CS 231A Computer Vision (Fall 2012)
Problem Set 3

Due: Nov. 13$^{\text{th}}$, 2012 (2:15pm)

1 Probabilistic Recursion for Tracking (20 points)

In this problem you will derive a method for tracking a point of interest through a sequence of images. In this problem there exists a true location of an object or interest point that you want to estimate in each frame. We observe this true location through a noisy image. These observations will then be used in conjunction with prior knowledge of the general position where our object or interest point is known to lie. Formalizing this description we can write down the following random variables

$x_k \in \mathbb{R}^d$: the ground truth location of our object or interest point at time k
$X_k = [x_1, \ldots, x_k]^T$: history of the ground truth location to time step k
$z_k \in \mathbb{R}^c$: our noisy position measurement at time k of x_k
$Z_k = [z_1, \ldots, z_k]^T$: history of our noisy position measurement of x_k to time step k

To gain our estimate for the position of the object or interest point at time step k we would like to solve for $p(x_k|Z_k)$. In addition, our estimate needs to be computationally efficient, and should be easily updated for when we wish to estimate the position at $k+1$. Thus we will calculate our estimate using a probabilistic recursion. To be able to compute this recursion it can only depend on distributions which are stored in memory. The distributions we have stored in memory are

$p(z_k|x_k)$: our measurement model
$p(x_k|x_{k-1})$: transition model
$p(x_{k-1}|Z_{k-1})$: the result of our recursion at the previous iteration

(a) In this part of the problem we will assume that Z_{k-1} and z_k are conditionally independent given x_k. Using this assumption derive an expression for $p(x_k|Z_k)$ in terms of $p(z_k|x_k)$ and $p(x_k|Z_{k-1})$. This is conditioned solely on our ground truth location at time k and measurements up to and including time $k-1$. Justify each step of your derivation (it should not be more than a few lines).

(b) The $p(x_k|Z_{k-1})$ term prevents our expression from being solely dependent on distributions that are stored in memory. We will now assume that x_k and Z_{k-1} are conditionally independent given x_{k-1}. Now using this assumption write our recursion as an expression which is solely dependent on distributions we have stored in memory. Justify your answer.

Remark: If both our measurement and transition model are Gaussian distributions, there exists a closed form solution for the recursion. In general, the recursion is approximated using numerical techniques known as Monte Carlo methods.

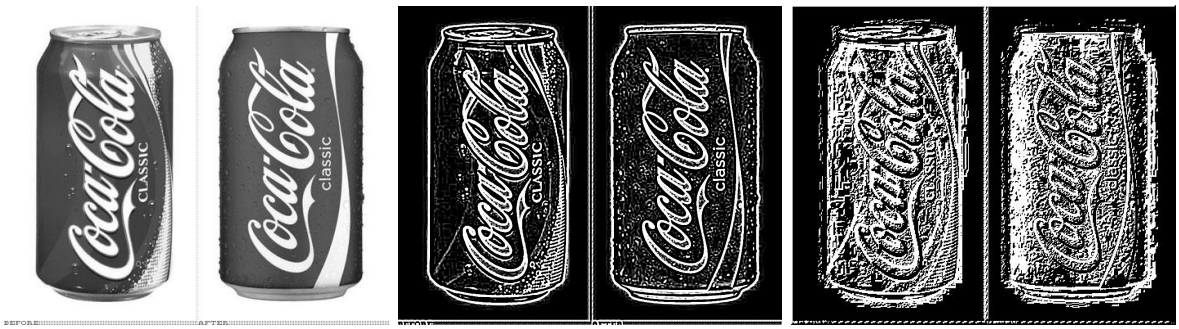
2 Sifting through SIFT (30 points)

SIFT has become a household name for anyone involved with computer vision. However, it is often naively or incorrectly applied when somebody simply downloads a compiled version from the internet and applies it to their task. The purpose of this problem is to illuminate subtleties of SIFT that can be overlooked, while also giving practical experience to the description and matching methods developed by Lowe et al.

