. An activity recognition approach that is based on logic programming handles complex events based on Bilattice framework (Ginsberg, 1988). The knowledge base consists of domain-specific rules, expressing complex events in terms of simple events. Each complex or simple event is associated with two uncertainty values, indicating a degree of information and confidence respectively. Another logic-based method that recognises user activities is proposed by (Filippaki et al., 2011). The method recognises complex events from simple events using rules that impose temporal and spatial constraints between simple events. Some of the constraints in complex events definitions are optional. As a result, a complex event can be recognised from noisy data, but with lower confidence. The confidence of a complex event increases when more of the optional simple events are recognised. Another method (Artikis et al., 2014) employs MLNs (Markov Logic Networks) that have formal probabilistic semantics, as well as an Event Calculus formalism to represent complex events. The key advantage of modelling complex events in Event Calculus is its ability to incorporate non-deterministic actions, concurrent actions, action preconditions and qualifications etc., explained more detailed in section 3.2.

Chapter 7

Datasets

For the development of detection and classification, the requirement of training datasets with a large number of images is a must. For the preparation of such datasets, objects are manually annotated by humans. During this manual annotation process, they draw bounding boxes around the objects. The annotators mostly prefer open-source manual annotation tools: LabelImg [1], LabelMe[2], Vatic[3], etc. Using such tools, they create bounding boxes for the localization of objects and add labels against the chosen regions. The annotated data can be stored in many different formats *xml, txt, json*, etc. The downside of these manual annotation tools lies in the fact they are extremely time-consuming and expensive. It requires much of human labor. In our work, we have tried to utilize these manual annotation tool e.g, Vatic to create a training set for performing automatic annotation of videos using state of the art object detection tool yolo discussed in section 4.2.

7.1 Traditional Datasets

Action recognition datasets traditionally used in the literature by the vision community consist of atomic actions. Among the commonly used spatio-temporal action recognition datasets today, typical examples include: (1)**KTH dataset** (Laptev et al., 2004) consist of some simple periodic actions, e.g, walking, running, waving. Figure 7.2 shows a simple action of the type “hand waving". It consists of 6 types of human actions: boxing, walking, running, jogging, hand waving, hand clapping. There are in total 599 action classes in this dataset. (2) **UT-Interaction dataset** (Ryoo and Aggarwal, 2009) consists of interactions between humans. For example Figure 7.1 shows human interactions of type “handshake”. The total number of human interactions in this dataset are of 6 types: handshake, hug, kick, point, punch and

[1] `https://awesomeopensource.com/project/tzutalin/labelImg`

[2] `labelme.csail.mit.edu`

[3] see `http://www.cs.columbia.edu/~vondrick/vatic/`

push. Here, each interaction consists of 10 videos, so total 60 videos, captured in different illumination conditions. (3) A more realistic and primarily used dataset by the vision community is **UCF Sports dataset** (Rodriguez et al., 2008) which comprises of human-human or human-object interactions, as shown in Figure 7.3 which contains a sports activity "weight lifting". It contains 101 action categories, with 13320 videos in total. Similarly other datasets Hollywood 2 dataset (Marszałek et al., 2009), UT Kinect3D dataset (Xia et al., 2012), HMDB51 dataset (Kuehne et al., 2011), Sports-1M dataset (Karpathy et al., 2014) are also used commonly for the recognition of actions. The fundamental limitation in the above mentioned datasets lies in the fact that most of them consists of atomic actions with limited number of action classes. Our goal is to work with interesting datasets from everyday activities, where interaction between objects leads to some meaningful information or an event. This motivates us to develop two custom data-sets from scratch for the applications of parking occupancy of the handicap slots in parking lots and classification of kicks in soccer videos.

Figure 7.1: UT-Interaction dataset, meeting (Ryoo and Aggarwal, 2009)

Figure 7.2: KTH dataset, handwaving (Laptev et al., 2004)

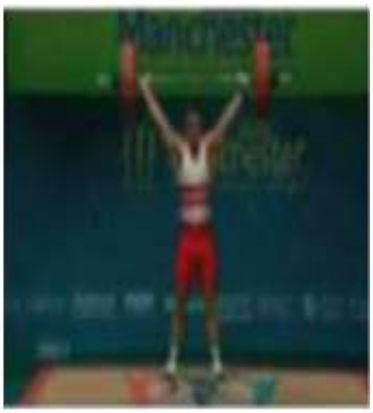


Figure 7.3: UCF Sports dataset, weight lifting (Rodriguez et al., 2008)

7.2 Experimental Dataset

For soccer kicks, in order to evaluate the performance of our system, we created a dataset of video clips trimmed from the *SoccerNet* dataset (Giancola et al., 2018). The overall SoccerNet dataset consists of 500 games from main European championships BundesLiga, Siere A, LaLiga and English Premier League. Each game is composed of 2 untrimmed videos, one for each half period. The videos are available in a variety of encoding (MPEG, H264), containers (MKV, MP4, and TS), frame rates (25 to 50 fps), and resolutions (SD to Full HD). We have used this dataset and created a

customized dataset of trimmed video clips. The trimmed video clips are depicting a specific class of action. Our sample dataset consists of 90 clips, categorized as 30 free kicks, 30 corner kicks and 30 goal kicks, where each clip is approximately 8 to 12 seconds.

