Report on Assignment 5

Rahul Bangar, Gururaj Mulay, Tariqul Sifat

December 16, 2018

Abstract

In this assignment we integrate the object detection and tracking systems from assignment 2 and 3 by applying a pre-trained deep convolutional network (VGG) on the attention windows and tracked windows to classify the objects within those windows. Finally, we use the classification labels from VGG, with their location and direction of motion to generate meaningful sentences describing the scene in the video.

1 Introduction

Our system consist of four major modules. First is the attention window detection from the background of the video frame. Here, the system captures the most interesting (attention grabbing) windows from the background of the video frame. It detects static attention windows such as house, lamp-posts, etc. Second is the foreground detection where a foreground detecting algorithm gives us a set of objects in the foreground. On the top of this we track the foreground objects using MOSSE tracking algorithm.

For every frame, both the modules produce a set of rectangular windows that are the 'most attentive.' Next, a trained VGG classier processes these attention windows and outputs the most probable label of that window which could be either from the static background or the dynamic foreground. This module writes a csv file with list of labels and their location, direction of motion and frame number where they were detected. Finally, we describe the video scene in sentences using the 'scene description' module. A short overview of all the implemented components in the system is given below. The next section will discuss the integration of the components that are mentioned above.

- **Attention Window Detection from Background:** PA2
  - SIFT: to detect the keypoints of interest
    - Threshold-based detection (naive, naive2)
    - Clustering-based detection (HCA, DBSCAN)
  - Saliency-based detection
- **Foreground Detection and Tracking:** PA3
  - Foreground Detection: MoG [6] (actually used), ViBE [7]
  - Window Tracking: MOSSE [4, 5]
- **Classification** PA4
  - Pre-trained VGG [1, 2]
- **Scene Description** PA5
  - Background and foreground description algorithm

For attention window detection, we are currently using DBSCAN to cluster the keypoints that form the window. In order to detect the foreground, we currently use Mixture of Gaussian approach [6]. A detailed description on why we selected a particular algorithm for that module and how the algorithms work is out of scope for this report and it can be found in the reports of earlier assignments.

2 Description

In this section, firstly we discuss newly implemented parts such as the scene description module, re-implementation of saliency-based attention window detection. Secondly, we describe the integration of the four modules, viz. attention window detection, foreground detection and tracking, image classification, and scene description module.

Also, we discuss about the post-processing of the captured attention windows from each frame of the test video.

2.1 Scene Description Module

The outputs produced by the VGG module is a csv file with following format.

`type, start_frame, end_frame, x1,y1, x2,y2, label, frequency(%)/activation`

For the moving objects the `type` will be 'moving' and for stationary objects it will be 'still.' The `start_frame` and `end_frame` represents the start and end frame number for tracked moving objects. For still objects both the values are the same since they are detected only for a single frame. `x1,y1, x2,y2` represent the upper-left and lower-right coordinates for 'still' type of objects. For moving objects it represents the centers of the tracked window at the 'start frame' and the 'end frame' respectively. `label` is the classification label outputted by the VGG network, for example, a barn or pickup truck. Finally, the `frequency(%)/activation` in 'moving' case represents the percentage of frames where the given 'label' was detected. In 'still' case, it gives the the activation level of the output label from VGG.

2.1.1 Current implementation:

The current approach to describe the video in sentences is a simple approach based on the `type` of the object from the csv file. We describe the video in terms of foreground and the background that are labels as 'moving' and 'still' in the csv file. For example, following is the scene description produced for the first test video (example1.mov) and the seconds test video (example2.mp4) respectively.

In the background, there is a barn in the middle left, a flagpole in the upper right, a boathouse in the middle left, a thresher in the middle left, a limousine in the middle left. In the foreground, a minivan (moving towards right slowly) AND a pickup (moving towards left slowly) are moving in opposite direction.

In the background, there is a racer in the center, a mobile home in the middle left, a maze in the center, a paddlewheel in the upper left, a aircraft carrier in the middle left, a warplane in the middle left, a liner in the upper left. In the foreground, a moving van (moving towards left slowly) , a limousine (moving towards left slowly) are moving in same direction.

