**Apply VQ on Classification Problem**

**Author Information**：

Tian Ye / [tye459@usc.edu](mailto:tye459@usc.edu) / [tian\_ye1994@hotmail.com](mailto:tian_ye1994@hotmail.com) (preferred) / 323-616-7117 (US)

**Description:**

The idea comes from a concept called ‘bag of words’. Suppose there is a dictionary containing basic elements to form something from a space, the usage of each element, or word, can form a vector, or histogram if you like. We can use this histogram vector as a feature vector to present the original object.

In this project, i apply this idea on att\_faces data set, which is made of 400 face images of 40 different people. 280 of them are set as training data and 120 of them are test data. In the following three section, a brief theory will be discussed, detail algorithm will be explained and i will show the result log.

**Section 1: Bag of Words**

Recall how we use VQ on image compression: divide the image to small blocks, view blocks as vectors cluster, find a few key vectors to present all of the vectors, set those key vectors as dictionary, and finally use the dictionary to reconstruct the image.

If we use random vectors to reconstruct the image, no way you can tell what is original image used to be. However, the visual effect of VQ is quite promising despite some blur depends on the setting. This tell us that the trained centers, or words, of VQ learns high level feature from the training image, which qualifies themself for acting as ‘basis’ of the images space.

In conventional machine learning, It is very common to convert an object to a feature vector and pass those vectors to a model, SVM for example. In this case, things can be simplified as setting the label to nearest training image’s label.

**Section 2: Detail of implementation**

**TRAIN:**

Step1：compute the mean of training images

Step2：subtract all training images with mean

Step3：perform 2x2 blocks operation on training images (overlap, stride = 1)

Step4：for each blocks, flatten them as a 1D array to form vectors

Step5：for each vector, find the minValue of that vector and subtract it from all four pixels

Step6：use K\_means to find k centers for all of those blocks

Step7：for each training image, use those centers to reconstruct themselves and record the frequency of each centers used to form the image.

Step8：convert those frequency as histogram to present each face.

**TEST: Assume a test image coming**

Step1: perform Step1~5 in training steps toward this test image

Step2: perform Step7 in training step using centers produced from training images

Step3: search all feature vectors of training images and assign the label with the nearest one’s label.

**Section 3: Result**

**Sample of trained center**

0.00304151 0.000338496 0.00744032 0.00332233

**Sample of feature vector**

0.042174 0.004257 0.002772 0.002178 0.00198 0 0.002772 0.00574201 0.004752 0.00594001 0.000792001 0.002079 0.000396 0.00653401 0.020592 0.001881 0.000594001 0.001485 0.002673 0.003267 0.00792001 0.002673 0.004554 0.003564 0.015939 0.001386 0.00297 0.012276 0.004257 0 0.000990001 9.90001e-05 0.004356 0.000792001 0.001386 0.00544501 0.024552 0.003465 9.90001e-05 0.003465 0.003663 0.011781 0.004257 0.000594001 0.022374 9.90001e-05 0.003267 0.001782 0.000990001 0.004554 0.002376 0.00297 0.010395 0.01584 0.00485101 0.00891001 0.037323 0.003762 0.003069 0.00702901 0.002079 0.001584 0.000891001 9.90001e-05 0.001782 0.002574 0.00584101 0.001188 0.00297 0.028908 0.000297 0.014355 0.00297 0.001584 0.002673 0.024651 0.003663 0.00663301 0.003069 0.001188 0.027423 0.030294 0.001386 0.00198 0.004356 0.000198 0.003564 0.00495001 0.000495001 0.003069 9.90001e-05 0.00920701 0.000594001 0.00495001 0.002673 0.0734581 0.002673 0.004653 0.000792001 0.002673 0.001386 0.003168 9.90001e-05 0.010989 0.00396 0.001089 0.000693001 0.000495001 0.00900901 0.000297 0.001188 0.00396 0.000495001 0.00643501 0.00198 0.003663 0.000990001 0.00554401 0.025641 0.018117 0.00594001 0 0 0.003069 0.002772 0 0.00514801 0.00603901 0.000891001 0.00584101 0.003762 0.001584 0.002871 0.002277 0.00653401 0.003663 0.003564 0.000297 0.003366 0.00613801 0.001584 0.002277 0.017325 0.000495001 0.002673 0.000495001 0.023661 0.016038 0.003168 0.003465 0.00584101 0.001485 0.004653 0.002673 0.00603901 0.001386 0.003267 0.00495001 0.011286 0.002376 9.90001e-05 0.004653 0.003465 0.012177 0.001584 0.002772 0.003366 0.020691 0.00396 0.002574 0.002178 0.004257 0.002475 0.00623701

