TWO PASS K-MEANS ALGORITHM FOR FINDING SIFT CLUSTERS IN AN IMAGE

Nenad Tomašev, Dunja Mladenić
Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
e-mail: nenad.tomasev@ijs.si, dunja.mladenic@ijs.si

ABSTRACT

This paper explores the ways to represent images as bags of SIFT feature clusters. SIFT features themselves are widely used in image analysis because of their properties of scale and rotation invariance. The usual way to group them is to segment the image into regions and then assign features to the corresponding image parts. When images themselves are not available for privacy reasons, this is not possible. We created a hybrid clustering algorithm which offers more flexibility than simple spatial k-means clustering. The algorithm parameters were optimized by a stochastic procedure. The impact of different elements of local representation on final clustering quality is discussed.

1 INTRODUCTION

Due to the constant increase in the amount of multimedia data, automatic image processing is becoming more and more important. For any sort of machine learning algorithms to be employed, images first need to be represented in a way which allows for easy manipulation and calculation. There exists a variety of local or global image features, capturing aspects of objects present in an image. One of the most frequently used local features are SIFT features. They are quite robust and invariant to translation, scaling and rotation [1].

Each SIFT feature has the following structure: (x,y) location in an image where it was found, scale at which it was found, orientation of the direction of the change it describes, as well as the most important part – the feature descriptor. The descriptor is a set of orientation histograms in the vicinity of the keypoint. Usually, a 16x16 pixel neighborhood is sampled, observing 16 histograms of dimensions 4x4, each having 8 bins. This results in the descriptor length of 128 values, so the descriptor is in fact quite high-dimensional. The keypoints themselves are usually found as scale-space extrema of difference of Gaussian image convolutions [2].

SIFT features can be used in various ways. For representing an entire image, it is customary to first quantize the most typical feature vectors and assign each feature on an image to its closest typical vector, thereby representing an image as a histogram of frequencies of these so-called *codebook* vectors. They are obtained as centroids of clusters of features from a stratified sample on a collection of images.

On the other hand, we might want to observe an image as a set of objects in a scene. Each object would then be represented as a group of spatially close SIFT features corresponding to the region in the image where the object is located [1]. In order to achieve this, features have to be grouped in a certain way. The ways of performing the grouping vary according to the specific image task. Sometimes it is beneficial to group features according to the descriptors and to observe an object as a small group of appearance clusters of features describing different textures [3]. More usual way to do is to find regions of interest by first segmenting the image at some level of detail, and then just assigning features to image segments based on their location. A simple K-means clustering based feature location in an image is also an option, though certainly not the most flexible choice.

Sometimes, there are certain privacy issues involved and it is not possible to obtain the images themselves, just some sort of feature representation. This means that there are times when it is not possible to rely on image segmentation for SIFT feature grouping. It is also noteworthy to mention that image segmentation does not necessarily lead to the most convenient image decomposition for purposes of SIFT-based object representation. Even if it does, one must still be careful with threshold parameters for segment merging/splitting, in order to get the desired level of detail.

The goal of this project was to make an algorithm for SIFT feature clustering in a single image, which works just with locally extracted information and which would be flexible enough to meet various preferential requirements for determining feature groups.

2 VIEWS ON SIFT FEATURE CLUSTERING

As previously mentioned, sometimes it is beneficial to group SIFT features before proceeding with the analysis. These feature clusters can then be represented either by their centroids in a simplistic manner, or mapped to corresponding codebook vectors to obtain histograms.

It would be ideal if the so-constructed clusters would match either whole objects or object parts perfectly. Naturally, this is not possible in general case. However, it should be our aim to try and fit those clusters as close as possible to the ideal case.

The first approach that comes to mind is to simply use one of the existing clustering algorithms, for instance k-means, and just define the distance measure between features conveniently. This metric should be defined so as to include information about the difference in keypoint location, feature descriptor and scale. We could also include information about colors in the vicinity of the points by calculating color histograms in some small pixel neighborhoods, if the images are provided. We could form a linear combination of these terms by weighting them according to their relative importance for the image mining task at hand. However, things are not all that simple.

Consider the image shown in Figure 1. There are two cars present in the image, which are exactly the same. If we think of how the SIFT clusters might look like in the end, it is clear that by using the above described approach, there will be big differences between the small and the big car. This is because the spatial distance is weighted the same in both cases and parts of the bigger car are further away from each other than in case of the smaller car. Hence, we will obtain different clusters for those two objects just because they are of different size.

In this particular example, normalizing the distance between each pair of keypoints by some function of the scales at which the respective SIFT features were detected could definitely improve the result. This is also not ideal, because some objects contain features of different scales. If the distribution of scales in the object's features has a big variance, any sort of direct distance normalization based on scale would induce a clustering on an object where features would most likely be broken up into two or more potentially spatially inconsistent groups.

