Haoran Man

ELEC345 HW6 Image Classification

1 Bag of Features Classification with SIFT Descriptors

1.1 Algorithm Implementation

1. I used vl_sift() to detect SIFT features in every image within each class and stored them in a struct 'train_imgs'.

2. For Clustering, I am using vl-kmeans() which is similar to kmeans but a bit faster. After combining all features of all classes into X(shown below), vl-kmeans() is applied to get the clusters. Each column is one of the cluster center.

```
>> whos X
Name       Size              Bytes        Class
X       128x126634        16209152  uint8
```

   Code I used: (N=1000)

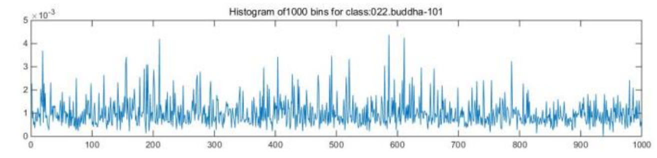
```
[C, A] = vl_kmeans(single(X),N,'distance', 'l1', 'algorithm', 'elkan');
```

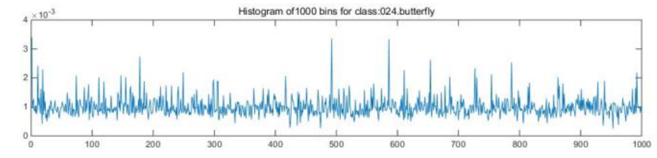
   Here, I used city block distance option and 'elkan' option to make the clustering faster.

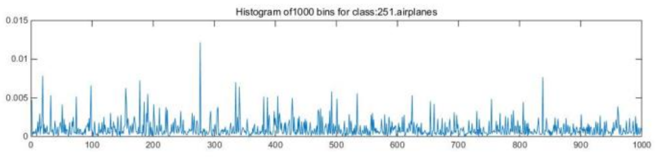
3. (a)For each class, I formed X1 that contains all features within that class. Then, I used Knnsearch() to get a distance metric. Then, for the result, I get the median of all distances and set 1.7*median as the threshold to get rid of the features that are far away from any clusters. (after different trial of threshold setting, I choose 1.7 here)
   Code:

```
[idx,D] = knnsearch(cluster{1,1}',X1','distance','cityblock');
```

   (b)Histogram is normalized and shown below:

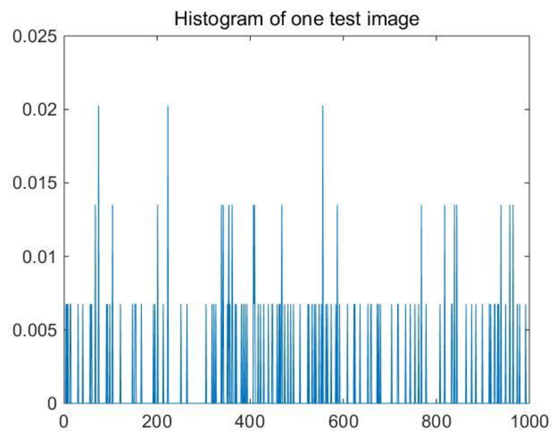






4. Use get SIFT features within one test image
5. Get the histogram and do the same thresholding method used before.
6. Normalize the histogram. Below is histogram of one test images.

Hist1 = Hist1./length(idx)



7. Use knnsearch to get the closest fit for the test image histogram within all 3 classes' histogram.

2. Technical write-up: Results and discussion

The result is shown below:

	Predicted			
Real Class		Buddha	Butterfly	Airplane
	Buddha	**71.4%**	21.4%	7.2%
	Butterfly	20%	**80%**	0%
	Airplane	12.5%	0%	**87.5%**

The result it pretty good. But the entire calculation took my computer ~40min to complete. This algorithm is definitely not appropriate for the second part of the assignment.
Clustering all 1000 clusters at once took a lot of time, I think do clustering at different levels with each much less cluster number for each level will be a more efficient way. In this way, we will be able to classify an image from coarse to fine levels, get the result much faster.

One interesting thing I observed: even if I do clustering on only one training class, after doing histogram for all three classes base on this single-class clustering, I can still get decent recognition rate: 78%, 70%, 93% for three classes with only class 1 is used for clustering. This means the clusters each class of image has is similar, only the histogram varies a lot. So another way of getting the algorithm faster is to get the cluster with less image input but use all images as input for histogram generation.

2. Classification on 25 image classes

Data preparation

For the test images, I grouped them manually to different folders for better testing.

Implementation

Total 25 classes of training images give us about 820,000 features. Clustering them at once to more than 1000 clusters will take forever. I once let it run for 3 hours and the clustering is not finished yet.