Chapter 8

Experimental Evaluation

In this chapter, we present experimental results for the *classification of soccer kicks*. Part of our experiments also aims to compare complex events generated using our proposed approach with a deep neural network approach (Adrian Rosebrock, 2019). Hence to compare the results of our hybrid approach, the implementation of a deep neural network for activity recognition in videos was performed. All experiments related to the visual processing pipeline and classification of videos using neural networks were performed on a GPU: we have used Nvidia Geforce GTX 1080 with 2560 cores running at 1733 MHz frequency on 8 GB onboard GDDR5X memory. We have used CUDA[1] version 8.0 to compile the code. Whereas, complex event detection pipeline was implemented in a system with 8 GB RAM and Intel Core i7, 2.20 GHz. The code for extracting simple events such as *closeness* and *movement* between objects is implemented in python. The main reasoning component, DLV[2] is used as an ASP solver taking into account its several properties. DLV offers frontends for most of the knowledge representation formalisms. It is free for academic and non-commercial purposes; one of the main strengths of the system is its ability to deal efficiently with different kinds of applications in the domain of artificial intelligence. Another advantage of DLV is its high expressive power to represent problems even with insufficient knowledge.

8.1 Evaluation on Complex Event Detection

The evaluation on complex events was performed for the classification of *soccer kicks*. Unfortunately, we could not perform evaluation for the occupancy of the handicap slots. The reason stands in the limitation of the dataset for this category, because such clips are not publicly available, and security clearance to record such clips from parking lots make them difficult to collect.

[1] `https://developer.nvidia.com/cuda-gpus`
[2] http://www.dlvsystem.com/

		PC		
		Corner kick	Free kick	Goal kick
Actual	Corner kick	22	6	0
	Free kick	3	24	0
	Goal kick	8	8	14

Table 8.1: Confusion matrix for complex event detection for our approach (PC = predicted class).

In the application of soccer event detection, our goal is to extract complex events such as: *free kicks*, *corner kicks* and *goal kicks* from spatio-temporal information extracted from the video clips like appearance, disappearance and movement of objects in the scene.

The clips have been processed using the two-step architecture described in previous chapters with more details: the information extracted from the visual processing pipeline provides simple events, compiled as facts in the final program. Given this information, complex events are then extracted by combining simple events and event conditions. The results of the detection of complex events are expressed in the form of a confusion matrix in Table 8.1. For the class of corner kick, it can be observed from Table 8.1, with 30 clips as input to the system; it successfully detects 22 as corner kicks, six as free kicks, whereas 2 of them do not match the definition of any of the event. For each video clip, rules get fired over the input evidence, and the occurrence of a complex event takes place once the event definition is fully satisfied. Intuitively, the conditions for free kicks and goal kicks are relaxations of the ones for corner kicks, and all three events include a kick event as in their definition.

We remark that these results in Table 8.1 are strongly dependent on the (quite simplified) definition of rules for simple and complex events we provided in this example use-case 2 in chapter 6: in other words, independently from the definition of our architecture, better results could be obtained e.g., by refining the algorithms used in the extraction of simple events and by imposing further conditions over other information in the definition of the datalog rules. Another limitation in the visual event pipeline stands in the ambiguities of the tracker output (e.g., multiple labeling of the same object, incorrect disappearance of objects) which produce unclean data at the end of the first step of the workflow. One solution to this problem would be to include a pre-processing step for data cleaning (possibly encoded as logical constraints based on the domain knowledge), which is able to resolve such ambiguities. Another possible solution could be to exploit other approaches where object detection and tracking are jointly addressed (Hou et al., 2017; Kang et al., 2017). To compare the results discussed for the classification of different soccer kicks, we use a deep neural network in the next section and see how it performs on every single video clip.

8.2 Video Classification

Video classification is a computer vision area, and it is about an automated understanding of the video, the input is a user labeled video stream, and the objective is to produce a label of event. A video is just a stack of images with the temporal aspect attached to them. As highlighted in the related work section the most commonly used techniques to deal with the temporal aspect of videos is the use of neural network architectures such as *Long short-term memory* (LSTMs). The major drawback attached to these approaches is the computational complexity and huge amount of data that they require when it comes to training such networks. The current generation of popular deep learning hardware is Nvidia graphics cards, and they are optimized to process data with extreme parallelism and speed, which CNNs utilize. LSTMs on the other hand, process things more sequentially, so the deep learning hardware does not increase its speed by much, especially during the training phase of the network. Each architecture has advantages and disadvantages that are dependent upon the type of data that is being modeled. When choosing one framework over the other, or creating a hybrid approach, the type of data and the job at hand are the most important points to consider. We had access to a relatively smaller dataset of regular clean 90 video clips with three different types of events. Mostly in literature, CNN is a top choice for image classification and more generally, computer vision applications. We worked to implement CNN based solution mentioned in (Karpathy et al., 2014) for the classification of videos in order to compare with our approach. In this paper (Karpathy et al., 2014), authors model videos with Convolutional Neural Networks in an almost similar way to how CNNs model images. They explore two components of the video classification: (1) how transfer-learning can be applied to video classification, (2) designing of CNNs which take into account the temporal connectivity in videos. Based on the discussed approach (Karpathy et al., 2014), the problem of video classification can also be viewed from the perspective of image classification. Which in some respect is true as videos can be understood as a series of individual images; and therefore, many practitioners from the vision community are at ease to treat video classification as performing image classification. The steps to classify videos following the approach discussed in (Adrian Rosebrock, 2019) are :

