CSC420 A2

vaghela2

February 20, 2017

Question 1

a.

```
function out = q1a(octave, scalePerOctave, sigma)
  img = imread('building_1.jpg');
  img = rgb2gray(img);

  k = 2^(1/scalePerOctave);
  I = img;

  for i = 1:octave
    % Reducing the image per octave
    if(i > 1)
      I = impyramid(I, 'reduce');
    end

    [imgHeight, imgWidth] = size(I);
    imgPyramid = zeros(imgHeight, imgWidth, scalePerOctave);
    diffPyramid = zeros(imgHeight, imgWidth, scalePerOctave-1);

    for j = 1:scalePerOctave
      % Smoothing the image for each scale
      imgPyramid(:,:,j) = imgaussfilt(I,(k^(j-1))*sigma);
      if(j > 1)
        % Taking the difference of gaussians
         diffPyramid(:,:,j-1) = (imgPyramid(:,:,j) - imgPyramid(:,:,j-1));
      end
    end

    % Finding all local maxima in particular octave
    localMaxima = [];
    for xPixel = 2:imgHeight-1
      for yPixel = 2:imgWidth-1
        for c = 2:scalePerOctave-2

          % Found a local maxima
          if (findLocalMaxMin(diffPyramid, xPixel, yPixel, c))
            xCoord = xPixel * (2^(i-1));
            yCoord = yPixel * (2^(i-1));
            scaleSigma = scalePerOctave * (k^(j-1));
            localMaxima = [localMaxima, xCoord, yCoord, scaleSigma];
          end
        end
      end
```

```
42      end % now you've found all the maxima in this octave
43
44      % drawing the points of interest on the image
45      [o, maximaSize] = size(localMaxima);
46      imshow(img);
47      for coord = 1:3:maximaSize
48        centers = [localMaxima(coord), localMaxima(coord+1)];
49        viscircles(centers, localMaxima(coord+2));
50      end
51    end
52  end
53
54
55  function out = findLocalMaxMin(diffPyramid, currentX, currentY, currentZ)
56    max = 1;
57    min = 1;
58    out = 1;
59    currentPixel = diffPyramid(currentX, currentY, currentZ);
60
61    % filtering bad maximum values
62    if (currentPixel > -2 && currentPixel < 2)
63      out = 0;
64      return;
65    end
66
67    % checking surrounding pixels for a max or min
68    for z = currentZ-1:currentZ+1
69      for y = currentY-1:currentY+1
70        for x = currentX-1:currentX+1
71
72          % don't compare pixel with itself
73          if (x == currentX && y == currentY & z == currentZ)
74            break
75          end
76
77          % current pixel isn't the max
78          if (diffPyramid(x,y,z) >= currentPixel)
79            max = 0;
80          end
81
82          % current pixel isn't the min
83          if (diffPyramid(x,y,z) <= currentPixel)
84            min = 0;
85          end
86
87          if (~min && ~max)
88            out = 0;
89            return;
90          end
91        end
92      end
93    end
94  end
```

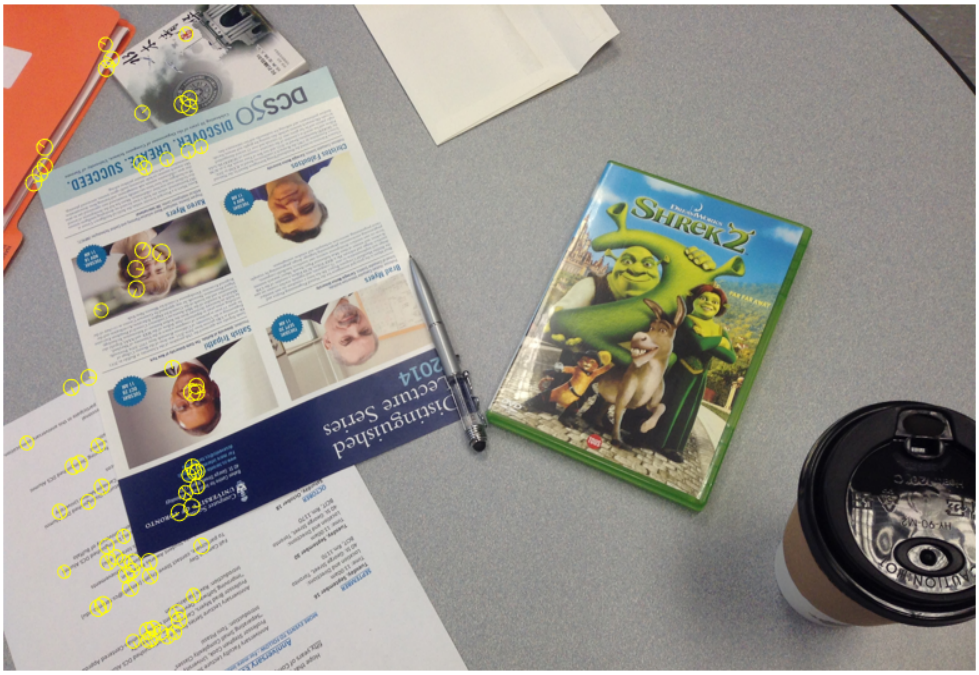
b.

