

**Introduction to Computer Vision**

**Coursework**

**Submission 2**

**Your name \_\_\_\_\_\_\_\_ HAO BAI \_\_\_\_\_\_\_\_**

**Student number \_\_\_\_\_\_\_\_201218765\_\_\_\_\_\_\_\_\_\_**

**Question 4(a)**

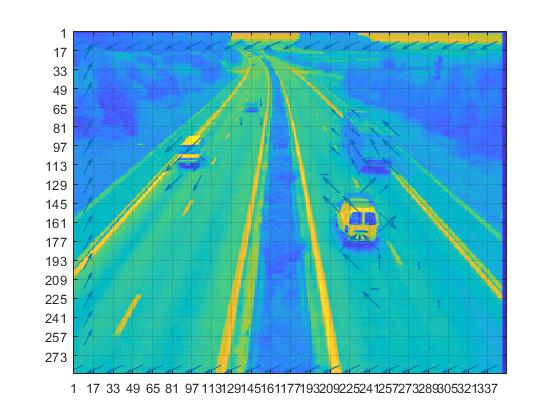
It+1



It



Motion field of It+1



**Question 4(b)**

Pt+1



It+1



**Your comments**

**How to use this function:**

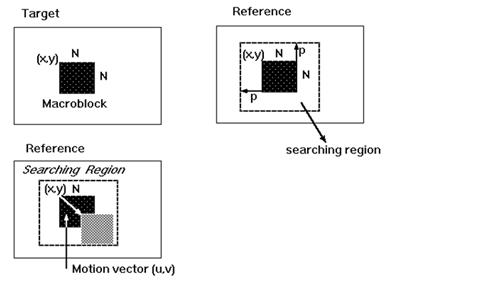
Run *cell 4-a* and *cell 4-b* in *ICV\_Assignment2\_main\_file.m* file.

**The algorithm in this cell is that:**

1. The function will load the video dataset(*DatasetC.mpg*) and find the 9th and 10th frame as frame It and It+1.
2. The block size is 16\*16 the search windows size is 20\*20.
3. Using the *ICV\_blockMatch ()* to estimatie the motions, this function required enter a the *It* and *It+1* images and *block size* and *search windows size*.
4. To predicte the *It+1* images, the ICV\_predictionOfFrame () can predicte the *It+1* images as *Pt+1*. The function required enter a the *It* and *It+1* images and *block size* and *search windows size*.

The algorithm of the using functions in these two cells:

*ICV\_blockMatch ()*



1. Convert the images matrix from *uint8* to *double*.
2. To divide the whole image into several equally sized non-overlapping blocks, the function will cut the extra part of the image.
3. The function cut the image into equally sized non-overlapping block according to the block size. The loop will begin with the first pixel of the image end with the last pixel of the image, the step size is the block size.
4. Each block matrix size is equal to *img(i:i+block\_size-1, j:j+block\_size-1)*
5. Store the block in *img\_block(block\_i, block\_j).matchBlock* , the block\_i and block\_j is the count that the number of the blocks. And store the image location in the *img\_block(block\_i, block\_j). loc.*
6. Find the size of the searching windows. The block is in the center of the searching windows, when the beginning of the block is 1 or the ending is the last number of the image, the search windows will out of bounds.
7. To fix that, the function will check the location of the block. if block is on the boundary of the image, the searching windows size will smaller size to fit the block window (delete the out of bound part).
8. The reference of the search windows is the *It+1* image.
9. The searching window moving the block *pixel by pixel*, and calculate each blocks’ mean square error.
10. And then find the minimum mean square error in the searching window, and it’s location.
11. Using the minimum mean square error location to calculate the pointer location.
12. Then after all block has been calculate, the function will be using *quiver* function to draw the motion estimate pointer. Using the pointer location and the block size. And show in image.

