**ASSIGNMENT A6**

Shantnu Kakkar

CS 6320, Spring 2016

March 09, 2016

**Section 1: Intro:**

This assignment is based on the edge tracking and feature detection (corners) techniques learnt in the lecture. I will be implementing various method, for example **Harris corner detector, orientation histogram, algorithm 5.2 and 5.3 from text**. The basic idea behind my approach is that as the light gets brighter or darker, the image will get brighter or darker. This means that the gradient scales with the image. This creates a problem for edge detectors since they rely on image gradient magnitude. To overcome that problem, we use orientation of the image gradient, which is unaffected by scaling. The following figure 1 from the text proves this. Orientation histogram is a good technique that will be used for this.

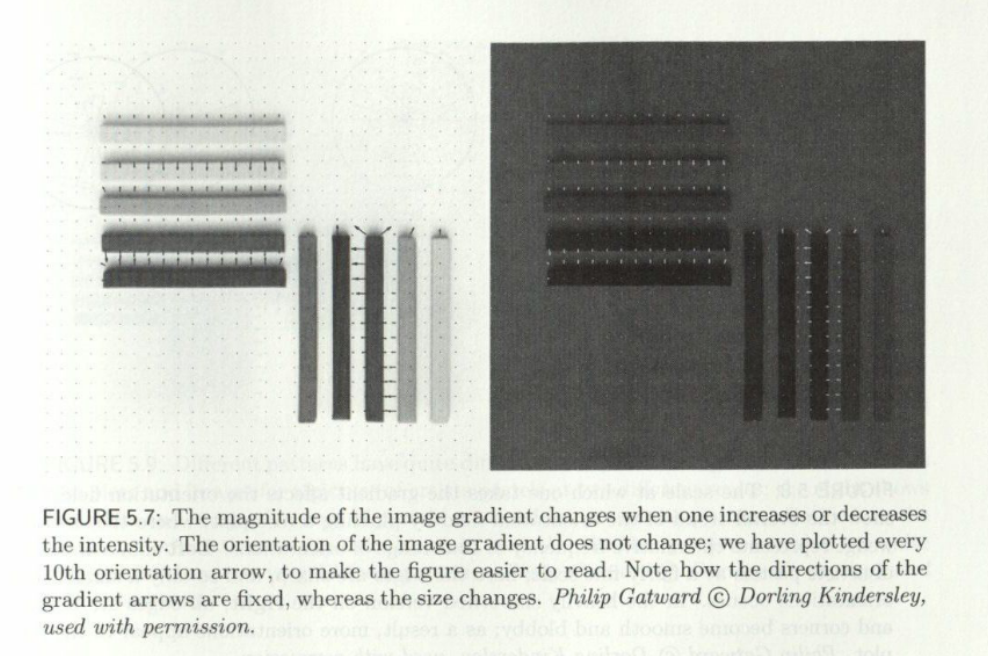


Figure 1 This figure forms the motivation for me to using image orientation for corner detection.

Another motivation is to use Laplacian of Gaussian since we need to estimate the radius (scale) of the circular patch. The radius estimate should get larger proportionally when the image gets bigger. We could center a blob of fixed appearance on the corner and then choose the scale to be the radius of the best fitting blob. An efficient way to do that us Laplacian of Gaussian.

For this problem, we are given two images and we need to find corners in them. We are required to implement algorithm 5.2 and 5.3 and compare them as feature detectors by implementing on the two given images. After the implementation, I will be answering the following question

* Are the two algorithms stable with respect to image rotation, translation and scaling?

**Section 2: Method:**

* Matlab is used to carry out the experiments.
* For the interest points I will try to shoot between 20-40 points.
* I will be doing the translation, rotation and scaling part as follows:

**Translation**:  
  
If the image is mxn, thenshift it right (with wraparound):  
  
   imt = [im(:,51:n),im(:,1:50)];  
  
**Rotation**:  
  
    im45 = imrotate(im,45);  
  
**Scaling**:  
  
   im2 = imresize(im,2);

* I will be using the combo function provided by professor to analyse my results for Harris.
* I will be using imregionalmax function instead of defining my own function since it worked better for me compared to my function. I was able to get stable rotation and translation by using this function. There were some problems when I tried with my own local\_max function. My own local max function will go into the critique section.

