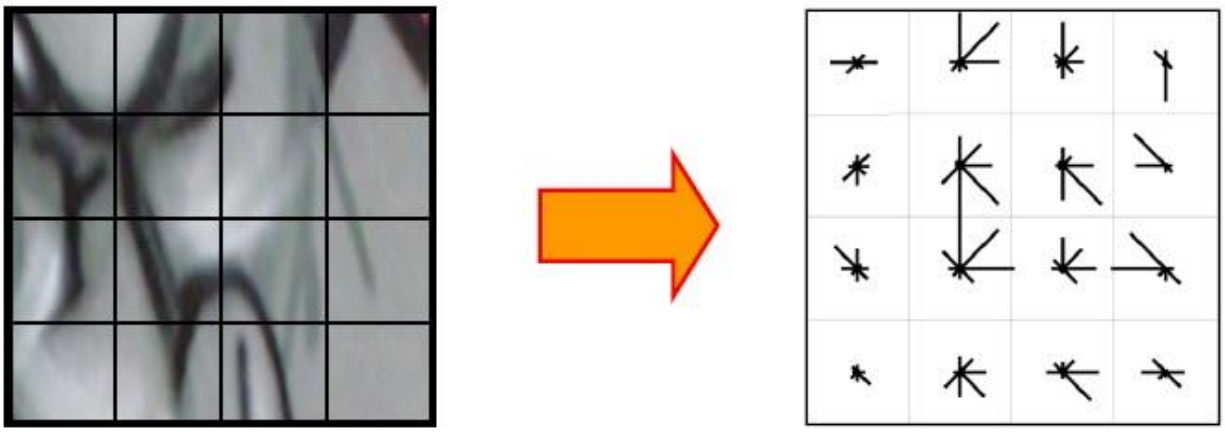
PS3 Solutions
Prepared By Ahsan Abdullah and Vivek Dubey

**SHORT ANSWER PROBLEMS**

**1.** While performing interest point detection using Laplacian of Gaussian, if we choose: (a) any local maxima in just scale-space while not also considering the maxima in position, then we may detect many interest points in a local spatial region in this image. This would make each interest point less distinctive, since each of these interest points could produce good matches to many different interest points. (b) Whereas if we choose interest points in any positions whose response to some filter exceeds a certain threshold, this could make the interest points less repeatable, since the same image content may not be detected in different images since the threshold will be different for different images (for example, the threshold may be too high for the same interest point to be detected in one of the images). This would also make the points less distinctive, for a similar region as in (a).

**2.** Since we already know the keypoint(with its assigned orientation, scale and location), the SIFT descriptor is used as a descriptor that's highly distinctive and yet invariant to the remaining variations. Each value recorded in a 128 dimensional SIFT keypoint descriptor corresponds to one of 8 quantized gradient directions in one of the 4x4 spatial bin.



Source: https://sensblogs.wordpress.com/2011/08/23/quick-reviews-on-local-descriptors-sift-and-single-object-recognition-by-fei-fei-li/

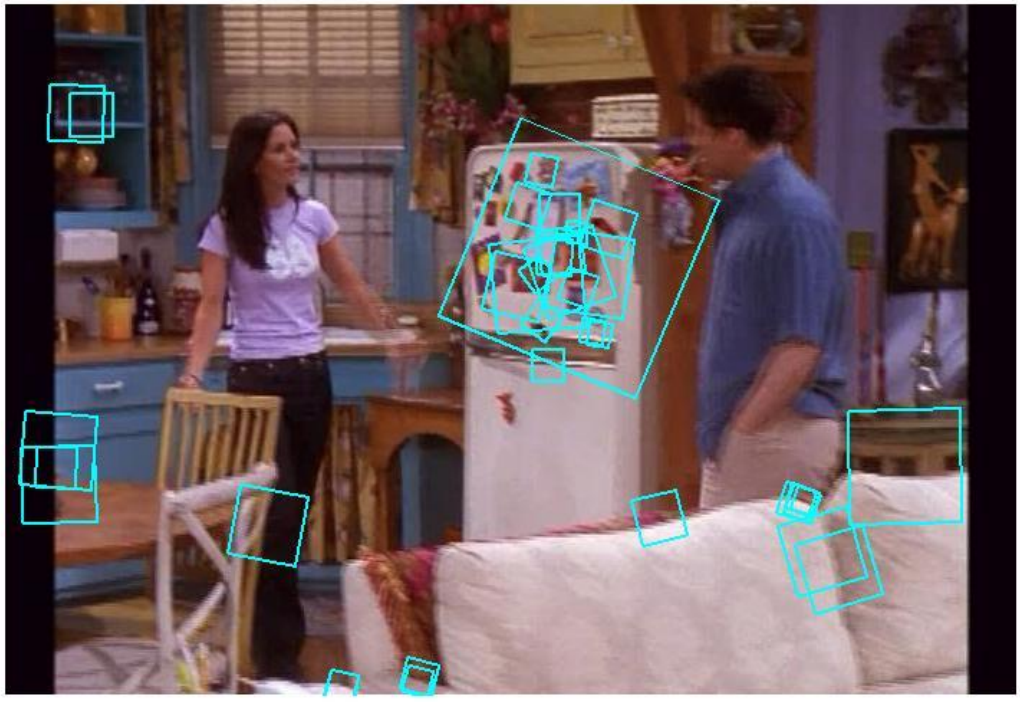
**3**. The Hough Parameter Space would be 4 dimensional. Generalized Hough Transform is a voting scheme. Each element in the SIFT descriptor space should vote for keypoints with following parameters: x position, y position, scale and orientation. This is analogous to each point in image space voting for the centers in the center parameter space(x and y positions of the center).

