

HW6



(Aaron) Chien-Cheng Lai
ChBE

I. Bag of Features Classification with SIFT Descriptors

Use SIFT to find features (and their descriptors)

By using `vl_sift` function to find the SIFT features of images in training dataset, I store sift description of each class into 'train'.

1x3 struct with 2 fields

Fields	description	label
1	128x92889 uint8	'024.butterfly'
2	128x11997 uint8	'051.cowboy-hat'
3	128x11838 uint8	'251.airplanes'

Cluster all the SIFT feature descriptors

1. Training

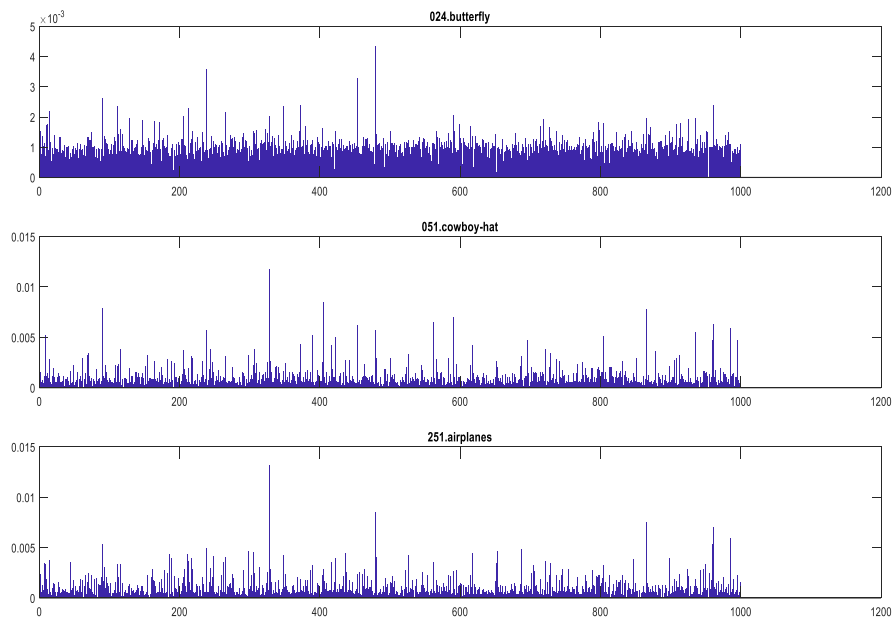
- Caculate kmeans:

Then in order to cluster all the SIFT feature descriptors, I use `vl_kmeans` with Elkan distance algorithm. I originally chose the Euclidean distances, but I found it has several disadvantages. Firstly, it could cause poor result to elongated clusters, or manifolds with irregular shapes. Secondly, it also tend to become inflated in high dimensions which is this case. Then, in `vl_feat` I thought Lloyd's algorithm which could be understood through the concept of Voronoi diagrams may be a good choice. However, I found that Elkan which using an acceleration technique based the triangular inequality is faster than Lloyd with equivalent algorithm. Thus, that is why I prefer Elkan.

- Form a histogram of N for each class:

The result of kmeans calculation is a 128×1000 matrix containing centers. With this center matrix and SIFT feature descriptors, I can find the corresponding center for each feature by computing the minimum distance between feature and center. If the feature is closest to $H(i)$, it will add one to $H(i)$. Then I create a histogram of $1000(=N)$ bins for each class.

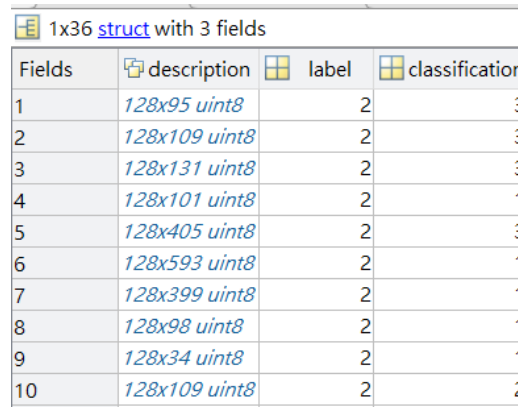
In this part, I use `vl_alldist2` function with Euclidean metric to compute the minimum distance. Because it is just a calculation between two vectors, Euclidean metric would be simple and fast. In the end, it's also normalized to $[0,1]$ by divided total number of features in each class .



2. Testing data

- **Use SIFT to find features (and their descriptors)**

In this homework, there are three testing data set including total 36 images. Again, I firstly create a struct 'test' with three columns (description, label and classification). Then I use `vl_sift` function to find the SIFT features description of images and put it into test description.



