ECSE-415 Introduction to Computer Vision

Final Project: Face Recognition and Tagging using Bag of Features

Due: Tuesday, April 14, 2015, 11:59pm

Overview

In this project, you will develop a software system for face recognition. Specifically, you will explore using the popular bag of quantized features and nearest neighbor classifier to classify unseen face images as being one of the training face images. Your resulting face recognition system will be used to tag members of the team in group photos. It is required that you work in teams of 3-4 students. Your project report should be in pdf format and is due on Tuesday, April 14, 2015 at 11:59pm on myCourses. All code must be handed in as well. Reports submitted up to 12 hours late will be penalized by 30%. After that, the student will be given a 0 grade for the project. The project will be graded out of a total of 100 points.

Bag of words models are a popular technique for image classification inspired by models used in natural language processing. Before we can represent our training and testing images as bag of feature histograms, we first need to establish a vocabulary of visual words. The visual word vocabulary is established by clustering a large corpus of local features. You will form this vocabulary by sampling many local features from our training set and then clustering them with K-means. The number of clusters is the size of our vocabulary. For example, if we cluster the feature space into K clusters, this partitions the continuous, D-dimensional feature space into K regions. For any observed feature, we can figure out which region it belongs to, as long as we save the centroids of our clusters. The centroids form our visual word vocabulary. Using these cluster centroids, we can then quantize the descriptors into one of the visual words. Therefore, using bag of words, instead of storing hundreds of descriptors for an image, we simply count how many of its descriptors fall into each cluster in our visual word vocabulary. Each of the training images can be represented by a histogram of visual words. Thus, if we have a vocabulary of 50 visual words, and we detect 220 features in an image, our bag of feature representation will be a histogram of 50 dimensions, where each bin counts how many times a descriptor was assigned to that cluster and sums to 220. The histogram should be normalized so that image size does not dramatically change the bag of feature magnitude.

We can represent any new test image using a histogram of the same visual words and use a nearest neighbor classifier to classify it into one of the training images. The nearest neighbor classifier is equally simple to understand. When tasked with classifying a test image, one simply first finds the histogram representation of the image and then uses the "nearest" training histogram (L2 distance is a sufficient metric) and assigns the test case the label of that nearest training example. Note that although the nearest neighbor classifier has many desirable features (it requires no training and it can learn arbitrarily complex decision boundaries), it is quite vulnerable to training noise. This can be alleviated by voting based on the k nearest neighbors (but you are not required to do so). Nearest neighbor classifiers also suffer as the feature dimensionality increases (e.g. SIFT features are 128-D), because the classifier has no mechanism to learn which dimensions are irrelevant for the decision. More sophisticated classification schemes are usually used to obtain better results (but you are not required to do so).

Part 1 – Acquiring Appropriate Datasets (Total points: 10)

For this project, you have to acquire two separate sets of images using your own cameras: a training set and a testing set. You will only use the training set for generating the SIFT- and LBP-based codebooks. The testing set is used to perform recognition.

Training Images

For the training dataset, you will need to acquire fifteen images from four different subjects[1] as shown in Fig 1 (total of 60 images). There are five different poses and three different scales. Note that in all cases, the pose variations are pure yaw rotations. Try to use a uniform white background while obtaining the images. Display your training images in the report, along with their scales and poses (**5 points** for reasonable acquisition parameters).

