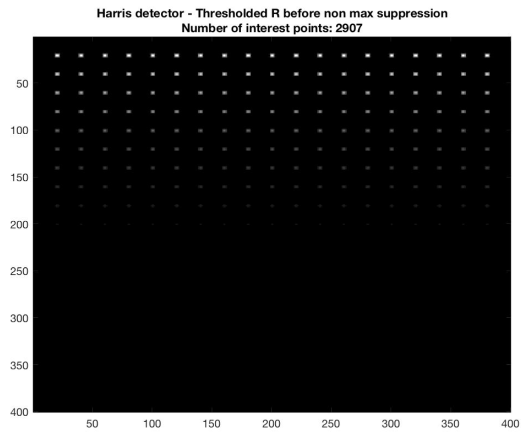
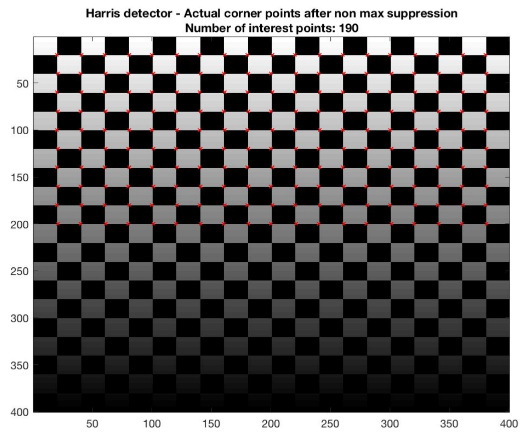
Computer Vision for HCI Abhinav Mahalingam (.15)

CSE 5524

Homework #7

1. Compute and display the Harris pixel-wise cornerness function R values for the image checker.jpg using a) Gaussian window/weighting function with a standard deviation of $\sigma_I = 1$ (use 3σ mask size), b) Gaussian Gx,Gy gradients with a standard deviation of $\sigma_D = 0.7$ (use 3σ mask size), and c) trace weighting factor of $\alpha = 0.05$. Give the values of R(17:23, 17:23) in your report. Next remove small and negative values in R (< 1,000,000). Display the thresholded R using imagesc. Lastly, do non- maximum suppression on R (for this version, keep a location only if a unique maximum is found in a 3x3 region) to identify the actual corner points and display them on the original image. (Note: use double() and not im2double() on checker.jpg) [5 pts]



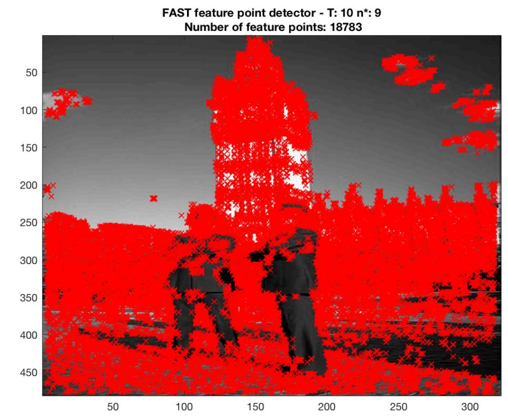


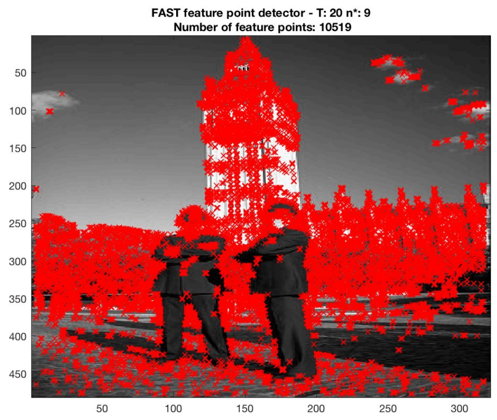
The left image shows the plot of thresholded R before non maximal suppression, where a total of 2907 interest points can be seen. The actual corner points identified by non maximal suppression plotted over the original image can be seen on the right image with a total of 190 corner points (marked by a red + symbol).

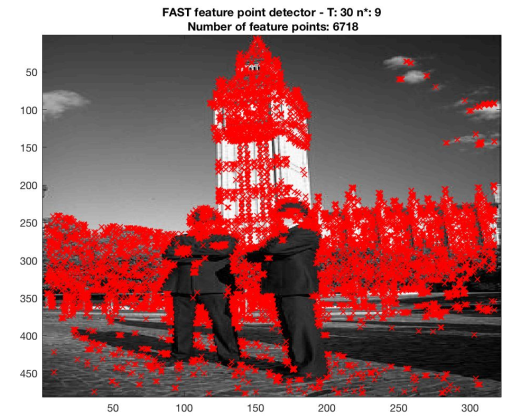
2. Implement the FAST feature point detector using a radius of *r* = 3 (you can hardcode the particular circle border locations), intensity threshold of *T* = 10, and a consecutive number of points threshold of *n** = 9. Run the detector on the image tower.jpg. Display the image and overlay the FAST feature points. Repeat with T = {20, 30, 50} and compare all four results. [6 pts]

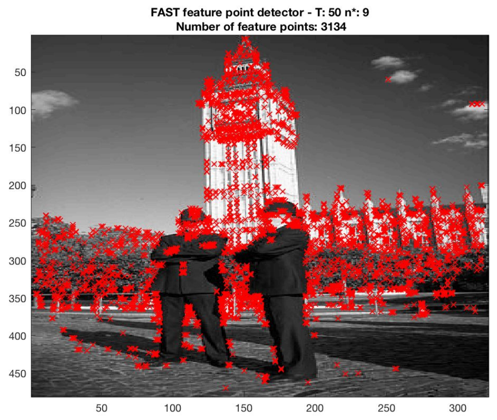
```
figure;
imshow(tower);
hold on;
plot(fastX,fastY,’rx’);
```

```
hold off;
```