**PROGRAMMING: VIDEO SEARCH WITH BAG OF VISUAL WORDS**

## 1. Raw Descriptor Matching

```
clc;clear;close all;
% Load given files and add required directories to path
addpath('provided_code\');
load twoFrameData

% Getting SIFT Descriptor indices lying inside selected region
discInds = selectRegion(im1, positions1);

% Finding distance between selected region SIFTs and other image SIFTs
distMat = dist2(descriptors1(discInds, :), descriptors2);

% Find minimum distance for each sift descriptor in image2
distMat = min(distMat);

% Descriptors with distances below a threshold are considered matches
matchedInds = find(distMat < 0.15);

% Showing Matched Patches
figure,
imshow(im2)
displaySIFTPatches(positions2(matchedInds, :), scales2(matchedInds),
orients2(matchedInds), im2);
```

The idea is to get the indices of the descriptors that lie in the region that is selected by the user through selectRegion.m and calculate the distance of those descriptors from all descriptors in frame 2. Once we get those, we can take the values above a certain threshold and display them. As the results show, there were a lot more descriptors that what would have been accurate but since SIFT descriptors carry no semantic meaning, this result is reasonable.



## 2. Build Visual Vocabulary and Show patches belonging to Words

### Step 1. Generate and Save Visual Vocabulary

```
clc;clear;close all;

% Add required folders to path
addpath('provided_code\');
addpath('sift\');
addpath('frames\');
siftDir = dir('sift\'); siftDir = siftDir(3:end);
framesDir = dir('frames\'); framesDir = framesDir(3:end);

% initialize matrices
allDescriptors = [];
allPositions = [];
allScales = [];
allOrients = [];
imID = [];

for i = 1:numel(siftDir)
    load(siftDir(i).name);

    % Randomly sampling at least 20 SIFT descriptors for each frame
    numDescsToSample = min(20, size(descriptors,1));
    descInds = randperm(numDescsToSample);

    % Saving all the required information needed to compute Visual Words
    % and showing Word - Patch correspondence
    allDescriptors = [allDescriptors; descriptors(descInds, :)];
    allPositions = [allPositions; positions(descInds, :)];
    allScales = [allScales; scales(descInds, :)];
    allOrients = [allOrients; orients(descInds, :)];
    imID(end+1: end+numDescsToSample) = i;
end

% K - Means to get 1500 visual words
[membership, kmeans, ~] = kmeansML(1500, allDescriptors');
kmeans = kmeans';

% Saving matrices into kmeans.mat
save kmeans.mat membership kmeans allPositions allScales allOrients imID
```

### Step 2. Load Visual Vocabulary and show Word – Patch Correspondence

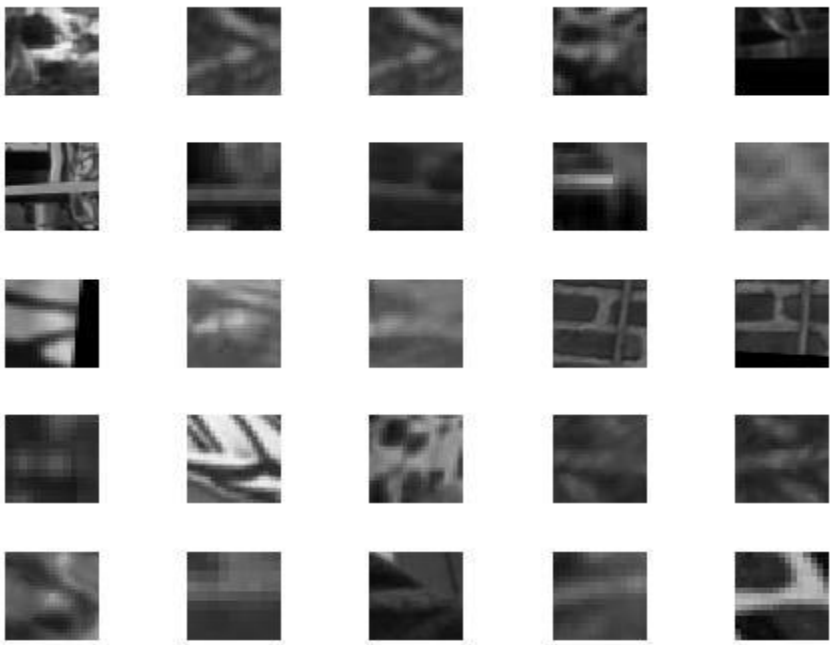
```
clc; clear; close all;

% Load kmeans.mat having Visual Words and other information
load kmeans

% Add required Directories to path
addpath('provided_code\');
addpath('sift\');
```