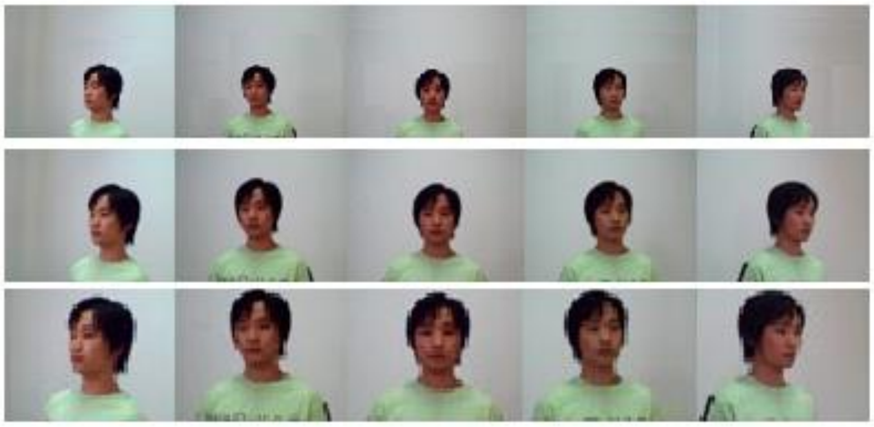


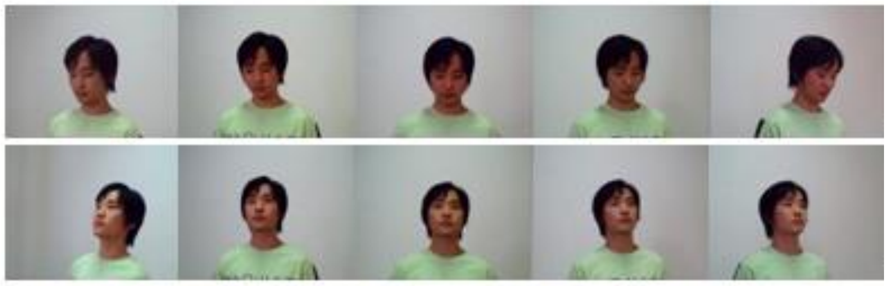
Fig 1: Training images[2]. Each row illustrates the same head pose in a different scale (First row: small scale; second row: normal scale; third row: large scale). Each column illustrates same head pose: (a) Head yaw rotation $\sim -45^\circ$. (b) Head yaw rotation $\sim -30^\circ$. (c) Frontal pose. (d) Head yaw rotation $\sim 30^\circ$. (e) Head yaw rotation $\sim 45^\circ$.

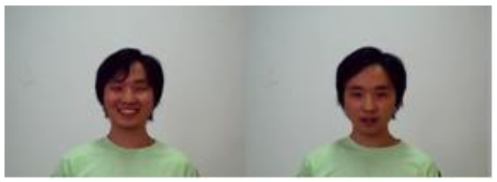
Testing Images

To assess the performance of the bag of words method in recognizing faces, you will need to acquire new, "unseen" images of the same subjects under different conditions (e.g. different head pose, different facial expression, and different accessories). The test dataset consists of fourteen images of each subject as shown in Fig 2 (total of 56 images). Note that all the test images should be acquired at the main scale. Fig 2 (a) includes new head pose variations; Fig 2 (b) illustrates new facial expressions; and Fig 2 (c) shows new accessory variations. Each of the above variations will be used to estimate the recognition rate of the bag of words method for unseen images. Display

[1] If you are in a group of three, ask another person to joint you in creating the datasets.
[2] All sample images are from http://mla.sdu.edu.cn/hmt/face.jpg

your testing images in the report, along with their scales and poses (**5 points** for reasonable acquisition parameters).


(a)


(b)

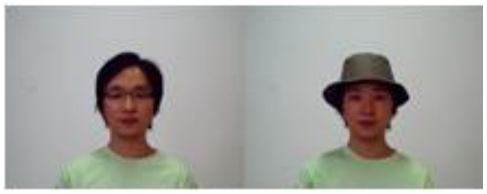

(c)

Fig 2: Testing images. All images are in the normal scale. (a) Pose variations. In addition to the yaw rotation of the training set, there a $\sim 30^{\circ}$ pitch rotation. (b) Expression variations (happy and surprised). (c) Accessory variations.

Part 2: Creating a Visual Codebook (Total points: 20)

For this project, you will generate two separate codebooks, one using SIFT and one using LBP. To create each codebook, please do the following:

1. For all the training images, extract features and compute descriptors from the face/head area as follows:
    - Manually define a rectangular bounding box around the face/head area on each image.
    - For SIFT features, detect all the key points in your images and then discard the features that are outside the face/head bounding box (the features that would not help identifying a person). Display the SIFT features on 10 selected training images (**5 points**).
    - For LBP, divide the face/head bounding box into $W \times H$ windows (e.g. 10x10). You will need to decide on W and H. For each window region, extract the LBP feature vectors and create a histogram of them to obtain the region descriptor.

Display 3 examples of resulting region histograms for one training image (**5 points**).

2. For all of the extracted descriptors in all of the training images (for each codebook):
    - Use K-means clustering in the descriptor space and find the cluster centroids; i.e. the code words.
    - Discretize the descriptors to the code words, using nearest neighbor and L2 distance as the metric.
    - Represent each training image with a histogram of code words. Display the histogram of code words for 2 training images using SIFT and LBP features (**10 points).**

Part 3: Recognition Results (Total points: 40)

To recognize a test image as being one of the trained face images, do the following steps after obtaining the codebooks:

1. Manually define a rectangular bounding box around the face/head area on the test image.
2. Extract features and descriptors in the test image and then do the following:
    - For the SIFT codebook, detect all the key points in the test image and then discard the features that are outside the face/head bounding box.
    - For the LBP codebook, divide the face/head bounding box into $W \times H$ windows. For each window region, extract the LBP feature vectors and create a histogram of them to obtain the region descriptor.
3. Discretize the descriptors to the code words, using nearest neighbor and L2 distance as the metric.
4. Represent the test image with a histogram of code words
5. Classify the test image by finding the nearest neighbor to its histogram of code words. Display the results of classification on 5 test images (**5 points**).

