ASSIGNMENT A4

Shantnu Kakkar

CS 6320, Spring 2016

February 27, 2016

Section 1: Intro:

This assignment is based on the Pattern Recognition techniques learnt in the lecture. Some of these techniques which would be implemented in this report are: **Normalized correlation, Gaussian Pyramid** and **edge detection**. All these techniques require the knowledge of filters and convolution. The basic idea behind filtering is to use weighted sums of pixel values using different patterns of weights (like Gaussian, uniform, derivative and finite differences, etc) to find different image patterns. I will be using a **linear filter** for this report meaning that the sum of outputs obtained for the sum of two images is the same as the sum of outputs obtained for the images separately. Filters offer a natural mechanism for finding simple patterns because filters respond most strongly to pattern elements that look like the filter The process of applying a filter is called **convolution**. It is shown in the below figure:

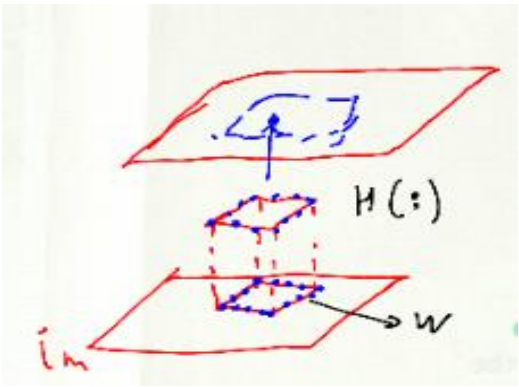


Figure 1 This figure shows the basic notion of convolution and filtering. It shows a linear filter H is applied to a sub image W. im in this figure is the image. We take a section of im and call it W. Then W is dot product with filter H pixel by pixel

One of the most important filter is the **Gaussian filter**. It is shown in figure 2. This filter is important because it ensures a blurring process that is circularly symmetric, has strongest response in the centre and die away near the boundary. This filter will be used to make Gaussian pyramid as explained in Section 2.

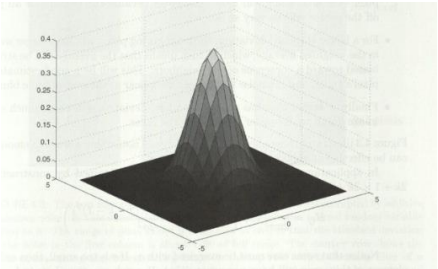


Figure 2 A Gaussian kernel

In this problem, we have been given four images. The images show some people in surrender with their hands raised. My aim is to detect those hands. Following question will be answered in the process:

- How can the success be optimized, where Success means that all raised hands in the image are found (compared to ground truth supplied by human analysis).
- How can efficiency be maximized, where Efficiency means lowest computational cost (i.e., perform correlation on the smaller images in the pyramid.
- Find the precision and recall measures for various images given and various method used.

Section 2: Method:

- Matlab is used to carry out the experiments.
- The template is made as follows – Crop 4 hands from each image by reading their pixel number using impixelinfo. Add all these templates and divide by 16. This template will serve as a **universal template**
- For each image **I will colour the portion as red if it is recognized by my method.**
- After it has been done red, I will be using the following formulas to find recall and precision for **success optimization**
  - **precision** = |{hands in image} intersect {hands retrieved}|/|{hands retrieved}
  - **recall** = |{hands in image} intersect {hands retrieved}|/{hands in image}|
- Finally, the **computational efficiency** is found by performing correlation on the smaller images in the pyramid. Tic and toc Matlab commands would be used for this.

I apply three methods in the following order:

- First I will apply simple normalized correlation to my image (gray image) and see the results.
- Then I will make a Gaussian pyramid of my image and template. I will then use a reduced template on the lower resolution images and see what results that gives.
- Finally, I would try something other than just a gray level hand image. E.g., I would take a hand image and compute an edge magnitude image and use that on an edge image of the image to be analysed.

