Discovering Visual Hierarchy through Unsupervised Learning

Haider Razvi
hrazvi@stanford.edu

1 Introduction:

We present a method for discovering visual hierarchy in a set of images. Automatically grouping and organizing images has applications for personal use where personal photographs can be arranged as overexposed or underexposed, urban or rural landscape, closeups or family gatherings (lots of people etc) besides various other useful hierarchies. This also has applications in the educational/research domain where it would be interesting and maybe useful to find out if a large dataset as the ImageNet [1] can be arranged in other interesting hierarchies. Effective hierarchies allow for efficient searching of images and better image recommendations (i.e. other images similar to a certain image of interest).

This problem is broken down into a problem of selecting appropriate features of interest from an image and then organizing images based on these features. We must be able to handle variations in scale, viewpoint, lighting, occlusion etc.

2 Extracting Points of Interest:

2.1 MSER:

The MSER [2] method is used to detect blobs in images, blobs are patches in the image that are either brighter or darker than their surroundings. Blob detectors work differently from edge detectors and provide complementary information about patches in the image, such as their shape etc. These can be used to obtain regions of interest for further processing. The MSER algorithm basically sorts pixels by intensity and considers pixels below a threshold to be black and those above it to be white. It does this for a range of thresholds and determines regions based on the stability of a group of pixels w.r.t to the change in thresholds. It is invariant to affine transformation of image intensities and is quite robust to viewpoint and scale changes.

Once the MSER method returns points of interest (also called region seeds), lists of pixels belonging to each region are extracted. The VLFeat library is used to do this [7]. Binary images are generated for each feature by setting pixels not belonging to that region as zero. Finally SIFT based descriptors for these features are generated. This is similar to the approach outlined in David Lowe's paper [4]. The descriptors are 128 dimensional vectors with values from 0-255.

The MSER method can sometimes give incorrect results where it considers multiple adjacent objects in the image as a single large feature/region. Two methods are used to guard against this,

1. All features which comprise more than 80% of the entire image's pixels are rejected.
2. All features which comprise between 50-80% of the image's pixels are attempted to be broken down into sub-features. For this purpose a bounding circle of the feature is calculated and SIFT is used to detect features within this bounding circle. Note: We

don’t re-run MSER on the area within the bounding circle as MSER would again consider any pixels falling within this bounding circle to be one feature. This almost always doesn’t give the correct shape so running MSER again isn’t very useful.

Figure-1 shows MSER features extracted for a sample image alongwith a binary image for one of the features. Figure-1E shows an example where MSER goes wrong and considers multiple objects as a single large feature. The calculated bounding circle is also shown. Figure-1F shows an example where MSER performs reasonably well.

To calculate the bounding circle for a detected feature, the binary image represented as a 2-D matrix is linearly indexed (using the same conventions as MATLAB) with the top left corner of the image taken as index1. A white pixel with the maximum linear index w_r gives the bottom pixel in the right-most column of the feature/region while a white pixel with the minimum linear index w_l gives the top pixel in the left-most column of the region. The pseudocode is then,