Question 2

a.

```
function [refFrame, refDescr, testFrame, testDescr] q2a()
    addpath('./sift');

    refImg = imread('reference.png');
    refImg = rgb2gray(refImg);
    testImg = imread('test.png');
    testImg = rgb2gray(testImg);

    [refFrame, refDescr] = sift(im2double(refImg));
    [testFrame, testDescr] = sift(im2double(testImg));

    imshow(refImg);
    hold on;
    % plotting only the first 100 points
    refI = plotsiftframe(refFrame(:,1:100));
    set(refI,'color','y','linewidth',1);
    hold off;

    imshow(testImg);
```

```
    hold on;
    % plotting only the first 100 points
    testI = plotsiftframe(testFrame(:,1:100));
    set(testI,'color','y','linewidth',1) ;
    hold off;

    [refFrame, refDescr, testFrame, testDescr] = [refFrame, refDescr, testFrame,
                                                  testDescr];
end
```



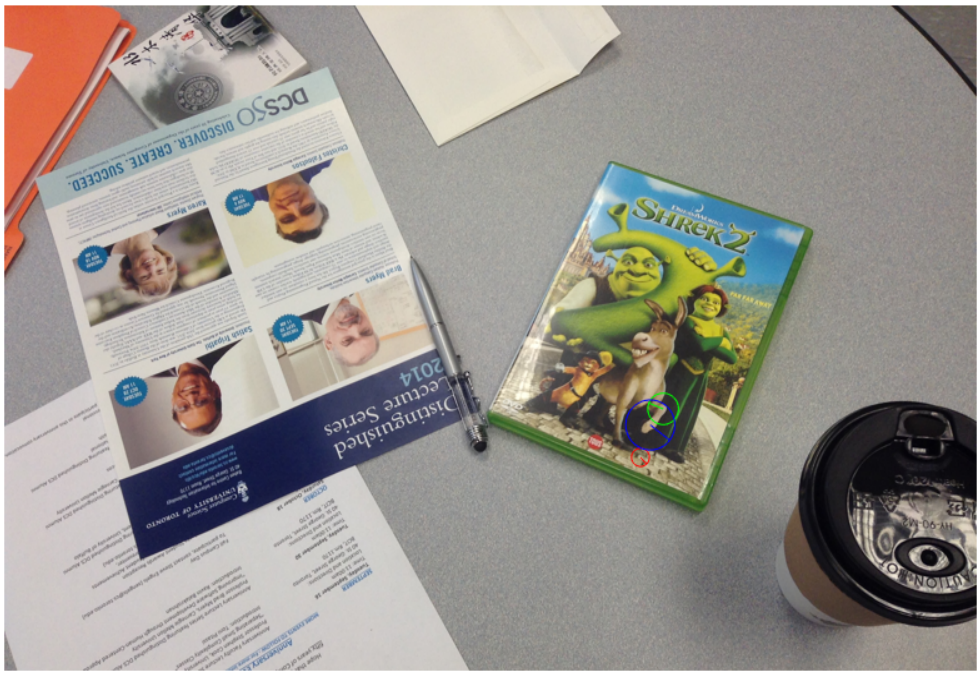
b.