- Read all the video frames by looping over the video file.
- Classify each frame individually after passing them through the CNN.
- Choose the label with the largest corresponding probability.

The problem encountered while practicing this approach for the task of video classification is called prediction flickering. It is a visible change in the prediction

label between cycles displayed on video displays. A simple solution to prevent this problem is to utilize a technique known as *rolling prediction average*. The use of rolling prediction averaging to reduce "flickering" works on the following assumptions.

- Loop over all frames in the video file.
- Pass each frame through the CNN and obtain the prediction.
- Maintain a list of the last L predictions, and compute the average of the last L predictions and choose the label with the largest corresponding probability.

A brief overview of how it works on the example use-case. Initially, the training script converts all the trimmed video clips from *SoccerNet* (Giancola et al., 2018) into images organized by class. Furthermore, it will only train with free kicks, corner kicks and goal kicks with 30 videos for each category, after segmenting dataset into training and testing splits using 80% of the data for training and the remaining 20% for testing. It then grabs the dataset class images, loads the *ResNet50* CNN, and applies fine-tuning of ImageNet weights to train the model. The training script generates two files: a fine-tuned classifier based on ResNet50 for recognizing different kicks and serialized label binarizer containing unique class labels. The prediction script loads an input video and proceeds to classify the video ideally using rolling average method. Here, after passing all frames through the CNN, it maintains a list of the last L predictions, and computes the average of the last L predictions and chooses the label with the largest corresponding probability. The assumption here is that subsequent frames in a video will have similar semantic contents. The averaging, therefore, enables us to smooth out the predictions and make for a better video classifier.

The images in Figure 8.1 show some of the results using the CNN based approach, it can be observed all three categories of soccer kicks are correctly classified as "free kicks", "corner kicks" and "goal kicks". Whereas, from Figure 8.2 shows some of the results where input videos are the wrong classified. Table 8.1 shows the result of the evaluation where out of 30 clips for each class, 20 are correctly classified as corner kicks, 21 as free kicks, and 13 as goal kicks. We remark that classification of videos into action categories will produce better results with larger datasets (Soomro et al., 2012; Karpathy et al., 2014).

		PC		
		Corner kick	Free kick	Goal kick
Actual	Corner kick	20	10	0
	Free kick	9	21	0
	Goal kick	11	6	13

Table 8.2: Confusion matrix for complex event detection (PC = predicted class).

Figure 8.1: Sequence of frames from input videos correctly classified as **Corner-Kick, Free-kick and Goal-kick** using the approach in (Adrian Rosebrock, 2019)

8.3 Our approach vs Neural Net

To this end, we compare the results of the presented hybrid approach in this thesis with the above mentioned deep neural network approach to classify videos based on predictive accuracy. We can see that in Table 8.3 and Table 8.4, our proposed methodology performs slightly better despite the limitations of the object tracker mentioned before. From Table 8.3, higher value on precision for corner kicks and goal kicks should not surprise, as we have strict rules for these classes, which get fired only when several conditions on the event description are met, whereas for free kicks, we have more relaxed rules. On the contrary, lower values on recall are partly justified because of the nature of the events we are taking into consideration: intuitively, the conditions for free kicks are relaxations of the ones for corner kicks and goal kicks, and they all include a kick event in their definition.

The comparison of results shows that in terms of precision and recall, our proposed approach outperforms the deep neural network-based video classification, taking into account we are dealing with datasets relatively smaller size. It is a well-known fact that a large amount of training data plays a critical role in making the deep neural network models successful. On the other hand, even smaller datasets are sometimes enough to train the state-of-the-art object detectors with

Figure 8.2: Sequence of frames from input videos incorrectly classified

specific number of object classes. Even though existing deep neural network-based action recognition approaches (Karpathy et al., 2014), (Simonyan and Zisserman, 2014), (Feichtenhofer et al., 2017), (Kong et al., 2018) have shown impressive performance on large-scale datasets in laboratory settings, it is really challenging to get the same optimal performance while dealing with relatively smaller datasets. Hence we argue, a hybrid solution can be a good trade-off between accuracy and semantic richness while defining events considering relatively smaller datasets. We also remark that the use of logical reasoning on the output of deep learning based solutions can surely makes the overall event recognition paradigm more explainable.