Fields	description	label	classification
1	128x95 uint8	2	3
2	128x109 uint8	2	3
3	128x131 uint8	2	3
4	128x101 uint8	2	1
5	128x405 uint8	2	3
6	128x593 uint8	2	1
7	128x399 uint8	2	1
8	128x98 uint8	2	1
9	128x34 uint8	2	1
10	128x109 uint8	2	2

(Note: label is the right class; while classification is the result after classification. If classification equals to label, it is successful.)

- **Compute feature histogram for each test image**

In order to know which class the test image should be in. I firstly compute feature histogram for each test image. Then, I could calculate the difference between this feature histogram and each three histogram for each class later. As before, those histograms are normalized by being divided total number of features in every test image.

(Note: I put those 36 normalized histogram into a 36*1000 matrix called `H_test`)

- **Classification**

In this step, I compute the difference between each test image histogram and every histogram for 3 classes. Then classifying the test image to the class which has minimum histogram difference. The result is put in the classification column in struct 'test'.

3. Results and Discussion

	Butterfly	Hat	Airplane	Total
Butterfly	100%	0%	0%	100%
Hat	50%	10%	'40%	100%
Airplane	12.5%	18.75%	68.75%	100%

For the above case, the accuracy is 59.5833333%.

In this case, I use SIFT for feature extraction, vl_kmean with Elkan algorithm for clustering all the SIFT feature descriptors and vl_alldist2 with Euclidean metric for form a histogram.

However, as you can see in this table and accuracy, the outcome does not perform well and also takes much time in computing. There are bunch of things I can improve both in feature extraction and cluster description. I think reducing the dimensions of features is a good way to save computational time and improve performance. We can try PCA-SIFT, SURF, Affine-SIFT, HOG and so on. Besides, we can try KD-trees and forests for clustering. Also, the distance matrix could be chosen more carefully, for example, ANN algorithm in kmeans may be a good method.

II. Algorithm Implementation Revisited

In part two, as I handle the 25 class dataset in its entirety, the goal is to implement alternate method to improve the performance in both time consuming and accuracy. I choose to change the way I used in feature extraction, using SURF (Speeded-Up Robust Features) instead of SIFT.

- Comparing SURF, SIFT and ORB

Based on the literature called “Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images”, It compared three different image matching techniques through different kinds of transformations and deformations such as scaling, rotation, noise, fisheye distortion, and shearing. It shows that.

(From <https://arxiv.org/ftp/arxiv/papers/1710/1710.02726.pdf>)

Table 1. Results of comparing the images with varying intensity.

	Time (sec)	Kpnts1	Kpnts2	Matches	Match rate (%)
SIFT	0.13	248	229	183	76.7
SURF	0.04	162	166	119	72.6
ORB	0.03	261	267	168	63.6

Table 2. Results of comparing the image with its rotated image.

	Time (sec)	Kpnts1	Kpnts2	Matches	Match rate (%)
SIFT	0.16	248	260	166	65.4
SURF	0.03	162	271	110	50.8
ORB	0.03	261	423	158	46.2

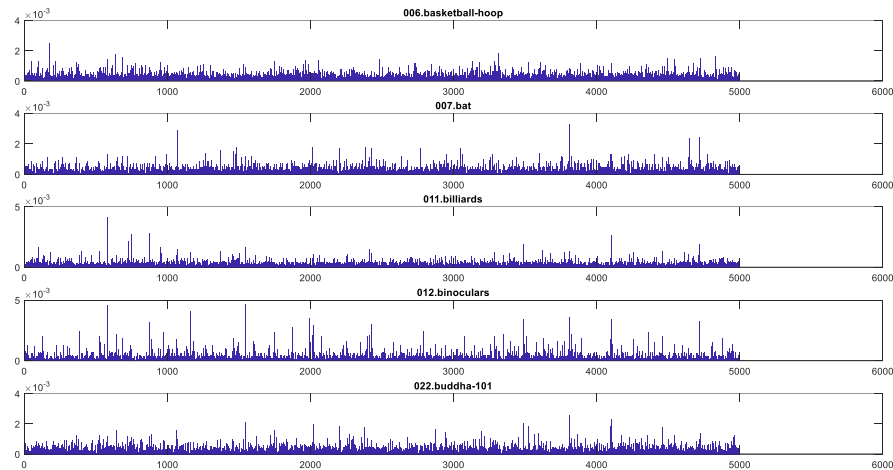
Table 3. Matching rate versus the rotation angle.

