Report on Assignment 2

Rahul Bangar, Gururaj Mulay, Tarequl Islam Sifat

February 2017

1 Introduction

The purpose of this assignment was to define an attention window per frame of a video, while also ensuring Inhibition of Return (IOR) for last 30 frames. In this context, our team tried a multitude of approaches, which are explained below.

2 Methods

2.1 Size-Response Sorting

SIFT can identify thousands of keypoints for every frame. Each feature is associated with an estimate of relevant neighbourhood and a response. Higher response implies greater stability for the keypoint. Assuming that we are trying to find attention windows that are large and correspond to highly responsive keypoints, the approach involved taking a product of both size and response, which would yield a score for each keypoint. However, such an approach has an inherent problem in that sizes could overshadow the responses, which are usually very small. To address this, the size and responses were both normalized to fall between 0 and 1. Having normalized these, a product was taken to yield a score for each keypoint. Then, the keypoint with the highest score that also satisfied the inhibition of return requirements was chosen to represent the attention window.

2.2 Clustering

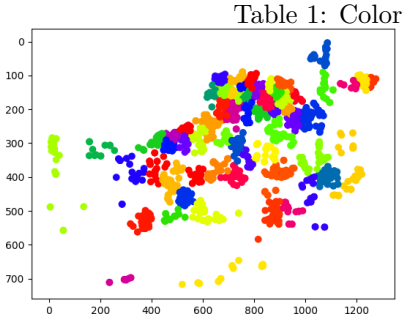
As discussed above, SIFT gave us a set of keypoints for every frame with attributes like location, size, response, octave, scale, etc. For every frame we get a large number of keypoints which are concentrated around 'interesting' regions in the frame. Selecting just a single, best keypoint based on a metric will disregard the other existing keypoints. So, to account for the influence of every keypoint, clustering of the keypoints based on their location in the frame was done. We implemented two methods of clustering viz., hierarchical and DBSCAN (available in sklearn python [4]). We tried a variety of distance metric and linkage criteria for keypoint locations and cluster sets as per the clustering documentations [5].

2.2.1 Hierarchical Clustering

Hierarchical clustering (HCA) has an advantage that we do not need to know the number of clusters a-priori. Figure shows the clusters formed by HCA applied on the filtered set of keypoints from SIFT. Filtering criteria (by size, response, scale, etc) are discussed in earlier sections. Pre-filtering removes the spurious keypoints from the blue sky and the grey ground that do not have any intersecting attention window to grab. Each color belongs to a cluster of keypoints closer together in terms of distance measure specified. The `fcluster(Z, t=2.5, depth=10)` method allowed to specify the depth of clustering dendogram. We observed that to get better clusters we need to fine-tune the parameters. However, these parameters are dependent every frame's contents.

We did clustering with SIFT applied on original grayscale image and the image outputted by applying Canny filtering on the grayscale image. Canny filtering removes higher frequency components and gives the

Table 1: Color Coded Clusters from HCA



edges in the frame. However, we observed that the normal method without Canny filtering retains some interesting high frequency clusters. So, we stick with that method henceforth.

Selecting the best cluster: For every frame we have a set of clusters from which we need to select the best cluster for displaying and windowing it as a output frame. So from the set of clusters, we sort them by the product of mean value of kepoint size and the mean value of response, in descending order. Now, for these sorted sets of clusters, we itererate over every cluster and find the mean in x and y direction to locate the center of the cluster. Finally, the window is found by adding to the center 0.5*standard deviations in x and y directions respectively. We check if this window is already existing and add it if not. So the clusters with highest mean sizes and responses will get prioprity over the others.

2.2.2 DBSCAN

Similar to Hierarchical clustering DBSCAN also does not require us to know the number of clusters beforehand. It has an added advantage of rejecting outliers as noise. We used pythons `sklearn` library for the DBSCAN clustering. We first grouped the keypoints according their scale (sigma value). We ran DBSCAN clustering on each of the groups of keypoints separately. Apart from using the distance between keypoints, we also used a histogram of a 11x11 size patch of image around the keypoints as a measure of similarity. We calculated histograms for all three color channels (BGR) and concatenated them. This approach is inspired by [1]. Then we concatenate the histogram with the position vector to create a feature vector for each of the keypoints. While clustering we put 100 times more weight on the position vector than on the histogram bins. The challenging part of this approach is to find the right ratio of weights between the position vector and the histogram bins. We choose different parameters for DBSCAN when clustering different groups of keypoints. Optimizing the parameters further will take more empirical studies. After we found a set of clusters from different groups of keypoints, we choose the cluster with highest mean response of it's keypoints as the best attention window.