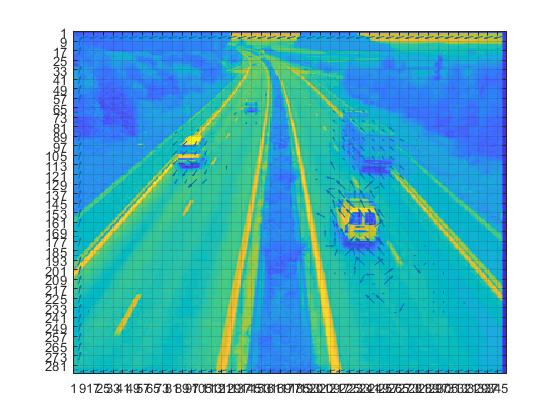
*ICV\_predictionOfFrame*

1. This function when doing the block matching and motion estimate is same as the *ICV\_blockMatch* function.

**Question 4(c)**

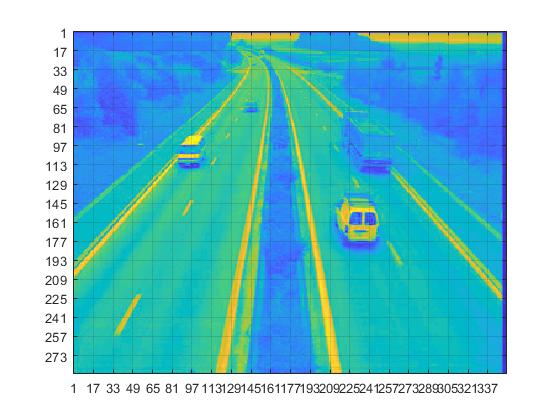
Pt+1

Block size = 8x8



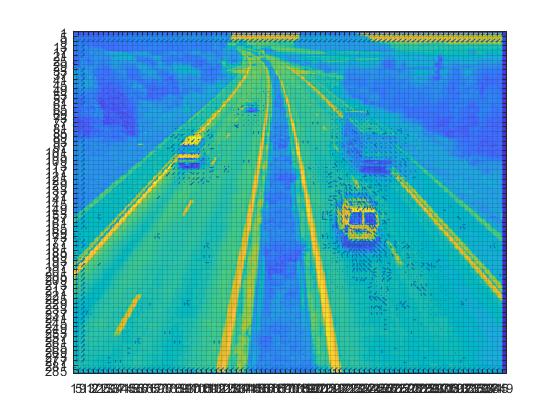
Pt+1

Block size = 16x16



Pt+1

Block size = 4x4

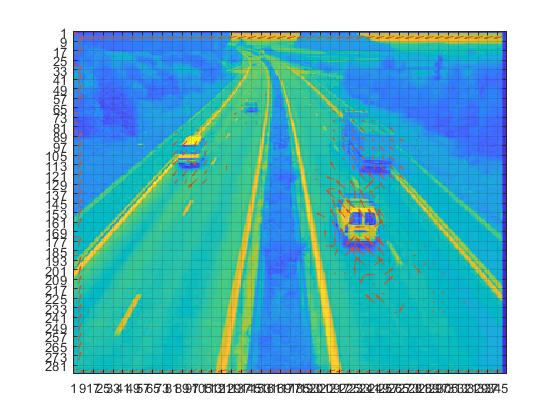


**Your comments:**

**Question 4(d)**

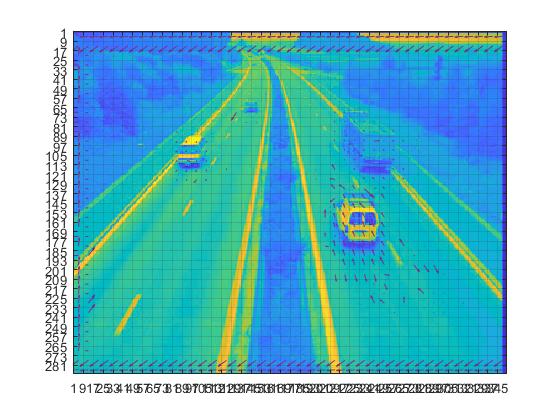
Pt+1

Window size = 16x16



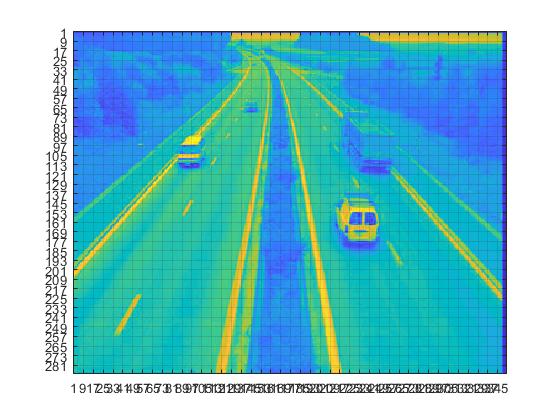
Pt+1

Window size = 32x32



Pt+1

Window size = 8x8



**Your comments:**

**Question 4(e)**

**Plot graphs:**

Time versus window size

Time versus block size

**Your comments:**

**Question 5(a)**

**Original frames:**

Selected frame 2



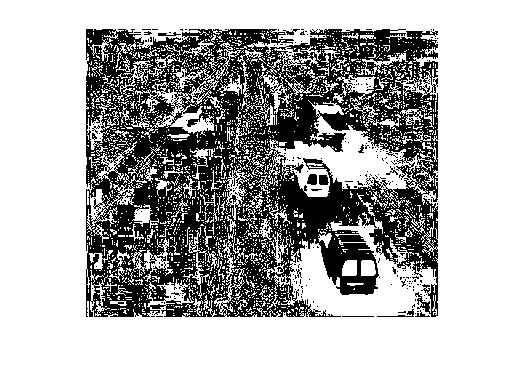
Selected frame 1