(a) It is desirable to locate image features that are stable across image instances and noise variation. Interest points are related to distinct changes in neighboring pixel values, but choosing the correct operator was the focus of many papers. For comparison, consider Fig. 1 which contains an input image, a difference of gaussians (DoG) of the input, and a gradient of the input. If you wanted to detect interest points that produce a stable signal across image variation and random noise, which operator would you use? Specifically, write down the mathematical formulation for both approaches (DoG and gradient) and identify properties that would support your decision. *hint: think about what these kernels look like in the frequency domain.*

(b) A large part of SIFT's immense popularity (10,000+ citations) is its excellent marriage of a keypoint detector and a descriptor. Choosing the keypoints involves many heuristics such as a spatial sampling, removing edge points, and contrast thresholding (described in sections 3 and 4 of the Lowe et al. 2004), whereas the descriptor is calculated in a straightforward manner. In this problem, we will give you a keypoint detector and ask you to create the descriptor. Please download `PS3_data.zip` from the course webpage to complete the rest of this problem.

(i) The keypoint descriptor of SIFT is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint



(a) Original Image (b) Difference of Gaussian of Image (c) Gradient of Image

Figure 1: Different metrics for stable interest points of a given image

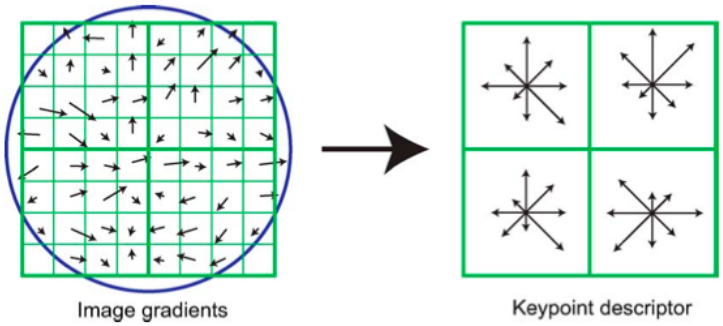


Figure 2: Diagram of keypoint descriptor formation. The blue circle is used to indicate the presence of a gaussian centered at the keypoint.

location, then collecting the results into orientation histograms. Please refer to Section 6.1 Lowe's 2004 Paper for details of implementation. Download `PS3_data.zip` from the course webpage and examine `sift_test.m` in the `PS3Prob2` subdirectory. The interest point detector is given to you, and it is your task to create the 128-d (4×4 array of histograms with 8 orientation bins per histogram) feature vector descriptor for each keypoint.

Follow the step by step instructions in `mySIFTdescriptor.m` to implement your SIFT descriptor and test your code using the given `sift_test.m` with the test image `elmo.jpg`. Print and turn in your `mySIFTdescriptor.m` and the plot of your result SIFT descriptors. The expected output is shown in figure 3.

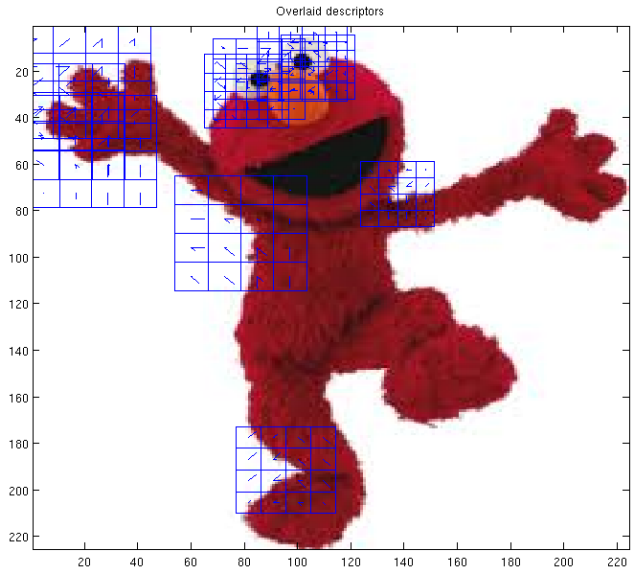


Figure 3: Result image of SIFT descriptors.