Angle →	0	45	90	135	180	225	270
SIFT	100	65	93	67	92	65	93
SURF	99	51	99	52	96	51	95
ORB	100	46	97	46	100	46	97

Although the matching rate of SURF is lower than SIFT, but it saves lots of computational time. In this part, we need to apply a huge dataset which would take too much time, so SURF is an excellent way to replace SIFT. The accuracy drops from using SURF instead of SIFT is about 13.5% while the time saved from using SURF instead of SIFT is about 25%. That is to say, we can save a lot of time by compromising some accuracy.

Comparing SURF to ORB, the time consuming are almost the same, but the performance in SURF is greater than performance in ORB. Therefore, that why I choose SURF to improve the algorithm.

- Using SURF for feature extraction for improvement for 25*25
The first five feature histograms shows below(N=3000)



The 25*25 confusion matrix is below,(N = 3000)

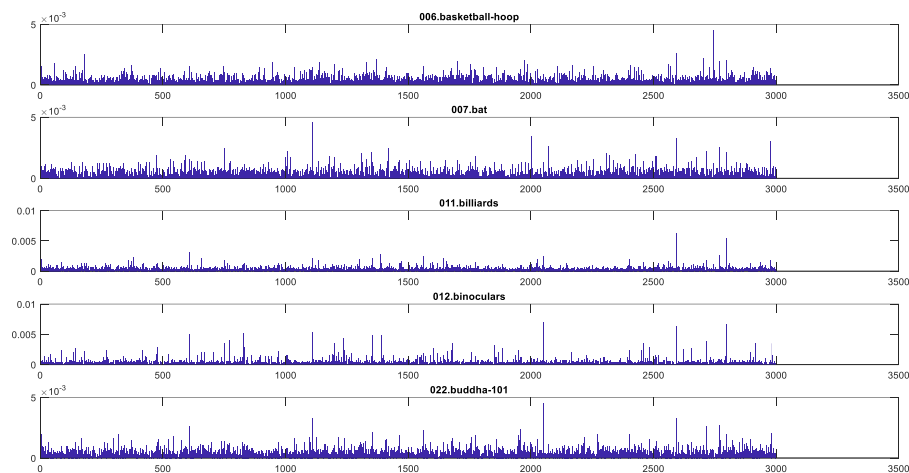
006.basketball-hoop	0.083	0	0	0	0.083	0	0	0	0.083	0	0	0.333	0.083	0.083	0	0	0.083	0	0	0	0.083	0.083	0	0	0
007.bat	0	0.143	0.071	0	0	0.214	0.071	0.143	0.071	0	0	0	0.071	0	0	0.071	0.143	0	0	0	0	0	0	0	0
011.billiards	0	0	0.3	0	0.1	0	0	0	0.1	0	0	0	0	0	0	0.4	0	0	0	0	0	0	0	0	0.1
012.binoculars	0	0	0.1	0.8	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0
022.buddha-101	0	0	0	0	0.286	0.071	0.071	0	0.071	0	0	0	0	0	0	0.071	0.286	0	0	0	0.071	0	0	0	0.071429
024.butterfly	0	0	0	0	0	0.4	0.1	0.3	0	0	0	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0
025.cactus	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.2	0	0	0	0	0	0	0	0
026.cake	0	0	0	0	0	0.2	0	0.3	0	0	0	0.1	0	0.2	0	0.1	0	0	0	0	0.1	0	0	0	0
028.camel	0	0	0	0.1	0.1	0	0	0.4	0	0	0	0	0.1	0.1	0.1	0.1	0	0	0	0	0	0	0	0	0
031.car-tire	0	0	0.067	0.067	0.067	0	0.133	0	0	0	0	0	0.067	0	0.067	0.2	0.067	0	0	0.067	0	0	0.133	0.067	0
037.chess-board	0	0	0	0	0	0.286	0.143	0	0	0	0.214	0	0	0.214	0	0	0.071	0	0	0	0	0	0.071	0	0
045.computer-keyboard	0	0	0	0	0	0.071	0	0	0	0	0	0.286	0.071	0.143	0	0	0.071	0.071	0	0	0	0.071	0.143	0	0.071429
051.cowboy-hat	0	0	0	0.3	0.1	0	0	0.1	0	0	0	0	0.3	0	0	0.1	0.1	0	0	0	0	0	0	0	0
054.diamond-ring	0	0	0	0	0	0	0.1	0.1	0	0	0	0	0	0.5	0	0.1	0	0	0	0	0.1	0	0	0	0.1
063.electric-guitar	0.3	0	0	0.1	0	0	0.1	0.1	0	0	0	0	0	0	0.3	0	0.1	0	0	0	0	0	0	0	0
072.fire-truck	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8	0	0	0	0	0	0	0.2	0	0
093.grasshopper	0	0	0	0	0	0	0.2	0.1	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0.2
102.helicopter	0.5	0	0	0	0	0	0.1	0	0	0	0	0	0.1	0	0	0	0	0.1	0	0	0	0.1	0	0	0.1
129.leopards	0	0	0	0	0	0.077	0.538	0.077	0	0	0	0	0	0.077	0	0	0.154	0	0	0	0.077	0	0	0	0
145.motorbikes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0	0	0	0.6	0	0	0	0	0
159.people	0	0	0	0	0	0.4	0.1	0	0.3	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0.1	0	0
171.refrigerator	0.063	0	0.25	0	0	0	0	0.125	0	0	0	0	0.188	0.063	0	0.063	0.125	0	0	0	0.063	0.063	0	0	0
178.school-bus	0	0	0	0	0.1	0	0.2	0	0	0.1	0	0	0	0	0	0.1	0	0	0	0	0	0	0.5	0	0
180.screwdriver	0.071	0	0	0.071	0	0	0	0.071	0	0	0	0	0.071	0	0.071	0.071	0.071	0	0	0	0	0	0.286	0.071	0.142857
251.airplanes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Below shows the accuracy on different cluster number in kmeans.