In order to describe the 'location' such as *upper left, lower right*, we divided the rectangular frame into 9 rectangular sections: 3 along the width and 3 along the height. Therefore the object lies in the upper right corner if the center_x and center_y of that object frame satisfies following condition `(2*width/3 < cx < width) and (0 < cy < height/3)`. Here is a snippet of the logic that assigns a location to every object.

```
if (0 < cx < width/3) and (0 < cy < height/3):
    position = "the upper left, "
elif (width/3 < cx < 2*width/3) and (0 < cy < height/3):
    position = "the upper middle, "
elif (2*width/3 < cx < width) and (0 < cy < height/3):
    position = "the upper right, "
    .
    .
```

Moreover, we say that two objects are moving in the same direction if they have same direction of their velocity vector after discretization (shown in the snippet below) the directions into four major direction (left, right, up, and down). On the other hand, if they have opposite direction for their velocity vectors then they are moving in opposite direction.

```
if np.abs(dx) > np.abs(dy) and dx > 0:
    direction = "towards right"
elif np.abs(dx) > np.abs(dy) and dx < 0:
    direction = "towards left"
elif np.abs(dy) > np.abs(dx) and dy > 0:
    direction = "towards lower edge"
elif np.abs(dy) > np.abs(dx) and dy < 0:
    direction = "towards upper edge"
```

Similarly, we say that an object is moving 'slowly' if its speed in pixels per frames is lesser than a threshold value.

```
motion = "slowly" if ((np.maximum(np.abs(dx), np.abs(dy))/frames_diff) < 4) else "↩
    relatively fast"
```

2.1.2 What we tried?

In order to generate a meaningful sentence given a set of words such as nouns (labels from VGG), verbs (moving or still), adverbs (slowly, rapidly), we decided to design a Markov Model that will predict the next most probable word given earlier set of words. We found the most frequently (frequency>10) occurring bigrams and trigrams from the gutenberg dataset [10]. This dataset is a corpus of books in various genres. From this corpus we extracted about 1M to 2M bigrams and trigrams each. The bigrams and trigrams that we found with more than 10 occurrences did not contain words like 'parking lot' or 'car moving right' etc. Therefore, we concluded that we need a corpus of sentences specialized for describing the outdoor traffic videos. Unfortunately, we did not find such corpus with the words that are appropriate to describe the videos that we are using for testing. Therefore, we decide to go with a simpler approach described earlier.

Finding color of the object: We tried histogram based methods to find the dominant color of the object in the attention window from PA2 and PA3. However, it turned out to be a difficult task to find the color of the actual object ignoring the background in the attention window. In HSV color space, the average Hue value of all the pixels from a particular attention window varied from 30 to 60 for every object. Ideally, it should vary from 0 to 180 to find the color based on the Hue value. So the color range of hue value from 30 to 60 did not provide any distinctive information about the color of various objects.

2.2 Re-implementation of saliency-based attention window detection

Visually salient parts of an image are a good candidate for finding attention windows. Current approaches for saliency usually work with static images instead of videos. To adapt it to a video, the idea was to pick the most salient object in a frame as the focus of our attention window while avoiding moving objects or static objects that are weakly salient. The basic approach is a simplification of the one described in [14].

To realize this, we used an accumulation and leakage layer on top of the saliency map output. First, we normalized the grayscale output of saliency map into 0.0 to 1.0. This output is accumulated into a global saliency map over time by simple addition. The maximum of the global saliency map is picked as the center of the attention window for that frame. At each step, however, there is also a leakage of saliency controlled by a leakage parameter between 0 to 1. The leakage parameter drains the global saliency at a fixed rate every frame. This avoids the accumulation of weakly salient objects over time, while also avoiding the detection of moving salient objects that are better handled by motion detection module. Whenever an attention window is chosen, the area corresponding to it is reset to 0.0 to avoid being detected in the next frame again.

For our implementation, we used Swig [12] to convert existing C++ implementation of Boolean Map Saliency[13] into a Python wrapper, which performs almost as fast as the original C++ implementation while also allowing us the ease of use of Python.