Following functions are implemented

- **CS5320_normcorr**: It finds the normalized correlation coefficients. Its inputs are template kernel and the gray level image and outputs is the correlation matrix. I have two normcorr functions by the name CS5320_normcorr_withMean(T,IM) and CS5320_normcorr_withoutMean(T,IM). **On searching the internet, I found that there are two ways to find normal correlation coefficient. One method (given on Wikipedia) subtracts the mean of the template and the sub image from the template and the sub image respectively, and then do the correlation. This method is given in the figure below. The other method (which professor suggested me to follow) directly does the correlation without doing any mean subtraction. What I observed was that method 2 works better for direct correlation between the template and the image without any Gaussian pyramid. But if we use reduced images and the reduced template then method 1 works better. It has also been observed that method 2 requires a very high threshold while making the pixel red(0.98-0.99), whereas method 1 requires a smaller threshold(0.4-0.5). By default the hands function call** CS5320_normcorr_withMean.
- **CS5320_G_pyramid**: It make the Gaussian pyramid. At the end of the function, you could uncomment the figures section to see the Gaussian levels and the down sampled images formed. Please note that this function has been hard coded to form 5 levels. Its Inputs are the gray image, size of smoothing window and the sigma to be used for making Gaussian filter. Its output is the Gaussian pyramid.
- **CS5320_hands**: It finds hands of surrender in image using Gaussian pyramid and normalized correlation. The hands and other returned portions are returned red coloured. At the end of the function you can comment out the figures section to see the red coloured output. The input to this function is the gray image in which hands you want to found and the output is the rgb image with red coloured hands. Please note that hands = cat(3, im, im, im) has been used to covert from gray image to a rgb. In the function you can choose which level of image in the Gaussian pyramid, and which level of template in the Gaussian pyramid you want.
- **CS5320_edge_method**: It finds hands of surrender in image using edge detection and normalized. There are various edge detection methods like **Sobel, zero crossing, Laplacian of Gaussian**, etc that have been tested in this function. You can uncomment the edge detection method that you want to use inside the function. The input to this function is the gray image in which hands you want to found and the output is the rgb image with red coloured hands. Please note that hands = cat(3, im, im, im) has been used to covert from gray image to a rgb.

In addition to the above functions there is a script named MakeUniversalTemplate to make a template which fits all the four given images. This script saves the template as a 'UniversalTemplate.mat' file. Hiwever please note that as directed by professor, the universal template has been hard coded inside the functions instead of using the 'load' matlab command to load it inside the script.

The following algorithm has been used for normalized correlation:

- Find number of rows and columns in image and template
  - *[num_rows_T , num_cols_T] = size(T);*
  - *[num_rows_IM , num_cols_IM] = size(IM);*

- *Find norm of T – MeanT*
  - *meanT = mean(T(:));*
  - *meanT = ones(size(T(:))) * meanT;*
  - *norm_T = norm((T(:) - meanT));*
- *Initialise C*
  - *C = zeros(num_rows_IM , num_cols_IM);*
- *Loop for r = 1: num_rows_IM - num_rows_T*
  - *Loop for c = 1:num_cols_IM - num_cols_T*
    - ➔ *Make window W = IM(r:r+num_rows_T-1,c:c+num_cols_T-1);*
    - ➔ *Find norm of W-MeanW:*
      meanW = mean(W(:));
      *meanW = ones(size(W(:))) * meanW;*
      norm_W = norm((W(:) - meanW));
    - ➔ *if norm_W ~=0*
      *C((r+floor(num_rows_T/2)), (c + floor(num_cols_T/2))) = dot(T(:)-meanT , W(:) - meanW)/(norm_T*norm_W);*

The following algorithm has been used for making the Gaussian pyramid:

- *Subsampling and Image by a factor of Two*
- *Apply a gaussian filter to the original image. I have done H = fspecial('gaussian',2*k+1,sigma);*
- *Create new image with dimensions half those of the old image*
- *Set the value of the i, jth pixel of the new image to the value of the 2i,2jth pixel of the filtered image*

The following method is used for finding hands in the image using Gaussian pyramid:

- Set the level in Gaussian pyramid that you want to set for template and image
  - levelT = 1; levelIM = 2;

- Convert your gray inout image to a rgb image
  - hands = cat(3, im, im, im);
- Make image pyramid
  - pyrIM = CS5320_G_pyramid(im,4,2);
- Load the universal template
  - load('UniversalTemplate.mat');
- Make template image pyramid
  - pyrT = CS5320_G_pyramid(T,4,2);
- Find size of T and image
  - [num_rows_T , num_cols_T] = size(T);
  - [num_rows_IM , num_cols_IM] = size(im);
- Find number of rows in the reduced template and image. Note that this will on the levelT and levelIM set.
  - NumRowsInReducedT = ceil(num_rows_T/(2^(levelT-1)));
  - NumColsInReducedT = ceil(num_cols_T/(2^(levelT-1)));
  - NumColsInReducedIM = ceil(num_cols_IM/(2^(levelIM-1)));
  - NumRowsInReducedIM = ceil(num_rows_IM/(2^(levelIM-1)));
- Find the normalized correlation
  C=CS5320_normcorr(pyrT(1:NumRowsInReducedT,1:NumColsInReducedT,levelT),pyrIM(1:NumRowsInReducedIM,1:NumColsInReducedIM,levelIM));
- Loop for i = 1: NumRowsInReducedIM
  - Loop for j = 1:NumColsInReducedIM