Class	**Precision**	**Recall**
Corner kick	66.6 %	73 %
Free kick	63.1 %	80 %
Goal kick	100 %	46.6 %

Table 8.3: Experimental Results for the proposed architecture

Class	**Precision**	**Recall**
Corner kick	50%	66.6 %
Free kick	56.8 %	70 %
Goal kick	100 %	43 %

Table 8.4: Experimental Results for the neural network

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this thesis, we presented a method to derive complex events from simple facts which are extracted from the visual recognition techniques. The overall goal of this work is the integration of knowledge representation and computer vision. In Chapter 1, we discussed the basics of event recognition, and we argue that logic-based event recognition systems have significant advantages over non-logic-based ones. In Chapter 2, we discuss the basic components that we need for the thesis. We start with techniques followed by the vision community to address the issue of event detection from videos and their limitations followed by the Hybrid approaches towards video event detection and finally the Integration of logical reasoning and deep learning. Chapter 3, explains the necessary Background material used in this thesis from object detection and tracking to logical framework of event calculus used for the representation of simple and complex events and rationale behind it. Finally, an implementation of the Event Calculus into answer set programs is discussed. Chapter 4, explains the workflow for our proposed architecture CEDEC (Complex event detection using event calculus), and also two fields of application: soccer event detection, and occupancy of handicap slots, finally high-level definitions of the respective events. Chapter 5, throws some light on how YOLO works in general, and how it performs for our applications of soccer event detection and handicap parking occupancy, finally how simple events are extracted from the Yolo output. In Chapter 6, we express our example scenarios in terms of the presented ASP encoding of the Event Calculus. In Chapter 7, we discuss the traditional and experimental datasets. In Chapter 8, we evaluate the overall performance of our proposed framework for event detection in videos. The results of our proposed methodology compare favorably with those achieved by the use of deep neural networks. We demonstrate that proposed approach for the detection of events in videos shows impressive performance in dealing with relatively smaller data sets.

9.2 Future work

There are several directions in which we intend to extend the work presented in this thesis. As discussed before one of the major limitations in our proposed architecture is the use of the object tracker which results in noisy data and restricts us to reason on events for the whole video. As a future work, we aim to manage these inaccuracies by a (possibly logic based) data cleaning step. In logical data cleaning, error detection and repair is typically performed using declarative cleaning programs (Han et al., 2011). For example, if a cleaning program asserts that a object detected for a specific class should have a single reading at any time point, it may also contain a strategy that replaces multiple readings with their average. In constraint-based cleaning, data dependencies are used to detect data quality problems. Data that is inconsistent with respect to the constraints can be repaired by finding a minimal set of changes that fixes the errors (Bohannon et al., 2005; Kolahi and Lakshmanan, 2009). An important advantage of such approaches is that they are able to find subtle data quality problems using sophisticated dependencies.

Currently we are utilizing YOLO for both detection and tracking. Another possibility can be to exploit more recent approaches where the problem of multi object detection and tracking is addressed, e.g., few recent works have attempted to tackle detection and tracking in end-to-end manner (Hou et al., 2017; Kang et al., 2017; Andriluka et al., 2018), and some works have further used such architectures for even the detection of high level action recognition (Hou et al., 2017). We strongly believe our proposed architecture would perform significantly better with clean data from the tracker output. We also want to apply and evaluate the presented method in different scenarios, such as CAVIAR[1] Video data set , EPIC-KITCHENS (Damen et al., 2018).

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

Appendix A

Author's Publications

1. Adnan Akbar, Abdullah Khan, Francois Carrez, Klaus Moessner, *"Predictive Analytics for Complex IoT Data Streams"*, in *IEEE Internet of Things journal, vol. 4, NO. 5, October 2017.*

2. Abdullah Khan, Beatrice Lazzerini, Gaetano Calabrese, Luciano Serafini *"Soccer event detection"*, in *4th International Conference on Image Processing and Pattern Recognition, Copenhagen, Denmark, May 2018.*

3. Abdullah Khan, Luciano Serafini, Loris Bozzato, Beatrice Lazzerini, *"Event Detection from Video Using Answer Set Programing."*, in *34th Italian Conference on Computational Logic. CILC 2019. Trieste, Italy, June, 2019.*

4. Abdullah Khan, Loris Bozzato, Luciano Serafini, Beatrice Lazzerini *"Visual Reasoning on Complex Events in Soccer Videos Using Answer Set Programming"*, in *5th Global Conference on Artificial Intelligence. Bolzano, Italy, September 2019.*

Bibliography

Adrian Rosebrock (2019). Video classification with keras and deep learning. https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/. Last checked on 2019-07-15.

Ahmad, K. and Conci, N. (2019). How deep features have improved event recognition in multimedia: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(2):39.