2.3 Integration of Components

Avoiding foreground detection in PA2 (foreground overlap checking): Since we are using SIFT features to find the attention window, our system finds attention windows from foreground as well as background. However, we only need the background objects, because the foreground objects will anyhow be detected and tracked using background subtraction algorithms like MoG, ViBE, etc. [6] [7] and the tracking algorithms like MOSSE [4]. Therefore, we ensure that a window detected from PA2 is from the static background of the video frame. For each frame we build a union of the foreground rectangles and make sure that no attention window overlap with the union of the foreground rectangles more than a certain threshold (30%).

2.4 Pre-processing Images and Testing with VGG

Detecting a large enough attention window: A lot of the attention windows detected by the our PA2 are very small even after clustering. We made sure that the smallest size of the attention windows is 224x224. We did not upscale the attention windows but grabbed a larger chunk of image from the frames if the original attention window is smaller than 224 in any dimension.

Avoiding Repetition of detected classes: Over the period of the whole video our PA2 (attention window detector) can end up detecting same object multiple time. We ensure that a window detected at a particular time frame is not detected again after a specific 'no repetition' period (30 frames) has passed. To that extend, we make sure that our top five strongest (in terms of activation) objects are unique (in terms of the class label).

3 Evaluation

We implemented a saliency-based attention window detection method. This method gives better results in detection of attention windows as compared to DBSCAN clustering of SIFT keypoints.
Re-evaluation of PA2: The metric measures the percentage of frames where an interesting attention window was captured. For older systems with DBSACN approach, the percentage was **34.24%** since remaining windows were predominantly detected from the trees. For the current system with saliency-based approach, the percentage of interesting attention windows went up to $152/219 =$ **69.40%.**
Re-evaluation of PA3: Since we did not change our 'foreground detection and tracking' module, the evaluation results did not change for PA3. They are as follows.

Using the 'Frame-based Metric' described earlier in PA3 report, the confusion matrix w.r.t. ground truth is shown in table 1. With respect to 'Window-overlap Metric,' the IOR–OOR mean increased from 0.2 & 0.1 to 0.70 & 0.63 for truck0 & truck1 respectively.

		Ground Truth		
		Positive	Negative	Total
SUT Output	Positive	$TP = 145$	$FP = 0$	$TP + FP = 145$
	Negative	$FN = 11$	$TN = 63$	$FN + TN = 74$
	Total	$TP + FN = 156$	$FP + TN = 63$	

Table 1: Confusion Matrix (on video example1.mov with overlapThreshold=0.25)

Re-evaluation of object classification using VGG-net (PA4): We tested the system on two videos, viz., example1.mov, example2.mp4 both of which are submitted along with the code. The csv files produced for both the videos (classes1.csv and classes2.csv) are also submitted.

VGG paper reports the top-1 error rate on their test dataset to be 28.07%. We tested the VGG net on the test images produced by our detection and tracking systems. As a ground truth, we hand-labeled the tracked images produced as the attention windows by the systems. In video1, the testing errors for the second system (tracked attention windows from PA3) was 4.86% and 98.7% for track0 and track1 respectively. For video2, the tracks were *Sedan*, *SUV*, and *SUV*. However, our system labeled them as moving van (95.14%), limousine (98.51%) respectively. It did not detect the third SUV. Thus, technically the error rate is close to 100%. But, among the 1000 labels of ImageNet there are no *Sedan* or *SUV*. So, VGG had no chance of detecting those car models. In this case, as long as VGG can identify that those are some type of car we can consider that as correct labeling. The third moving object (SUV) did not get detected because the foreground pixels generated by it did not create a large enough window to be detected in the short video.

Output Label	Ground Truth	Activation
barn	barn/house	0.82
flagpole,flagstaff	pole	0.7
pole	pole	0.56
boathouse	house	0.5
thrasher	house	0.47

Table 2: Video1 (example1.mov) activation levels for top 5 attention windows from PA2

Output Label	Ground Truth	Activation
racing car	car	0.74
mobile home	mobile cabin	0.61
maze	home	0.6
paddle wheel	home	0.5
aircraft career	parking lot	0.38

Table 3: Video2 (example2.mp4) activation levels for top 5 attention windows from PA2

Outputs of classification on PA2: Table 2 and 3 lists the top 5 (in terms of activation level) objects classified by the VGG net for video 1 and 2 respectively.
Outputs of classification on PA3: Table 4 and 4 lists all the tracks detected by the MOSSE algorithm. The match % indicates the percentage of the frames where classifier output matched with the ground truth (top-1 sense). Moreover, for the second track, VGG classified it as minivan 76.67% of the times as indicated by the value in the parenthesis. However, it is classified as a pickup truck (which is the ground truth) only 1.3% of the times.

