A Keypoint Descriptor Inspired by Retinal Computation

Bongsoo Suh, Sungjoon Choi, Han Lee

Stanford University
{bssuh,sungjoonchoi,hanlee}@stanford.edu

Abstract. The main goal of our project is to implement a keypoint descriptor inspired by the human retinal computation. To simulate the human visual system, we use the sampling patterns of Fast Retina Keypoint (FREAK) proposed in [1]. In FREAK, only two fields are used to compute a descriptor. To improve the discriminative power, we employ more than two fields for descriptor computation and carefully select the fields for computation by using the knowledge of the receptive fields of retinal ganglion cells. After we construct descriptors from a large dataset, we train them to reduce the number of features to 53. We show that our descriptor successfully matches and recognizes various objects including the test images provided in the problem set 3.

1 Introduction

Computers and machines cannot recognize and classify objects as human eyes can. The secret to Humans' ability to recognize different objects lies in how their retina encodes the objects in the scene. FREAK [1], an efficiently implemented keypoint descriptor, was inspired by the retinal computation. It is compact and robust, using a binary descriptor constructed from a sampling pattern that is similar to retinal configuration. Even though it outperforms other descriptors, such as SIFT, we find that understanding of the retinal receptive field is not fully utilized in the model. Receptive fields of the retinal ganglion cells are known to have center-surround spatial receptive fields, measuring correlation of neighboring areas. This is different from FREAK implementation, which they select fields randomly. Therefore, in our project, we propose to improve FREAK by applying topologies similar to ganglion cells' receptive field, labeling which topology works the best for objects with certain features. We trained our

model to find significant topologies with training data sets, and then verified our model with new images to see if biological constraints provide improvements.

2 Related Work

2.1 Review of previous work

Inspired by the human retina, the authors of FREAK paper proposed to use a retinal sampling grid to compute a sequence of one-bit Difference of Gaussians. The retinal sampling grid is circular with inner circles symmetrically distributed, having higher density near the center. We find that FREAK algorithm is based on a simple computation, selecting two circles randomly and then looking for the pairs that give more information. The authors showed that FREAK descriptor out performed SIFT in both accuracy and speed.

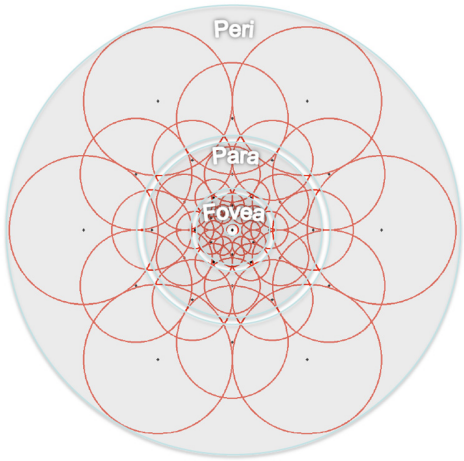

Fig. 1. Sampling Pattern in FREAK implementation. The inner circles, or fields, represents receptive field of a ganglion cell, and pairs of fields are chosen. By comparing the mean intensity, it generates binary string.

2.2 Key ideas and contributions of our work

Authors' idea of using retinal ganglion cells' population density variation was very insightful. However, we considered the case where retinal ganglion cells have locally limited spatial receptive fields, mostly computing nearby scenes.

Thus, we removed the random selection of two Gaussian kernels, which are far apart. This allowed us to understand how spatial structure improves the performance, because we can remove the number of irrelevant features. We constrained the selection of pairs to be within its neighbors. In addition, we not only tested for two circles, but also test for one and up to three neighboring circles.

3 Technical Details

3.1 Overview

We used the same sampling pattern the authors used. However, as described above, we plan to constrain the selection of circles, or topology, to be neighbors and up to three circles. For each image patch (or keypoint), we stored all possible summation and subtraction combinations for each topology into an N-dimensional binary string array, where N is the number of possible topologies. Thus, for all the test image patches (or keypoints), M, we produce M×N matrix of 0s and 1s as shown in Figure 4. The binary string array for each image is found by weighted summation of Gaussian filtered areas. The area integral is similar to the FREAK paper, computing average intensity, but we use Gaussian weighting whereas FREAK use a box filter. Using this matrix, we find the most significant topologies in distinguishing the different image patches by using machine learning, which will be described in Section 3.3.