Following functions are implemented

* CS5320\_Harris – This function detects the corners(interest points). It takes input as the gray image and the desired window size, and the output is the response matrix defined as:

R (mxn array): corner response <=0: homogeneous , >0 and small: edge, large: corner

* CS5320\_LoG\_interest – It finds the interest points as well as the radius (maxima sigma scale) at those points. Its input is the gray image and a value p which helps us to set the threshold to reduce the number of threshold points. It outputs the interest points as 1 and other points as 0. It also outputs the corresponding max sigma scale at that location.
* CS5320\_gradient\_histogram – It gets the histogram of gradient orientations. Its input is the gray image, the row of center of patch, column of center of patch, radius of pixels to consider, minimum gradient magnitude to consider, and a Boolean which tells which tells whether to use magnitude of gradient as weight in histogram. It outputs orientation counts in 20-degree bins
* CS5320\_corner\_patches – It produces patches using Harris ( Alg. 5.2 from text). See below for detailed description if this function.
* CS5320\_LoG\_patches – It produces patches using LogInterest (Alg. 5.3 text). ). See below for detailed description if this function.

In addition to the above functions there is a script named verification which contains how I call every function. Please note that I have divided this script into various cells (sections). Please run individual section for every function. I have commented out imshow in this script everywhere. Please uncomment that to see the answers. For translation, rotation and scaling, uncomment from the corner patches and

loG patches section to see the stability analysis.

**The following algorithm has been used for Harris:**

* *Initialize R = zeros(size(im,1),size(im,2));*
* *Find gradient[dx,dy] = gradient(double(im));*
* *Loop for r = 1+k:size(im,1)-k*
* *Loop for c = 1+k:size(im,2)-k*
* *a = dx(r-k:r+k,c-k:c+k);*
* *b = dy(r-k:r+k,c-k:c+k);*
* *a = a(:);*
* *b = b(:);*
* *pts = [a,b];*
* *M = pts'\*pts;*

**The following algorithm has been used for log interest**

* *Make sigma vector sigma = 0.3:0.01:6;*
* *Initializations*

*scale = zeros(nr,nc);*

*C = zeros(nr,nc,num\_sigmas);*

*interest\_pts = zeros(nr,nc);*

*maxResponses = zeros(nr,nc);*

*temp1\_scale = zeros(nr,nc);*

*temp2\_scale = zeros(nr,nc);*

* *Loop for s\_index = 1:num\_sigmas*

*Make template T = fspecial('log', 21, sigma(s\_index));*

*Find response C(:,:,s\_index) = abs(filter2(T,im));*

* *Loop for r = 1:nr*
* *loop for c = 1:nc*

*temp1 = C(r,c,:);*

*temp1 = temp1(:);*

*[temp2(r,c),idx] = max(temp1);*

*if idx ~=1 && idx ~= num\_sigmas*

*temp1\_scale(r,c) = sigma(idx);*

*maxResponses(r,c) = temp2(r,c);*

* *Find points above threshold*

*maxvalues = p\*max(max(maxResponses));*

*for r = 1:nr*

*for c = 1:nc*

*if maxResponses(r,c)<maxvalues*

*maxResponses(r,c) = 0;*

* *Find local maxima interest\_pts = imregionalmax(maxResponses);*
* *Update scale = interest\_pts.\* temp1\_scale;*

**The following method is used for finding hands in** CS5320\_gradient\_histogram function**:**

* Initialize *H = [0;0;0;0;0;0;0;0;0];*
* *Find window/patch = im(r-ceil(radius):r+ceil(radius),c-ceil(radius):c+ceil(radius));*
* *Gradient [dx,dy] = gradient(double(patch));*
* *Loop for r = 1:size(patch,1)*
* *for c = 1:size(patch,2)*

*magnitude = sqrt( dx(r,c)^2 + dy(r,c)^2);*

*if magnitude > thresh*

*orientation = (atan2d(dy(r,c),dx(r,c)));*

*if orientation <0*

*orientation = orientation + 180;*

*end*

*bin = ceil(orientation/20);*

*if bin > 0*

*if w==0*

*H(bin) = H(bin) + 1;*

*else*

H(bin) = H(bin) + magnitude;

