CSE 152 Assignment 2
Winter 2019
Due Fri, Mar 1, 5pm PST

Instructions :

- Attempt all questions.
- Please comment all your code adequately.
- In this assignment, you are required to use ipython notebook (`https://jupyter.org/`) to run your code.
- Please install following packages in order to run the code: OpenCV, matplotlib, scikit-learn
- There are two .ipynb files (`hw2_stereo.ipynb`, `hw2_bow.ipynb`) which contain codes for problem 1 and 2 respectively.
- You must submit a ZIP file containing two .ipynb files and a PDF file containing your ipython notebook codes and results. You have to export ipython notebooks to a PDF which can demonstrate your code and figures clearly.
- You must submit both files (.pdf and .zip) on Gradescope. You must mark each problem on Gradescope in the PDF.
- Please write your code at the "WRITE YOUR CODE HERE" prompt in the .ipynb files.

1 Disparity from Stereo Pair Images

In this problem, we will implement SSD (Sum Squared Distance) and NCC (Normalized Cross Correlation) and use them to calculate disparity from stero pair images.

(i) SSD and NCC Matching: In this part, you have to write two functions ssd_match and ncc_match that implement the computation of the matching score for two given windows with SSD and NCC metrics, respectively. **[5 points]**

(ii) Calculate disparity for stereo pair images: Now you have to implement stereoMatching to calculate disparity from given two sets of stereo pair images. You can use the plane sweep algorithm mentioned in class. **[5 points]**

(iii) Visualize disparity map: Now it's time to visualize the disparity result of stereo matching. You have to show your results with different parameters, that is, window sizes: 3, 7, 11 and matching metrics: SSD and NCC. Finally, you have to show the results for each pair of parameters. Your outputs should look similar to the provided result figures. **[5 points]**

2 Image Classification Using Bag-of-Words

2.1 Review of bag-of-words for image classification

We will solve image classification, with the relatively simple but effective bag-of-words approach. We treat an image as a set of regions and ignore the spatial relationships between different regions. The image classification problem can then be solved by counting the frequencies of different regions appearing in an image. Image classification using bag-of-words includes the following steps:

- **Region Selection:** Select some regions in the images so that we can extract features from those regions in the next step. The regions can be selected by feature detection methods like edge or corner detection, or you can just uniformly sample the image.
- **Feature Extraction:** We extract features from the selected regions. One commonly used feature is SIFT. We extract features from every image in the training set. These features are collected to compute the visual vocabulary for image representation.
- **Building visual vocabulary:** Once we have the features extracted from training images, we build a visual vocabulary by grouping them together to form a few clusters. We will use the k-means algorithm to group the features. The reason why we need this step is to reduce the redundancy in feature space and have a more concise feature representation of images.
- **Learning and recognition:** Given the visual vocabulary, each training image is represented by a histogram, where the features in the image populate bins that correspond to the visual word closest to them. Thereafter, we will use k-nearest neighbors to perform classification for a test image, also represented by a histogram using the same vocabulary.

2.2 Simple face classifier

We will make a classifier to tell whether there is a face in a given image. The dataset contains 200 images with faces and 200 images without faces. We will pick 100 images from each group for training and the other 100 images for testing. The skeleton code is given.

(i) **Extract interest points from image**: You will now try two methods for this: [**2 points**]

  (a) Uniformly sample the images. You can divide the image into regular grids and choose a point in each grid and then uniformly choose `nPts` number of interest points.

  (b) Sample on corners. First use the Harris Corner detector to detect corners in the image and then uniformly choose `nPts` number of interest points. (This has already been implemented for you as an example.)

(ii) **Extract features**: You are required to try two kinds of features: [**2 points**]

  (a) SIFT feature. Here we use the SIFT implementation in OpenCV package (this has already been implemented for you as an example).

  (b) Image Patch feature. Extract a small image patch around each feature point. You can decide the size of each patch and how many pixels it should cover on you own.

(iii) **Build visual vocabulary**: Use k-means clustering to form a visual vocabulary. You can use the python k-means package. You can decide the number of clusters yourself. The default number of cluster centers in k-means is 50. You can try different metrics to do k-means clustering and see which gives the best results. [**2 points**]

(iv) **Compute histogram representation**: Compute the histogram representation of each image, with bins defined over the visual words in the vocabulary. These histograms are the bag-of-words representations of images that will be used for image classication. **[2 points]**

(v) **K-nearest neighbor classifier**: After building the visual vocabulary, we now do image classication using the nearest neighbors method. Given a new image, first represent it using the visual vocabulary and then find the closest representation in the training set. The test image is assigned the same category as its nearest neighbor in the training set. Next, to make the algorithm more robust, find the first K-nearest neighbors (for K = 3 and 5). You may try different distance metrics when doing K-nearest neighbors search. **[4 points]**

(vi) **Calculate testing accuracy**: Using the settings above that give the best accuracy to report results as the following 2D table. You should report the accuracy of each method on both positive and negative testing samples. Some of the methods may have poor accuracy. But that is fine, don't worry too much about accuracy. You will get full credit as long as you can correctly implement and reason about the various methods. **[8 points]**

	Uniform Sampling		Edge Detection	
	Positive	Negative	Positive	Negative
SIFT Feature				
Image Patch				

(vii) **Explain your implementation**: Briefly explain how you implemented the methods and which parameters you find to have a significant impact on the final performance. **[5 points]**