EE569 – HW4

Mozhdeh Rouhsedaghat

2726554211 rouhseda@usc.edu

Problem 1

Abstract and Motivation:

Texture classification and segmentation has always been an important tool to understand images. Before CNN, laws filter was a relatively successful approach for this goal. There are also other methods for this approach which all of them try to extract specific and important features of different types of patterns. With classification of the extracted output features we can differentiate between different patterns and separate them.

Approach and Procedures:

Texture Classification

We have used 25 2D laws filter to extract features of textures. For constructing these filters, we have computed the tensor product of the below 1D filter:

Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5(Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

As you see each of these filters can detect one of the important texture features (Edge, Wave, Ripple, …). We calculate their correlation with each 5*5 window in the image as the filter output for the center pixel of the window. Then we compute the average energy of each filter. It is the sum of the square of the filter output of each pixel and we use it as a feature for the given input texture. The next step is to normalize the resulting 25 feature vector of input texture. Then we can classify the obtained feature vectors into 4 cluster using k-means clustering algorithm.

For applying K-means we first initialize a random center for each cluster and find the nearest center to each vector to assign the vector to that cluster. Then we compute the average of all the feature vectors which are assigned to a cluster as the new center of that cluster. We continue this process till the location of the new centers stop changing or change smaller than a threshold. At this point all the vectors are assigned to an appropriate cluster.

Texture Segmentation

First, we calculate the correlation of each of the 25 filters with each 5*5 window in the image as the filter output for the center pixel of the window. Then we compute the average energy of the window with a specific size surrounding each pixel of each filter. So, we have a 25-dimension feature vector for each pixel and we can use k-means to classify pixels into 7 groups. But before that we have to normalize the feature vectors, we can also use the energy of $L5^{T}l5$ to normalize other features at each pixel.

Improving Texture Segmentation

There are several methods for improving texture segmentation results. One of them is to use PCA to reduce the dimension of 25d to a lower dimension. This operation can remove unwanted noise in the texture and leads to smoother counters. Another method is to enhance the boundary by giving only the boundary region to k-means algorithm to classify them into two groups, as we are sure there are only two clusters at that regions. Some other post processing techniques like using morphological processes to remove small unwanted regions can also be used if needed. One other operation which leads to a better segmentation result is to use only the 15 filters mentioned below instead of 25. As 10 of 25 filters work like other 10 filters and using all these 20 filters only leads to an unwanted emphasis to their detected features.

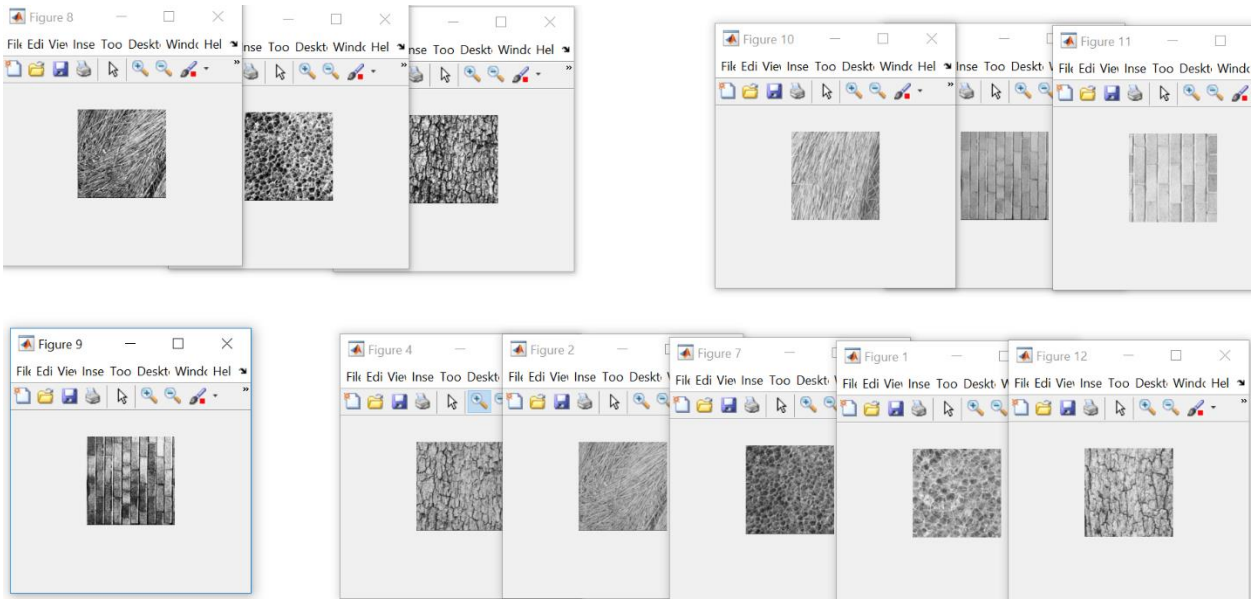
L5L5	L5E5/E5L5	E5S5/S5E5
E5E5	L5S5/S5L5	E5W5/W5E5
S5S5	L5W5/W5L5	E5R5/R5E5
W5W5	L5R5/R5L5	S5W5/W5S5
R5R5	W5R5/R5W5	S5R5/R5S5