➔ if C(i,j)>threshhold

```
hands(i*(2^(levelIM-1)),j*(2^(levelIM-1)) ,1) = 256;
hands(i*(2^(levelIM-1)),j*(2^(levelIM-1)) ,2) = 0;
hands(i*(2^(levelIM-1)),j*(2^(levelIM-1)) ,3) = 0;
```

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

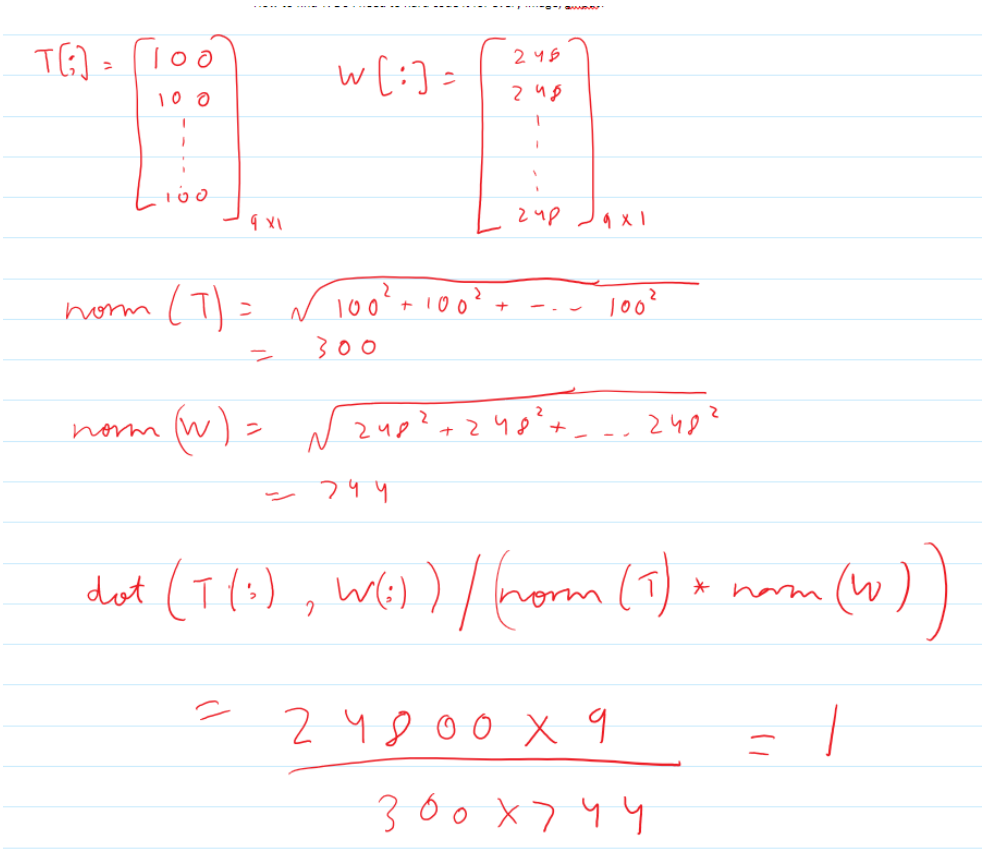
- Make an edge image of the original image. Various methods can be used for this sich as sobel, zerocrossing, and Laplacian. Function **edge** is used for this, example im = edge(im,'loG');
- Likewise make an edge image of the universal template.
- Do normalized correlation between the two edge images
- 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 the image1
- 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.

```
Command Window
>> T = [100,100,100;100,100,100;100,100,100;];
imshow(T);
s1g = imread('s1.jpg');
s1gGray = rgb2gray(s1g);

C = CS5320_normcorr_withoutMean(T,s1gGray);
fx >>
```