A possible matching algorithm for this can begin by using sift to detect key points on both reference and test. Then find the closest match for each point by calculating the distance. Take the first and second closest matches and compute a ratio of the distances. If the ratio is below some threshold, then its a proper match.

```
function [refFrame, testFrame, refIndices, testIndices] = q2b()
    addpath('./sift');

    refImg = imread('reference.png');
    refImg = rgb2gray(refImg);
    testImg = imread('test.png');
    testImg = rgb2gray(testImg);

    % detecting keypoints for both images
    [refFrame, refDescr] = sift(im2double(refImg));
    [testFrame, testDescr] = sift(im2double(testImg));

    % calculating distance between points on both images
    distance = dist2(refDescr.', testDescr.');
    [n, m] = size(distance);
    threshold = 0.8;

    matches = [];
    ratios = [];

    % if ratio of distances between two points is less than threshold its a match
    [distSort, distIndex] = sort(distance, 2);
    for i = 1:n
        closestMatch = distIndex(i,1);
        ratio = distSort(i,1)./distSort(i,2);
```

```
        if ratio < threshold
            matches(i) = closestMatch;
            ratios(i) = ratio;
        else
            matches(i) = 0;
            ratios(i) = 1;  % ignore this match
        end
    end

    [ratioSort, ratioIndex] = sort(ratios);
    testMatches = [];
    % Getting the top 3 matches
    for i = 1:3
        index = ratioIndex(1, i);
        testMatches(index) = matches(index);
    end

    indices = find(testMatches > 0);

    % plotting the matches
    imshow(imRef);
    hold on;
    ref1 = plotsiftframe(refFrame(:, indices(1):indices(1)));
    set(ref1,'color','r','linewidth',1);
    ref2 = plotsiftframe(refFrame(:, indices(2):indices(2)));
    set(ref2,'color','g','linewidth',1);
    ref3 = plotsiftframe(refFrame(:, indices(3):indices(3)));
    set(ref3,'color','b','linewidth',1);
    hold off;

    imshow(imTest);
    hold on;
    test1 = plotsiftframe(fTest(:, testMatches(indices(1)):testMatches(indices(1))));
    set(test1,'color','r','linewidth',1);
    test2 = plotsiftframe(fTest(:, testMatches(indices(2)):testMatches(indices(2))));
    set(test2,'color','g','linewidth',1);
    test3 = plotsiftframe(fTest(:, testMatches(indices(3)):testMatches(indices(3))));
    set(test3,'color','b','linewidth',1);
    hold off;

    [refFrame, testFrame, refIndices, testIndices] = [refFrame, testFrame,
    [indices(1), indices(2), indices(3)], [testMatches(indices(1)),
    testMatches(indices(2)), testMatches(indices(3))]];
end
```