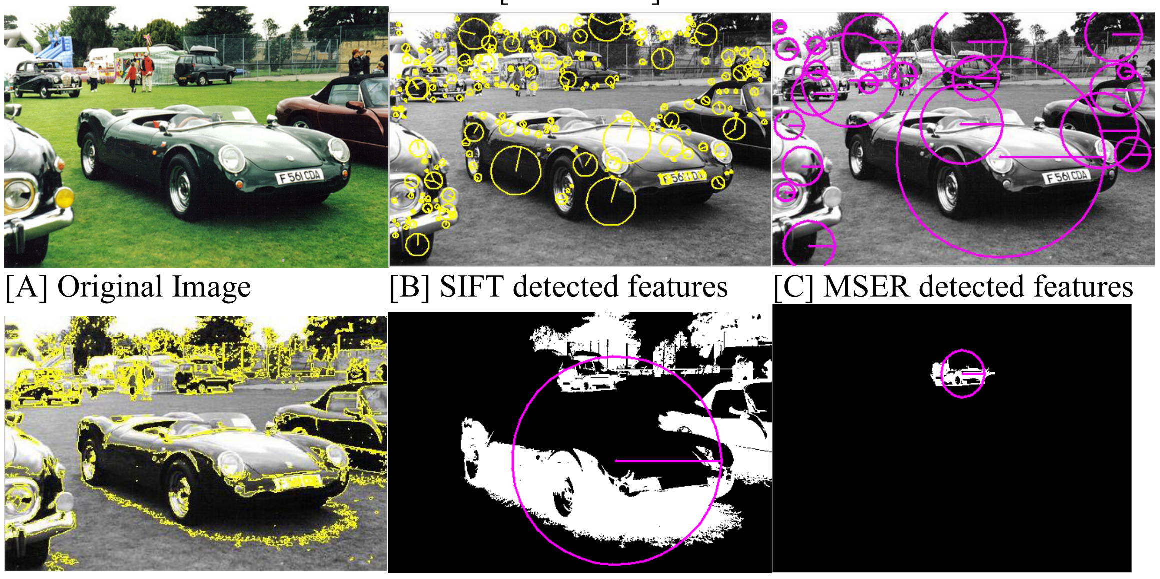
```
    $( x_r , y_r ) \leftarrow$ x,y coordinates of $w_r$
    $( x_l , y_l ) \leftarrow$ x,y coordinates of $w_l$
    $C_x \leftarrow ( x_r + x_l ) / 2$
    $C_y \leftarrow ( y_r + y_l ) / 2$
    $C_r \leftarrow \alpha * \text{sqrt}( (C_x - x_r)^2 + (C_y - y_r)^2 )$
```

Where α is a scaling factor and is empirically selected to be 0.667

2.2 SIFT:

SIFT [3] is a method for detecting and also encoding features in an image. It is invariant to scale, orientation, affine distortion and is partially invariant to illumination changes. It applies a difference of Gaussians function to a series of smoothed and resampled images defining the minima and maxima of the result as points of interest.

[FIGURE-1]



[A] Original Image [B] SIFT detected features [C] MSER detected features

[D] MSER region contours [E] MSER detects all this as a single large feature [F] MSER correctly detects the truck in the background

The SIFT method tends to detect a lot of very small features in the image. Its two parameters the peak threshold and the (non) edge threshold are set to limit detection of lots of very small points as features. Figure-1B shows SIFT based features detected for an image, these are shown as circular patches where the line from the center of the circle shows the feature's orientation.

4 DataSet & Data Biasing:

My dataset consists of 200 images across 5 categories {Cars, Scooters, Faces, Buildings, Yachts}. There are 40 images per category. This is a relatively easy dataset as the categories are quite distinct from each other. Scooters and cars do share some similarity in their features (shape of the wheels, in some cases headlights etc). This is specifically done so as to see whether the algorithm will be able to distinguish between objects that have some percentage of similar features but are distinct overall.

To improve clustering performance I initially take 30 images from each category and for the categories Cars and Scooters I manually segment out the object (car/scooter). For Faces, Buildings and Yachts I don't do any segmentation but pick out images which have less cluttered backgrounds so that the majority of the features detected in the image will actually belong to Buildings, Faces and Yachts. I call this data biasing and run K-means clustering on only these 150 images to cluster the SIFT features into 50 and MSER features into 20 clusters as discussed previously. These cluster models are saved. Figure-2 shows an example of a segmented out image (car) and a non-segmented out image (building). Note the building image also has a few cars and people in it.

