EE569 HW4

Problem 1:

Motivation:

Texture analysis is a typical type of work of image processing. For this problem, we need to cluster a number of different texture images into 4 group. For different type of texture, they have different image features, and these features can be extracted with the help of different Laws kernels. We can let the image convolute with the laws filter and get the high and low frequency features of each image. Then we can use these feature to cluster textures.

A Texture Classification

Approach:

- Extend the image boundary
- Get the 25 Laws filters created by the tensor product of the 1D Kernels.
  Sample filter $E5^{T}L5$:

$$\begin{bmatrix}-1\\-2\\0\\2\\1\end{bmatrix}\times\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix}=\begin{bmatrix}-1 & -4 & -6 & -4 & -1\\-2 & -8 & -12 & -8 & -1\\0 & 0 & 0 & 0 & 0\\2 & 8 & 12 & 8 & 2\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

- Remove the illuminance effect
  Instead of subtracting the mean value of each image, I use the histogram-equalization method to make the 12 texture images to have the same pixel value distribution.
- (step 4) For each image, we use 25 5×filters to get 25 filtered image mats. Each of these mats represents one feature of the original image.
- (step 5)Calculate the energy of the filtered image, using the following equation:

$$E=[\sum_{each\_pixel}(\text{filtered pixel value})^2]/N,\ N \text{ is the total pixel numbers}$$

- Repeat the step 4 & 5 for 12 images to get a 12*25 Feature Matrix.
- Because some of the feature dimension have very large feature values and some of them have very small feature values, so we should do the feature normalization. For each feature dimension, we should subtract this dimension's mean value and then divide by this dimension's standard deviation.
- Some of the feature dimension have strong discriminant power, some do not, so we can reduce the 25D to a lower dimension (3D). With the help of OpenCV.
- Finally, we use the K-means algorithm to cluster the images.

Results:

The true result should be:

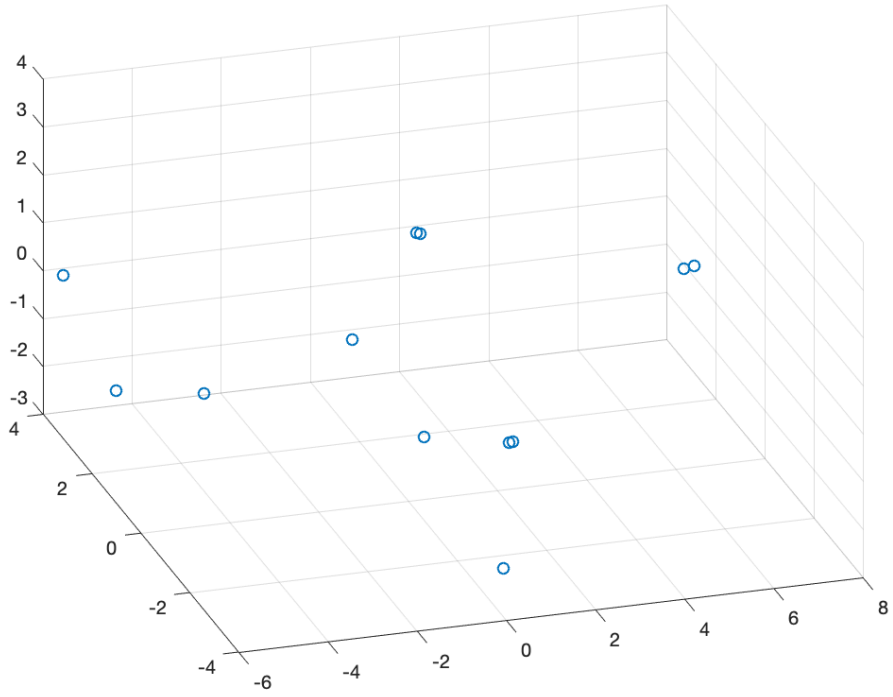
	Image #
Bark(1)	4, 6, 12
Straw(2)	2, 8, 10
Brick(3)	3, 9 , 11
Bubbles(4)	1, 5, 7