Figure 1: *An image of a car, containing a small copy of itself in the upper left corner.*

One might consider the above example slightly artificial because both cars were the same, but the same argument goes for having similar objects of the same type in several places in an image, not all of them having the same size. In Section 4 we will propose one solution for this problem.

3 RELATION BETWEEN IMAGE SEGMENTS AND SIFT FEATURE CLUSTERS

Image segmentation can be a useful way to obtain SIFT feature clusters after decomposing the original image into subsegments. However, this can also sometimes lead to redundant feature clusters and more complex representations of the original image. Observe the images in Figure 2. Even though it is possible to interpret every single leaf as an individual object for further analysis, it is hardly necessary. It would be quite acceptable having a few of the leaves clustered together, the more the better. They bear nearly identical SIFT features and even have similar colors. However, the shown segmentation has produced 64 different segments in the image. This is much more than what would be required for further processing.

Figure 2: *An image of an autumn tree branches and leaves and the corresponding segmentation obtained by SRM segmentation algorithm.*

Since image segments usually represent either objects or parts of objects, it would be reasonable to assume that the features contained within a single segment should be relatively homogenous with respect to their cluster labels. This doesn't mean that all the features within a single segment must come from a single SIFT feature cluster. In the case of the leaves depicted above, it would also be acceptable to have some slight overlap of clusters within segments. Also, image segments tend to be of quite irregular shapes, sometimes extending along the edge of the entire image. Since distance between keypoints plays a role in the clustering, it is difficult to produce such irregular clusters, especially if the density of keypoints within is low, meaning that the region is not highly textured. Hence, it is also acceptable to have several feature clusters within the segment and vice versa. What is best depends on the image

in question and there is no universal answer to cover all the possible cases.

4 PROPOSED ALGORITHM

There are as many ways to approach the problem as there are clustering algorithms in general. In this particular case, we would like to propose a simple extension to the k-means algorithm, which is usually used to quickly find spatial feature clusters when more time-consuming methods are not applicable. This is the procedure we will be considering in the rest of the paper.

- INITIALIZATION:
  - Perform K-means based on x,y coordinates of the features in order to place initial centroids at spatially reasonable locations
  - Calculate local color histograms for neighborhoods of all keypoints
- LOOP:
  - Calculate local color histograms for all centroids
  - Calculate separately distances in coordinates, scale, color histograms and feature descriptors between all keypoints and all centroids
  - For each keypoint:
    - Rank centroids separately according to distance from the keypoint in either of the metrics (in an ascending way, the closest centroid having rank of 1). Mark these ranks $r_{xy}(c_i)$, $r_c(c_i)$, $r_s(c_i)$, $r_d(c_i)$ for centroid c_i, respectively
    - Calculate distance from the keypoint to the centroid as a linear combination of these ranks: $d(X_i, c_i) = r_{xy}(c_i) + \alpha r_c(c_i) + \beta r_s(c_i) + \gamma r_d(c_i)$
    - Assign keypoint to the closest centroid
  - If there were no reassignments or the error change is below threshold, END
  - Calculate new centroids and go back to start of the loop

The use of ranks instead of explicit distances addresses the issue raised in Section 2 of dealing with similar objects on various scales. This way, there is no need to try explicitly combining individual distances which might behave differently in different situations. Certainly, some of the information is lost this way, but hopefully it's compensated enough by overcoming some shortcomings of the direct approach. It should also be noted that instead of combining continuous variables, here there is only a discrete set of rank combinations for a centroid, which could also be ordered in a more general way.

Since the initial spatial K-means is done in the initialization step, this second K-means pass with the modified rank-based metric is there as sort of cluster refinement.

Parameters allow for setting the algorithm to favor either of the aspects of a keypoint when deciding on cluster assignment. It is clear that the choice of parameters should reflect the context in which the clustering is performed. If it is known that the images in the dataset are not very colorful, then local color histograms definitely won't contribute, they could even worsen the end configuration. The three parameters α, β, γ represent relative importance of color, scale and feature descriptor compared to significance of keypoint (x,y) coordinates when determining feature clusters. Hence, the influence of coordinate distance is assumed to be non-zero, naturally.

5 EXPERIMENTAL SETUP

When dealing with a specific task, it is necessary to set the rank weights well. We mentioned image segmentation as a way of grouping SIFT features, which has its own advantages and disadvantages. Here we check which setup of the rank weights seems to be most consistent with respect to image segmentation. We approach this as a stochastic optimization problem.

A small set of 50 images was handpicked for evaluation from the IMAGENET challenge dataset [4]. The images were chosen from several different categories, namely plants, animals, landscapes, people, cars, gears, etc. We chose those images which had neither too simple a segmentation, nor too complex one. The aim wasn't to try solving the generalized problem, but just to gain some insight into which setup could be expected to produce clusters which follow the structure of image segments.