Ahmad, K., Mekhalfi, M. L., Conci, N., Melgani, F., and Natale, F. D. (2018). Ensemble of deep models for event recognition. *ACM TOMM*, 14(2):51.

Akbar, A., Carrez, F., Moessner, K., and Zoha, A. (2015). Predicting complex events for pro-active iot applications. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 327–332. IEEE.

Akbar, A., Khan, A., Carrez, F., and Moessner, K. (2017). Predictive analytics for complex iot data streams. *IEEE Internet of Things*.

Akman, V., Erdoğan, S. T., Lee, J., Lifschitz, V., and Turner, H. (2004). Representing the zoo world and the traffic world in the language of the causal calculator. *Artificial Intelligence*, 153(1-2):105–140.

Al Machot, F., Mayr, H. C., and Ranasinghe, S. (2018). A hybrid reasoning approach for activity recognition based on answer set programming and dempster–shafer theory. In *Recent Advances in Nonlinear Dynamics and Synchronization*, pages 303–318. Springer.

Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., and Torroni, P. (2008). Verifiable agent interaction in abductive logic programming: the sciff framework. *ACM Transactions on Computational Logic (TOCL)*, 9(4):29.

Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202.

Amato, G., Carrara, F., Falchi, F., Gennaro, C., and Vairo, C. (2016). Car parking occupancy detection using smart camera networks and deep learning. In *IEEE ISCC 2016*, pages 1212–1217. IEEE.

Andriluka, M., Iqbal, U., Insafutdinov, E., Pishchulin, L., Milan, A., Gall, J., and Schiele, B. (2018). Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5167–5176.

Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., and Studer, R. (2011). Etalis: Rule-based reasoning in event processing. In *Reasoning in event-based distributed systems*, pages 99–124. Springer.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307.

Arenas, M., Baier, J. A., Navarro, J. S., and Sardina, S. (2018). On the progression of situation calculus universal theories with constants. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Artikis, A., Makris, E., and Paliouras, G. (2019). A probabilistic interval-based event calculus for activity recognition. *Annals of Mathematics and Artificial Intelligence*, pages 1–24.

Artikis, A., Sergot, M., and Paliouras, G. (2014). An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908.

Artikis, A., Skarlatidis, A., and Paliouras, G. (2010). Behaviour recognition from video content: a logic programming approach. *International Journal on Artificial Intelligence Tools*, 19(02):193–209.

Bacchus, F., Halpern, J. Y., and Levesque, H. (1995). Reasoning about noisy sensors in the situation calculus. *Contract*, 49620(91-C):0080.

Badea, L. (2000). Learning trading rules with inductive logic programming. In *European Conference on Machine Learning*, pages 39–46. Springer.

Banbara, M., Inoue, K., Kaneyuki, H., Okimoto, T., Schaub, T., Soh, T., and Tamura, N. (2017). catnap: Generating test suites of constrained combinatorial testing with answer set programming. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 265–278. Springer.

Batusov, V., De Giacomo, G., and Soutchanski, M. (2019). Hybrid temporal situation calculus. In *Canadian Conference on Artificial Intelligence*, pages 173–185. Springer.

Besold, T. R., Garcez, A. d., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kühnberger, K.-U., Lamb, L. C., Lowd, D., Lima, P. M. V., et al. (2017). Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*.

Bhaskar, J., Sruthi, K., and Nedungadi, P. (2015). Hybrid approach for emotion classification of audio conversation based on text and speech mining. *Procedia Computer Science*, 46:635–643.

Bohannon, P., Fan, W., Flaster, M., and Rastogi, R. (2005). A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 143–154.

Bragaglia, S., Chesani, F., Mello, P., Montali, M., and Torroni, P. (2012). Reactive event calculus for monitoring global computing applications. In *Logic Programs, Norms and Action*, pages 123–146. Springer.

Brendel, W., Fern, A., and Todorovic, S. (2011). Probabilistic event logic for interval-based event recognition. In *CVPR 2011*, pages 3329–3336. IEEE.

Brewka, G., Eiter, T., and Truszczyński, M. (2011). Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103.

Bry, F. and Eckert, M. (2007). Rule-based composite event queries: the language xchange eq and its semantics. In *International Conference on Web Reasoning and Rule Systems*, pages 16–30. Springer.

Budvytis, I., Badrinarayanan, V., and Cipolla, R. (2010). Label propagation in complex video sequences using semi-supervised learning. In *BMVC 2010*, pages 27.1–12.

Cabalar, P., Kaminski, R., Schaub, T., and Schuhmann, A. (2018). Temporal answer set programming on finite traces. *Theory and Practice of Logic Programming*, 18(3-4):406–420.

Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., and Storari, S. (2009). Exploiting inductive logic programming techniques for declarative process mining. In *Transactions on Petri Nets and Other Models of Concurrency II*, pages 278–295. Springer.

Cugola, G. and Margara, A. (2010). Tesla: a formally defined event specification language. In *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 50–61. ACM.

Dal Palù, A., Dovier, A., Formisano, A., Policriti, A., and Pontelli, E. (2016). Logic programming applied to genome evolution in cancer. In *CILC*, pages 148–157.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE.

Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*, pages 428–441. Springer.

Damen, D., Doughty, H., Maria Farinella, G., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., et al. (2018). Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 720–736.

De Giacomo, G., Lespérance, Y., and Levesque, H. J. (2000). Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1-2):109–169.

De Vos, M., Crick, T., Padget, J., Brain, M., Cliffe, O., and Needham, J. (2005). Laima: A multi-agent platform using ordered choice logic programming. In *International Workshop on Declarative Agent Languages and Technologies*, pages 72–88. Springer.

Doherty, P., Gustafsson, J., Karlsson, L., and Kvarnström, J. (1998). Temporal action logics (tal) language specification and tutorial. *Electronic Transactions on Artificial Intelligence (http://www. etaij. org)*, 3.

Donadello, I., Serafini, L., and Garcez, A. D. (2017). Logic tensor networks for semantic image interpretation. *arXiv preprint arXiv:1705.08968*.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Dousson, C. and Le Maigat, P. (2007). Chronicle recognition improvement using temporal focusing and hierarchization. In *IJCAI*, volume 7, pages 324–329.

Dovier, A., Formisano, A., and Pontelli, E. (2009). An empirical study of constraint logic programming and answer set programming solutions of combinatorial problems. *Journal of Experimental & Theoretical Artificial Intelligence*, 21(2):79–121.

Eiter, T., Ianni, G., and Krennwallner, T. (2009). Answer set programming: A primer. In *Reasoning Web International Summer School*, pages 40–110. Springer.

Erdem, E., Gelfond, M., and Leone, N. (2016). Applications of answer set programming. *AI Magazine*, 37(3):53–68.

Erdem, E. and Oztok, U. (2015). Generating explanations for biomedical queries. *Theory and Practice of Logic Programming*, 15(1):35–78.

Erdem, E., Patoglu, V., and Saribatur, Z. G. (2015). Integrating hybrid diagnostic reasoning in plan execution monitoring for cognitive factories with multiple robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2007–2013. IEEE.

Erdem, E., Patoglu, V., Saribatur, Z. G., Schüller, P., and Uras, T. (2013). Finding optimal plans for multiple teams of robots through a mediator: A logic-based approach. *Theory and Practice of Logic Programming*, 13(4-5):831–846.

Erdem, E. and Türe, F. (2008). Efficient haplotype inference with answer set programming. In *AAAI*, volume 8, pages 436–441.

Etzion, O., Niblett, P., and Luckham, D. C. (2011). *Event processing in action*. Manning Greenwich.

Falcionelli, N., Sernani, P., Brugués, A., Mekuria, D. N., Calvaresi, D., Schumacher, M., Dragoni, A. F., and Bromuri, S. (2017). Event calculus agent minds applied to diabetes monitoring. In *Agents and Multi-Agent Systems for Health Care*, pages 40–56. Springer.

Falkner, A., Friedrich, G., Schekotihin, K., Taupe, R., and Teppan, E. C. (2018). Industrial applications of answer set programming. *KI-Künstliche Intelligenz*, 32(2-3):165–176.

Feichtenhofer, C., Pinz, A., and Wildes, R. P. (2017). Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4768–4777.

Filippaki, C., Antoniou, G., and Tsamardinos, I. (2011). Using constraint optimization for conflict resolution and detail control in activity recognition. In *International Joint Conference on Ambient Intelligence*, pages 51–60. Springer.

Gan, C., Wang, N., Yang, Y., Yeung, D.-Y., and Hauptmann, A. G. (2015). Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2568–2577.

Garcez, A. d., Gori, M., Lamb, L. C., Serafini, L., Spranger, M., and Tran, S. N. (2019). Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint arXiv:1905.06088*.

Gebser, M., Guziolowski, C., Ivanchev, M., Schaub, T., Siegel, A., Thiele, S., and Veber, P. (2010). Repair and prediction (under inconsistency) in large biological networks with answer set programming. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*.

Gebser, M., Kaminski, R., and Schaub, T. (2011). aspcud: A linux package configuration tool based on answer set programming. *arXiv preprint arXiv:1109.0113*.

Ghallab, M. (1996). On chronicles: Representation, on-line recognition and learning. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 597–606. Morgan Kaufmann Publishers Inc.

Giancola, S., Amine, M., Dghaily, T., and Ghanem, B. (2018). Soccernet: A scalable dataset for action spotting in soccer videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1711–1721.

Ginsberg, M. L. (1988). Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational intelligence*, 4(3):265–316.

Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., and Russell, B. (2017). Actionvlad: Learning spatio-temporal aggregation for action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 971–980.

Girshick, R. (2015). Fast R-CNN. In *IEEE ICCV 2015*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE CVPR 2014*, pages 580–587.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Havur, G., Ozbilgin, G., Erdem, E., and Patoglu, V. (2014). Geometric rearrangement of multiple movable objects on cluttered surfaces: A hybrid reasoning approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 445–452. IEEE.