C <183x275 double>

	1	2	3	4	5	6
1	0	0	0	0	0	
2	0	1	1	1	1	
3	0	1	1	1	1	
4	0	1	1	1	1	
5	0	1	1	1	1	
6	0	1	1	1	1	
7	0	1	1	1	1	
8	0	1	1	1	1	
9	0	1	1	1	1	
10	0	1	1	1	1	
11	0	1	1	1	1	
12	0	1	1	1	1	
13	0	1	1	1	1	

- 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

```
Command Window
>> T = [100,100,100;100,100,100;100,100,100;];
>> C = CS5320_normcorr_withoutMean(T,T)

C =

   0   0   0
   0   1   0
   0   0   0

fx >>
```

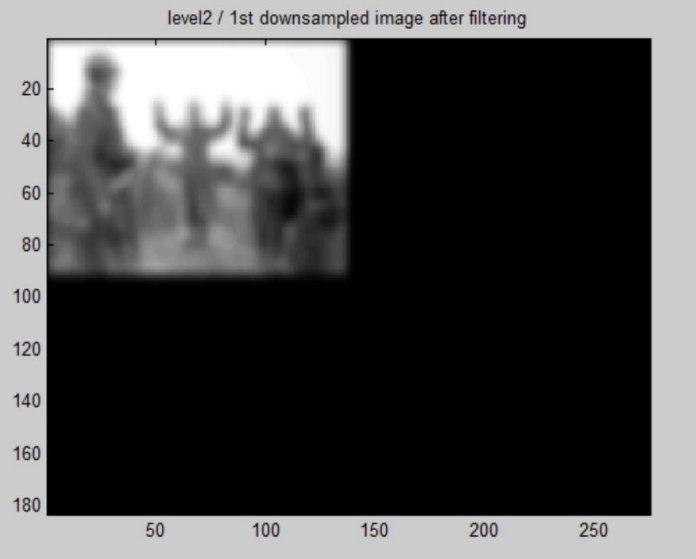
Testing **CS5320_G_pyramid**

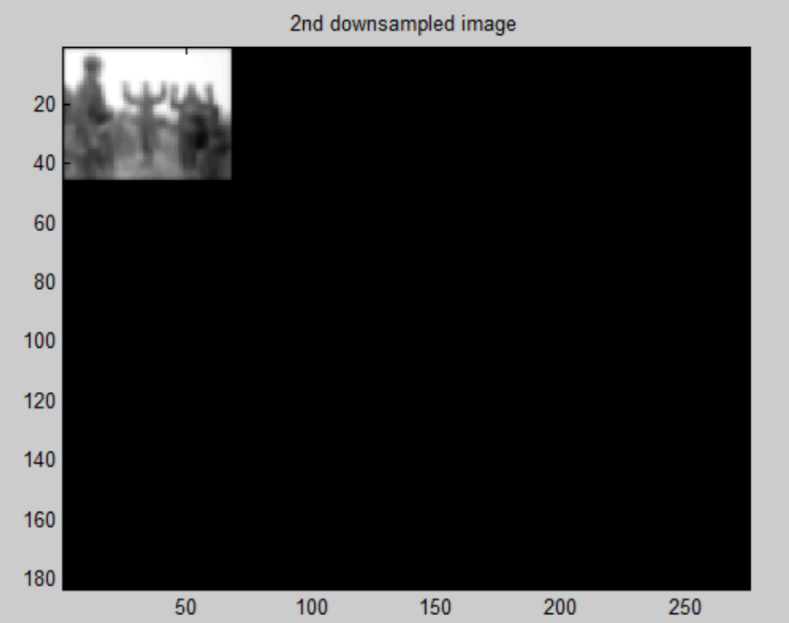
- The image should be down sized correctly and it should be blurred. As we can see the below images this is happening. It's size is decreased to exactly half and blurring is taking place.