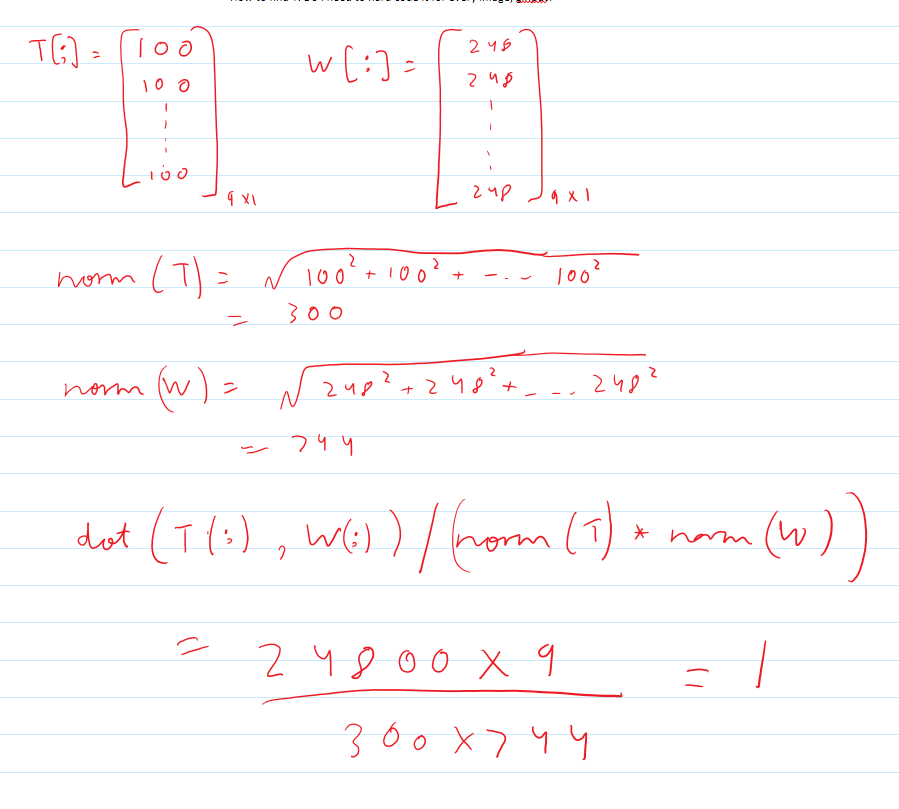
**The following method is used for finding hands in the image using edge detection:**

* Make an edge image of the original image. Various methods can be used for this sich as sobel, zerocrossing, and Laplacian. Function **edge** is used for this, example im = edge(im,'loG');
* Likewise make an edge image of the universal template.
* Do normalized correlation between the two edge images
* If for any pixel the correlation is greater than a certain threshold, then change the colour of that pixel of the original image.

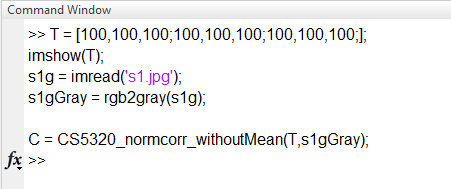
**Section 3: Verification:**

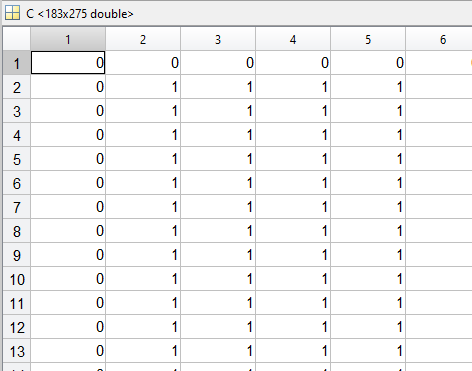
Testing **CS5320\_normcorr\_withoutMean.**

* Take a 3\*3 random template, say [100,100,100; 100,100,100; 100,100,100;] and send it to the **CS5320\_normcorr\_withoutMean** function along with theimage1
* Now since there are 9 elements in T which is 3\*3, the window(subimage) will be a 3\*3 matrix.
* Let’s compute the answer of this dot product between the template and **the first subimage (top left most)** using hand.
* The formula used in matlab is: dot(T(:), W(:) )/(norm\_T\*norm\_W);