Test#	From one dimension # to another			Clusters #		Remove luminance effect method				
	25D	15D	3D	K=4	K=5	mean	Histogram-equalization	Result (Red wrong)	accuracy	**discuss**
1	From		to		√		√	[4; 1; 0;	If group the	Only the "texture10.raw" is

								2; 4; 2; 4; 1; 0; 3; 0; 2]	group 1 & group 2 together, the accuracy is 100%	separate from the "straw" group. Other 3 groups are all correct.
2	From		to	√		√		[0, 0, 0, 0, 1, 1, 0, 2, 3, 0, 0, 0]	41%	Most of the images are clustered in the first group.
3		from	to	√			√	[3; 1; 0; 2; 3; 2; 3; 1; 0; 2; 0; 3]	83.3%	Compare to test #6, the accuracy is higher, because if we reduce the 25D to 15D, when we apply the PCA process, the 15D will lead to a better (less error) result due to the lower dimension.
4	From to				√		√	2; 1; 0; 4; 2; 4; 2; 1; 0; 3; 0; 4]	100%	
5	From to			√			√	[2; 1; 0; 2; 2; 2; 2; 1; 0; 3; 0; 2]	66.7%	
6	From		to	√			√	[2; 1; 0; 2; 2; 2; 2; 1; 0; 3; 0; 2]	66.7%	
7	Form to			√		√		[0, 0, 0, 0, 1, 1, 0, 2, 3, 0, 0, 0]	41%	Same with test #2

The strongest discriminant power is $S5^T*S5$
The weakest is $L5^T*L5$: kernel all positive

The plot of the PCA result:



The PCA procedure can largely reduce the calculation load, and then get a faster calculation speed. But as we can see from the test #5 & #6(with pca) or #2(with pca) & #7, the accuracy of the output with PCA doesn't have a large increase. This phenomenon may due to the small number of the test images.

B Texture Segmentation

The main idea is similar to part a, we still use the 25 Laws filters.

- We apply 25 Laws filters to the image "comb.raw", and get 25 filtered image. (should do padding first)
- For each of the 25 filtered image, calculate each pixel's energy within the neighbor area(with a window size which we can set different values)
- Because the output of the Filter $L5^{T}*L5$ is not a useful feature, so we should get rid of it. We can use this feature to normal all other feature.
- At last, we still use k-means to denote the segmented regions.