Results:

1-a)

here is labels for 25d feature vectors, you can see the clusters of images below:

2 2 4 2 3 3 2 3 1 4 4 2



here is labels for 3d feature vectors:

1 1 1 1 3 3 1 2 4 1 1 1

While the correct classification result (by eye) is:

1 2 3 4 1 4 1 2 3 2 3 4

As you see this result is highly dependent on illumination feature of each input image. For example, in the labels for 25d feature vectors, class label 4 clusters bright images and class label 3 clusters high contrast input images. So, laws filter is highly dependent of the image illumination for the clustering operation and if we don't do contrast adjustment for input images its clustering result for pattern detection is not reliable.

After illumination adjustment the resulting labels are as follows:

labels for 25d feature vectors:

1 4 2 3 1 3 1 4 2 3 2 3

labels for 3d feature vectors:

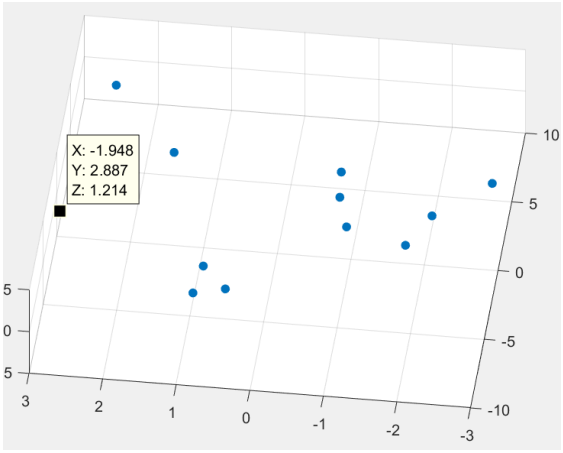
4 1 2 3 4 3 4 1 2 2 2 3

This result is much more satisfactory. All the clustering results are correct expect for input image 10.

To find the strongest and weakest filter, for each feature, we calculate the intra-distance to inter-distance ratio for all pairs of points and sum the ratios up. The feature with the larger ratio is the one with a weaker discriminant power and vice versa. I found that $R5^{T}R5$ has the strongest discriminant power with intra-distance to inter-distance of 0.0163 and $L5^{T}R5$ has the weakest discriminant power with intra-distance to inter-distance of 0.1494.

You can see the plot of the reduced 3-D feature vector in the feature space below:

As you can see different texture patterns can be detected in this plot.



Dimension reduction didn't change the clustering results by keeping the most important data. Even in some cases it may improve the clustering result by eliminating small noises which may affect the clustering. Besides it speeds up the k-means convergence, as less computation is needed to converge with less feature dimension.