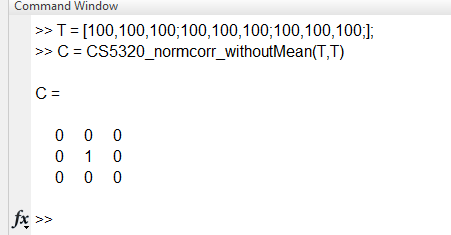


* Thus we see that we are getting a 1. This one should be saved in the middle element of the 3\*3 matrix. Now we run the matlab to see whether the result is a 3\*3 matrix with a middle element 1 or not.





* From the above two images we can see that for the dot(T(:), W(:) )/(norm\_T\*norm\_W) calculated between the window and the template, we are getting a value 1 in the pixel number (2,2) for both theory and matlab simulation. Hence the function is correct.
* Likewsie, if I do normal correlation between the template itself, I get a matrix with 1 in the middle

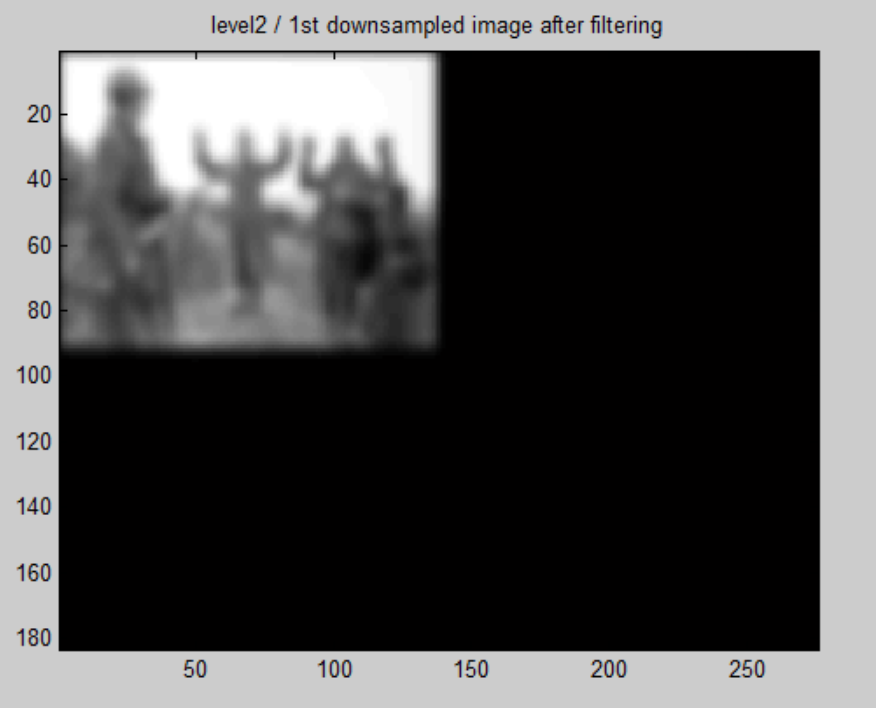


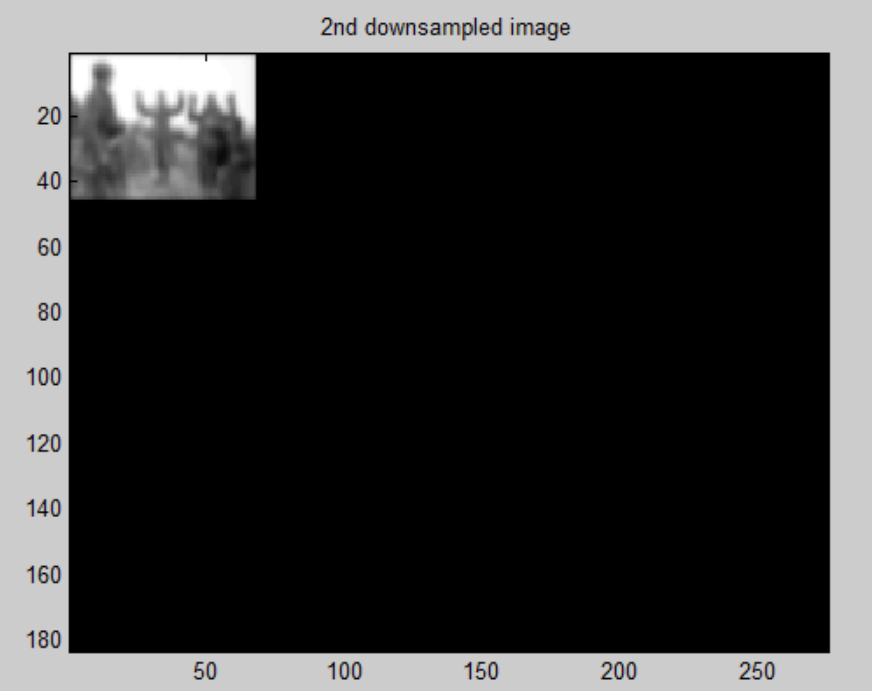