Clustering at different levels as I mentioned in the first part can be the way to go to reduce the calculation time. As suggested in this part, vocabulary tree is the algorithm I am looking for. For this algorithm, the traditional 128 SIFT descriptor is used, so the loading part of part one can be used for this part directly.

Clustering

The following reference helped on the following part of the assignment.

https://www.mpi-inf.mpg.de/fileadmin/inf/d2/HLCV/HLCV_2014/sb_exercise4_0.pdf

Luckily for us, the vocabulary tree clustering is already implemented in VLFeat. I used vl_hikmeans() for the clustering, which is Hierarchical integer k-means function in VLFeat.

```
[tree,A]=vl_hikmeans(uint8(X),K,N)
```

In the code, K is the branch for each level and N is the minimum clusters in the vocabulary tree.

Using this algorithm, clustering takes less than 10 min, which is much faster than ordinary clustering.

Creating histogram

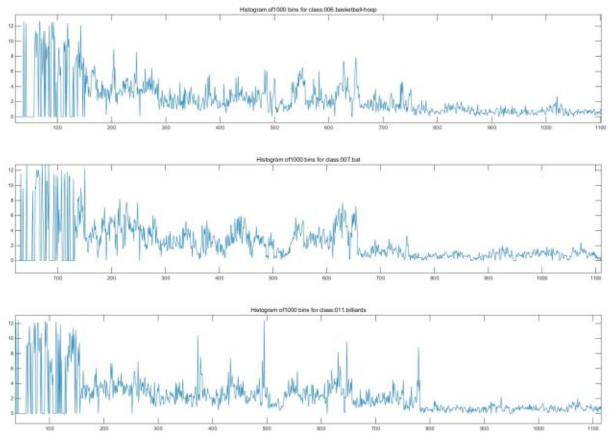
To get the histogram for each class, I used the following code:

```
path = vl_hikmeanspush(tree,X1) ;

H = vl_hikmeanshist(tree,path)';
```

X1 is the combination of descriptors within one class and then H is the histogram for this class on the vocabulary tree clusters.

Below is the histogram for the first 3 classes based on N=1000 and K=5 after getting rid of the high values.

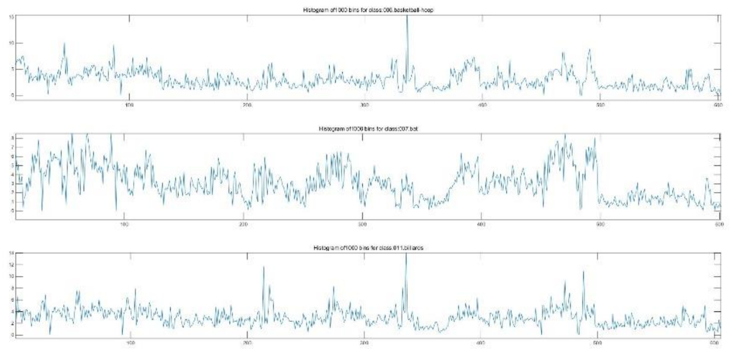


We can see there is a significant anomaly below ~160. I tried using the entire histogram to do recognition and the result is not good. Instead, I used the histogram from 160:end for the comparison. Note that the cutoff will change when K and N change, approximately the later 4/5 of the histogram is generally good for classification.

For this part of the histogram, no thresholding is applied. Then, I used the mean value of the histogram as the normalization factor.

Hist1 = Hist1/(mean(Hist1));

After the normalization, I get the histogram as follows:



As we can see, different classes are very distinctive. For testing below, I used this 'threshold' method.

Testing

The result is shown in the confusion matrix below. Average recognition rate is 32.46%.