**Accuracy of test images**

real label: 1 predicted label: 1 distance: 0.033015

real label: 1 predicted label: 39 distance: 0.0389419

real label: 1 predicted label: 1 distance: 0.036379

real label: 2 predicted label: 2 distance: 0.0136153

real label: 2 predicted label: 2 distance: 0.0139664

real label: 2 predicted label: 2 distance: 0.0133756

real label: 3 predicted label: 3 distance: 0.0126921

real label: 3 predicted label: 3 distance: 0.0219307

real label: 3 predicted label: 3 distance: 0.0249818

real label: 4 predicted label: 4 distance: 0.0124339

real label: 4 predicted label: 4 distance: 0.0136009

real label: 4 predicted label: 4 distance: 0.0263801

real label: 5 predicted label: 5 distance: 0.0251947

real label: 5 predicted label: 5 distance: 0.0180604

real label: 5 predicted label: 5 distance: 0.0273072

real label: 6 predicted label: 6 distance: 0.0215265

real label: 6 predicted label: 6 distance: 0.0182853

real label: 6 predicted label: 6 distance: 0.0141019

real label: 7 predicted label: 7 distance: 0.0185825

real label: 7 predicted label: 7 distance: 0.0127944

real label: 7 predicted label: 7 distance: 0.0143533

real label: 8 predicted label: 8 distance: 0.0177246

real label: 8 predicted label: 8 distance: 0.0262191

real label: 8 predicted label: 8 distance: 0.0214417

real label: 9 predicted label: 9 distance: 0.0156658

real label: 9 predicted label: 9 distance: 0.0164221

real label: 9 predicted label: 9 distance: 0.0229808

real label: 10 predicted label: 10 distance: 0.0153728

real label: 10 predicted label: 10 distance: 0.0240853

real label: 10 predicted label: 10 distance: 0.0231415

real label: 11 predicted label: 11 distance: 0.0149106

real label: 11 predicted label: 11 distance: 0.0149683

real label: 11 predicted label: 11 distance: 0.0165434

real label: 12 predicted label: 12 distance: 0.0188485

real label: 12 predicted label: 12 distance: 0.0167219

real label: 12 predicted label: 12 distance: 0.0194545

real label: 13 predicted label: 13 distance: 0.0123453

real label: 13 predicted label: 13 distance: 0.0176759

real label: 13 predicted label: 13 distance: 0.016509

real label: 14 predicted label: 14 distance: 0.0184156

real label: 14 predicted label: 14 distance: 0.0258449

real label: 14 predicted label: 14 distance: 0.0249516

real label: 15 predicted label: 15 distance: 0.0190266

real label: 15 predicted label: 15 distance: 0.0136283

real label: 15 predicted label: 15 distance: 0.0176836

real label: 16 predicted label: 16 distance: 0.0174201

real label: 16 predicted label: 16 distance: 0.0194887

real label: 16 predicted label: 16 distance: 0.0135257

real label: 17 predicted label: 17 distance: 0.018915

real label: 17 predicted label: 17 distance: 0.0191698

real label: 17 predicted label: 17 distance: 0.0122088

real label: 18 predicted label: 18 distance: 0.0163521

real label: 18 predicted label: 18 distance: 0.0157195

real label: 18 predicted label: 18 distance: 0.017433

real label: 19 predicted label: 19 distance: 0.0152145

real label: 19 predicted label: 19 distance: 0.0379345

real label: 19 predicted label: 19 distance: 0.0149611

real label: 20 predicted label: 20 distance: 0.0140636

real label: 20 predicted label: 20 distance: 0.0142126

real label: 20 predicted label: 20 distance: 0.0177572

real label: 21 predicted label: 40 distance: 0.0245947

real label: 21 predicted label: 21 distance: 0.0157351

real label: 21 predicted label: 21 distance: 0.0246639

real label: 22 predicted label: 22 distance: 0.0156276

real label: 22 predicted label: 22 distance: 0.0192193

