PA-4 : Object Labeling using VGG

Avinash Konduru, Nikhila Chireddy
Department of Computer Science,Colorado State University
04-17-2017

I. Introduction:

This paper presents a method to label the objects that are obtained from attention windows extraction as well as object motion detection and motion tracking.

For identifying the attention windows, we have used the SIFT[1] algorithm which uses the on-center off-surround methodology. Using this algorithm, we extract one unique attention window per frame. The extraction of attention windows is explained in detail in Section – II.

To detect and track the moving objects, we have used the MoG[2] method (*a la* Stauffer and Grimson) and the MOSSE[3] filter. By using MoG method we identify foreground pixels and then group them to obtain objects. For each foreground object detected, we track it using MOSSE filter. Motion detection and tracking are explained in detail in Section – III.

After extracting the attention windows and detecting the moving objects, we apply VGG[4] to each of them to obtain a label. Along with the label, VGG also returns a value which represents the probability that the label assigned is correct. VGG method is explained in detail in Section – IV.

Section – V covers the experimental results and Section –VI concludes the paper.

II. Attention Windows Extraction:

There are several approaches for identifying the attention windows. Most common among them are SIFT(Scale Invariant Feature Transform), SURF(Speeded Up Robust Features), Saliency detection, etc. The SIFT algorithm is used to detect and describe the local features in images and then identify the same object in different images. In our approach, for each frame we first identify the key points and sort them based on the diameter. We now pick top 20% of the keypoints (keypoints with higher diameter), draw a rectangle around them and then add them to a list. We also pick the bottom 20% of keypoints and group the keypoints which are close to each other. These grouped keypoints are also appended to the list. In this way we make sure that we don't completely avoid the keypoints with lesser diameter. From this list, we now pick the rectangle which has the highest dimension.

To be sure that we are extracting unique attention windows, we maintain an attention windows list. Each time an attention window is obtained, we check if this has already occurred in any of the previous 30 frames. If the window has occurred previously, we drop it and select another attention window and if it has not occurred and append it to the list. In this way we make sure that the attention windows are not repeated for 30 frames.

We need to consider another case where we pick the moving objects as attention window. Since the moving objects are already being picked, we try to avoid picking the same objects as attention window. To do this, we take the list of attention windows and the list of moving objects identified in a frame, and then calculate the distances among their centers. If the distance is lesser than the radius of both the objects, then we drop the window and go for the next window.

Thus we ensure that the attention windows obtained are not repeated for 30 frames and also are not identifying the moving objects.

III. Moving object Detection and Tracking:

To detect the moving objects, we need to find the foreground pixels and group them. To do this we have many approaches like – MoG(Mixture of Gaussians), extracting the median image and subtracting it, statistical sampling models(like ViBE[5]), etc. In our approach, we have used the Mixture of

Gaussians method. For each frame, we apply the OpenCV MoG method and obtain the foreground mask. We then group the nearby foreground pixels to form objects. To do this, we have used the OpenCV findContours() method.

For each frame, to detect new objects and avoid detecting the same objects repeatedly, we maintain an object list. Whenever a moving object is detected, we compare it with the objects in the object list and if a match is found we ignore it, otherwise add it to the list. To check for a match, we first use the SIFT algorithm and obtain the keypoint descriptors of the objects we are comparing. We then calculate the distance among these descriptors and check for a match using OpenCV flann matcher. If a match is found, we ignore the object, otherwise we add it to the object list.

After detecting the moving objects, we start tracking them. For this purpose we have implemented a MOSSE filter with certain references[6]. For each object in the object list, we initialize a MOSSE object and this tracks the object till it leaves the frame. In this way, a track is created for all the objects.

IV. Assigning Labels to objects:

For all the attention windows obtained and the moving objects detected, we try to label them using the VGGnet. VGG can categorize about 1000 different items. It takes a 224*224 image as input and produces a label and the correctness percentage of that label. To make our windows of size 224*224, we apply affine transformation on the image. The obtained image is then passed to VGG and thus we obtain a label.

After obtaining all the labels, we output the 5 best still images and all the moving objects. The 5 best still images are those which have the highest probability.

V. Experimental Results:

The labels of all the objects identified are maintained in a CSV file. This file consists of 7 fields. For still objects it stores the - type (still), frame #, frame #, x (upper left), y (upper left), object label, and activation level. And for moving objects it stores- type (moving), start frame #, end frame #, x (upper left, first frame), y (upper left, first frame), object label, and percent of frames matching that label.
The CSV file looks like:

StartFrame#	EndFrame#	X(UL)	Y(UL)	ObjectLabel	ActivationLevel	ObjectType
59	59	45	211	barn	0.621336	STILL
21	21	257	254	planetarium	0.527987	STILL
74	74	965	66	flagpole, flagstaff	0.288741	STILL
11	11	964	74	pole	0.276077	STILL
46	46	-65	210	sliding door	0.264637	STILL

Table 1: Output CSV file

Our system works well for still objects. It gives unique attention windows in each frame(when compared with last 30 frames) and also avoids detecting the moving objects as still objects. The detection of moving objects also works well but then sometimes it fails to detect few objects. This happens when there are two similar objects. For example, if there was a car of model A and color B which was already identified and after few frames another car with the same feature comes in, our system fails to identify it as a new object. Also we need to identify the keypoint descriptors of the object every time and then calculate the distances among them. To avoid these computations we have tried using the OpenCV MatchTemplate method for comparing objects. We tried to search for the object in the previous frame

and if the object was found and detected in the previous frame, we would ignore it. Though this approach worked well initially, there were few problems with it. Even if a match is not found, the MatchTemplate would return some location where there is a minimum match and thus it would not get added into the list. One way to avoid this is to keep a threshold value for the minimum value returned by the MatchTemplate method. But this minimum value varies from video to video and we were not able to identify one standard threshold value. Also the results produced by flann matching were better compared to this. So we decided to proceed with the previous approach i.e., flann matching.

VI. Conclusion:

To detect the still objects we have used the SIFT method which gives the keypoint detectors. After obtaining the keypoints, we select the keypoints which have higher diameter as the attention window. To detect the moving objects, we have used MoG method to identify the foreground pixels and then we cluster them to form objects. For every object detected, we track it using the MOSSE filter. After obtaining the still objects and the moving objects, we try to label them using the VGGnet. To do this, we resize out objects to 224*224 and then pass them as input to VGG. VGG searches for the best match and returns a label as well as the probability of the label being correct.

Our system currently takes about 10 minutes to process a 7 seconds video without labeling. For labeling using VGG(with 1.4GHz Intel core I5, 4GB 1600MHz DDR3), it takes about 1hour. The motion detection module takes some time for execution since it has to get the keypoint descriptors and then calculate the distances. So in future we may consider using the MatchTemplate and find a fix to avoid the objects which do not match.

VII. References:

1. Distinctive Image Features from Scale-Invariant Keypoints – David G. Lowe
2. Adaptive background mixture models for real-time tracking – Chris Stauffer, W.E.L. Grimson
3. Visual object tracking using Adaptive Correlation Filters – David S. Bolme, J. Ross Beveridge, Bruce A. Draper, Yui Man Lui
4. Very Deep Convolutional Networks for Large-Scale Image Recognition – Karen Simonyan, Andrew Zisserman
5. ViBe : A visual background subtraction algorithm for Video Sequences – Olivier Barnich, Marc Van Droogenbroeck
6. https://github.com/opencv/opencv/blob/master/samples/python/mosse.py
7. http://docs.opencv.org/3.2.0/d4/dc6/tutorial_py_template_matching.html
8. http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html
9. http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html