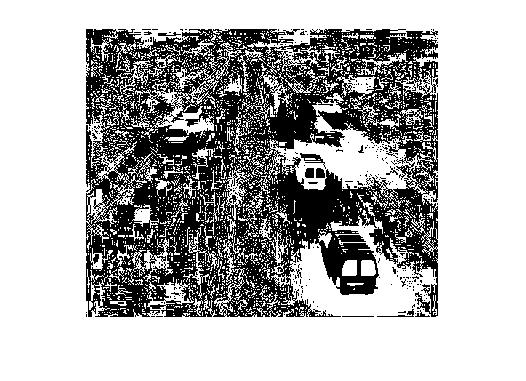


Reference frame

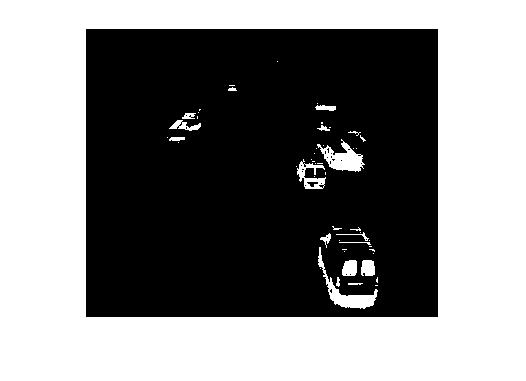


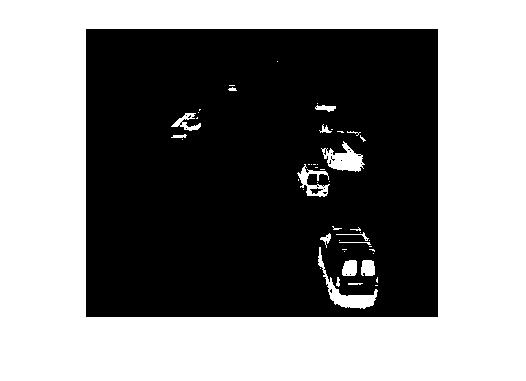
**Frame differencing:**





**Threshold results:**





**Your comments:**

**Threshold = 40**

**Question 5(b)**

**Original frame:**

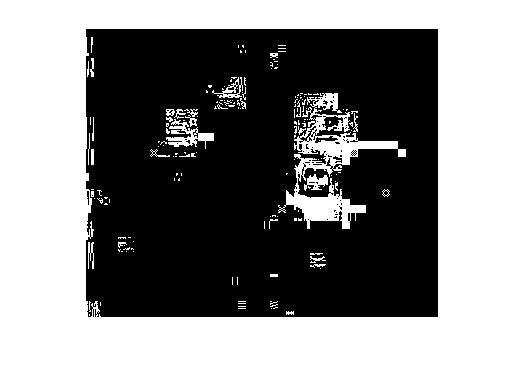
Selected frame 1

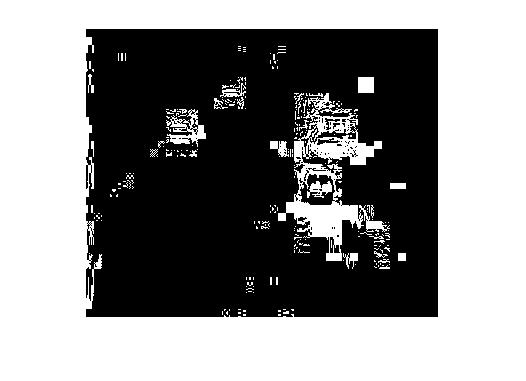


Selected frame 2

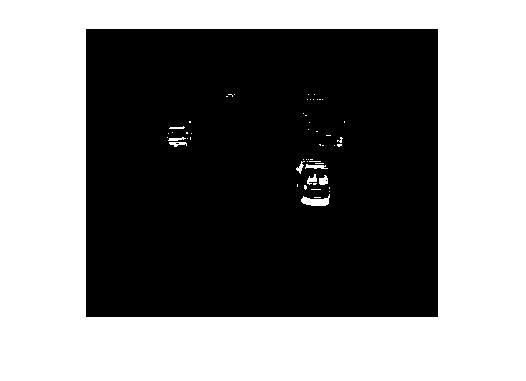


**Frame differencing:**





**Threshold results:**





**Your comments for 5a,5b:**

**Question 5(c)**

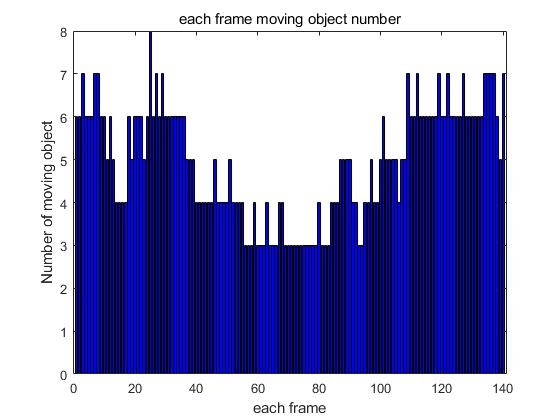
Generated background



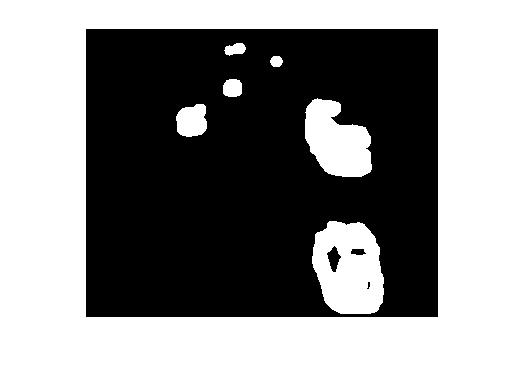
**Your comments:**

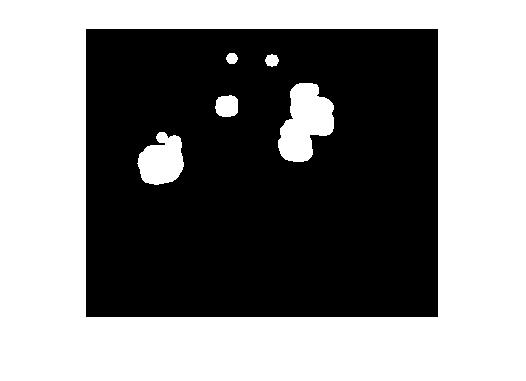
**Question 5(d)**

Bar plot



**Your comments:**

****

****

**Question 6(a)**

**Three non-consecutive windows**

W3



W2

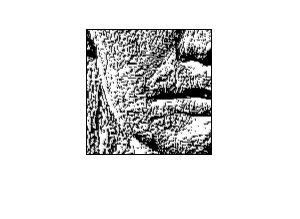


