CS676A Introduction to Computer Vision

Assignment 2

Interest Point Based Matching

Saurav Kumar, 12641
Shreyash Pandey, 12683

In this assignment, we have implemented an interest point based matching algorithm, consisting of these 3 steps: 1) Harris Corner Detection to detect interest points, 2) Descriptor (histogram method) to describe interest points, and 3) Matching the interest points based on SSD. 3 sets of images were used in this assignment to test the implementation. We present the overview of our implementation, the experiments we ran, and the results that we obtained.

Part 1: Harris Corner Detector

Implementation

To compute f-value in Harris Corner Detection technique, we needed the Hessian matrix corresponding to the point, which we computed by averaging the gradients in the neighbourhood of the point. We pre-processed the gradient for each point and stored in a matrix, using Sobel Filter[4]. For each point, f-value was computed and corresponding plot of the matrix was formed as image, as shown in 2nd row of each set in Result section of Part 1.

Harris function:

```
for i within x-window:
     for j within y-window:
          Ix  = Gx(i, j)
          Iy  = Gy(i, j)
          H  = H + [Ix^2 IxIy; IyIx Iy^2]
     end
end
H = H / window-size
f = det(H) / trace(H)
```

Gradient using Sobel Filter:

S_x = [-1 0 1; -2 0 2; -1 0 1]
S_y = [-1 -2 -1; 0 0 0; 1 2 1]

G_x = conv2(G, Sx, 'same');
G_y = conv2(G, Sy, 'same');

In the next step, after having obtained f-value for each point, call it matrix F, we now discretize F to get D. To set threshold, we have used the value mean + stddev of all values in F. After obtaining D, we just consider all the local maxima in D as our interest points. Initially, we tried finding local maxima ourselves, but eventually found a nice trick which gives all local maxima very efficiently and quickly.

Discretization Step:

```
D = zeros(width, height)
T = mean2(F) + sqrt(var(F(:)))
for i = 1 : width
    for j = 1:height
        if F(i, j) > T
            D(i, j) = F(i,j);
        end
    end
end
```

Local Maxima Step:

```
InterestPts = D > imdilate(D, [1 1 1;
                               1 0 1;
                               1 1 1])
```

Results

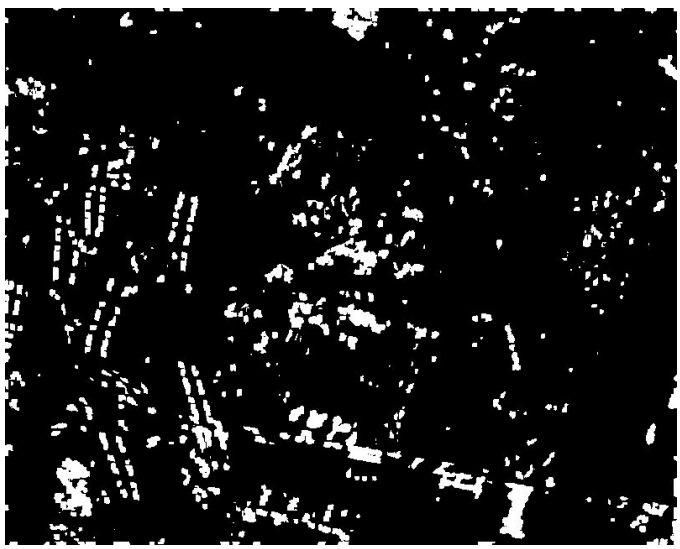
1. Image Set 2 (Order : Original, f-value Image, Interest Points)