DREAMWORKS
SHREK 2
FAR FAR AWAY
DVD

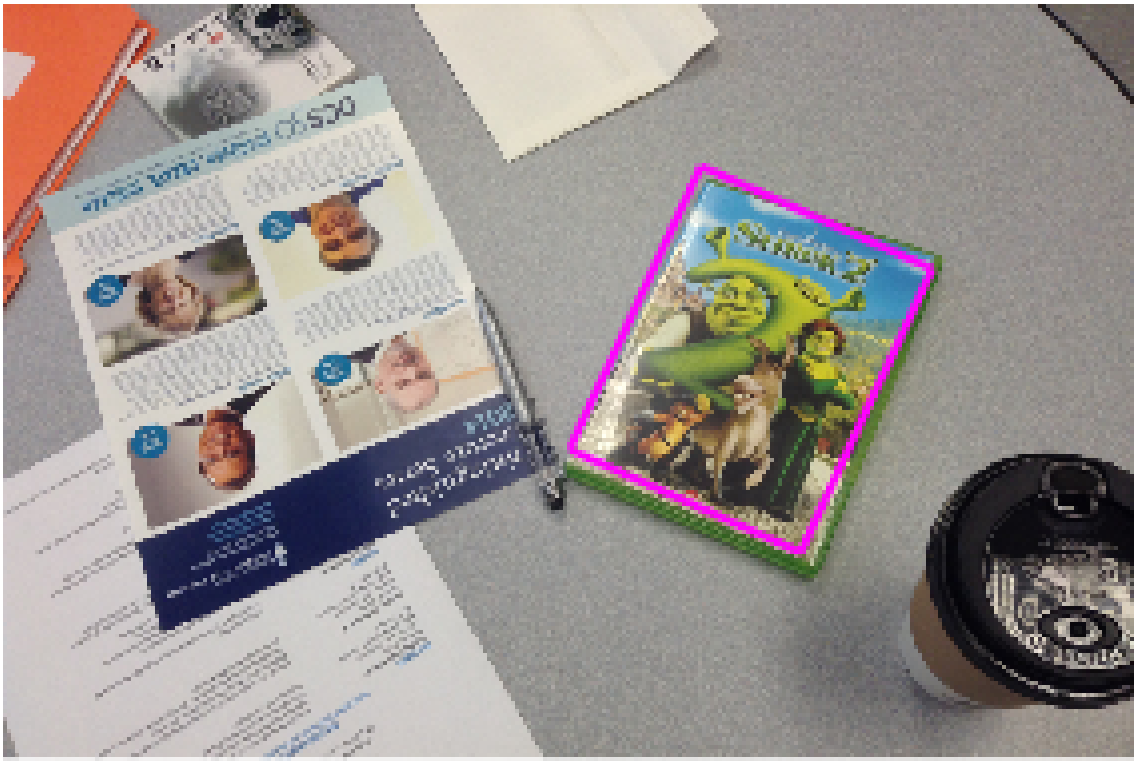

DCSSO DISCOVER. CREATE. SUCCEED.
Distinguished
Lecture Series
2014
SHREK 2

c.

```
function out = q2c()
    addpath('./sift');
    [refFrame, testFrame, refIndices, testIndices] = q2b();

    refPoints = [refFrame(1:2, refIndices(1):refIndices(1)),
                 refFrame(1:2,refIndices(2):refIndices(2)), refFrame(1:2,
                 refIndices(3):refIndices(3))];

    testPoints = [testFrame(1:2, testIndices(1):testIndices(1)),
                  testFrame(1:2, testIndices(2):testIndices(2)),
                  testFrame(1:2, testIndices(3):testIndices(3))];

    % building the [x y 0 0 1 0; 0 0 x y 0 1] matrix for 3 points
    r1a = [refPoints(1, 1), refPoints(2, 1), 0, 0, 1, 0];
    r1b = [0, 0, refPoints(1, 1), refPoints(2, 1), 0, 1];

    r2a = [refPoints(1, 2), refPoints(2, 2), 0, 0, 1, 0];
    r2b = [0, 0, refPoints(1, 2), refPoints(2, 2), 0, 1];

    r3a = [refPoints(1, 3), refPoints(2, 3), 0, 0, 1, 0];
    r3b = [0, 0, refPoints(1, 3), refPoints(2, 3), 0, 1];

    P = [r1a; r1b; r2a; r2b; r3a; r3b];

    % build the [x;y] matrix for test 3 points
    pPrime = [testPoints(1,1); testPoints(2,1); testPoints(1,2);
              testPoints(2,2); testPoints(1,3); testPoints(2,3)];

    % computing (p^-1)*p
    out = inv(P)*pPrime;
end
```

d.

```
refImg = imread('reference.png');
testImg = imread('test.png');
[h, w] = size(imRef);

% getting edges for parallelogram
P = [1, 1, 0, 0, 1, 0; 0, 0, 1, 1, 0, 1;
     1, h, 0, 0, 1, 0; 0, 0, 1, h, 0, 1;
     w, 1, 0, 0, 1, 0; 0, 0, w, 1, 0, 1;
     w, h, 0, 0, 1, 0; 0, 0, w, h, 0, 1;]

affine = q2c();
PP = P*affine; % computing affine transformation
imshow(testImg);
hold on;
line([PP(1), PP(3)],[PP(2),PP(4)],'Color','y', 'Linewidth', 2);
line([PP(1), PP(5)],[PP(2),PP(6)],'Color','y', 'Linewidth', 2);
line([PP(3), PP(7)],[PP(4),PP(8)],'Color','y', 'Linewidth', 2);
line([PP(5), PP(7)],[PP(6),PP(8)],'Color','y', 'Linewidth', 2);
hold off;
```


Shrek 2