Result

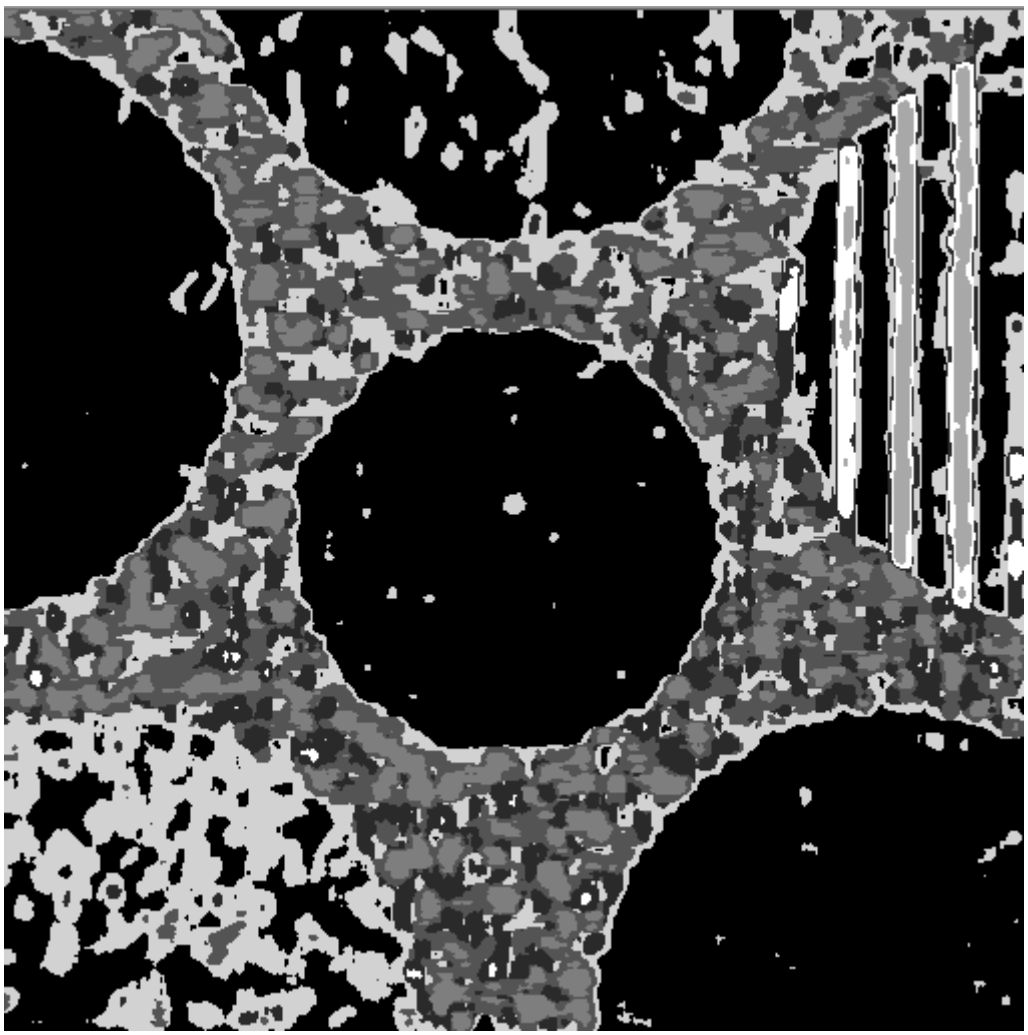


Figure1-3: Energy window size = 9

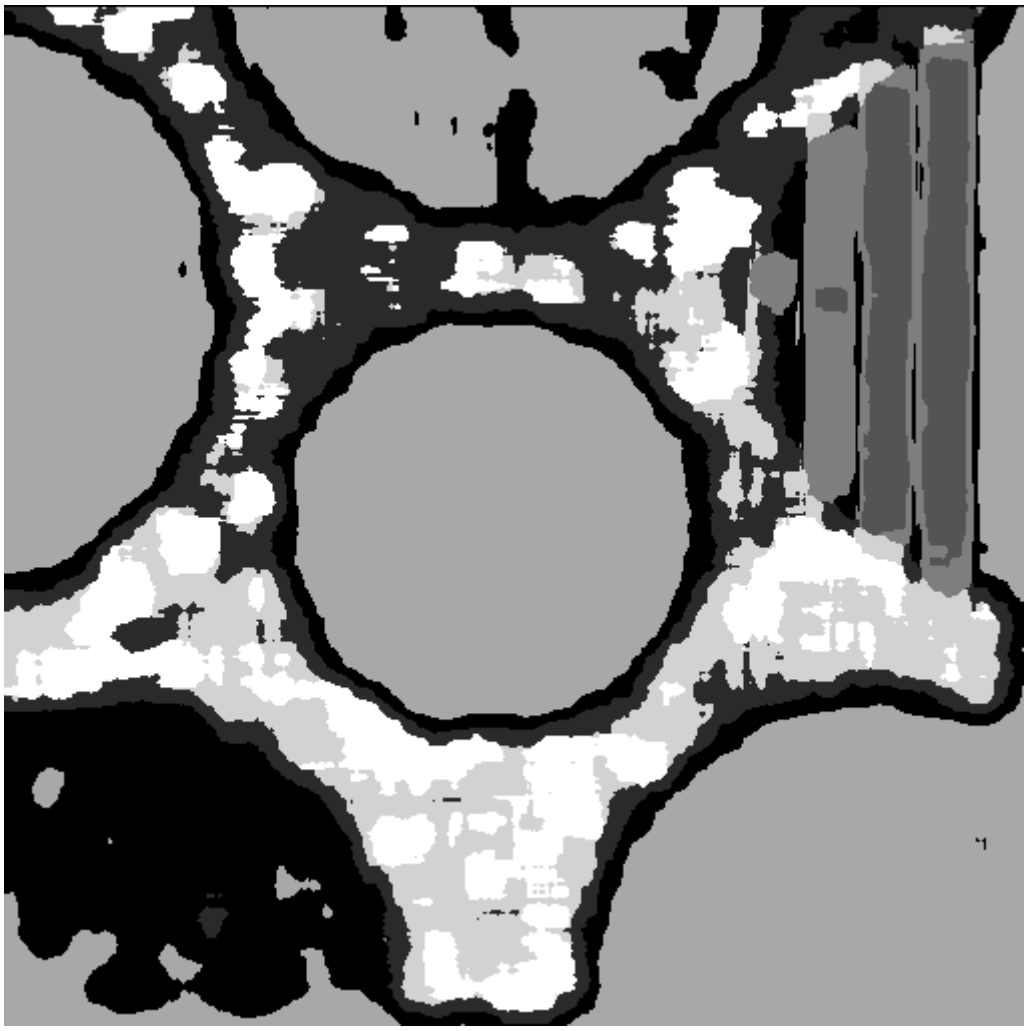


Figure1-4: Energy window size = 25

There are 4 circle regions denoted as the same color, that is because the algorithm denoted several clusters to the texture edge part. The edge region's energy is much higher than the texture part, so the 4 same color circle region's energy difference is much lower than the difference between the edge part and texture part.

With a large energy window, the continuous texture parts have less spotted effect, but the edge part still take too much cluster center points.

C Advanced Texture Segmentation Techniques

- In order to improve the segmentation, we can focus only on the 4 same color circle region (stage two segmentation), and do the k-means (k=4) again.
- After the stage two segmentation, for each pixel, if the pixel value is in the minority part (we can set a threshold) the neighborhood pixel value, we change the pixel value to the majority one.

And the output is:

Problem 2:Image Feature Extractor

SIFT is used to extract the image feature, with the help of GoD. This method transform the image into a large collection of the local feature vectors, and is invariant to image translation, scaling and rotation, and partially invariant to illumination changes and 3D projection.

A SIFT(use OpenCV)

i. Translation, scaling, rotation

ii. Translation: this is the easiest and basic feature to have strong robust output, the accuracy to translation test should be close to 100%, since all the descriptor features should almost be the same.

Scaling: The scale-invariant features are efficiently identified by using a staged filtering approach. The first