real label: 22 predicted label: 22 distance: 0.0187364

real label: 23 predicted label: 23 distance: 0.0129361

real label: 23 predicted label: 38 distance: 0.0255949

real label: 23 predicted label: 23 distance: 0.0226548

real label: 24 predicted label: 24 distance: 0.0200484

real label: 24 predicted label: 24 distance: 0.0217886

real label: 24 predicted label: 24 distance: 0.0251787

real label: 25 predicted label: 4 distance: 0.0279787

real label: 25 predicted label: 25 distance: 0.014848

real label: 25 predicted label: 25 distance: 0.0200479

real label: 26 predicted label: 4 distance: 0.0288235

real label: 26 predicted label: 26 distance: 0.0201416

real label: 26 predicted label: 26 distance: 0.0288221

real label: 27 predicted label: 27 distance: 0.0164465

real label: 27 predicted label: 27 distance: 0.0179423

real label: 27 predicted label: 27 distance: 0.017713

real label: 28 predicted label: 28 distance: 0.0219383

real label: 28 predicted label: 28 distance: 0.0192539

real label: 28 predicted label: 28 distance: 0.0196225

real label: 29 predicted label: 29 distance: 0.0156006

real label: 29 predicted label: 29 distance: 0.0240898

real label: 29 predicted label: 29 distance: 0.0147573

real label: 30 predicted label: 30 distance: 0.0155073

real label: 30 predicted label: 30 distance: 0.023102

real label: 30 predicted label: 30 distance: 0.0191907

real label: 31 predicted label: 31 distance: 0.0174566

real label: 31 predicted label: 31 distance: 0.0215224

real label: 31 predicted label: 31 distance: 0.0152762

real label: 32 predicted label: 32 distance: 0.020953

real label: 32 predicted label: 32 distance: 0.0229488

real label: 32 predicted label: 32 distance: 0.0200142

real label: 33 predicted label: 33 distance: 0.0176042

real label: 33 predicted label: 33 distance: 0.0156483

real label: 33 predicted label: 33 distance: 0.0161701

real label: 34 predicted label: 34 distance: 0.0187704

real label: 34 predicted label: 34 distance: 0.023191

real label: 34 predicted label: 34 distance: 0.0245847

real label: 35 predicted label: 35 distance: 0.0128151

real label: 35 predicted label: 35 distance: 0.0185408

real label: 35 predicted label: 35 distance: 0.0180153

real label: 36 predicted label: 36 distance: 0.0159098

real label: 36 predicted label: 36 distance: 0.018761

real label: 36 predicted label: 12 distance: 0.0304074

real label: 37 predicted label: 37 distance: 0.0148011

real label: 37 predicted label: 37 distance: 0.0118263

real label: 37 predicted label: 37 distance: 0.0145977

real label: 38 predicted label: 38 distance: 0.0217174

real label: 38 predicted label: 38 distance: 0.0208385

real label: 38 predicted label: 38 distance: 0.0167008

real label: 39 predicted label: 39 distance: 0.0127276

real label: 39 predicted label: 21 distance: 0.0273125

real label: 39 predicted label: 39 distance: 0.0200768

real label: 40 predicted label: 40 distance: 0.0139214

real label: 40 predicted label: 5 distance: 0.0291252

real label: 40 predicted label: 40 distance: 0.0272184

**Total number of test cases:(test) 120**

**# correct: 112 test accuracy: 0.933333**

**Section 4: Others**

**To run the code, highly recommend you recompile it with release mode or this will be extremely slow. (Also OpenCV3 as I/O)**

**When doing matching during produce feature vector, the program can be accelerated by TSVQ which is implemented and explain in “VQ\_and\_TSVQ” published in my page.**

**Program will generate a binary file called ‘codebook.dat’ in codebook folder, clear that directory to generate new codebook.**