OpenCV provides a nice framework for detectors, descriptors, and matchers to work together in unison, including the original implementation of FREAK. We initially planned to modify a part of FREAK code, but we figured out that it would be much simpler for ourselves to write our own code that computes the descriptor. Most of our work was written in MATLAB code.

3.2 Topology construction

We construct binary descriptors by thresholding the sum of plus or minus intensities among the set of receptive fields. The intensities are smoothed by corresponding Gaussian kernel. Unlike the original FREAK, we use multiple receptive fields, not just pairs. Thus, the resulting descriptors depends on the mechanism of receptive field selection.

How do we select multiple receptive fields among 43 receptive fields? There are more than 10,000 combinations if we randomly choose three fields. Examining

all combinations is not feasible and is not relevant to the configuration of pixels on a 2D image nor the human visual system. Computing differences between adjacent fields is desirable to get maximum discriminative power. We first modeled the position of receptive fields as vertices in a planar graph. The vertices are located at the center of receptive fields. The edges in the graph describe that if two nodes are connected by an edge, they are close enough and can be used to compute difference.

Next question is how to construct edges from the set of vertices. Our approach is to subdivide the planar graph into triangles. The planar graph can be triangulated by adding random edges until no edges can be inserted without intersecting other edges. However, the initial triangulation does not represent geometrically correct adjacencies. We performed *Delaunay Triangulation* to get a nice set of triangles. The result of triangulation is depicted in Figure 2.

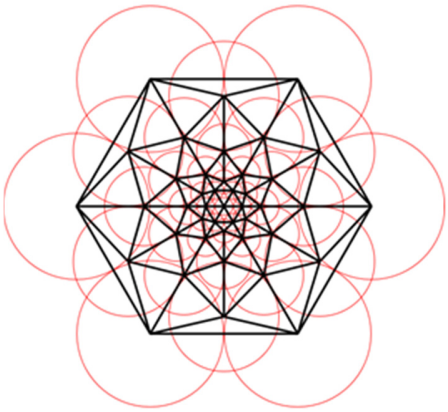

Fig. 2. Delaunay Triangulation. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation.

Next, we generated a set of receptive fields as following. The example cases are shown in Figure 3.

One receptors. It is trivial case. Each vertex in the graph is selected once.

Two receptors. Select each edge and group the vertices on both ends.

Three or more receptors. Perform depth-first search. Visit a vertex only when one of its neighbors are visited before.