stage identifies key locations in scale space by looking for locations that are maxima or minima of a difference-of-Gaussian function. Each point is used to generate a feature vector that describes the local image region sampled relative to its scale-space coordinate frame. The features achieve partial invariance to local variations, such as affine or 3D projections, by blurring image gradient locations.

Rotation: Each key location is assigned a canonical orientation so that the image descriptors are invariant to rotation.

iii. Robustness to illumination change is enhanced by thresholding the gradient magnitudes at a value of 0.1 times the maximum possible gradient value. This reduces the effect of a change in illumination direction for a surface with 3D relief, as an illumination change may result in large changes to gradient magnitude but is likely to have less influence on gradient orientation.

iv. **LoG:**

Costly (need more multiplication and add operarion);

very noise sensitive;

DoG:

DoG is an approximation of LoG;

Both are used in blob detection, and both perform essentially as band-pass filters.

With the separability and cascadability of the DoG, we can achieve more efficient implementation.

v. 160

B Image Matching

The main procedure is following:

- For this problem, we have to transform the image data form RGB space to BGR space first, since the OpenCV use the BGR space.
- Set a SIFT detector, and detect the key points in image 1 and image 2
- And then, for each key points, compute the key point descriptor attributes (the 128D orientation value vector)
- Do the key points matching based on the distance between two descriptor attributes. For each key point in image 1, there is one optimal key point matching with it. (But for each key point in image 2, there may be 0 or several key points in image 1 matching with it.