2.3 Saliency

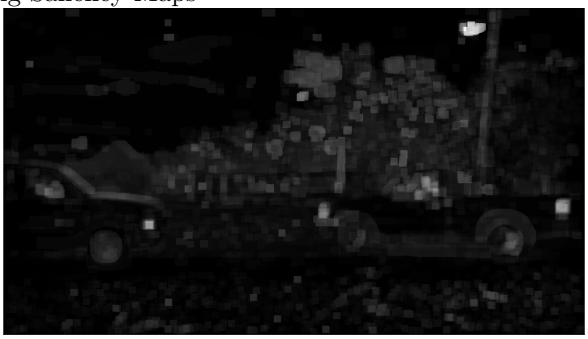
There are many saliency based attention models available. We opted to try **Boolean Map based Saliency** [2] approach, as the authors assert state of the art performance. This method borrows from the Boolean Map theory of attention, which postulates that an observer's momentary conscious awareness of a scene can be represented by a Boolean Map. The model assumes these boolean maps to be a function of image's "feature maps", which can be color channels, depth maps, orientations, etc. Each feature map is thresholded with a configurable step size to generate boolean maps. Each boolean map is also accompanied by an inverse boolean map, which is simply the original boolean map, but inverted. These boolean maps are then smoothed out using morphological opening operations.

The boolean maps generated from all the feature maps are used to create attention maps. Each attention map marks a set of surrounded regions in the image. 1s (0s) bounded by a boundary of 0s (1s) are defined as being surrounded. Under this definition, the connected components at the boundary are the only ones that are not surrounded. The surrounded regions so found are dilated and normalized, and then linearly combined to form a mean attention map. The mean attention map can be post-processed to create a saliency

map, or binary thresholded for creating a mask for image segmentation.

Since a ready-made implementation of Boolean Map based Saliency wasn't available, we implemented it from scratch using Python and OpenCV [3]. However, it seems that output of our algorithm (a Numpy array) is not incompatible with most OpenCV functions, except `imshow`. Lacking time to investigate the issue further, we opted to modify the parameters (opening width, normalization, etc.) so that the algorithm produced a binary image output instead. The parameters for this were chosen as suggested by the authors. The binary image so produced is used as a mask for the SIFT algorithm to limit the search for keypoints to only salient regions. These keypoints were then processed in the same fashion as Size-Response Sorting to yield the attention windows.

Table 2: Generating Saliency Maps



3 Results

Lacking an objective standard for comparison, we only provide subjective comments. We note that saliency databases provide eye-fixation heatmaps, which might be a good way to compare our own attention windows. AUC and shuffled-AUC are commonly used for this purpose.

Using basic size-response sorting to find relevant keypoints seems to give attention windows as required by the base part of the assignment. However, it is easily seen from the results that such an approach suffers a lot by giving attention windows of uniform-textures like sky, ground, etc. On the plus side, it is fast.

HCA provided a majority of windows that were detected in from trees in the background since the higher frequencies in the trees was detected frequently by SIFT. Use of Canny filtering doesn't help to smooth or remove the keypoints from trees. Also, the windows were significantly overlapping. We attribute this to the outlier keypoints from the cluster that caused the size of windows to grow. We need to address this by devising a better approach to crop a window around a cluster.

While retrieved attention windows from our DBSCAN implementation are distinguishable, seems like a good amount of time we seem to get a window from top of the trees. The lampposts and the headlight of the right car seem to pop up several times.

Saliency based approach should provide us with the most salient objects in the scene, which would be a good candidate for an attention window. Due to issues described earlier, however, we could not pursue this further, and instead, settled for thresholding the saliency map to a binary mask to be used with SIFT. The resulting attention windows, in our opinion, weren't much better than the ones given by SIFT itself. This can be explained by the fact that using a saliency map as a binary mask forfeits the advantages of the saliency map itself, which benefits from a fine grain representation. We hope to resolve these issues, and use the saliency map itself for finding attention windows in future.

References

[1] Francisco J. Estrada, PascalFua, Vincent Lepetit, Sabine S, "Appearance-based Keypoint Clustering", University of Toronto at Scarborough.

[2] Jianming Zhang and Stan Sclaroff. "Saliency Detection: A Boolean Map Approach." In Proc. IEEE Internetional Conference on Computer Vision (ICCV), 2013

[3] http://www.opencv.org

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html #scipy.cluster.hierarchy.linkage