For segmentation, statistical region merging method was used [6]. SIFT features were extracted by SiftWin application developed by David Lowe [7].

Optimization was performed by simulated annealing [5]. It is a method which gives a trade-off between exploration and exploitation in the search space, controlled by the temperature parameter. The non-negative probability of choosing the worse solution during the search helps escaping local optima and is given by the following formula:

$$p(s_{i+1} = o_i \mid Err(o_i) > Err(s_i)) = e^{(Err(s_i) - Err(o_i))/Td}$$

In the above equation, the current solutions are denoted by *s*, the configurations being currently checked by *o*, *T* is the temperature parameter and *d* the average change caused by mutations to the solutions, which is calculated in the first few steps.

For this particular case, it was assumed that the greatest influence should be exhibited by the coordinate distance, in order to get spatially consistent clusters. Therefore, the bounds were set on all three parameters α, β, γ to fall into the interval [0,1].

Fitness of configurations was estimated as the average within-segment entropy of cluster labels. Minimizing this entropy would lead to more homogenous solutions.

Two different optimization runs were performed, one by setting a fixed number of clusters (K=20) on all images, the other by stating that for every image the number of clusters is equal to number of segments on that image. 300 iterations were performed in each of these runs.

Images in the dataset had a varying number of segments, ranging from 13 to 76, with an average of a little over 34 segments per image. The number of SIFT features on an image ranged from 487 to 3062 and the average was around 1410 features per single image.

6 RESULTS

It turns out there was little difference between the two optimization runs in terms of the final suggested configurations.

	Descriptor rank weight	Color rank weight	Scale rank weight
$n_c = n_{segments}$	0	0.06	0.62
$n_c = 20$	0	0.04	0.63

Table 1: *Weighting schemes that were most aligned with SRM segmentation*

It was clear beforehand that descriptor similarity should not be given much weight if the goal was to achieve segmentation-like clustering of the keypoints, because similar features can be located anywhere in an image, so a high weight of feature similarity would certainly have disrupted the desired grouping. However, it still came as a surprise to see that in this particular case of statistical region merging segmentation, any use of the descriptor information whatsoever led to clustering configurations which were less aligned with the observed segmentations.

It is also intuitive that color should be given some significance, but not too much since there can be many regions in an image having same or similar colors. High scale weight suggests that features of similar scales tend to be grouped together in same segments often enough for it to become important.

Even with the described optimal parameter setting, the produced clustering of SIFT features differs greatly from given segmentation. Average within-segment entropy is roughly 50% of the maximum possible entropy for the used number of clusters. This can be explained by the fact that some image segments do not contain SIFT keypoints at all, so the entropy naturally gets a bit higher when more clusters are used than is actually needed. However, it is not entirely clear how one should go about guessing the proper number of clusters beforehand in the k-means setting. It is of course possible to run clustering for a range of values of k and pick the best one. Performing all these runs would have taken much more time, so we limited the experiments to this less-than-ideal case.

The parameters obtained via the described optimization process do not extend to the general case. The choice of parameters is context-dependent, and sometimes it is even advisable to give greatest weight to feature descriptor.

Both the weighting scheme and the final entropy estimates relate only to comparison with SRM segmentation.

7 CONCLUSION

We explored the possibilities of using a two-pass k-means approach when clustering SIFT features on an image, first performing spatial clustering for initialization and then refining the clustering in the second pass by using as distance a linear combination of centroid ranks obtained with respect to some selected individual distance measures. We performed stochastic optimization on parameters of the algorithm to see which configuration leads to best alignment with image segmentation. Most important was naturally the coordinate distance, followed by scale and color, while the feature descriptors proved to be of no importance in this particular case. Achieving segmentation-like clustering of keypoints is only one possible approach, so different weighting schemes could be chosen for other tasks.

8 ACKNOWLEDGEMENTS

This work was supported by the Slovenian Research Agency and the IST Programme of the EC PASCAL2 (IST-NoE-216886).

References

[1] Z. Zhang and R. Zhang *Multimedia Data Mining*. Chapman & Hall, 2009.

[2] D. Lowe. Object recognition from local scale-invariant features, *Proceedings of the International Conference on Computer Vision*. pp. 1150-1157. 1999.

[3] K. Mikolajczyk, B. Leibe, B. Schiele. Local Features for Object Class Recognition. *In Proc. ICCV'05, Tenth IEEE International Conference on Computer Vision*. Beijing. China. 2005.

[4] IMAGENET. http://www.image-net.org/

[5] Franco Busetti. Simulated Annealing overview. www.geocities.com/francoburseti/saweb.pdf

[6] R. Nock, F. Nielsen. Statistical Region Merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 1452-1458. 2004.

[7] D. Lowe. SIFT Keypoint Detector. http://www.cs.ubc.ca/~lowe/keypoints/