Machine Learning Engineer Nanodegree

Capstone Proposal

Juan José Madrigal Martínez
January 30, 2017

Madrid, Spain
(+0034) 600 86 32 48
juanjomadrigal326@gmail.com
LinkedIn GitHub Kaggle Udacity

Domain Background

This project aims at building a video analysis system for surveillance purposes. It basically finds and indexes the movement events filmed by a fixed camera.

Video surveillance has become a major and widely used tool for multiple issues [1] and is supported by many companies [2] [3]. But the huge ammount of information which is usually dealt with has led to the need to use Machine Learning techniques to extract patterns, predictions and other refined information. This approach is being implemented [4] and there is much work for Machine Learning engineers to do in this field.

An efficient implementation of Machine Learning techniques to video surveillance would prevent users from dealing with a sequential (and manual) search through the (perhaps many hours long) video source, which is rather inefficient, boring and error prone, thus providing a major tool for a wide range of purposes.

[1] Wikipedia - Surveillance / Cameras

[2] VideoSurveillance.com

[3] Tyco

[4] Briefcam

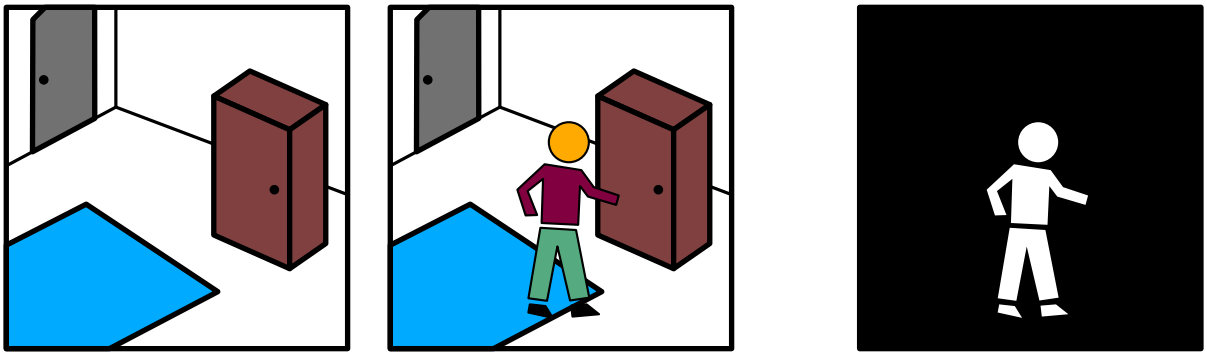
Problem Statement

To apply unsupervised Machine Learning clustering algorithms to detect and quantify the movement events filmed by a fixed camera. The algorithms are to be applied to a tridimensional binary array obtained from the original video after some careful video-preprocessing (see below). Further refinements may also be tackled, such as finding time-parametric curves accurately fitting each movement event.

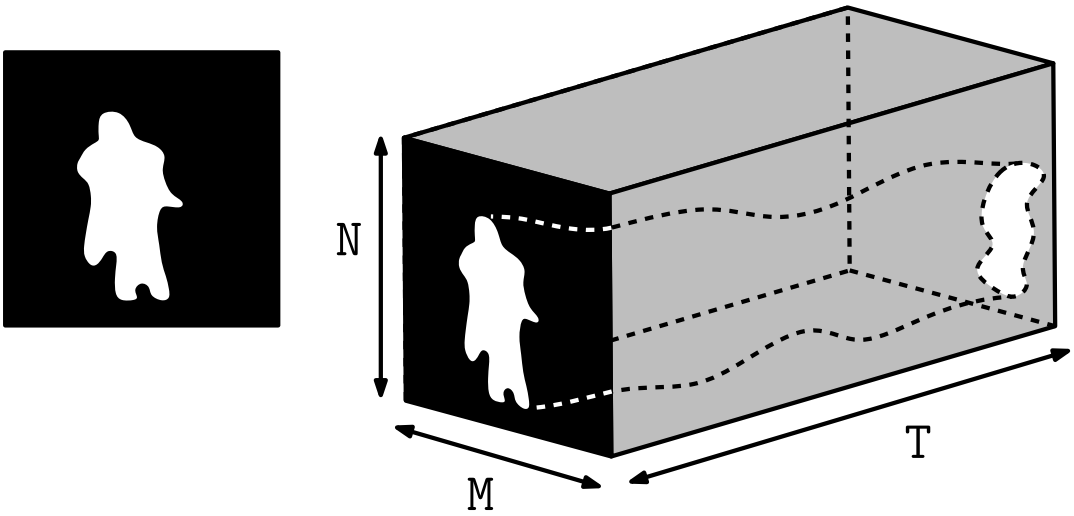
Datasets and Inputs

The datasets and inputs for this project will be obtained after some video preprocessing made on sample videos. The selected videos are located in capstone_project/data/unprocessed_videos and they have been downloaded from VIRAT Video Dataset [1]. The video preprocessing is carried out using the free packages SciKitImage and SciKitVideo and has been generously provided by WardenAutomation.

The video preprocessing takes as input a `.avi` $\mathtt{M} \times \mathtt{N}$ pixels video and a `.png` $\mathtt{M} \times \mathtt{N}$ pixels picture (*background*). For each frame in the video, the frame and the background are compared pixel-wise, and the difference is recorded in a $\mathtt{M} \times \mathtt{N}$ binary array, where 1 denotes difference (above some threshold) and 0 denotes no difference.