Result image:

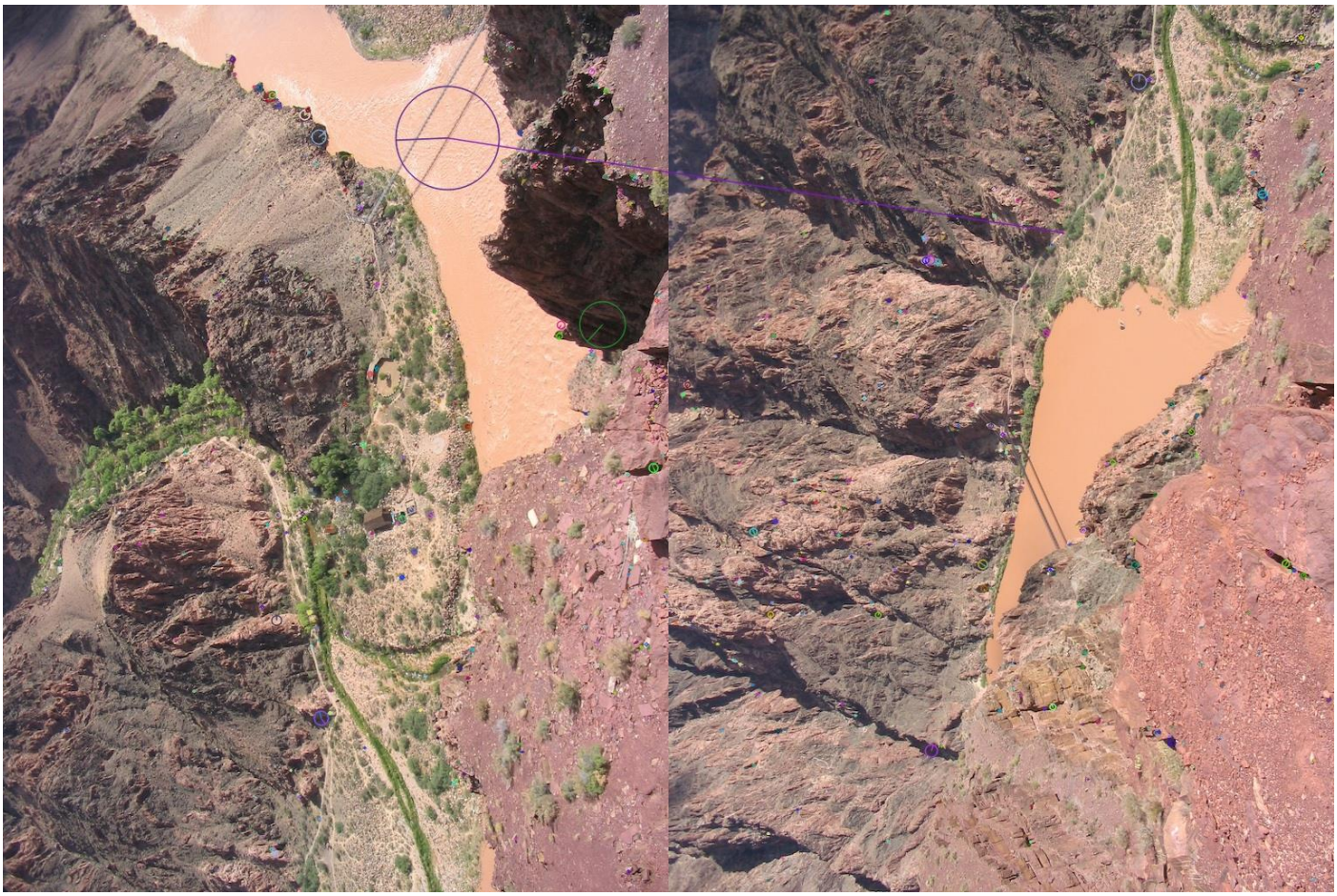


Figure2-1: Key point with largest scale in image 1 and its matching key point in image 2

