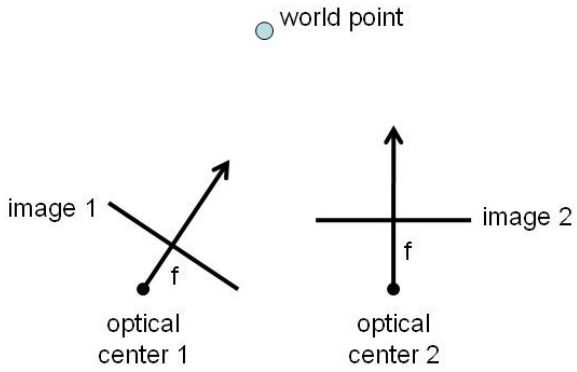
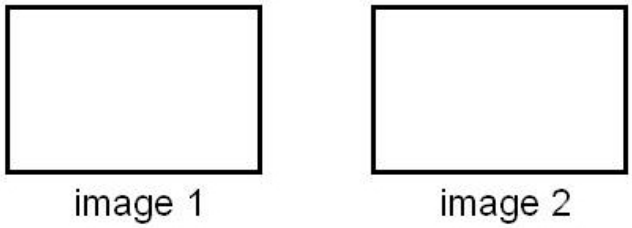
CS 378 Computer Vision
Problem Set 3
Out: Tuesday, Oct 28
Due: Tuesday, Nov 11, 11:59 PM

I. Short answer problems: stereo [30 points]

1. This figure depicts the top view of a stereo rig. *f* denotes the focal length. Considering this camera configuration, determine where the epipolar lines would fall in each view for various 3d world points.



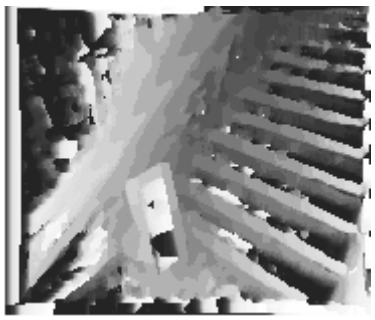
Annotate each of the views for these cameras with at least 3 epipolar lines, and mark any visible epipoles.



2. Consider the image on the right. This is the left view of a stereo pair. Explain two things specifically about this example that may cause errors in the computed disparities when using a window-based dense matching approach.

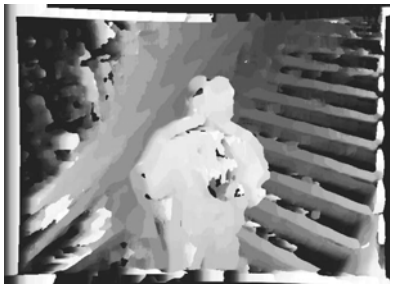
3. The example images below show one view of a stereo pair (left) and the resulting disparity map (right). Black values in the disparity map denote positions with invalid disparity values, where the window-based dense stereo algorithm failed.

Left image of stereo pair



Disparity map

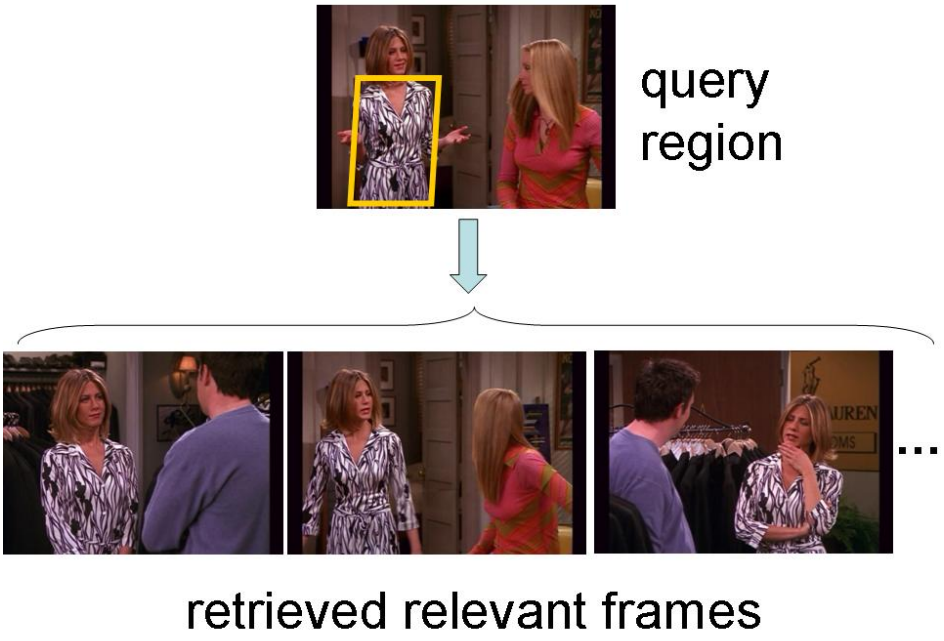
Left image of stereo pair



Disparity map

Design a simple method to detect foreground objects *using only depth information obtained from a stereo camera*. You can assume the stereo camera will be stationary and fixed on a particular scene (e.g., this stairwell), and the operator of the system will be able to collect some initial video while the scene contains no foreground objects. When running, the input to the system will be a stereo image pair, and the output should be a single image showing the scene and marking pixels on objects believed to be foreground. Write down a concise summary of the steps needed to do this.

II. Programming problem: Video search with bags of visual words [70 points]



For this problem, you will implement a video search method to retrieve relevant frames from a video based on the features in a query region selected from some frame. Use the Video Google method of Sivic & Zisserman as a model. (The paper is linked from the class web page.)