To evaluate the recognition performance of the method, you will use two quantitative measures of performance: (a) recognition rate and (b) confusion matrix for pose estimation. To compute the recognition rate, you first find the class label of each test images and compare that with the ground truth. Then, the recognition rate could be obtained by $\frac{number\ of\ correct\ recognitions}{total\ number\ of\ test\ images} \times 100$.

Since the training images contain pose variations, we can also find a pose label for the test images in addition to finding the subject label. Note that the pose label for each test image would be the pose label of the corresponding closest training image. Since we have five different training poses, the confusion matrix for pose estimation would be a 5×5 matrix, with its ijth element illustrating the normalized number of times the ith pose is classified to the jth pose. Note that each row of the matrix sums to one. Note that a perfect pose estimation would result in a diagonal confusion matrix.

To evaluation the performance of each codebook, follow these steps:

- Do the following for each code book
  - For $K = 5, 10$ and 20, do the following
    - Use the training images in both training and evaluation: (**15 points**)
      1) Compute the recognition rate of the method.
      2) Compute the confusion matrix for pose estimation.
    - Use the training image for training and the testing images for evaluation: (**15 points**)
      3) Compute the recognition rate of the method.
      4) Compute the confusion matrix for pose estimation.
- Determine the best performing codebook along with its number of code words. Use only the result of recognition rate for unseen images in your reasoning. (**5 points**)

Part 4: Face Tagging (Total points: 20)

For this part of the assignment, you will use your best performing face recognition codebook along with a face detector to tag faces in group photos. You can use Matlab or OpenCV built-in face detector to find all the faces in the images. Display a bounding box around each face (**10 points**). After detecting and locating all faces in the input image (note that there might be some undetected faces and some misdetections), you can then treat each detected face as a separate image and use your face recognition method to find the label for each face and print the name beside the head on the image.

For this part, you should obtain five different group photos (each image should contain all the trained subjects). Try to have pose variations (note that the face detectors usually fail for large head rotations), facial expressions and scale variations in these images (try to be creative!). Display the original and the tagged images (**10 points**).

Part 5: Organization of Report (Total points: 10)

The report should be organized and easy to read. Marks will be deducted for lack of organization and clarity.

Notes on writing your code

For writing your code, you are free to choose between OpenCV and Matlab (and Matlab MEX interface). Note that you should submit all your images along with your code. Submit a readme file along with you code, report and images and explain the required steps to run your code in there.

If you are using Matlab, your code should be compiled without any error and ready to be used. If you are using Matlab Mex interface, make sure to include all the source files, include file and libraries and a build script so that your Mex functions could be built on a new platform. If you are using C++, include all the source files, header files and library files required to rebuild your code on a new platform. Marks will be deducted if compile-time or run-time errors occurs.

You can use the built-in capabilities of both Matlab and OpenCV environments while doing each step of the project. You cannot, however, use the built-in capabilities to perform the face recognition. If you are using a third party code/library to find features and descriptors (besides OpenCV), specify that on your report and make sure to submit that along with your code.

Your submitted code should be commented and easy to follow. Try to use a distinct function for every part of the code; e.g.

- Main function
  - Initialize variables and load all the training and testing images
  - Training for the SIFT codebook
    - Extracting SIFT features and computing descriptors for training images
    - Clustering the descriptors to find the code words
    - Descriptor discretization
    - Representing each training image with a histogram of code words
  - Training function for the LBP codebook
    - Dividing each training image into $W \times H$ windows
    - Extracting LBP features and computing the region descriptors for each region on each training images
    - Clustering the descriptors to find the code words
    - Descriptor discretization
    - Representing each training image with a histogram of code words
  - Compute the recognition rate for training images for each codebook and each K
  - Compute and display the confusion for pose estimation for training images for each codebook and each K
  - Compute the recognition rate for testing images for each codebook and each K
  - Compute and display the confusion for pose estimation for testing images for each codebook and each K
  - Face tagging
    - Load group images
    - Face detection

- Use the best codebook to classify each detected face
- Label faces and generate the output images