Testing **CS5320\_G\_pyramid**

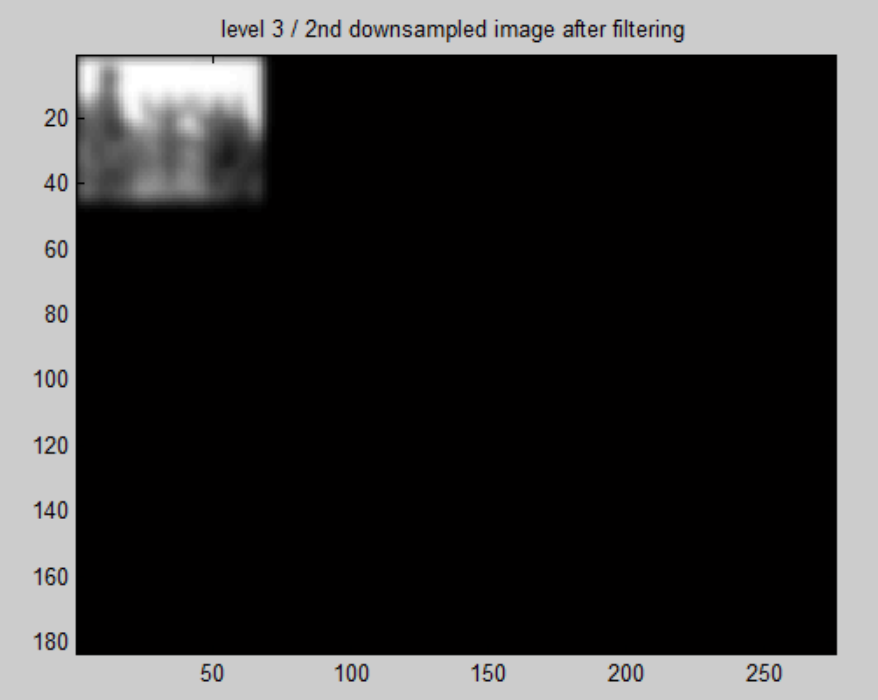
* The image should be down sized correctly and it should be blurred. As we can see the below images this is happening. It’s size is decreased to exactly half and blurring is taking place.









**Like wise Level 4 and level 5 are being obtained correctly.**

Testing **CS5320\_hands**

* This function serves to make hand red between reduced template and reduced image. By many observations, it has been observed that the color of the hand does change when normal correlation (done between reduced template and reduced image) within in this function comes out to be greater than the threshold. The following image is the proof.