	N=1000	N=3000	N=5000
Accuracy	33.56	33.9	32.88

Based on the table, accuracy has maximum value around 3000.

- Using SURF for feature extraction for improvement for 12*12
- The first five feature histograms shows below(N=3000)



The 12*12 confusion matrix is below,(N = 3000)

006.basketball-hoop	0.417	0	0	0	0.083	0	0.083	0	0.083	0	0	0.333
007.bat	0	0.071	0.071	0.071	0	0.143	0.071	0.286	0.286	0	0	0
011.billiards	0.2	0	0.4	0	0.1	0.1	0	0	0.1	0.1	0	0
012.binoculars	0	0	0.1	0.9	0	0	0	0	0	0	0	0
022.buddha-101	0	0.071	0	0.071	0.5	0	0.071	0.143	0.143	0	0	0
024.butterfly	0	0	0	0	0	0.6	0.1	0.2	0.1	0	0	0
025.cactus	0.1	0	0	0	0	0.2	0.6	0	0.1	0	0	0
026.cake	0.1	0.1	0	0	0	0.2	0	0.5	0.1	0	0	0
028.camel	0.1	0	0	0.1	0.2	0	0.1	0.1	0.4	0	0	0
031.car-tire	0.2	0	0.067	0.133	0.067	0.067	0.2	0	0.067	0.2	0	0
037.chess-board	0.071	0	0.071	0	0	0.286	0.286	0	0.071	0	0.214	0
045.computer-keyboard	0.071	0	0.071	0.143	0.071	0.071	0.071	0	0.071	0	0	0.429

Below shows the accuracy on different cluster number in kmeans.

	N=1000	N=3000	N=5000
Accuracy	38.46	41.26	40.56

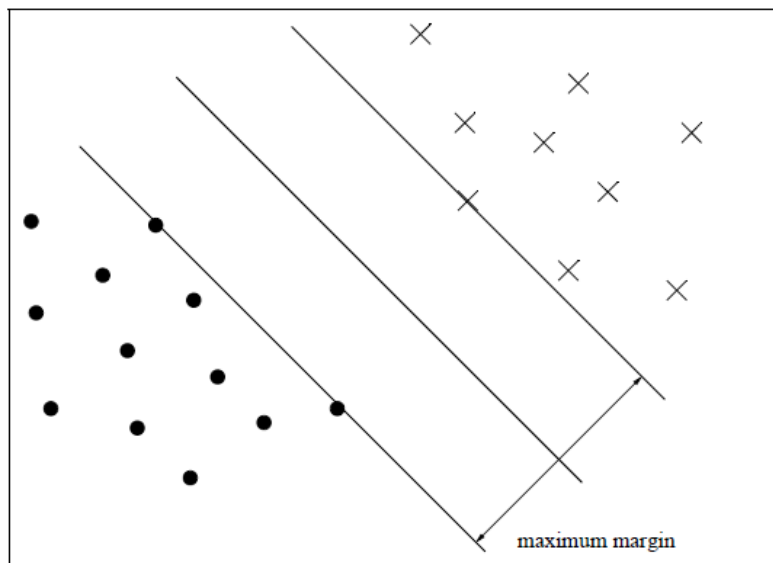
Based on the table, accuracy has maximum value around 3000.