Unfortunately, for video2 the predicted label is different from the ground truth label causing the 'Match %' to be very close to 0%. If we consider the fact that VGG does not have the labels that we picked as ground truth and the best VGG can do is tell us that it was some type of car, we can argue that the accuracy is at least 66.67% (2 out of the 3 moving objects detected).

Track Label (%)	Ground Truth	Match %
pickup truck (95.14)	pickup truck	95.14
minivan (76.67)	pickup truck	1.3

Table 4: Video1 (example1.mov) percent match of tracked windows with ground truth labels

3.1 Evaluation of Scene Description (Sentence Generation) (PA4):

The modules used for constructing a scene description system were: DBSCAN clustering on SIFT keypoints (PA2), MOG with MOSSE tracker (PA3), and VGG classifier (PA4).

The sentence generation module describes the still background and moving foreground objects as they are labeled by the VGG-net. Given the fact that the scene description algorithm is not completely 'unsupervised,' it does a satisfactory job of describing the background and foreground, with object labels, their location in the scene, and the approximate direction in which they are moving.

4 Conclusion

We improved the background attention window detection method by implementing a saliency based approach. The scene description sentences produced by our algorithm gives a general description on the objects present in the background and foreground along with their location in the video frame and direction of motion. This algorithm is generic and can be applied to any video to describe the background and foreground. However, the current algorithm provides a **'text template'** which is then filled in depending on the number of objects (moving or still) in the video. Therefore, the scene description is rigid in the sense that it does not show any 'intelligence' in describing a scene.

In future, we could improve the description by describing the colors of the objects, overall context of the scene (such as parking lot, highway). Moreover, we need to understand the 'depth'

Track Label (%)	Ground Truth	Match %
moving van (95.14)	Sedan	0%
limousine (98.51)	SUV	0%
(untracked)	SUV	0%

Table 5: Video2 (example2.mp4) percent match of tracked windows with ground truth labels

property of every object to understand their relative positions. Ultimate goal would be to develop a generic algorithm that describes the scene in a non-rigid and intelligent manner without having a 'text template' to start with.

References

[1] Simonyan K. and Zisserman A., Very Deep Convolutional Networks for Large-Scale Image Recognition https://arxiv.org/pdf/1409.1556/

[2] Pre-trained VGG http://www.cs.colostate.edu/~cs510/yr2017sp/more_assignments/vgg.zip

[3] Itseez, Open Source Computer Vision Library, 2015, https://github.com/itseez/opencv

[4] David S., Bolme J., Ross Beveridge, Bruce A. Draper, Yui Man Lui, Visual Object Tracking using Adaptive Correlation Filters, Computer Science Department, Colorado State University, Fort Collins, CO 80521, USA

[5] Open Source Computer Vision Library Samples (Python), Steven Puttemans, 2016, https://github.com/opencv/opencv/blob/master/samples/python/mosse.py

[6] Stauffer C., Grimson W., Adaptive background mixture models for real-time tracking, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139

[7] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. In IEEE Transactions on Image Processing, 20(6):1709-1724, June 2011.

[8] A. Godil, et al., Performance Metrics for Evaluating Object and Human Detection and Tracking Systems, NISTIR 7972, July 2014

[9] F. Bashir, Performance Evaluation of Object Detection & Tracking Systems, CVPR, June '06

[10] http://www.cs.colostate.edu/ cs435/datafiles/PA1/gutenberg.tar

[11] http://web.mit.edu/vondrick/vatic/ + https://dbolkensteyn.github.io/vatic.js/

[12] http://swig.org/

[13] https://github.com/rcrowder/OpenCV-Attention-and-Saliency

[14] L. Itti, C. Koch and E. Niebur, "A Model of Saliency Based Visual Attention for Rapid Scene Analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 11, 1998, pp. 1254-1259. doi:10.1109/34.730558