		Predicted Class																								
	Class Name	basketball-hoo		billiards		buddha		cactus		camel		chess-board		cowboy-hat		electric-guitar		grasshopper		leopards		people		school-bus		airplan
	Class Name		bat		binoculars		butterfly		cake		car-tire		computer-key		diamond-ring		fire-truck		helicopter		motorbikes		refrigerator		screwdriver	
	basketball-hoop	0.083	0.083	0.083	0	0	0.083	0.083	0.167	0	0	0	0	0	0	0	0.083	0.167	0	0	0	0	0	0	0.083	0.083
	bat	0	0.143	0.071	0	0	0.071	0.214	0	0.071	0	0	0	0.071	0.071	0	0	0	0	0.071	0	0.214	0	0	0	0
	billiards	0	0	0.1	0	0.1	0.1	0	0.2	0	0	0	0	0	0	0	0	0	0.1	0	0	0.1	0	0.2	0.1	0
	binoculars	0	0	0.1	0.6	0	0	0	0.1	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0.1	0
	buddha	0	0.071	0	0.071	0.5	0	0.143	0.071	0	0	0	0	0.071	0	0	0	0.071	0	0	0	0	0	0	0	0
	butterfly	0	0.1	0	0	0.2	0.5	0	0	0	0	0	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0
	cactus	0	0	0	0	0.1	0.1	0.5	0	0.1	0	0	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0
	cake	0	0.1	0	0	0	0	0.1	0.3	0	0	0	0	0	0.1	0	0	0	0.1	0	0	0.2	0	0.1	0	0
	camel	0	0	0	0.2	0	0	0.1	0.1	0.2	0	0	0	0	0	0	0	0.1	0	0	0	0.1	0	0.1	0.1	0
	car-tire	0	0.133	0	0.067	0.067	0	0.067	0.067	0	0.133	0	0.067	0	0	0	0.067	0	0	0	0	0.067	0	0.2	0	0.067
	chess-board	0.071	0.071	0	0	0	0.071	0.357	0.143	0	0	0.214	0.071	0	0	0	0	0	0	0	0	0	0	0	0	0
	computer-keyboard	0	0	0	0.143	0	0	0	0	0.071	0	0	0.429	0	0	0.071	0.143	0.071	0.071	0	0	0	0	0	0	0
Real Class	cowboy-hat	0	0	0	0.2	0	0	0	0.1	0	0	0	0	0.2	0.1	0	0	0	0	0	0	0.1	0.1	0	0	0.2
	diamond-ring	0	0	0.1	0	0.1	0	0.2	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0.1	0.1	0	0	0.1
	electric-guitar	0	0	0	0.2	0	0	0	0.1	0	0	0	0	0	0	0.3	0	0.2	0	0.1	0.1	0	0	0	0	0
	fire-truck	0.1	0	0	0	0	0	0.3	0	0.1	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
	grasshopper	0	0	0.1	0	0	0	0.2	0.1	0	0	0	0	0	0.1	0	0	0.1	0	0	0	0	0	0.1	0.1	0.2
	helicopter	0	0	0	0	0	0	0	0.1	0	0.1	0	0	0.1	0	0	0	0.2	0.3	0	0	0	0.1	0	0	0.1
	leopards	0	0.077	0	0	0	0.077	0	0	0	0	0	0	0	0	0	0	0	0	0.692	0	0.077	0.077	0	0	0
	motorbikes	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6	0	0	0.1	0	0.1
	people	0	0	0	0	0.2	0.2	0.1	0	0.1	0.2	0	0	0	0	0	0.1	0	0	0	0	0.1	0	0	0	0
	refrigerator	0	0	0.063	0.125	0.063	0	0.125	0.063	0	0.063	0	0	0	0	0.063	0	0.125	0	0	0	0	0.125	0	0.188	0
	school-bus	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0.2	0.1	0	0	0.1	0	0.5	0	0
	screwdriver	0	0	0.143	0.286	0.071	0	0	0	0	0	0	0	0.071	0	0.071	0.071	0	0	0	0	0	0.071	0.143	0.071	0
	airplanes	0	0	0	0.063	0	0	0	0.125	0	0	0	0	0.063	0	0	0	0	0	0	0	0.063	0.063	0	0	0.625
	SUM	0.255	0.779	0.76	1.954	1.401	1.403	2.489	1.735	0.743	0.496	0.214	0.767	0.577	0.671	0.605	1.164	1.235	0.671	0.864	0.7	1.22	0.636	1.443	0.742	1.475

Changing K and N in the clustering will change accuracy as well.
K=5, N=1000, rate 32.46%
K=7,N=2000, rate 30.79%
K=10,N=5000, rate 28%

Certain classes such as binoculars, leopards, motorbikes and airplanes are identified with success rate larger than 60%. Clearly these classes have more distinctive features than other classes like people. Binoculars, motorbikes, airplanes have a much clearer background than other images. For leopards, it has obvious and unique feature vectors that could make it easier to be identified.
To acquire more accurate classification, more training images will be needed.

Personal thoughts

Computation power is always limited. A good algorithm will help me much more than a faster computer.

Machine learning is interesting. I might take classes on machine learning later.

I am surprised I am able to implement this algorithm after taking this class. I am having a great time on this class.

Code Instruction：

HW6.m has 2 modes, 1 is for reduced set and 2 is for expanded set. Hit run and you will be able to get the final confusion matrix for either case.