If the video has `T` frames, then the video preprocessing gives as output a $\mathtt{M} \times \mathtt{N} \times \mathtt{T}$ binary array, which is supposed to capture the movement in our scene and that will be the primary input for the project. These arrays are saved not as arrays as such but as `.avi` files, and are located in capstone_project/data/processed_videos.



The total ammount of data in this latter folder sums up to 10 videos (and a exploratory `video1.avi`) whose duration ranges between half and three minutes. This should be enough to tune our algorithm (see below) for the purposes of similar (fixed-camera) videos.

[1] **A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video**, Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J.K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xiaoyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai, *Proceedings of IEEE Comptuer Vision and Pattern Recognition (CVPR), 2011*.

Solution Statement

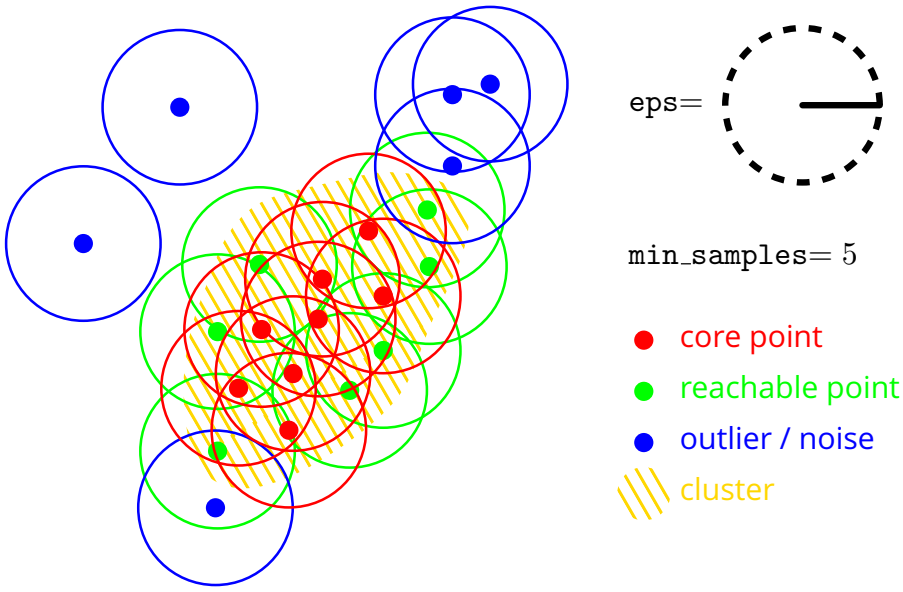
We propose the use of DBSCAN clustering algorithm, which is implemented in SciKitLearn. This clustering algorithm is well-suited for our generic clusters, which are uneven in size and non-convex.

DBSCAN clustering algorithm (*Density-based spatial clustering of applications with noise*) takes primarily two parameters

- `min_samples`, integer
- `eps`, float

and classifies the points in a metric space into three groups

- **Core points**: points whose `eps`-neighbourhood contains at least `min_samples` points
- **Reachable points**: points that are not core points themselves, but that cointain some core point in their `eps`-neighbourhood
- **Outliers or noise**: points that are not core points nor reachable points



Each group of mutually `eps`-path-connected core points with the associated reachable points forms a cluster. The way in which the algorithm is implemented may lead to some indeterminisms, such as reachable points reached from two different clusters, but the number of clusters and their intrinsic shape are well-determined.

This algorithm has some major advantages that make it well-suited for our clustering problem:

- The number of expected clusters is not to be specified. This is crucial, since our goal is to determine the number of clusters / movement-events
- The clusters may be of any shape

- The algorithm deals well with noise (our arrays, generated from video processing, always have noise)

[1] DBScan - Wikipedia

Benchmark Model

A major Benchmark Model may be found at Actions as Space-Time Shapes.

This work deals with the recognition of activities in a single movement-event. For this movement-event to be analysed, spectral clustering methods are used [1]. The performance is measured in terms of misclassifications (2.17%, 7.91% and 36.40% for different methods).

[1] Spectral Clustering - Wikipedia

Evaluation Metrics

The direct metric to tune the main parameters of DBScan (`eps` and `min_samples`) will be the Sum of Square Errors according to the expected value of clusters, which is known on beforehand for each of the training videos.

Once the algorithm is accurately tuned, the metric(s) used to estimate the quality of our clustering will comprise

- Homogeneity
- Completeness
- V-measure
- Adjusted Rand Index
- Adjusted Mutual Information
- Silhouette Coefficient

Project Design

A theoretical workflow for approaching a solution would include

1. To apply the video preprocessing techniques to the selected videos to get the tridimensional binary arrays, which are the main input for our Machine Learning work.
2. To apply DBSCAN clustering algorithm to each array for a first contact with the potential results.
3. To compare the previous results with a manual detection and indexing for a small set of training videos. This may lead to a major refinement of the main algorithms. Together with this manual checking, the metrics mentioned above will be used and compared with the latter, thus finding which metrics work best for our video surveillance context.
4. To apply the refined algorithms and selected metrics to a large variety of videos. Further tuning may be implemented.

5. To provide code for spatial visualisation of all the previous processes.
6. From all the extracted data, to make conclussions about the success of our study and devise new ways or fields for further work.