FAST feature point detector - T: 10 n*: 9
Number of feature points: 18783


FAST feature point detector - T: 20 n*: 9
Number of feature points: 10519


FAST feature point detector - T: 30 n*: 9
Number of feature points: 6718


FAST feature point detector - T: 50 n*: 9
Number of feature points: 3134

As the value of threshold increases, we can see that the number of feature points detected changes drastically. For the threshold value of 50, we can see that the FAST detector performs reasonably well in detecting unique corner points.

```
% Abhinav Mahalingam
% CSE5524 - HW7
% 10/7/2017
```
HW7.m
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 1
Im = double(imread('input/checker.png'));
sigD = 0.7;sigI = 1;
alpha = 0.05;
[cornerPoints, R] = harris_detector(Im, sigD, sigI, alpha);
imagesc(R);
title(sprintf('Harris detector - Thresholded R before non max suppression\nNumber of interest points: %d',size(R(R>0),1)));
colormap('gray');
pause;
imagesc(Im);
hold on;
plot(cornerPoints(:,1),cornerPoints(:,2),'rx','MarkerSize',3);
hold off;
title(sprintf('Harris detector - Actual corner points after non max suppression\nNumber of interest points: %d',size(cornerPoints,1)));
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Problem 2
Im = double(imread('input/tower.png'));
thresholds = [10 20 30 50];
nStar = 12;
for threshold = thresholds
  featurePoints = fast_detector(Im, threshold, nStar);
  imagesc(Im);
  colormap('gray');
  hold on;
  plot(featurePoints(:,1),featurePoints(:,2),'rx');
  hold off;
    title(sprintf('FAST feature point detector - T: %d n*: %d\n Number of feature points: %d',threshold,nStar,size(featurePoints,1)));
  pause;
end
```

fast_detector.m

```
function[featurePoints] = fast_detector(Im, threshold, nStar)
circle = [0 -3; 1 -3; 2 -2; 3 -1; 3 0; 3 1; 2 2; 1 3; 0 3; -1 3; -2 2; -3 1; -3 0; -3 -1; -2 -2; -1 -3];
featurePoints = [];
[height, width] = size(Im);
radius = 3;
for x = radius+1:width-radius-1
   for y = radius+1:height-radius-1
      intensity_circle = zeros(1, size(circle, 1));
      current_pixel_value = Im(y,x);
      for n=1:size(circle,1)
         intensity_circle(1,n) = Im(y+circle(n,2),x+circle(n,1));
      end
      aboveT = intensity_circle > current_pixel_value + threshold;
      longestSeqAboveT = longest_non_zero_seq(aboveT);
      belowT = intensity_circle < current_pixel_value - threshold;
      longestSeqBelowT = longest_non_zero_seq(belowT);
      if longestSeqAboveT >=nStar || longestSeqBelowT >= nStar
         featurePoints = cat(1,featurePoints, [x y]);
      end
   end
end
end
```

gaussDeriv2D.m

```
function [Gx, Gy] = gaussDeriv2D(sigma)
   maskSize = 2*ceil(3*sigma)+1;
   Gx = zeros(maskSize,maskSize);
   xo = 1 + ceil(3*sigma);yo = xo;
   const1 = (2*pi*sigma^4);
   const2 = (2*sigma^2);
   for x=1:maskSize;
      for y=1:maskSize;
         Gx(y,x) = -((x - xo)/const1)*exp(-((x-xo)^2 + (y-yo)^2)/const2);
      end
   end
   Gy = Gx';
end
```

harris_detector.m

```
function[cornerPoints,R]=harris_detector(Im, sigD,sigI, alpha)
[Gx,Gy] = gaussDeriv2D(sigD);
Ix = imfilter(Im,Gx,'replicate');
Iy = imfilter(Im,Gy,'replicate');
g_filter = fspecial('gaussian',2*ceil(3*sigI)+1,sigI);
g_Ix2 = imfilter(Ix.^2, g_filter,'replicate');
g_Iy2 = imfilter(Iy.^2, g_filter,'replicate');
g_IxIy = imfilter(Ix.*Iy, g_filter,'replicate');
R = (g_Ix2.*g_Iy2) - (g_IxIy.^2) - alpha.*((g_Ix2+g_Iy2).^2);
R(R<1e6) = 0;
newIm = non_maximal_suppression(R);
[y,x] = find(newIm > 0);
cornerPoints = [x y];
end
```

longest_non_zero_seq.m

```
function[longestSeqLen] = longest_non_zero_seq(row_vector)
length = size(row_vector,2);
seqLen = 0;
longestSeqLen = 0;
for n=1:length
   if row_vector(1,n) == 1
      seqLen = seqLen +1;
   else
      if seqLen > longestSeqLen
         longestSeqLen = seqLen;
      end
      seqLen=0;
   end
end
if seqLen~=0
   for n=1:length
      if row_vector(1,n) == 0
         break;
      end
      seqLen=seqLen+1;
   end
   if seqLen > longestSeqLen
```

```
        longestSeqLen = seqLen;
    end
end
end
```

non_maximal_suppression.m

```
function[out] = non_maximal_suppression(Im)
[height,width] = size(Im);
out = zeros(height,width);
for x = 2:width-1
    for y=2:height-1
        max_in_nighborhood = max(max(Im(y-1:y+1,x-1:x+1)));
         if Im(y,x)==max_in_nighborhood
             out(y,x) = max_in_nighborhood;
         end
    end
end
end
```