Event Detection from Video using Answer Set Programming

Authors: Abdullah khan, Luciano Serafini, Loris Bozzato, Beatrice Lazzerini
Outline

Objective

Recognition of complex events from a simple events in videos.

Methodology

1. Object detection and tracking in videos
2. Logical Framework (Event Calculus) for event recognition
3. Answer set programming (reason about the logical rules).
What is event recognition?

Given an input video/image, perform some appropriate processing, and output the “action label”.

- **gestures**
- **actions**
- **human-object interactions**
- **interactions**
- **group activities**
State of the art in video event detection

- Video data
- Convolutional Neural Networks (CNN)

**Learn** millions of internal parameters from **Data**
- Representations optimized for the training data
YOLO Object detection and tracking?

Divide image into SxS grid

Within each grid cell predict:

Bboxes: 4 coordinates + confidence

Direct prediction using a CNN
Datasets: UCF / J-HMDB

- **UCF-Sports**
  - 10 action categories
  - 150 videos
  - Trimmed

- **J-HMDB-21 (subset)**
  - 21 action categories
  - 928 videos
  - Trimmed

- **UCF-101-24 (subset)**
  - 24 action categories
  - Multiple instances, not whole video duration
Use-case (Handicap Parking Detection)

- 4 min long video, consisting of approximately 6.5k manually annotated frames.

- Objects are detected and tracked from every single frame using the state-of-the-art object detector (YOLO).
Proposed Architecture

1. Simple event detection pipeline
   - Video
   - Object Detection (YOLO) → Object Tracking (Optical Flow) → Recognition of simple events

2. Complex event detection pipeline
   - Recognition of complex events
   - Answer set programming (solvers)
   - Event Calculus Axioms
YOLO (You Only Look Once)

Input video

YOLO (Object Detection/Tracking (YOLO))

https://github.com/AlexeyAB/darknet
YOLO (Continued)

Input video

YOLO (Object Detection/Tracking)

https://github.com/AlexeyAB/darknet
Logical reasoning on Complex events (Event Calculus)

EC distinguishes three kind of objects. *Events*, *fluents*, *time-points*.

*Fluents* are relations whose truth values varies with time.

<table>
<thead>
<tr>
<th>Basic Predicates</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>holdsAt($f, t$)</td>
<td>fluent $f$ is true at time-point $t$</td>
</tr>
<tr>
<td>happens($e, t$)</td>
<td>event $e$ occurs at time-point $t$</td>
</tr>
<tr>
<td>initiates($e, f, t$)</td>
<td>if event $e$ occurs at time-point $t$, then fluent $f$ will be true after $t$.</td>
</tr>
<tr>
<td>terminates($e, f, t$)</td>
<td>if event $e$ occurs at time-point $t$, then fluent $f$ will be false after $t$.</td>
</tr>
</tbody>
</table>
# Simple and complex events

<table>
<thead>
<tr>
<th>Simple event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{appearsCar}(A, T)$</td>
<td>The object corresponding to car $A$ enters the scene at time $T$</td>
</tr>
<tr>
<td>$\text{disappearsCar}(A, T)$</td>
<td>The object corresponding to car $A$ leaves the scene at time $T$</td>
</tr>
<tr>
<td>$\text{appearsSlot}(L, T)$</td>
<td>The object corresponding to parking slot $L$ appears in the scene at time $T$</td>
</tr>
<tr>
<td>$\text{disappearsSlot}(L, T)$</td>
<td>The object corresponding to parking slot $L$ disappears from the scene at time $T$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Complex event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{covers}(A, L, T)$</td>
<td>The object car $A$ covers the slot $L$ at time $T$</td>
</tr>
<tr>
<td>$\text{uncovers}(A, L, T)$</td>
<td>The object car $A$ uncovers the slot $L$ at time $T$</td>
</tr>
</tbody>
</table>
We are currently assuming a simple scenario with one car and one slot in the scene.
Encoding of simple and complex events using EC

Complex events derived from simple events using EC formalism

\[
\text{happens}(\text{covers}(A, L), T) \leftarrow \text{agent}(A), \text{location}(L), \text{time}(T), \\
\text{happens}(\text{disappearsSlot}(L), T), \\
\text{holdsAt}(\text{visibleCar}(A), T).
\]

\[
\text{happens}(\text{uncovers}(A, L), T) \leftarrow \text{agent}(A), \text{location}(L), \text{time}(T), \\
\text{happens}(\text{appearsSlot}(L), T), \\
\text{holdsAt}(\text{visibleCar}(A), T).
\]
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*. 

$$parking(A, L, T_1, T_2) \leftarrow \text{happens}(\text{covers}(A, L), T_1), \text{happens}(\text{uncovers}(A, L), T_2), \text{parkingframes}(N), T_3 = T_1 + N, T_2 \geq T_3.$$
Simple and complex events via Timeline

- Happens(covers(car, hp_slot))
- Happens(uncovers(car, hp_slot))
- Happens(appearsCar(car))
- Happens(disappearsSlot(hp_slot))
- HoldsAt(visible(hp_slot))
- Happens(appearsSlot(hp_slot))
- parking(car, hp_slot)
Query on basic facts from tracker Output

holdsAt(visibleSlot(hp_slot), 0).
happens(appearsCar(car), 1).
happens(disappearsSlot(hp_slot), 2).
happens(appearsSlot(hp_slot), 4).
happens(disappearsCar(car), 5).

Query: if there is a parking in the video? which objects and at what time?

parking(A,L,T1,T2) ?

car, hp_slot, 2, 4.
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 (e.g. 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.
Conclusion

The overall goal of this work is the integration of knowledge representation and computer vision:

1) Visual processing pipeline for detection-based object tracking, leading to the extraction of simple events.

(2) Answer set programming-based reasoning to derive complex events

Future work

For the future work we aim to manage inaccuracies of the tracker output by a (possibly logical based) data cleaning step. We also want to apply and evaluate the presented method in different scenarios e.g. (sports videos)
THANK YOU