Bonus: I have implemented K-means algorithm in my code.

1-b)

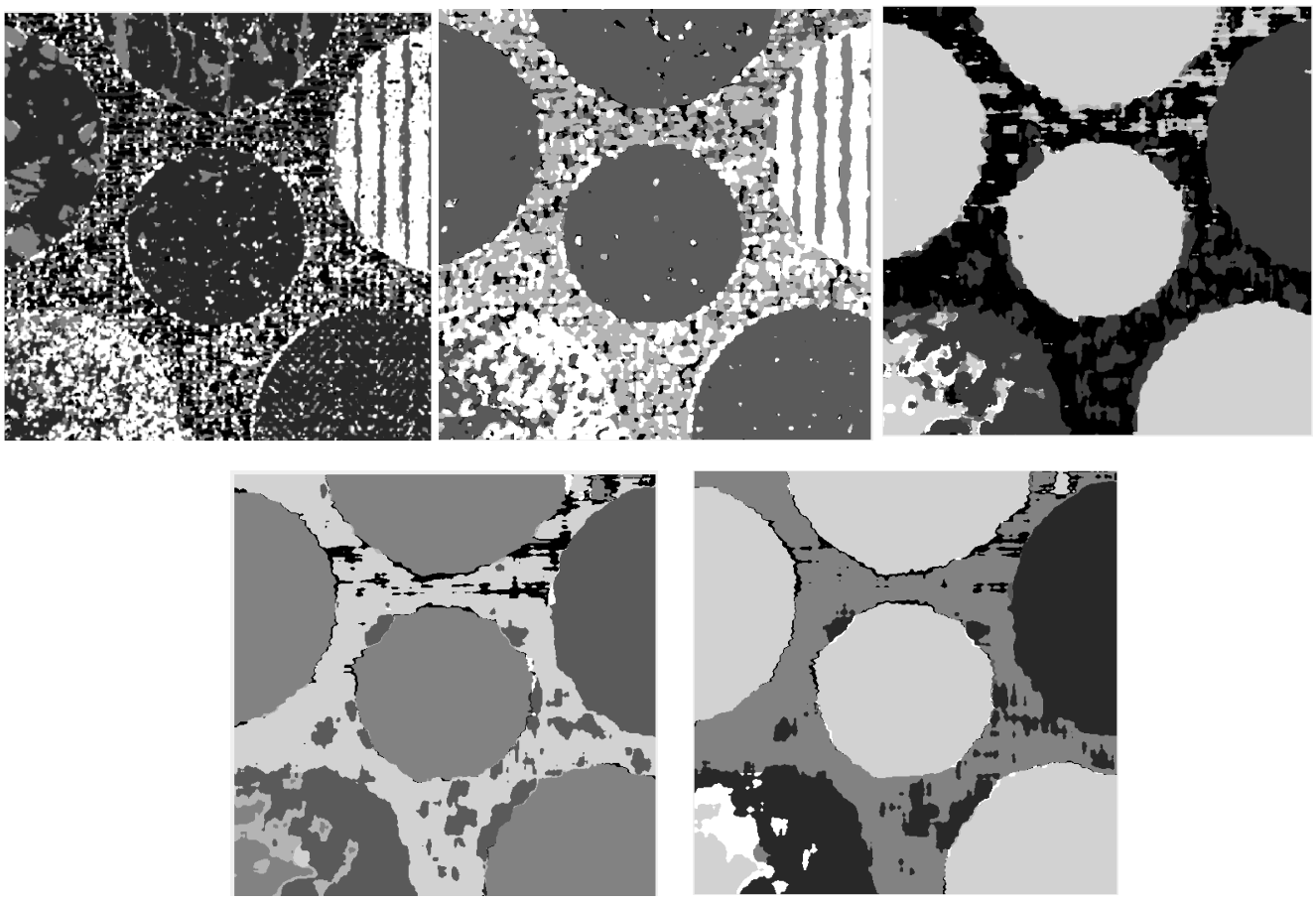


Fig3: Texture segmentation result of Com.raw with window size of 3 (upper left), 7 (upper middle), 15 (upper right), 23 (lower left), and 31 (lower right)

As you see by increasing window size to 31 the output segmentation result gets better and smoother. I couldn't try bigger window size because of Hard Ware limitation as increasing window size increases computation time significantly. (Computing the output with W=31 takes more than one hour and a half with my laptop.)