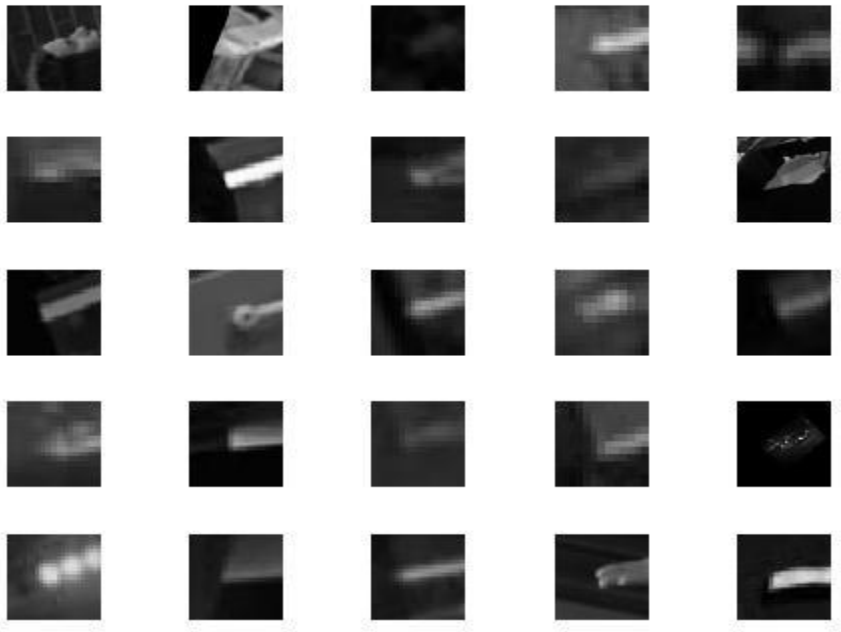
```
addpath('frames\');
siftDir = dir('sift\'); siftDir = siftDir(3:end);

% Randomly Choose 2 Visual Words
randWords = randperm(size(kmeans, 1), 5);

for i = randWords
    % Find 25 SIFT features corresponding to this visual word
    figure,
    matchingDescInd = [];
    numPlotted = 0;
    randDataPerm = randperm(numel(siftDir));
    % From each image select patches that correspond to selected word. Keep
    % iterating over images until we find 25 matches.
    for j = randDataPerm
        try
            load(siftDir(j).name)
        catch e
            continue;
        end
        % Getting membership of each descriptor to corresponding word
        distMat = dist2(kmeans, descriptors);
        [~, minInd] = min(distMat);
        % Getting patches corresponding to random word i
        matchingDescInd = find(minInd == i);
        for k = matchingDescInd
            numPlotted = numPlotted + 1;
            im = imread(imname);
            patch = getPatchFromSIFTParameters(positions(k,:), scales(k), ...
                orients(k), rgb2gray(im));
            subplot(5,5,numPlotted)
            imshow(patch)
            if numPlotted == 25
                break;
            end
        end
        if numPlotted == 25
            break;
        end
    end
end
```

In Problem 2, we sampled out 20 descriptors from each frame available. We had a total of 131733 SIFT descriptors in our 'allDescriptors' matrix. After this, we clustered all these descriptors into 1500 clusters using the k-means algorithm, using the efficient implementation provided to us in kmeansML.m. The cluster centers that we obtained as a result are our visual words. We will use these words in future to represent any frame in its bag-of-words form.

To find 25 patches corresponding to a visual word we go to each image, find sift patches belonging to that word and display the patches. We stop as soon as we find the desired number of patches.

Another way to visualize the 25 patches belonging to a visual word is that we use allDescriptors that we had constructed earlier. Along with all descriptors, we also had their respective positions, scales and orientations stored too, in matrices ‘allPositions’ etc. We can then sample out 25 sift patches for each word.