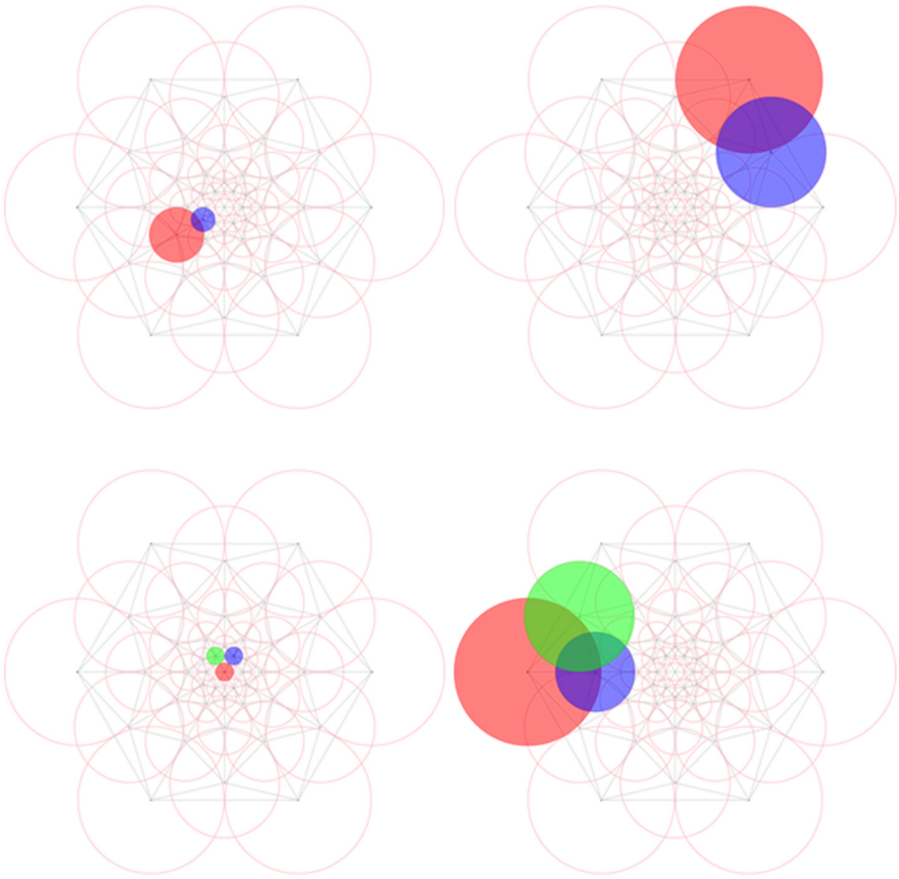

Fig. 3. Examples of produced topologies with two fields (top) and three fields (bottom). The radii of circles indicate the sizes of Gaussian kernel.

3.3 Feature matrix construction and Machine Learning

Intensity between the fields are compared, similar to FREAK, but we constrained it to be the neighboring fields. We compute the feature values by comparing the mean intensity. For the case of pairs, it's just the difference between two. For the case of three circles, we consider all combinations of different choice of plus and minus signs.

$$T(P_a) = \begin{cases} 1 & \text{if } (I(P_a^{r_1}) - I(P_a^{r_2}) > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$T(P_a) = \begin{cases} 1 & \text{if, } (\pm I(P_a^{r_1}) \pm I(P_a^{r_2}) \pm I(P_a^{r_3})) > 0 \\ 0 & \text{otherwise} \end{cases}$$

We computed the feature matrix by computing T for each pair or triple. Each field is computed in gray and RGB, making four features for each field. There are total 1,416 features, and we used 53 features that has the highest entropy where the average value is close to 0.5.

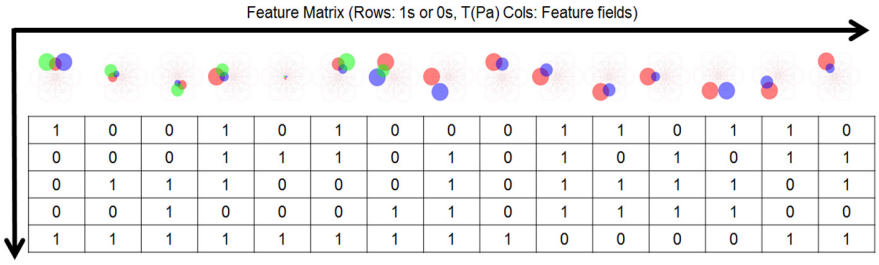

Fig. 4. Produced feature matrix. Each row indicates a binary descriptor for a test image where the values are T for each topology.

While considering different topologies, some had more significant features than others. For example, one topology (selected from the top row) has evenly distributed values of 0s and 1s for different image patches (keypoints; corresponding row values of that column); the corresponding values are computed from different image patches (keypoints) for that topology. In this case, average is close to 0.5. On the other hand, when the feature values are 0 for most of the time or 1, such topologies is not as significant as other topologies. In our computations, we used only these significant topologies, selecting 53 features, and we determined the best topologies from our training data set based on the aforementioned criteria.

3.4 Training

We used the same training data as the original FREAK algorithm did [3]. For each image, we found keypoints using the SIFT detector. Then, we created a

matrix, in which the number of rows is the same as the number of keypoints (or image patches we computed) and the number of columns is the same as the number of topologies. For each keypoint, we computed the scores for different topologies, and write the results to a vector. We performed this for every key point in an image, and as the last step, we averaged the scores for the same topology in all keypoints (average of a column vector) and put each average value in a topology score vector. We repeated the same procedure for all images in the training data set, and averaged the topology score vectors to create the final topology score vector. Lastly, we sorted the topologies by how close the score is to 0.5. We started from 128 features (topologies) close to 0.5, and reduced so that we can improve the speed.