III. Support Vector Machines for Image Classification

Review paper

This paper are focusing on the potential application of SVM. That is, imaging there are two categories and our data have two features as showing below.

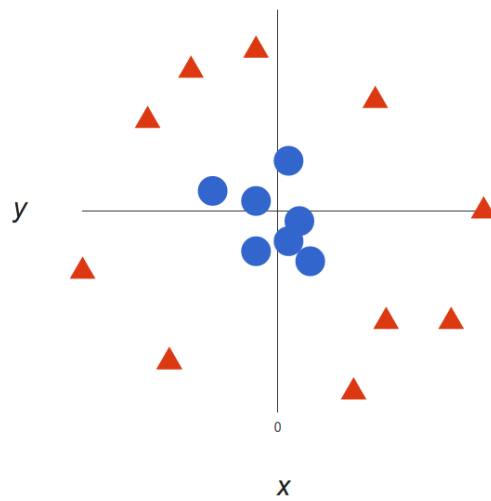
In order to find out the best hyperplane, it computes the distance between the hyperplane and the nearest element of each category. The hyperlane with the largest value of distance are what we looking for. The distance to the closest point is $1/||w||$ while the margin is $2/||w||$.



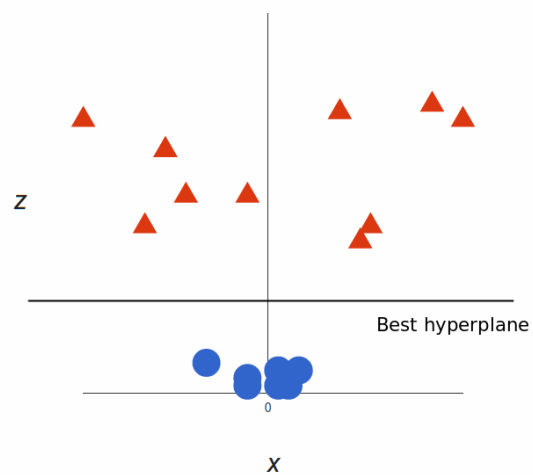
In addition, for nonlinear data, the data are not always as simple as this. In fact, it is usually difficult to draw a simple linear decision boundary. Therefore, it will add a dimension to solve the nonlinear data. For example, if we have x and y dimensions, we would add z as third one. By setting $z = \phi(x)$, we have

$$W(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

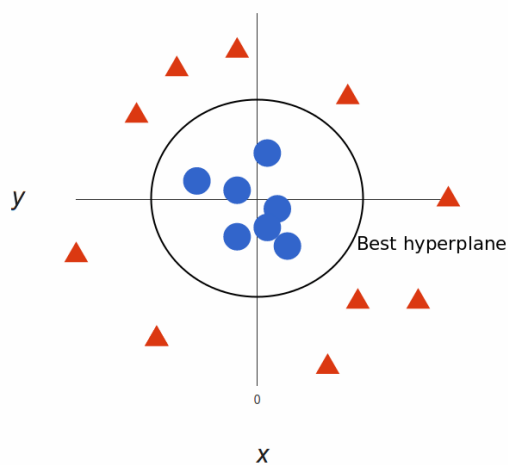
For example, if our data looks like this.



Then we add third dimension to decide the hyperplane.



Then we transform it back to two dimensions.



However, although we are able to classify nonlinear data through the way that we map the space to higher dimensions, it comes another big issue. This transformation will cost too much resource when it comes to too many dimensions.

To achieve a cheaper computing solution, it introduces the kernel trick. By kernel trick, we do not even have to compute the vectors. We could compute the dot product among these vectors. That allow us to do the transformation in less computational cost.

- **SVM for image classification**

Since the SVM is based on binary classification, we need change the process of SVM or combine several binary one in order to apply to multi-class classification.

As the experiment result shows in the paper, a kernel based on χ^2 based performs very well. Therefore, we can also apply this general ideal to other issues.

In conclusion, SVM allows us to classify nonlinear data with kernel method. Besides, χ^2 based kernel is outstanding performing in the experiments.

- **Train SVM on 3 class dataset**

To apply SVM to our data set, I use the matlab function, including bagOfFeatures for feature extraction, trainImageCategoryClassifier for SVM application and evaluate for testing data.

The confusion matrix shows below,

	PREDICTED		
KNOWN	1 024	051	251
024	1 1.00	0.00	0.00
051	1 0.40	0.50	0.10
251	1 0.00	0.00	1.00

Besides, average accuracy is 0.83. It's faster and more accurate than I did previously.