You can access all the video data and SIFT features, as well as some provided Matlab code from the readable CS directory here: **/v/filer4b/v26q003/pset3data/**. The provided data files include both the original images (/v/filer4b/v26q003/pset3data/frames/, in .jpeg files) and their associated detected SIFT descriptors (/v/filer4b/v26q003/pset3data/sift/, in .mat files). The provided code files are in /v/filer4b/v26q003/pset3data/code. Note that you should copy over the .m files to your local directory, but you do not need to copy the image or .mat files. There is a lot of data, and it is best to point to the above paths directly in your code.

Each .mat file corresponds to a single image, and contains the following variables, where *n* is the number of detected SIFT features in that image:

descriptors	*n*x128	double	// the SIFT vectors as rows
imname	1x57	char	// name of the image file that goes with this data
numfeats	1x1	double	// number of detected features
orients	*n*x1	double	// the orientations of the patches
positions	*n*x2	double	// the positions of the patch centers
scales	*n*x1	double	// the scales of the patches

First make sure that you can read these variables in a script, and display the images and the features [see the provided `displaySIFTPatches.m` function to draw specified patches onto an image].

The basic framework will require these components:

- **Form a visual vocabulary.** Cluster a large, representative random sample of SIFT descriptors from some portion of the frames using k-means. Let the k centers be the prototype visual words. The value of k is a free parameter; for this data something like k=500 should work. [see Matlab's `kmeans` function, or provided `kmeansML.m` code]

- **Map a raw SIFT descriptor to its visual word.** The raw descriptor is assigned to the word vector (cluster center) it is nearest to in terms of Euclidean distance. [see provided `dist2.m` code for fast distance computations; note that `[minvals, mininds] = min(D, [], 2);` returns a vector containing the minimum value per row of the matrix D along with the column indices where each of the mins are found.]

- **Map an image's features into its bag-of-words histogram.** The histogram for image I_j is a *k*-dimensional vector: $F(I_j) = \left[freq_{1,j}, freq_{2,j}, \ldots, freq_{k,j} \right]$, where each entry $freq_{i,j}$ counts the number of occurrences of the *i*-th visual word in that image, and *k* is the number of total words in the vocabulary. In other words, a single image's list of *n* SIFT descriptors yields a *k*-dimensional bag of words histogram. [Matlab's `histc` is a useful function]

- **Compute the tf-idf weighting for a bag of words histogram.** Given the raw frequency counts of each word (the histogram *F* in the bullet above), re-weight the entries based on the term frequency-inverse document frequency approach. Watch out for division by zero.

- **Compute similarity scores.** Compare two bag-of-words histograms using the normalized scalar product. Compute the inner product between two weighted histograms, and normalize by the product of their norms.

- **Sort the similarity scores** between a query histogram and the histograms associated with the rest of the images in the video. Pull up the images associated with the *M* most similar examples. [see Matlab's `sort` function]

- **Form a query from a region within a frame.** Select a polygonal region interactively with the mouse, and compute a bag of words histogram from only the SIFT descriptors that fall within that region. Weight it with tf-idf. [see provided `selectRegion.m` code]

After implementing these components, write programs to do the following, and display and report on the results.

1. **Visualizing the vocabulary [20 pts]:** Display example image patches associated with three of the visual words. Choose three words that are distinct to illustrate what the different words are capturing, and display enough patch examples so the word content is evident (e.g., say 40 patches per word displayed) Explain what you see. [See provided `getPatchFromSIFTParameters.m` for mapping the saved parameters to the extracted patch from the original image.]

2. **Full frame queries [20 pts]:** Choose 10 different frames from the entire video dataset to serve as queries. Display the *M*=8 most similar frames to each of these queries (in rank order) based on the normalized scalar product between their bag of words histograms. Run the queries both with the tf-idf weighting and without; compare and explain the results.

3. **Region queries [30 pts]**: Select queries regions from within 10 frames (which may be different than the 10 used above) to demonstrate the retrieved frames when only a portion of the SIFT descriptors are used to form a bag of words. *Include examples where the same object is found in the most similar M frames but amidst different objects or backgrounds, and also include one or two failure cases*. Explain the results, including possible reasons for the failure cases.

III. OPTIONAL : Extra credit (up to 10 points each, max 20 points total)

- **Inverted file index**. Implement an inverted file index to retrieve the frames. Compare the efficiency of the linear scan search used above, and the retrieval when using the index.

- **Spatial consistency**. Implement some form of a spatial consistency check to post-process and re-rank the shortlist produced based on the normalized scalar product scores. Demonstrate a couple query examples where this improves the results.

- **Stop list**. Implement a stop list to ignore very common words. Discuss and illustrate the impact on your results.

Submission instructions: what to hand in

Create a tar file with

- Your documented Matlab code. Include your name at the top of each file in a comment.

- A **single pdf** file containing the following

    - Your name at the top

    - Your answers to Part I, numbered.

    - A brief explanation of your implementation strategy for Part II: a short paragraph or two describing in English what you have computed.

    - Your responses and image results for each question in Part II, numbered. Insert image figures in the appropriate places.

    - (optional): any results and descriptions for extra credit portions. For max credit explain what you implemented and also interpret the results.

1. Name the tar file with <username>_pset3.tar and submit via the turnin program: `turnin --submit harshd pset3 <filename>`

2. Also submit a hardcopy of the pdf file. You can submit the hardcopy in class. Otherwise, leave it at CSA 114.