Hou, R., Chen, C., and Shah, M. (2017). Tube convolutional neural network (t-cnn) for action detection in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5822–5831.

Ielpa, S. M., Iiritano, S., Leone, N., and Ricca, F. (2009). An asp-based system for e-tourism. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 368–381. Springer.

Inoue, K., Bando, H., and Nabeshima, H. (2005). Inducing causal laws by regular inference. In *International Conference on Inductive Logic Programming*, pages 154–171. Springer.

Janiesch, C., Matzner, M., and Müller, O. (2011). A blueprint for event-driven business activity management. In *International Conference on Business Process Management*, pages 17–28. Springer.

Ji, S., Xu, W., Yang, M., and Yu, K. (2012). 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231.

Jiang, H., Lu, Y., and Xue, J. (2016). Automatic soccer video event detection based on a deep neural network combined cnn and rnn. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 490–494. IEEE.

Jiang, Y.-G., Wu, Z., Tang, J., Li, Z., Xue, X., and Chang, S.-F. (2018). Modeling multimodal clues in a hybrid deep learning framework for video classification. *IEEE Transactions on Multimedia*, 20(11):3137–3147.

Jonathan Hui (2018). Real-time object detection with yolo. `https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088/`. Last checked on 2018-03-18.

Kang, K., Li, H., Yan, J., Zeng, X., Yang, B., Xiao, T., Zhang, C., Wang, Z., Wang, R., Wang, X., et al. (2017). T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732.

Khan, A., Lazzerini, B., Calabrese, G., and Serafini, L. (2018). Soccer event detection. In *IPPR 2018*, pages 119–129.

Khan, A., Serafini, L., Bozzato, L., and Lazzerini, B. (2019). Event detection from video using answer set programing. In *CILC 2019*, volume 2396 of *CEUR Workshop Proceedings*, pages 48–58. CEUR-WS.org.

Kim, T.-W., Lee, J., and Palla, R. (2009). Circumscriptive event calculus as answer set programming. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Kolahi, S. and Lakshmanan, L. V. (2009). On approximating optimum repairs for functional dependency violations. In *Proceedings of the 12th International Conference on Database Theory*, pages 53–62.

Kong, Y., Gao, S., Sun, B., and Fu, Y. (2018). Action prediction from videos via memorizing hard-to-predict samples. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kowalski, R. A. and Sergot, M. J. (1986). A logic-based calculus of events. *New Generation Comput.*, 4(1):67–95.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563. IEEE.

Laptev, I., Caputo, B., et al. (2004). Recognizing human actions: a local svm approach. In *null*, pages 32–36. IEEE.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2002). The DLV system for knowledge representation and reasoning. *CoRR*, cs.AI/0211004.

Levesque, H., Pirri, F., and Reiter, R. (1998). Foundations for the situation calculus.

Lévy, F. and Quantz, J. (1997). Representing beliefs in a situated event calculus. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*. Citeseer.

Li, X., Wang, K., Wang, W., and Li, Y. (2010). A multiple object tracking method using kalman filter. In *The 2010 IEEE international conference on information and automation*, pages 1862–1866. IEEE.

Lifschitz, V., Porter, B., and Van Harmelen, F. (2008). *Handbook of Knowledge Representation*. Elsevier.

Liu, T., Lu, Y., Lei, X., Zhang, L., Wang, H., Huang, W., and Wang, Z. (2017). Soccer video event detection using 3d convolutional networks and shot boundary detection via deep feature distance. In *International Conference on Neural Information Processing*, pages 440–449. Springer.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *ECCV 2016*, pages 21–37. Springer.

Lorenzo, D. (2002). Learning non-monotonic causal theories from narratives of actions. In *NMR*, pages 349–355.

Lorenzo, D. and Otero, R. P. (2000). Learning to reason about actions. In *ECAI*, pages 316–320.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Luckham, D. (2002). *The power of events*, volume 204. Addison-Wesley Reading.

Ma, J., Miller, R., Morgenstern, L., and Patkos, T. (2013). An epistemic event calculus for asp-based reasoning about knowledge of the past, present and future. In *LPAR (short papers)*, pages 75–87.

Ma, S., Sigal, L., and Sclaroff, S. (2016). Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1942–1950.

Marszałek, M., Laptev, I., and Schmid, C. (2009). Actions in context. In *CVPR 2009-IEEE Conference on Computer Vision & Pattern Recognition*, pages 2929–2936. IEEE Computer Society.

McCarthy, J. (1963). Situations, actions, and causal laws. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE.

Miller, R. and Shanahan, M. (2002). Some alternative formulations of the event calculus. In *Computational logic: logic programming and beyond*, pages 452–490. Springer.

Mueller, E. T. (2008). Discrete event calculus reasoner documentation. *Software documentation, IBM Thomas J. Watson Research Center, PO Box*, 704.