Increasing window size creates smoother energy feature vector so regional changes won't affect the final clustering result of a texture's pixels and let them fall into the same texture cluster.

1-c)

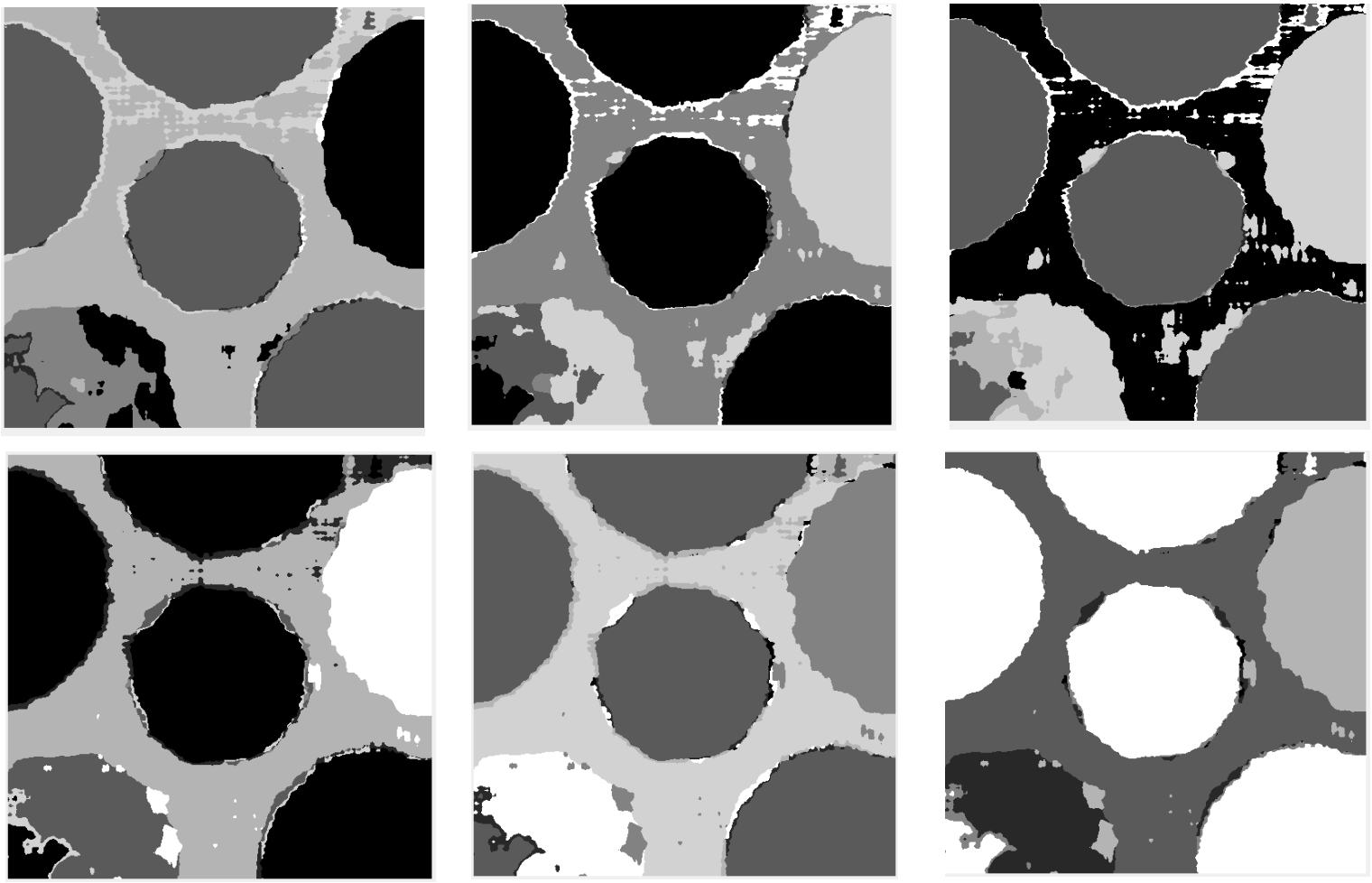


Fig4: Texture segmentation result of Com.raw with W=31, 3-D PCA of 25-D filters (upper left), W=31, 5-D PCA of 25-D filters (upper middle), W=31, 10-D PCA of 25-D filters (upper right), W=31, 3-D PCA of 14-D filters (lower left), W=31, 3-D PCA of 15-D filters (lower middle), and final improved result with W=31, 3-D PCA of 14-D filters with merging 2 classes as background texture (lower right)

As you see laws filter failed to distinguish between 4 different textures patterns and mapped them to a single cluster.

I have used PCA to reduce the dimension from 25 to 3, 5, and 10. It seems reducing the dimension to 3 leads to a better result. Overall PCA improves the clustering result by eliminating small noises which may affect the clustering. Besides it speeds up the k-means convergence, as less computation is needed to converge with less feature dimension. Then I used the 15 filters I explained in the procedure section instead of 25. (Fig4, lower middle image). As you see it improved the segmentation result and gave a smoother result, it also reached a much better segmentation result for the lower left part pf the image (smoother and distinct from other