2. Image Set 1 (Order : Original, f-value Image, Interest Points)

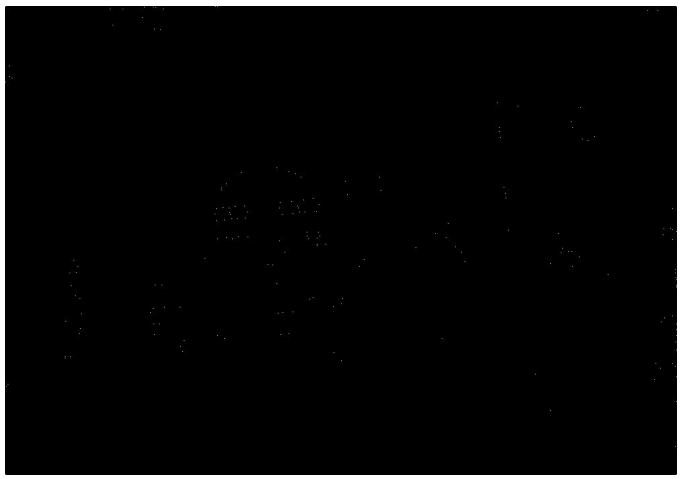


2. Image Set 3 (Order : Original, f-value Image*, Interest Points*)





* For the non blurred image, window size= 6. For the blurred one window size = 12

Part 2: Image Point Descriptor

After obtaining the interest points in the image, the next step is to describe the interest points using vectors which should contain some informations which can be used to match against very similar points. This is done using histograms (16 for this assignment) of gradients for the 16 patches (4 x 4 subpatches). The size of the each patch is one of the parameters that we used for tuning the matching quality.

Implementation

Using the ideas from SIFT technique[1], we assign each keypoint an orientation corresponding to the dominant orientation within the patch. We begin with finding the average gradient of a subpatch, relative to the orientation of the point itself as described in [1]. This orientation (relative) is assigned a bin (one of the 16 bins, corresponding to the histogram). This histogram serves as the descriptor of the interest point.

```
SIFT Descriptor:
M = []
θ0 = atan2(Gy(x,y), Gx(x,y))
for i in x-subPatch
   for j in y-subpatch
      h = zeros(16, 1)
      for k in x-pixels-in-subpatch
         for l in y-pixels-in-subpatch
              θ = atan2(Gy(k,l), Gx(k,l)) -
θ0           if θ < 0 then θ += 2xpi
             binIndex = floor(16xθ/2xpi) +
      1
             h[binIndex] += 1
         end
      end
      M = [M; h/norm(h)]
   end
end
```

We store the resulting descriptors for each interest points and use it to compare with other images. We have varied the size of the patch and obtained the results shown in results section of Part 3. Three values (8, 16 and 32) were used as patch size.

Part 3: Matching

Implementation

Matching step uses the stored descriptors from part 2 above. It takes 2 images and finds the interest points most similar to each interest point in the other image. In order to match the points, we use sum of squared distances (SSD) as score. We use the technique of taking ratio of best and second best score in order to reject false matches. Finally, a python script marks the matching points with circles of same color. Colors are chosen randomly across the interest points to visually distinguish the matching points.

Matching:

```
M = []
for item1 in image1.descriptors
   for item2 in image2.descriptors
      score = getSSDScore(item1, item2)
      if (score < item1.bestScore)
         item1.best2Score = item.bestScore
         item1.best2Item  = item.bestItem
         item1.bestScore  = score
         item1.bestItem   = item2
      end
   end
   if (item1.bestScore/item1.best2Score <
τ )
      M = [M; item1.point item2.point]
   end
end
```

Coloring:

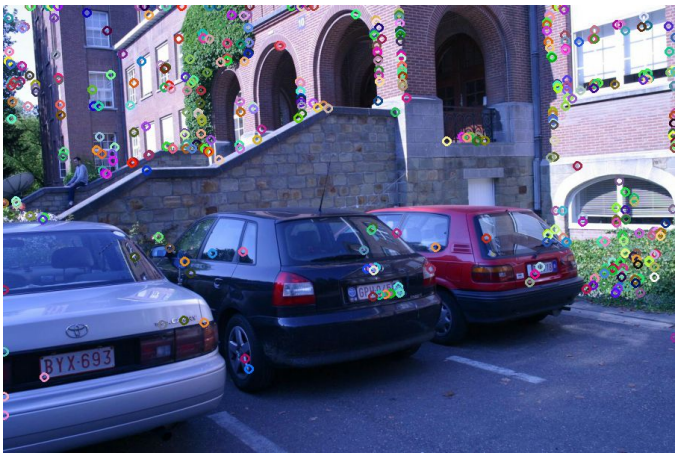
```
img1 = cv2.imread(imageName1)
img2 = cv2.imread(imageName2)

for (p1, p2) in matchList
   color = randomColor()
   cv2.circle(img1, p1, rad, color)
   cv2.circle(img2, p2, rad, color)

def randomColor():
   b = randint(0, 255)
   g = randint(0, 255)
   r = randint(0, 255)
   return (b, g, r)
```

We tried 3 values for the threshold τ, 0.8, 0.85 and 0.9 and obtained the following results.