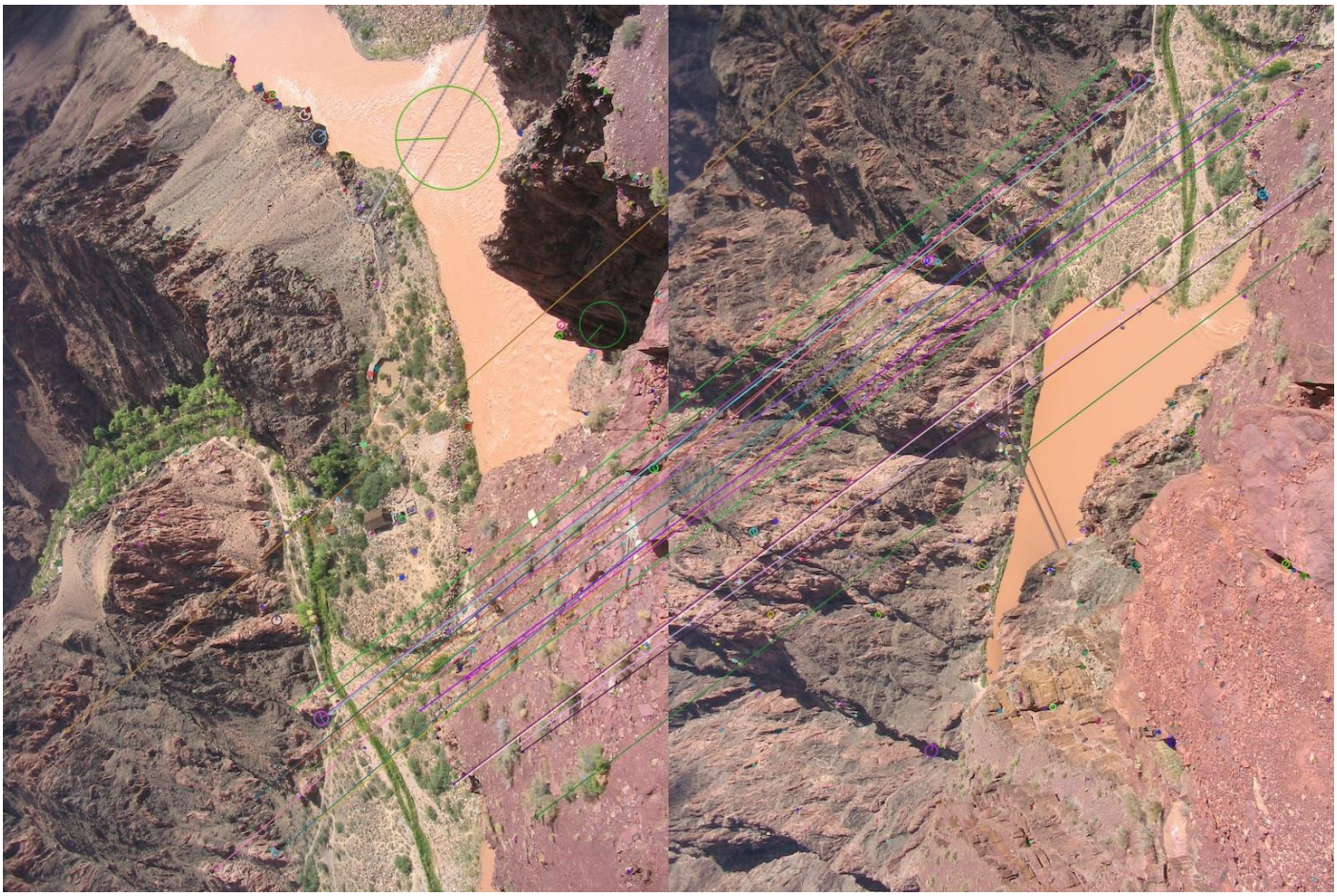


Figure2-2: Matched points pairs with high matching accuracy

Discussion:

- In the program, I set the key point # = 400 in each image.
- Most of the key points are shown around the corner area
- The diameters of circles in the Figure above denote the **orientation** of the key points' descriptor. In addition, the area of the circle denote the meaningful neighborhood area. From the direction of the line from the center of the circle to the edge, we can know which direction of this key point has the largest gradient value.
- Form the input image, we can find that only the left-bottom part of the "river1.raw" is approximately the same with the upper-right part of the "river2.raw". Therefore, in Figure2-2, only these parts' key points are matching to the 2nd image.
- In Figure 2-1, we can find that around the key point with the largest scale, the texture is quite smooth, so this key point have a large meaningful neighborhood area.
- **Orientation** calculation procedure:
  For each key point, the meaningful neighborhood area is divided into 16*16 segment. For each segment, there is a segment gradient. Group the segment into 16 group (4*4 each group, as shown in the Figure2-3), and calculate the total gradient of 8 direction (for each 45°). Therefore, there are 128 gradients, which form the 128-D **orientation** attribute vector.