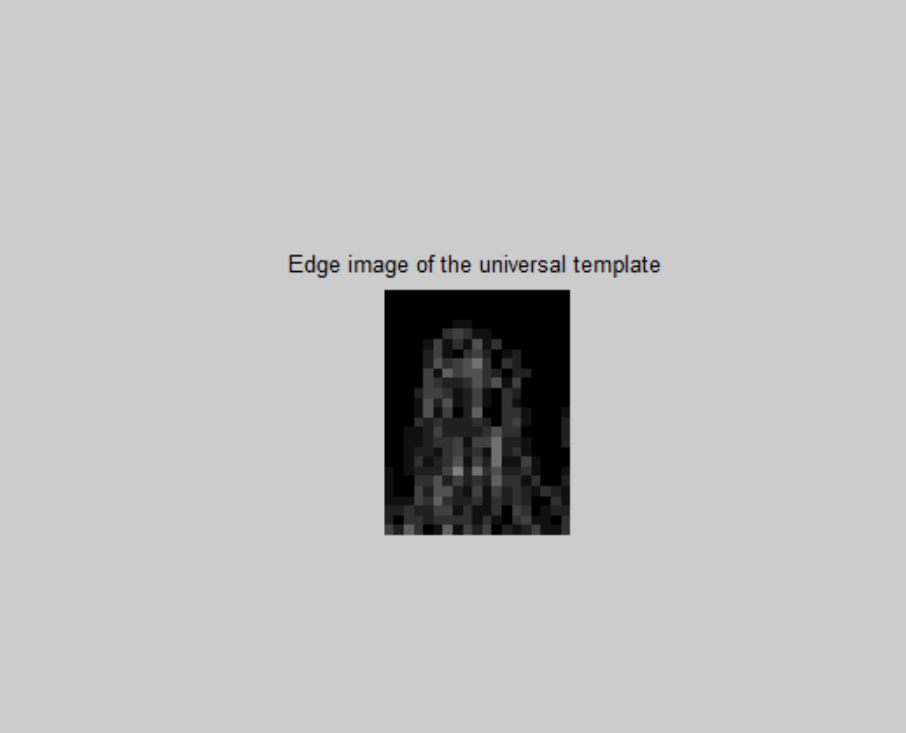


**Section 4: Data:**

Following figure shows the universal template that I extracted from all the images. It’s size is 25\*19. It has been hard coded in hands and other functions that require a template, but have no template as input.



Following is the edge image of the universal template that I am hard coding. It is giving good results. I made it by cropping 4 hands from each image, making edge image of each cropped hand and averaging them.



Now **I will be showing the various results that I am getting as follows. In the analysis section, results for each of these will be calculated mathematically.**

1. Directly doing the normal correlation between the images and the universal template using the CS5320\_normcorr\_withoutMean function. Please note that I had to vary the threshld for getting good results with this direct normalization correlation technique, since it is not a good technique. Other techniques like Gaussian pyramid and edge detection will have threshold fixed.









1. Directly doing the normal correlation between the images and the universal template using the CS5320\_normcorr\_withMean function. Please note that I had to vary the threshld for getting good results with this direct normalization correlation technique, since it is not a good technique. Other techniques like Gaussian pyramid and edge detection will have threshold fixed.