textures). In another try, I removed $L5^TL5$ from the mentioned 15 features to see the effect of $L5^TL5$ (lower left). As you see $L5^TL5$ has a weak discriminant power and removing it doesn't change the result considerably. To reach a netter result I noticed the background region is clustered into two segments, so I merged them as one cluster and the final result (lower right) seems satisfactory.

Discussion

Laws filters is not a very strong means for texture classification and segmentation. In many cases it fails to distinguish between different textures, as it uniformly divides the frequency region into several frequency bands (In case of 5*5 filters 25 frequency bands) and works fine if the highest energy region of each texture falls in one of these regions, while in many cases the highest energy of several textures falls in the same frequency band. In this case Laws filter often fails to distinguish between textures.

Problem 2

Abstract and Motivation:

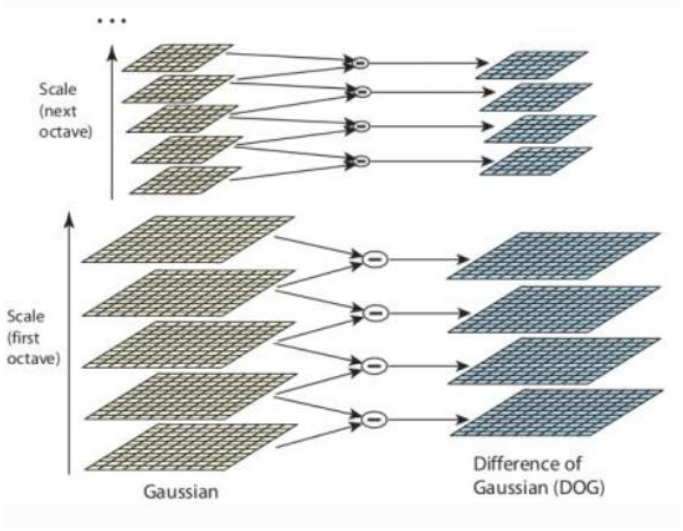
Key points of an image are important point which represent the image characteristics and we can distinguish the image using them. We can use them for image classification of image matching. There are several approaches for extracting key points of images. One of them is SIFT which is scale and rotation invariant and partially invariant to changes in illumination and 3d camera view point.