		Predicted Class																								
	Class Name	basketball-ho(		billiards		buddha		cactus		camel		chess-board		cowboy-hat		electric-guitar		grasshopper		leopards		people		school-bus		airplan
	Class Name		bat		binoculars		butterfly		cake		car-tire		computer-key		diamond-ring		fire-truck		helicopter		motorbikes		refrigerator		screwdriver	
Real Class	basketball-hoop	0.083	0.083	0.083	0	0	0.083	0.083	0.167	0	0	0	0	0	0	0	0.083	0.167	0	0	0	0	0	0	0.083	0.083
	bat	0	0.143	0.071	0	0	0.071	0.214	0	0.071	0	0	0	0.071	0.071	0	0	0	0	0.071	0	0.214	0	0	0	0
	billiards	0	0	0.1	0	0.1	0.1	0	0.2	0	0	0	0	0	0	0	0	0	0.1	0	0	0.1	0	0.2	0.1	0
	binoculars	0	0	0.1	0.6	0	0	0	0.1	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0.1	0
	buddha	0	0.071	0	0.071	0.5	0	0.143	0.071	0	0	0	0	0.071	0	0	0	0.071	0	0	0	0	0	0	0	0
	butterfly	0	0.1	0	0	0.2	0.5	0	0	0	0	0	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0
	cactus	0	0	0	0	0.1	0.1	0.5	0	0.1	0	0	0.1	0	0	0	0.1	0	0	0	0	0	0	0	0	0
	cake	0	0.1	0	0	0	0	0.1	0.3	0	0	0	0	0	0.1	0	0	0	0.1	0	0	0.2	0	0.1	0	0
	camel	0	0	0	0.2	0	0	0.1	0.1	0.2	0	0	0	0	0	0	0	0.1	0	0	0	0.1	0	0.1	0.1	0
	car-tire	0	0.133	0	0.067	0.067	0	0.067	0.067	0	0.133	0	0.067	0	0	0	0.067	0	0	0	0	0.067	0	0.2	0	0.067
	chess-board	0.071	0.071	0	0	0	0.071	0.357	0.143	0	0	0.214	0.071	0	0	0	0	0	0	0	0	0	0	0	0	0
	computer-keyboard	0	0	0	0.143	0	0	0	0	0.071	0	0	0.429	0	0	0.071	0.143	0.071	0.071	0	0	0	0	0	0	0
	cowboy-hat	0	0	0	0.2	0	0	0	0.1	0	0	0	0	0.2	0.1	0	0	0	0	0	0	0.1	0.1	0	0	0.2
	diamond-ring	0	0	0.1	0	0.1	0	0.2	0	0	0	0	0	0	0.3	0	0	0	0	0	0	0.1	0.1	0	0	0.1
	electric-guitar	0	0	0	0.2	0	0	0	0.1	0	0	0	0	0	0	0.3	0	0.2	0	0.1	0.1	0	0	0	0	0
	fire-truck	0.1	0	0	0	0	0	0.3	0	0.1	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
	grasshopper	0	0	0.1	0	0	0	0.2	0.1	0	0	0	0	0	0.1	0	0	0.1	0	0	0	0	0	0.1	0.1	0.2
	helicopter	0	0	0	0	0	0	0	0.1	0	0.1	0	0	0.1	0	0	0	0.2	0.3	0	0	0	0.1	0	0	0.1
	leopards	0	0.077	0	0	0	0.077	0	0	0	0	0	0	0	0	0	0	0	0	0.692	0	0.077	0.077	0	0	0
	motorbikes	0	0	0	0	0	0.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0.6	0	0	0.1	0	0.1
	people	0	0	0	0	0.2	0.2	0.1	0	0.1	0.2	0	0	0	0	0	0.1	0	0	0	0	0.1	0	0	0	0
	refrigerator	0	0	0.063	0.125	0.063	0	0.125	0.063	0	0.063	0	0	0	0	0.063	0	0.125	0	0	0	0	0.125	0	0.188	0
	school-bus	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0.2	0.1	0	0	0.1	0	0.5	0	0
	screwdriver	0	0	0.143	0.286	0.071	0	0	0	0	0	0	0	0.071	0	0.071	0.071	0	0	0	0	0	0.071	0.143	0.071	0
	airplanes	0	0	0	0.063	0	0	0	0.125	0	0	0	0	0.063	0	0	0	0	0	0	0	0.063	0.063	0	0	0.625
	SUM	0.255	0.779	0.76	1.954	1.401	1.403	2.489	1.735	0.743	0.496	0.214	0.767	0.577	0.671	0.605	1.164	1.235	0.671	0.864	0.7	1.22	0.636	1.443	0.742	1.475