### 3. Full Frame Queries

```
clc;clear;close all;

% Load kmeans.mat having Visual Words and other information
load kmeans

% Add required Directories to path
addpath('provided_code\');
addpath('frames\');
addpath('sifts\');
framesDir = dir('frames\'); framesDir = framesDir(3:end);
siftDir = dir('sift\'); siftDir = siftDir(3:end);

% Select 3 random query frames
randQueryFrames = randperm(size(framesDir, 1), 3);

for i = randQueryFrames
    load(siftDir(i).name);

    % Get normalized Bag of Words (BOW) histogram for query image
    queryImBOWhist = getBOWHistForIm(descriptors, kmeans);

    % Randomly sample some images out of which the best matches are to be
    % found. Ideally we should search through all frames.
    randTestFrames = randperm(size(framesDir, 1), 1500);
    allTestImBOWhist = [];
```

```
    for j = randTestFrames
        load(siftDir(j).name);
        testImBOWhist = getBOWHistForIm(descriptors, kmeans);
        % concantenate normalized BOW histograms for test images
        allTestImBOWhist = [allTestImBOWhist; testImBOWhist];
    end

    % Find distance between query image histogram and all test image
    % histograms
    distVec = dist2(queryImBOWhist, allTestImBOWhist);
    % Sort Results
    [~, sortedInd] = sort(distVec);
    % Get best five matches
    matchedFrames = randTestFrames(sortedInd(1:5));

    % Plot all of them
    figure,
    subplot(2,3, 1)
    imshow(imread(framesDir(i).name));
    for j = 1 : 5
        subplot(2,3, j+1)
        imshow(imread(framesDir(matchedFrames(j)).name))
    end
end

function bowhist = getBOWHistForIm(imdescs, words)
    % The function returns BOW normalized histogram for the given image
    % SIFT descriptors.

    distMat = dist2(words, imdescs);
    [~, wordInd] = min(distMat);
    for i = 1:size(words, 1)
        bowhist(i) = numel(find(wordInd == i));
    end
    bowhist = bowhist ./ norm(bowhist);
end
```

For full frame queries, we used the bag-of-words representation of all the frames. For any given frame, we make its bag-of-words histogram using a separate function called getBOWHistForIm.m. So, to compare the best matching frames (among all frames present, excluding the query frame) from a given query frame, we can compare the distances between their histograms. The top 5 most similar frames are displayed in a result as the following.

Note that the distances that we mentioned above are normalized – which means that the histogram vectors for each frame were divided by their corresponding l2-norm. This is a really important step to get good results.

Sample result of a full frame query. The top left image is the QUERY image and rest are the best 5 sorted matches.






## 4. Region Queries

```
clc;clear;close all;

% Load kmeans.mat having Visual Words and other information
load kmeans

% Add required Directories to path
addpath('provided_code\');
addpath('frames\');
addpath('sift\');
framesDir = dir('frames\'); framesDir = framesDir(3:end);
siftDir = dir('sift\'); siftDir = siftDir(3:end);

% Select random query frames
numQueryFrames = 3;
randQueryFrames = randperm(size(framesDir, 1), numQueryFrames);

for i = randQueryFrames
    load(siftDir(i).name);

    % Only the below 3 lines are different from fullFrameQueries.m
    im = imread(framesDir(i).name);
    indsInRegion = selectRegion(im, positions);
    queryImBOWhist = getBOWHistForIm(descriptors(indsInRegion, :), kmeans);
```

```
    % Randomly sample some images out of which the best matches are to be
    % found. Ideally we should search through all frames.
    randTestFrames = randperm(size(framesDir, 1), 1500);
    allTestImBOWhist = [];

    for j = randTestFrames
        load(siftDir(j).name);
        testImBOWhist = getBOWHistForIm(descriptors, kmeans);
        % concantenate normalized BOW histograms for test images
        allTestImBOWhist = [allTestImBOWhist; testImBOWhist];
    end

    % Find distance between selected region histogram and all test image
    % histograms
    distVec = dist2(queryImBOWhist, allTestImBOWhist);
    % Sort Results
    [~, sortedInd] = sort(distVec);
    % Get best five matches
    matchedFrames = randTestFrames(sortedInd(1:5));

    % Plot all of them
    figure,
    subplot(2,3, 1)
    imshow(imread(framesDir(i).name));
    for j = 1 : 5
        subplot(2,3, j+1)
        imshow(imread(framesDir(matchedFrames(j)).name))
    end
end
```

For problem 4, we have to make only one change- instead of making a bag-of-words representation using the query image using all its descriptors, we need to make it using the descriptors that lie in the region selected by the user. The representations of all the other test frames remain the same. Following is one of our results.

Query Region

Region based Query Search Results.