Mueller, E. T. (2014). *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Nascimento, J. C., Abrantes, A. J., and Marques, J. S. (1999). An algorithm for centroid-based tracking of moving objects. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 6, pages 3305–3308. IEEE.

Neubauer, K., Wanko, P., Schaub, T., and Haubelt, C. (2017). Enhancing symbolic system synthesis through aspmt with partial assignment evaluation. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 306–309. IEEE.

Nguyen, T. H., Son, T. C., and Pontelli, E. (2018). Automatic web services composition for phylotastic. In *International Symposium on Practical Aspects of Declarative Languages*, pages 186–202. Springer.

Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., and Barry, M. (2001). An a-prolog decision support system for the space shuttle. In *International symposium on practical aspects of declarative languages*, pages 169–183. Springer.

Nouman, A., Yalciner, I. F., Erdem, E., and Patoglu, V. (2016). Experimental evaluation of hybrid conditional planning for service robotics. In *International Symposium on Experimental Robotics*, pages 692–702. Springer.

Parameswaran, L. et al. (2013). A hybrid method for object identification and event detection in video. In *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pages 1–4. IEEE.

Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., and Theodoridis, Y. (2017). Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427.

Patroumpas, K., Artikis, A., Katzouris, N., Vodas, M., Theodoridis, Y., and Pelekis, N. (2015). Event recognition for maritime surveillance. In *EDBT*, pages 629–640.

Prapas, I., Paliouras, G., Artikis, A., and Baskiotis, N. (2018). Towards human activity reasoning with computational logic and deep learning. In *SETN 2018*, page 27. ACM.

Quiniou, R., Callens, L., Carrault, G., Cordier, M.-O., Fromont, E., Mabo, P., and Portet, F. (2010). Intelligent adaptive monitoring for cardiac surveillance. In *Computational Intelligence in Healthcare 4*, pages 329–346. Springer.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE CVPR 2016*, pages 779–788.

Reiter, R. (2001). *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press.

Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., and Leone, N. (2012). Team-building with answer set programming in the gioia-tauro seaport. *Theory and Practice of Logic Programming*, 12(3):361–381.

Rodrigues, C., Gérard, P., and Rouveirol, C. (2010). Incremental learning of relational action models in noisy environments. In *International Conference on Inductive Logic Programming*, pages 206–213. Springer.

Rodriguez, M. D., Ahmed, J., and Shah, M. (2008). Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, volume 1, page 6.

Ryoo, M. S. and Aggarwal, J. K. (2009). Spatio-temporal relationship match: video structure comparison for recognition of complex human activities. In *ICCV*, volume 1, page 2. Citeseer.

Schindlauer, R. (2006). *Answer-set programming for the Semantic Web*. na.

Serafini, L. and Garcez, A. S. d. (2016). Learning and reasoning with logic tensor networks. In *Conference of the Italian Association for Artificial Intelligence*, pages 334–348. Springer.

Shinde, S., Kothari, A., and Gupta, V. (2018). Yolo based human action recognition and localization. *Procedia computer science*, 133:831–838.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576.

Skarlatidis, A., Paliouras, G., Artikis, A., and Vouros, G. A. (2015). Probabilistic event calculus for event recognition. *ACM TOCL*, 16(2):11.

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Suchan, J., Bhatt, M., Wałega, P., and Schultz, C. (2018). Visual explanation by high-level abduction: on answer-set programming driven reasoning about moving objects. In *AAAI 2018*.

Thielscher, M. (2005). Flux: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, 5(4-5):533–565.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497.

Tran, S. N. (2017). Propositional knowledge representation and reasoning in restricted boltzmann machines. *arXiv preprint arXiv:1705.10899*.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

Van Dongen, B. F. and Van der Aalst, W. M. (2004). Multi-phase process mining: Building instance graphs. In *International Conference on Conceptual Modeling*, pages 362–376. Springer.

Van Gemert, J. C., Jain, M., Gati, E., Snoek, C. G., et al. (2015). Apt: Action localization proposals from dense trajectories. In *BMVC*, volume 2, page 4.

Wali, A. and Alimi, A. M. (2010). Hybrid approach for event detection from video surveillance sequences. In *2010 5th International Symposium On I/V Communications and Mobile Network*, pages 1–6. IEEE.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

Xia, L., Chen, C.-C., and Aggarwal, J. K. (2012). View invariant human action recognition using histograms of 3d joints. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 20–27. IEEE.

Xu, L., Gong, C., Yang, J., Wu, Q., and Yao, L. (2014). Violent video detection based on mosift feature and sparse coding. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3538–3542. IEEE.

Yalciner, I. F., Nouman, A., Patoglu, V., and Erdem, E. (2017). Hybrid conditional planning using answer set programming. *Theory and Practice of Logic Programming*, 17(5-6):1027–1047.

Zhang, X., Zhang, H., Zhang, Y., Yang, Y., Wang, M., Luan, H., Li, J., and Chua, T.-S. (2015). Deep fusion of multiple semantic cues for complex event recognition. *IEEE Transactions on Image Processing*, 25(3):1033–1046.