Approach and Procedures:

Scale Invariant Feature Transform (SIFT)

Key Point Detection:
We use gaussian filter with different sigma, as gaussian kernel with low σ gives high output for small corner and high σ gives high output for larger corner. Then we do the same for several levels of down sampled image and build a gaussian pyramid. Then we compute the difference of the output and to find the DoG. We use it instead of LoG as it is costly. The potential key point is

an extremum within the 8-connectivity neighbor areas and the neighboring 9 points of two neighbor images in a 3-D DoG space.



Key Point Localization:

We must verify whether the selected point are actual key points or not. We use several methods to eliminate low-contrast key points. One of them is to compare the intensity of the extrema with a threshold called Contrast Threshold and discard weak extrema points. DoG has a higher response for edges, so another step is to remove key points detected on edges. After these steps only more reliable and valid key points are left.

Key Point Orientation:

An orientation is assigned to each key point to achieve invariance to image rotation. Depending on the scale, the gradient magnitude and orientation in a window of 3*scale is calculated. We represent the gradient direction by dividing 360 degree into 36 bins and stating the grad direction by the bin it belongs to. Then we weight each grad orientation by grad magnitude and gaussian weighted with 3*scale and draw their histogram. The orientation with largest assigned value is selected as the main orientation and for those with a value greater than 80% of the peak value we create key points with the same location and scale but different direction.

Key Point Description:

For the 16×16 neighborhood of each key point, we divide it into sixteen 4×4 sub-blocks, then rotate its coordinate to fit the direction of the orientation of the key point and calculate 8 bin orientation histogram of each sub-block and use all values of 128 bins as the descriptor

Image Matching

With a Brute Force algorithm we can find the match of a key point by finding a key point with minimal l2 distance.

Bag of Words

After finding key point description of training images we can use K-means algorithm to cluster all of them and obtain K centroids and the distribution of each class of inputs. Then we can cluster a test data by clustering its key point description using he obtained centroids to reach its distribution.

Results:

2-a)

I. It is invariant to scale, rotation, and translation. (Other aspects of robustness: It is also partially invariant to illumination change and affine or 3d projection)

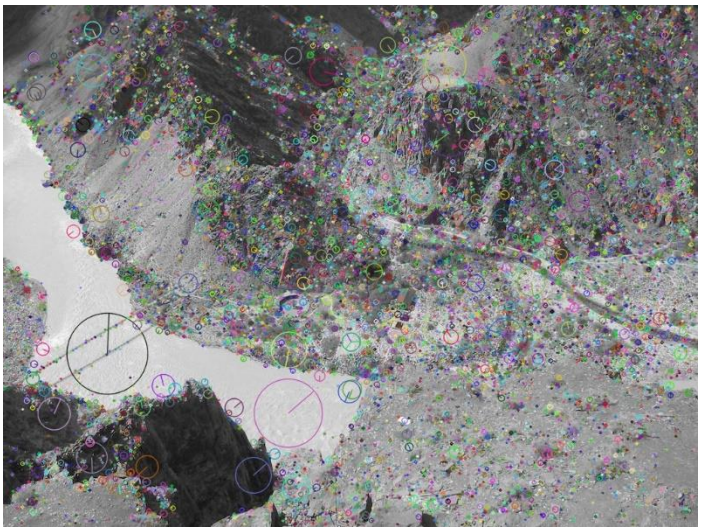
II. It uses multi-scale filtering; means it compute the gaussian filter with different sigma for each octave. Using different sigmas helps us to detect key points at any scale; as low sigma gives high value for small centers and high sigma gives high values for large corners. As we locate the potential key point by finding the local extrema of DoG output which is not dependent to orientation or location of it. We also assign a canonical orientation to each key point and specify its descriptor relative to this orientation, so changing the rotation won't change the descriptor of that key point.