Results

1. patch size = 8, ratio threshold = 0.8

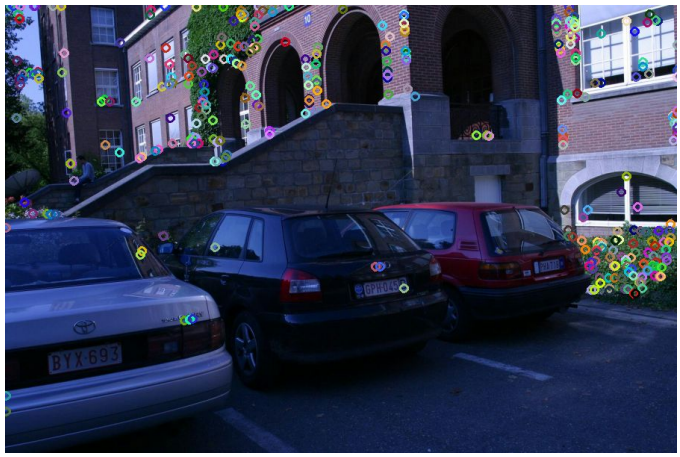

2. patch size = 8, ratio threshold = 0.85




3. patch size = 16, ratio threshold = 0.8

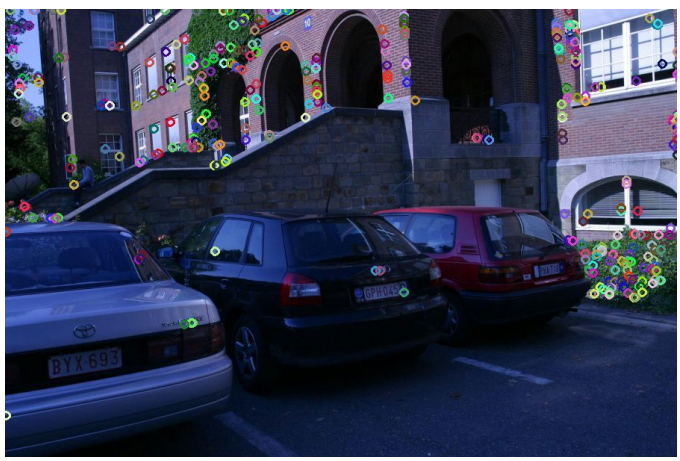

4. patch size = 16, ratio threshold = 0.85


5. patch size = 16, ratio threshold = 0.9

6. patch size = 32, ratio threshold = 0.8


7. patch size = 32, ratio threshold = 0.85

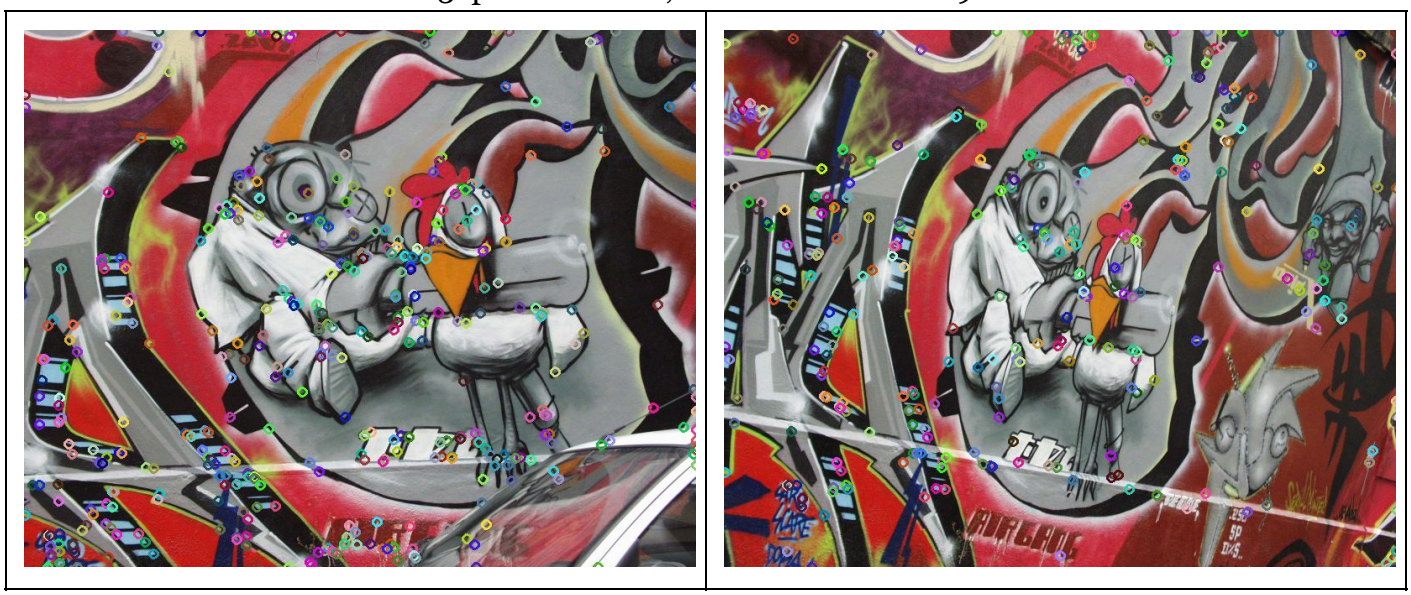



8. patch size = 32, ratio threshold = 0.9


1. patch size = 8, ratio threshold = 0.8


2. patch size = 8, ratio threshold = 0.85

3. patch size = 8, ratio threshold = 0.9

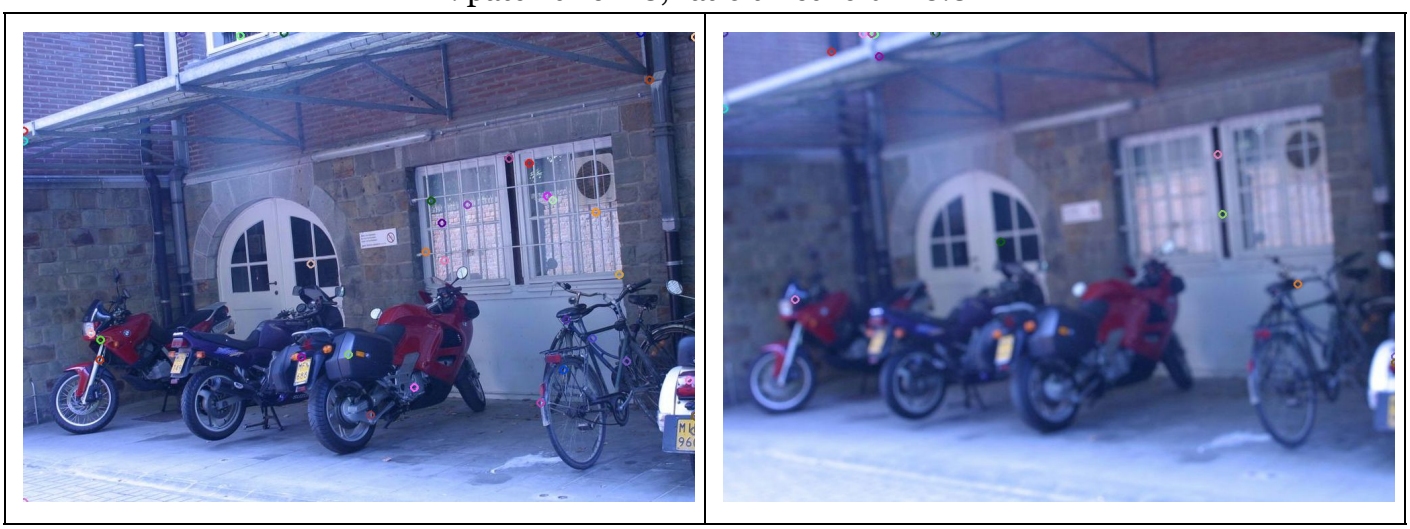


4. patch size = 16, ratio threshold = 0.85

5. patch size = 16, ratio threshold = 0.9

6. patch size = 32, ratio threshold = 0.85

1. patch size = 8, ratio threshold = 0.8

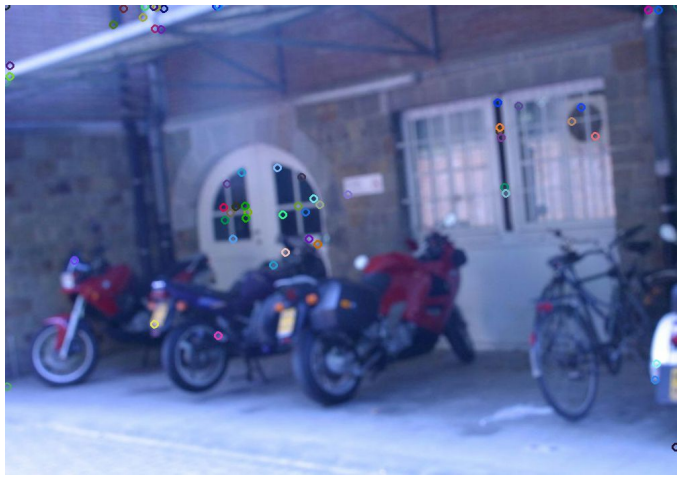


2. patch size = 8, ratio threshold = 0.85


3. patch size = 16, ratio threshold = 0.85


4. patch size = 16, ratio threshold = 0.9


5. patch size = 32, ratio threshold = 0.9

Conclusion

In order to obtain the desired match between interest points in the two images, for various pairs of images, we varied the patch size and threshold of ratio of distances to get the results as shown above. Larger patch-size implies a larger neighbourhood considered for the descriptor of each interest point, implying a more detailed and accurate descriptor. While this adds distinctiveness to the interest point, it also implies that the matching results are fewer but of better quality.
A smaller threshold has the same effect on the matching results. We get fewer number of matches, and the ambiguity is also reduced.
As expected, our algorithm fails to match interest points in the bike image, primarily due to a difference in scale between the two images. Lack of sharpness in the second image can be interpreted as blurring introduced due to scaling up of a scaled down image. This difference in scale can be handled by SIFT algorithm, which we will implement in the next assignment.

References and Acknowledgments

1. http://people.csail.mit.edu/hasinoff/320/sift-notes.txt
2. https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
3. https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
4. https://en.wikipedia.org/wiki/Sobel_operator
5. https://en.wikipedia.org/wiki/Corner_detection
6. Matlab, Mathworks