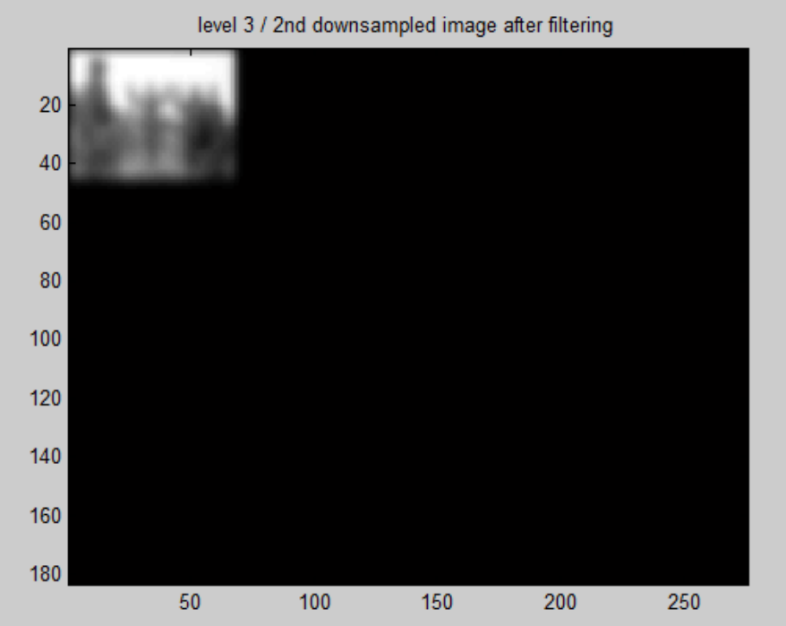

Original image
20
40
60
80
100
120
140
160
180
50
100
150
200
250


level1 / Original image after filtering
20
40
60
80
100
120
140
160
180
50
100
150
200
250


1st downsampled image
20
40
60
80
100
120
140
160
180
50
100
150
200
250


level2 / 1st downsampled image after filtering
20
40
60
80
100
120
140
160
180
50
100
150
200
250





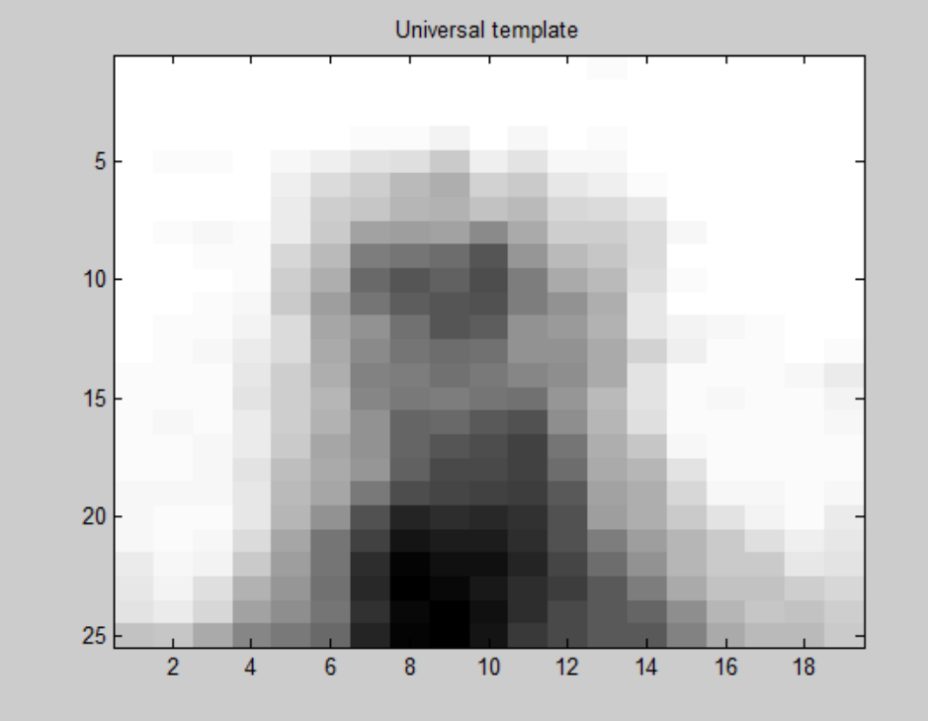
Like wise Level 4 and level 5 are being obtained correctly.