III. Illumination robustness achieved by setting a threshold for the gradient magnitudes and discard them if their values are less than 0.9 times the maximum possible gradient value. Canonical orientation is also determined by the peak in a histogram of local image gradient orientation. These values are relative so by changing illumination they also change relatively and won't change the resulting key point features.

IV. LoG is costly and needs more time to be computed while DoG is fast and is a good approximation of LoG.

V. 8*4*4+8*2*2=160 elements

2-b)



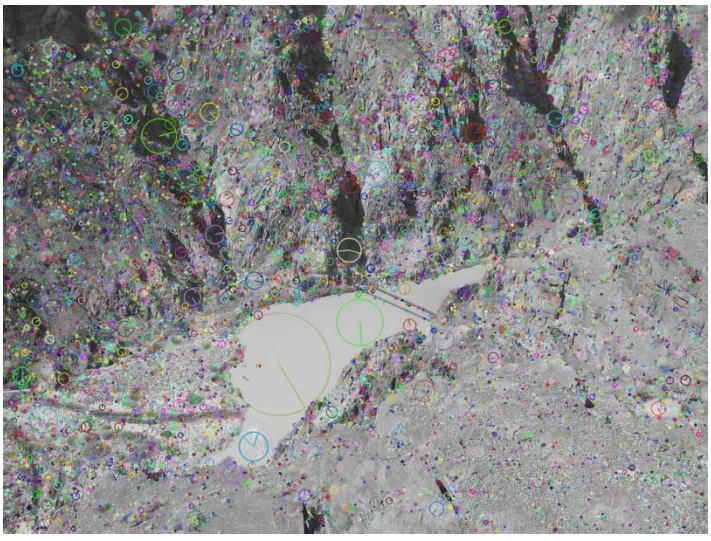


Fig 6: Key points of the river1.raw (left) and river2.raw(right)

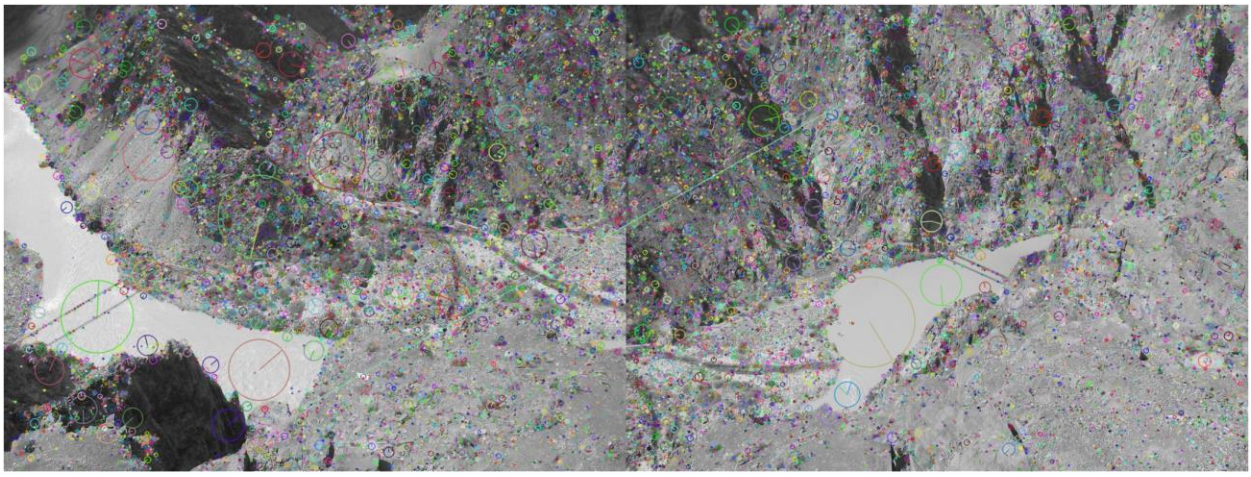


Fig 7: The resulting match

Fist I searched the key points of river1 image to find the key-point with largest l2 norm and then searched the key points of river2 image to find the nearest key point to it. (vector with least l2 norm of the difference of their descriptors). Then I plotted the resulting match.