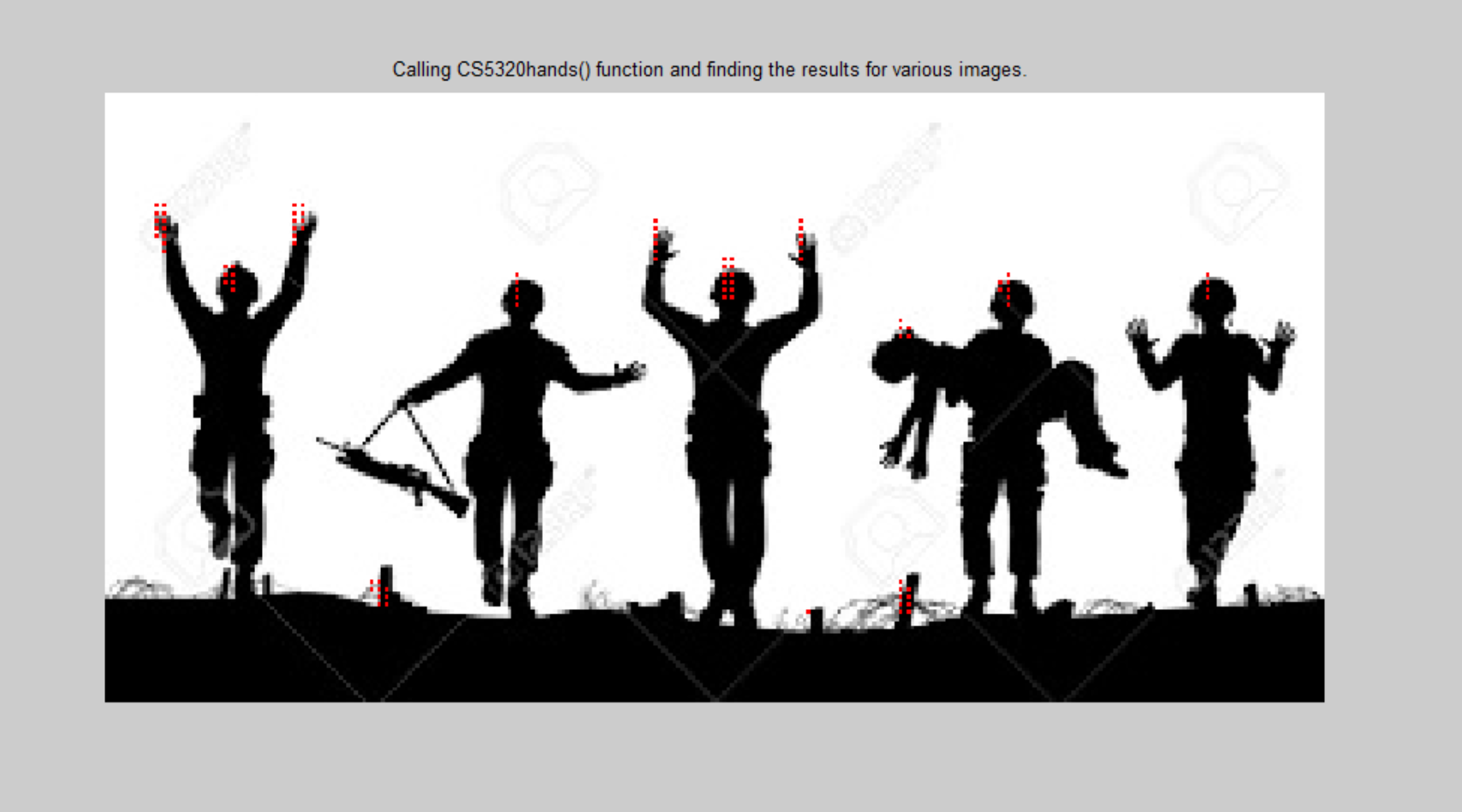


1. Calling CS5320\_hands() function and finding the results for various images. In CS5320\_hands function, CS5320\_normcorr\_withMean function has been used. Also, **level 2 of image pyramid and level 2 of template pyramid has been used**. Threshold is fixed to 0.7 for all the images.









1. Calling CS5320\_edge\_method() function and finding the results for various images. In CS5320\_hands function, CS5320\_normcorr\_withMean function has been used. Threshold is fixed to 0.26 for all the images.









**Section 5: Analysis:**

**The following is the summary of Precision, recall and computational efficiency for various omages and various methods. Please note that the running time is varying for the tic toc for each run. I do not know the reason for varying.**

1. **Directly doing the normal correlation between the images and the universal template using the CS5320\_normcorr\_withoutMean function**. Please note that I had to vary the threshold for getting good results with this direct normalization correlation technique, since it is not a good technique. Other techniques like Gaussian pyramid and edge detection will have threshold fixed. Here, tic toc has been done as follows:

*C = CS5320\_normcorr\_withoutMean(T,s1g);*

*or i = 1: num\_rows\_IM*

*for j = 1:num\_cols\_IM*

*if C(i,j)>0.7*

*s1g(i,j,1) = 256;*

*s1g(i,j,2) = 0;*

*s1g(i,j,3) = 0;*

*end*

*end*

*end*

*time1 = toc*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Figure 1** | **Figure 2** | **Figure 3** | **Figure 4** |
| **Computational Time** | **0.6712** | **0.5746** | **0.5832** | **0.5481** |
| **Precision** | **4/6** | **6/10** | **4/5** | **5/5** |
| **Recall** | **1** | **6/11** | **4/16** | **5/12** |

1. **Directly doing the normal correlation between the images and the universal template using the CS5320\_normcorr\_withMean function.** Please note that I had to vary the threshold for getting good results with this direct normalization correlation technique, since it is not a good technique. Other techniques like Gaussian pyramid and edge detection will have threshold fixed. Here, tic toc has been done as follows:

*tic;*

*C = CS5320\_normcorr\_withMean(T,s1g);*

*for i = 1: num\_rows\_IM*

*for j = 1:num\_cols\_IM*

*if C(i,j)>0.7*

*s1g(i,j,1) = 256;*

*s1g(i,j,2) = 0;*

*s1g(i,j,3) = 0;*

*end*

*end*

*end*

*time1 = toc*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***Figure 1*** | ***Figure 2*** | ***Figure 3*** | ***Figure 4*** |
| ***Computational Time*** | ***1.2290*** | ***1.3251*** | ***1.2790*** | ***1.2765*** |
| **Precision** | **4/5** | **6/11** | **1** | **4/9** |
| **Recall** | **1** | **6/11** | **6/16** | **4/12** |

1. **Calling CS5320\_hands()** function and finding the results. In CS5320\_hands function, CS5320\_normcorr\_withMean function has been used. Also please note that, **level 2 of image Gaussian pyramid and level 2 of template Gaussian pyramid** has been used. Threshold is fixed to 0.7 for all the images. Here tic toc has been done like this:

*s1g = imread('s2.jpg');*

*s1gGray = rgb2gray(s1g);*

*tic*

*hands = CS5320\_hands(s1gGray);*

*time1 = toc*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Figure 1** | **Figure 2** | **Figure 3** | **Figure 4** |
| **Computational Time** | **0.6836** | **0.6774** | **0.6829** | **0.6844** |
| **Precision** | **4/6** | **5/10** | **6/9** | **4/12** |
| **Recall** | **1** | **5/11** | **6/16** | **4/12** |

1. **Calling CS5320\_edge\_method()** function and finding the results for various images. In CS5320\_hands function, CS5320\_normcorr\_withMean function has been used. Threshold is fixed to 0.35 for all the images. Here, Here tic toc has been done like this:

*s1g = imread('s2.jpg');*

*s1gGray = rgb2gray(s1g);*

*tic*

*hands = CS5320\_hands(s1gGray);*

*time1 = toc*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Figure 1** | **Figure 2** | **Figure 3** | **Figure 4** |
| **Computational Time** | **1.1723** | **1.4429** | **1.1005** | **1.0669** |
| **Precision** | **4/5** | **10/13** | **5/9** | **6/11** |
| **Recall** | **1** | **10/11** | **5/16** | **6/12** |

**Section 6: Interpretation:**

The following are my observations:

* Template has the most vital role to play in ensuring proper detection of hand It is very important to form an appropriate Template.
* CS5320\_normcorr\_withoutMean function and CS5320\_normcorr\_withMean function are giving really good results, however their drawback is that I need to play around a lot with the threshold. For Gaussian pyramid and edge detection method, I was able to fix my threshold.
* CS5320\_normcorr\_withoutMean function works twice as fast as CS5320\_normcorr\_withMean. Also, CS5320\_normcorr\_withoutMean requires a higher threshold.
* For the CS5320\_hands() method, I observed that if I correlate levels above level 2 of the reduced template and the reduced image, then I am not getting good results.
* A very good observation is that even though some time is wasted in making the pyramids for the image and the template, the overall time is reduced while doing correlation with reduced template and reduced image.
* I tried various edge detection methods like sobel, zero crossing, laplacian of Gaussian, etc. However, I observed that Laplacian of Gaussian gave the best results. Hence I used it for my edge method.
* Contrary to my expectations, the edge detection method consumed more time than Gaussian pyramid method. This could be because I used Laplacian of Gaussian. Sobel was quick, but didn’t give good results. Zero crossing was average.
* For image 1, edge detection method has better precision. Recall is same for both.
* For image 2, edge detection is better both for precision and recall
* For figure 3, Gaussian pyramid is better both for precision and recall
* For figure 4, edge detection performs better for both recall and same.
* From above four points, it turn out that Edge detection of image with Laplacian of Gaussian performs better than Gaussian pyramid method.

**Section 7: Critique:**

The experiment could be improved by following ways:

* Making a better template
* Playing more with upper levels of the Gaussian pyramid for reduced template and reduced image.
* Drawing plots of precision vs recall to know better how far am I from the ideal situation.

**Section 8: log**

20+ hours total