Testing **CS5320_hands**

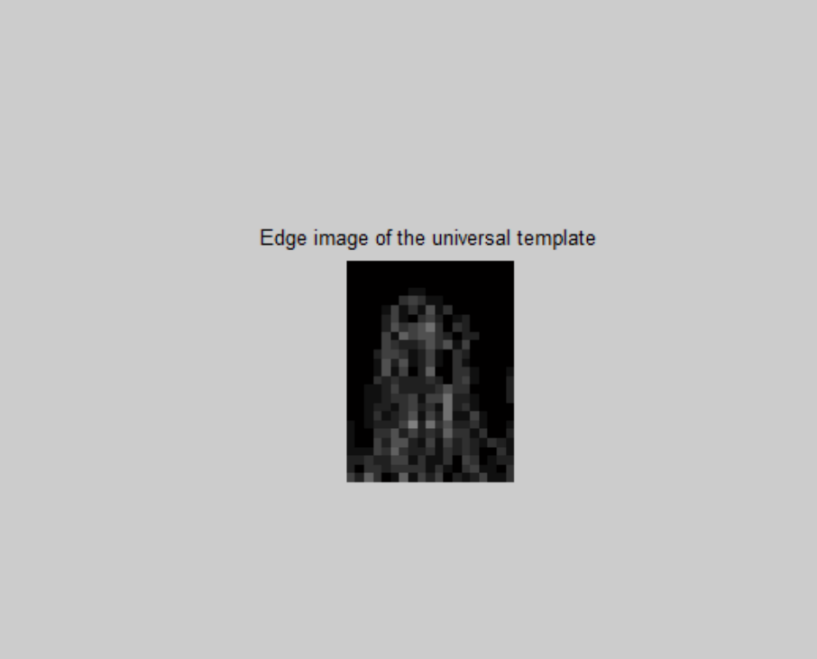
- This function serves to make hand red between reduced template and reduced image. By many observations, it has been observed that the color of the hand does change when normal correlation (done between reduced template and reduced image) within in this function comes out to be greater than the threshold.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

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithoutMean() function

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithoutMean() function

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithoutMean() function

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithoutMean() function

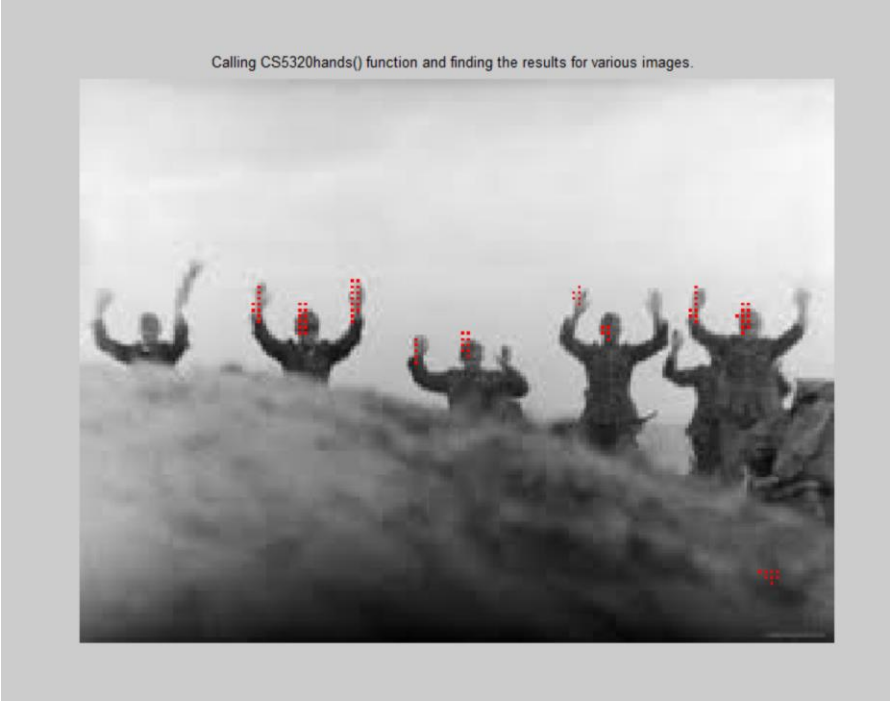
2) Directly doing the normal correlation between the images and the universal template using the CS5320_normcorr_withMean function. Please note that I had to vary the threshld for getting good results with this direct normalization correlation technique, since it is not a good technique. Other techniques like Gaussian pyramid and edge detection will have threshold fixed.

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithMean() function

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithMean() function

Directly doing the normal correlation between the images
and the universal template using CS5320normcorrwithMean() function

3) 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.



Calling CS5320hands() function and finding the results for various images.

Calling CS5320hands() function and finding the results for various images.

4) 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.

Calling CS5320edgemethod() function and finding the results for various images

Calling CS5320edgemethod() function and finding the results for various 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

2) **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**

3) **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**

4) **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