(ii) Consider the situation where you are trying to detect a rigid object that has a con-

sistent orientation across object instances (Refer to Fig. 1 for a practical example). What change to the orientation assignment of the original SIFT algorithm could improve detection of the specified object? *hint: what invariances does SIFT have and what are the tradeoffs?*

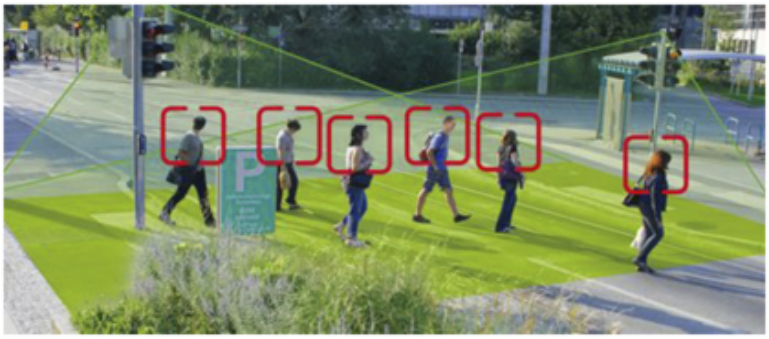


Figure 4: In the example problem of detecting pedestrians with a stationary camera, it is a safe assumption that they will always be of similar orientation.

3 Single Object Recognition Via SIFT (30 points)

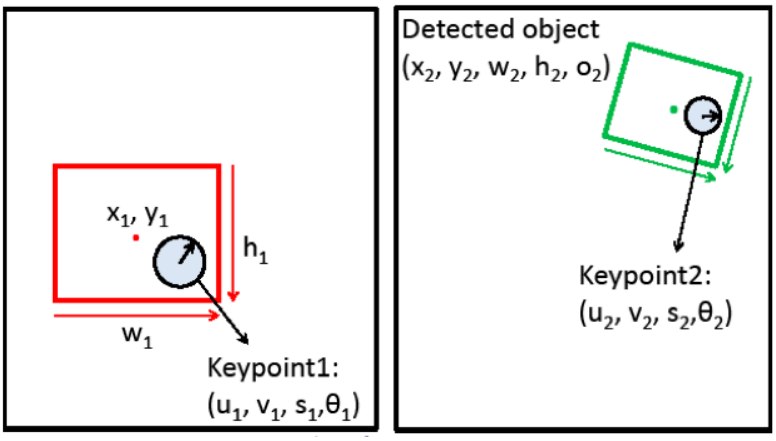
In his 2004 SIFT paper, David Lowe demonstrates impressive object recognition results even in situations of affine variance and occlusion. In this problem, we will explore a similar approach for recognizing and locating a given object from a set of test images. It might be useful to familiarize yourself with sections 7.1 and 7.2 of the paper, which can be found on the course website under the reading. The code and data necessary to solve this problem can be found in `PS3Prob3.zip` on the course webpage.

Figure 5: Sample Output, showing training image and keypoint correspondences.

(a) Given the descriptor g of a keypoint in an image and a set of keypoint descriptors from another image $f_1 ... f_n$, write the algorithm and equations to determine which keypoint in $f_1 ... f_n$ (if any) matches g. Implement this matching algorithm in the given function `matchKeypoints.m` and test its performance using the `matchObject.m` skeleton code. Load the data in `PS3_Prob3.mat` and run the system with the following line:

```
>>matchObject(stopim{1}, sift_desc{1}, keypt{1}, obj_bbox, stopim{3}, ...
 sift_desc{3}, keypt{3});
```

Note that the SIFT keypoints and descriptors are given to you in `PS3_Prob3.mat` file. Your result should match the sample output in Fig. 5. [Turn in your code and a sample image similar to Fig. 5]



(a) Image 1 (b) Image 2

Figure 6: Two sample images for part (b)

(b) Now given an object in an image, we want to explore how to find the same object in another image by matching the keypoints across these two images.