The orientation of matching key points is different, but their descriptors are constructed with respect to the orientation of the key point, so their descriptors are very similar.

With this approach, the resulting match is often false, as we use a brute force algorithm and will find a match for each key point. But the proper match may not exist for a key point. To address this issue, we would better set a threshold and if the l2 norm of the difference of a key point descriptor with its match is larger than the threshold we reject this pair of key points to only keep more reliable matches.

2c)

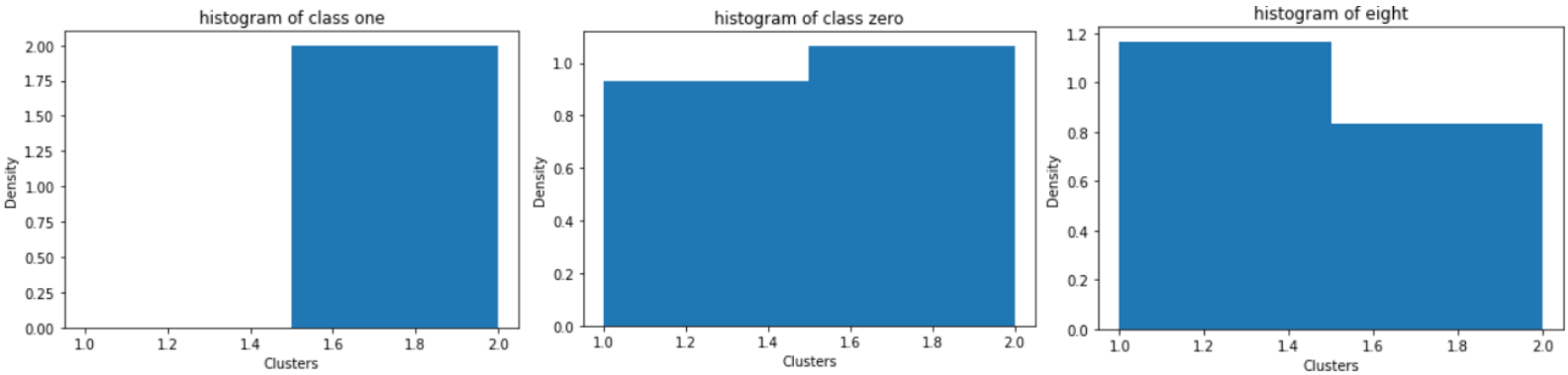


Fig 8: Density histogram of class one, class zero and input image eight

When we cluster the key points of each 0 or 1 input image, the resulting density diagram is very similar to its corresponding class histogram in most cases. (When the number of detected key points is very small (or 0), it is more likely for the classifier to produces false results.)

The resulting histogram of 8 is more similar to class zero; it is because of their similar curves inside their visual structure; SIFT detects these regions and they look like . (8 is like to zeros; one on top of another).

Similarity with class one: ~ 0+0.8/2=0.4

Similarity with class zero: ~0.9/1.2+0.8/1.1=1.48

Discussion

With a brute force approach for match detection for SIDT features, the resulting match is often false, as the algorithm will always find a match for each key point. But the proper match may not exist for a key point. To address this issue, we would better set a threshold and if the l2 norm of the difference of a key point descriptor with its match is larger than the threshold we reject this pair of key points to only keep more reliable matches.

BoW has two main weaknesses: First, visual words are not discriminant features, BoW cannot differentiate between key features of a class and useless ones. It takes all of what is found into account with the same degree of importance. Second, the spatial relationship between visual words is lost. If we detach different parts of a bicycle and give it to BoW it still classifies it as a bicycle as it doesn't notice the relation of words with each other but only their existence.