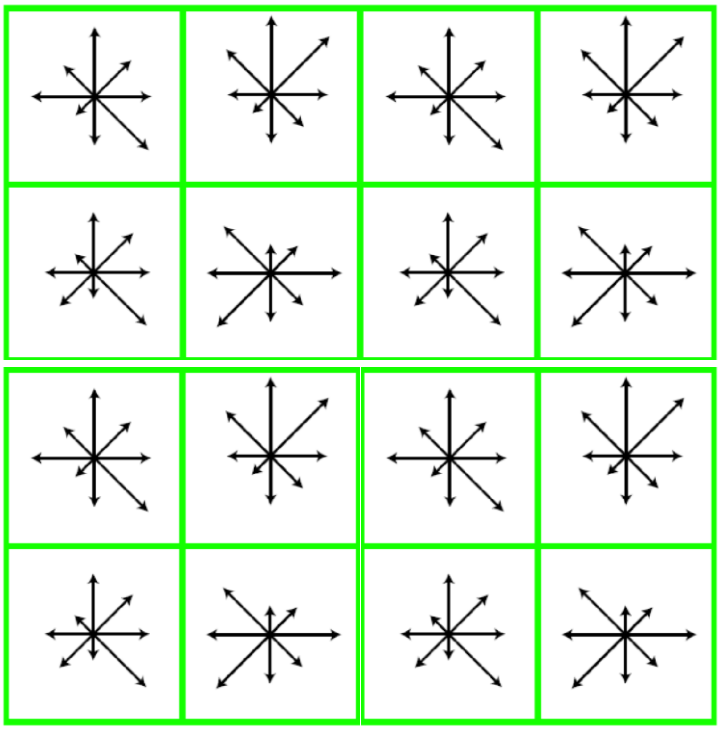


Figure2-3: orientation segment

C bag of words

- # of key points of each image is 20, and the descriptor # is less than 20.
- For all the pictures in his problem, we should resize the 28*28 image to a larger scale, to get a better performance. For example, set width and height both to 2, 4, 10 times of original length.
- For each "zero" and "one" image, we need to find the 20 key points with 20 descriptors. And cluster them into 2 bins (each bin denoted with a 128-D descriptor vector).
- We calculated the 20 key points of the "8" image, along with 20 descriptor. For each of the descriptor, we calculate the distance from the 2 bins' descriptor vector, and put this key point into the bin with smaller distance.
  Output data:

	Feature 1	**Feature 2**	result
2 (resize scale)	**15**	0	Feature 1 is more close to "1"
"0" 1st	3	**8**	

“0” 2nd	**15**	0	The “8” has more “1” feature
“0” 3rd	0	**3**	
“0” 4th	0	**17**	
“0” 5th	1	**7**	
“1” 1st	**14**	4	
“1” 2nd	**9**	0	
“1” 3rd	**16**	2	
“1” 4th	**3**	1	
“1” 5th	**12**	4	
4 (resize scale)	**11**	3	Feature 1 is more close to “1” The “8” has more “1” feature
“0” 1st	4	**6**	
“0” 2nd	**10**	0	
“0” 3rd	0	**6**	
“0” 4th	0	**18**	
“0” 5th	4	**5**	
“1” 1st	**14**	1	
“1” 2nd	**8**	1	
“1” 3rd	**9**	6	
“1” 4th	**4**	2	
“1” 5th	**15**	0	
10 (resize scale)	0	**13**	Feature 1 is more close to “0” The “8” has more “1” feature
“0” 1st	**6**	**6**	
“0” 2nd	0	**10**	
“0” 3rd	**5**	1	
“0” 4th	**16**	0	
“0” 5th	**11**	1	
“1” 1st	6	**9**	
“1” 2nd	**4**	**4**	
“1” 3rd	1	**16**	
“1” 4th	2	**7**	
“1” 5th	**8**	3	

- and the output show that “8” image has more segment similar to the “1” bin.