(i) A keypoint is specified by its coordinates, scale and orientaion (u, v, s, θ). Suppose that you have matched a keypoint in the bounding box of an object in the first image to a keypoint in a second image, as shown in figure 6. Given the keypoint pair and the red bounding box in Image 1, which is specified by its center coordinates, width and height (x_1, y_1, w_1, h_1), find the predicted green bounding box of the same object in Image 2. Define the center position, width, height and relative orientation $(x_2, y_2, w_2, h_2, o_2)$ of the predicted bounding box. Assume that the relation between a bounding box and a keypoint in it holds across rotation, translation and scale.

(ii) Once you have defined the five features of the new bounding box in terms of the two keypoint features and the original bounding box, briefly describe how you would utilize the Hough transform to determine the best bounding box in Image 2 given n correspondences.

(c) Implement the function `getObjectRegion.m` to recover the position, scale, and orientation of the objects (via calculating the bounding boxes) in the test images. You can use a coarse Hough transform by setting the number of bins for each dimension equal to 4. Use the line in (a) to test your code and change all the 3's to 2, 4, 5 to test on different images. If you are not able to localize the objects (this could happen in two of the test images), explain what makes these cases difficult. [Turn in your `getObjectRegion.m` and matching result images.]

4 Single Object Matching Via Shape Context (20 points)

Depending on a given problem, some feature descriptors may be better suited than others for object recognition. In contrast to other detectors studied in class, Shape Context [Belongie et al 2002] measures similarity between shapes and exploits it for classification. The methodology also calculates a transformation that would maximally align two potential matches. It will be useful to familiarize yourself with section 3.1 as well as the introduction of section 3 in the paper, which can be found on the class website.

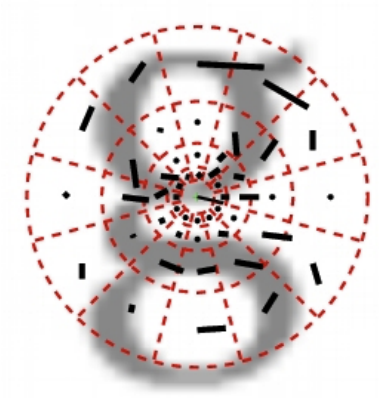


Figure 7: Visualization of polar-coordinate binning

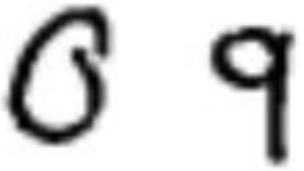
(a) One natural application of Shape Context is to match handwritten digits. In this problem we have provided the data, interest point detection, and matching code for you.

   (i) Write a function `compute_shape_context.m` to generate the shape context feature to be used when matching. Specifically your function should take in the minimum radius, maximum radius, number of bins for radius, number of bins for angle, the input data from the edge detection, the mean radius and outliers. Your code should output the mean radius and the shape context feature for each data point. Write your code in `compute_shape_context.m`. In this file there is a description of the input and output data structures, as well as detailed skeleton code on what to do. This code will take in data from the interest point detector and output data to be used in the matching process for shape context. Run the `test_shape_context.m` script to verify your implementation. Turn in your code for `compute_shape_context.m`.

   (ii) Run the `compare_digits.m` script and turn in the error for each of the 3 matches. The error for each match is the sum of the squared distances. We calculate the error for you and print it out to a figure after running the `compare_digits.m` script.

(b) Now we consider different images and matching scenarios other than handwriting to test how shape context performs.

   (i) We will use preprocessed versions of the images in Fig 9 to test your shape context descriptor. Run the `extend_shape_context.m` script. Turn in the resulting error values and warped figure (figure 3 generated from the code) and indicate what properties of shape context yield this performance.



(a) (b)

Figure 8: Example data from handwritten digit dataset

(a) (b)

Figure 9: Other test images for shape context

(ii) Considering the invariances of Shape Context when would you expect Shape Context descriptor to have poor performance? Give specific circumstances citing the invariances of Shape Context, and use the previous football result as evidence. Given that Shape Context performs well for the digit data, when else would we expect shape context to perform well?