Depending on the number of topologies, the matching accuracy varied. When we included more topologies, only few keypoints were matched, but with very high accuracy. As we decreased the number of topologies, more keypoints were matched, with lower accuracy. We converged to using 53 features (topologies), finding that this number gave both good accuracy and good speed. Importantly, using this number, our model showed good performance of object recognition.

4 Experiments

Figure 5 shows our approach works well on the simplest case matching all the keypoints on the same image.

Fig. 5. Preliminary result. Both left and right images are same. We experimented this for the purpose of verification.

Figure 6 shows the comparison between our approach and SIFT. Our approach is not always perfect. We think we need to tune parameters further so that it can be comparable to SIFT.

Fig. 6. Correspondences of our approach (left) and SIFT (right). Our approach has a few outliers. We believe this can be reduced by tuning parameters.

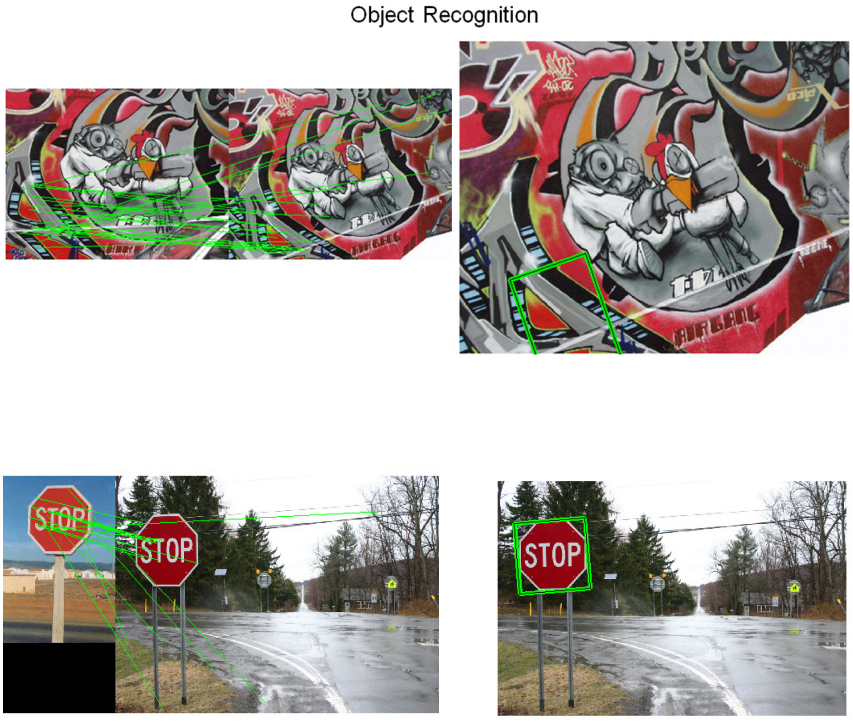

Fig. 7. Object Recognition. The given image patches are successfully recognized by using our descriptor, which only uses 53 features.

5 Conclusions

Considering the number of features we used, we achieved relatively good performance. We used only 53 features, which was significantly less than both SIFT and the original FREAK. Since there are less features, it takes less time to compute descriptors. Thus, in the case when fast approximation of object detection and recognition is needed, our approach can be used with less features achieving almost online computation.

For some test images, our model worked well in recognition. However, it performed poorly when the object was transformed significantly. Since we are just considering the simplest case which is a linear averaging of the intensities of the

neighboring fields, the limitation in performance seems to be coming from the simple approximation of biophysical computation.

In this project, we used the neighboring fields of retinal sampling pattern inspired by human visual properties. Biology, however, is more complicated. Our work might be improved by including higher order features from retinal computational properties, having more complex and nonlinear features.

References

1. A. Alahi, R. Ortiz, and P. Vandergheynst: FREAK: Fast Retina Keypoint. In: CVPR. (2012)
2. FREAK software: http://lts2www.epfl.ch/software
3. Oxford Dataset: http://www.robots.ox.ac.uk/~vgg/research/affine