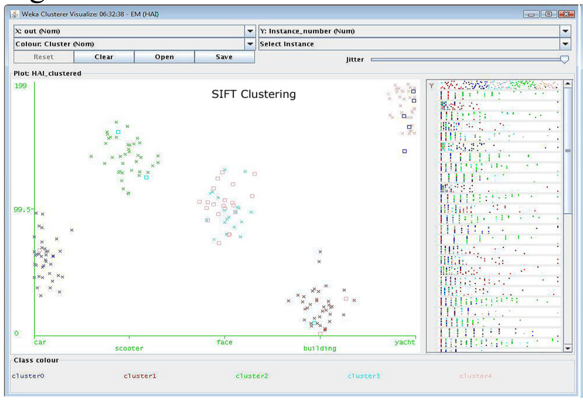
[FIGURE-2]

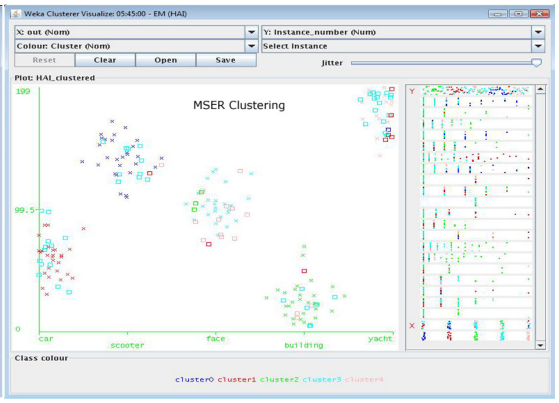
5 Representing Images as Feature Vectors:

For each of the 200 original images (note, these are not the segmented images but rather their original counterparts) SIFT and MSER features are detected. Taking each image individually, its SIFT features are clustered into 50 clusters based on the saved SIFT cluster model above. A 50 dimensional feature vector S is generated for this image. S_i then contains the number of SIFT features of the image that mapped to cluster i. Similarly a 20 dimensional feature vector is also generated for an image based on its detected MSER features and the saved MSER cluster model above. Thus there is one 50 dimension SIFT feature vector and one 20 dimensional MSER feature vector produced for each image.

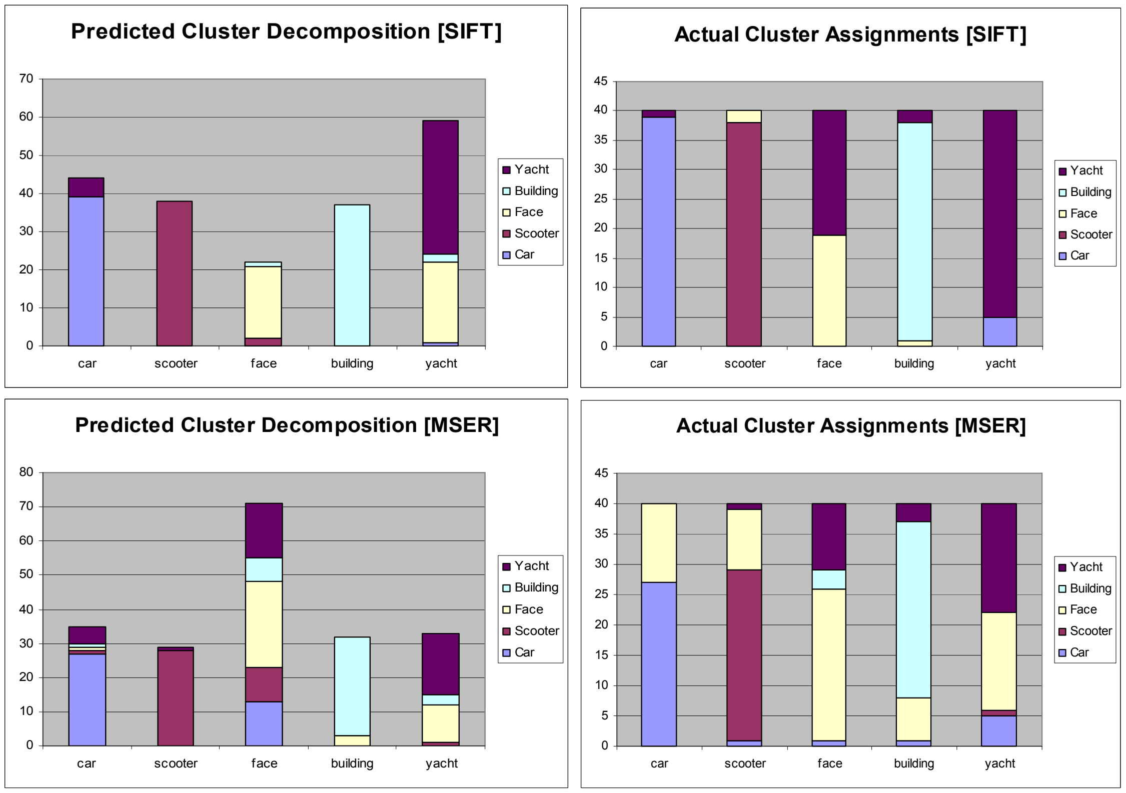
6 Clustering Images:

The EM algorithm is used to cluster images based on only their 50 dimensional SIFT feature vectors. A cluster is given an identity based on the category that has the maximum number of images falling into that cluster. Figure-3A shows the clustering produced. The crosses [x] represent images that were correctly clustered while the squares represent incorrectly clustered images. Similarly the EM algorithm is used to cluster images based on only their 20 dimensional MSER feature vectors. The resulting clustering is shown in Figure-3B.





[FIGURE-] 3A 3B



[FIGURE-4]

Figure-4 is another important figure. Under the "Predicted Cluster Decomposition" heading it shows the number of images that were assigned by EM to each cluster and the color coding shows which categories these assigned images actually belonged to. For example in “Predicted Cluster Decomposition [SIFT]” 22 images were assigned by EM to the Face category. 19 of these images were actually Faces while 2 were Scooters and 1 was a Building. The complementary figures labeled “Actual Cluster Assignments” show for the 40 images belonging to each category how many ended up assigned to each category. For example for “Actual Cluster Assignments [SIFT]” 19 of the Face images were assigned to the Face category but 21 of them were assigned to the Yacht category. Similar results are also shown for MSER.

7 Combining SIFT & MSER:

In the figures above note that SIFT is conservative about assigning Face Images to the Face category. It has high precision for that cluster but low recall as most Face images end up in the Yacht category. MSER behaves differently, the face cluster is the largest and most face images do land into this cluster. However while MSER has higher recall for this cluster it has lower precision with images from Yachts, Cars etc also falling into this cluster. I aim to combine the MSER and SIFT features in an attempt to achieve better results. The simplest approach I tried first was to concatenate the SIFT and MSER feature vectors for an image so that an image is now represented by one 70 dimensional feature vector and then cluster images using EM. This worked really well for Cars and Images which had perfect results. All 40 cars ended up in one cluster with no images from any other category and the same results were also true for Scooters. Faces improved too, however accuracy for buildings especially suffered. The overall accuracy (percentage of images correctly clustered) was 73% compared with 63.5% for MSER only EM clustering and a nicer 84% accuracy for SIFT only EM clustering. Probably simply combining SIFT & MSER feature vectors isn’t a good approach and I’m looking into other schemes and consider this future work.
Figure-5 gives clustering as produced by EM run on SIFT only feature vectors. For shortage of space I did not include all 200 images. The mosaic contains most of the negative examples (incorrectly clustered images) but fewer than actual correctly clustered images.

8 References:

[1] ImageNet, http://www.image-net.org/
[2] J. Matas, O. Chum, M. Urban, and T. Pajdla. *Robust wide baseline stereo from maximally stable extremal regions*. In BMVC, 2002.
[3] Lowe, David G. (1999). *Object recognition from local scale-invariant features*. Proceedings of the International Conference on Computer Vision. **2**. pp. 1150–1157
[4] Per-Erik Forss´en, David G. Lowe: *Shape Descriptors for Maximally Stable Extremal Regions.* ICCV 2007.
[5] G. Csurka, C. Bray, C. Dance, L. Fan: *Visual categorization with bags of keypoints*. In Workshop on Statistical Learning in Computer Vision, ECCV, 2004.
[6] D. Martin, C. Fowlkes, J. Malik. *Learning to detect natural image boundaries using brightness and texture*. In NIPS 2003.
[7] VLFeat, http://www.vlfeat.org/

[FIGURE-5]