W1

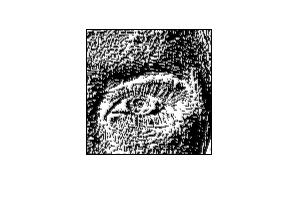


**LBP of windows**

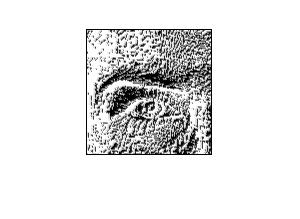
LBP3



LBP2

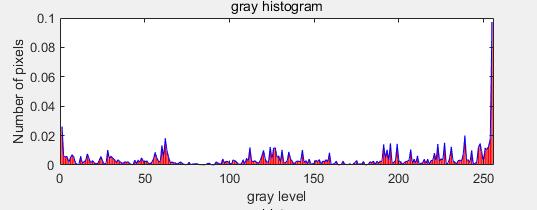


LBP1

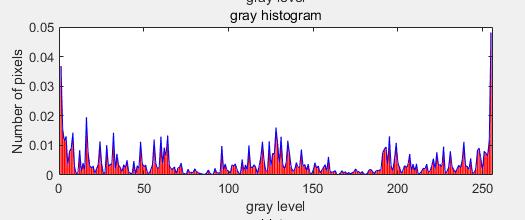


**Histograms of LBPs**

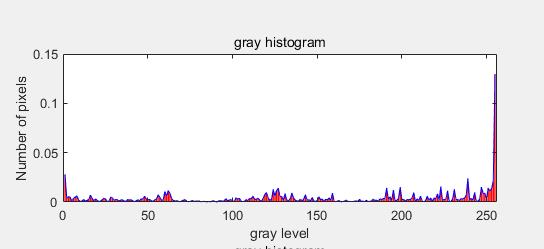
H3



H2



H1



How to show the result:

Run the ***cell 6- a*** in ***ICV\_Assignment2\_main\_file.m*** file.

The algorithm in this cell is that:

1. Load the face image(*face-2.jpg*) and convert to grayscale mode using *rgbgray* function
2. Set the *Block size* = 128, that means each block windows is 128 pixels \* 128 pixels. That divided the face image into 4 small windows.
3. Using *ICV\_divideIntoNonOverlapping* functions. That required input an *image* and the *block size.* And return a group of non-overlapping images that divided from the images.
4. Using *size ()* function to get how many blocks, the image has been divided.
5. The loop is to read each blocks and handle.
6. The *ICV\_LBPfunction* is a function that required input a image and return a LBP image.
7. Then function *ICV\_hisgram* will print the histogram of the frequency of the numbers, the required input is the block images and number of the all block images and the current count number. That will help the function draw the bar chart together.
8. Each of the histogram will as the feature descriptor of each block.

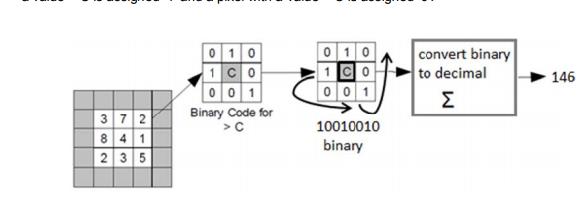
The algorithm of the using functions:

*ICV\_divideIntoNonOverlapping:*

1. This function will divide the big image into small piece of the non-overlapping images as the return value accordong to the block size needed.
2. The function will get to know the length and width of the image
3. sometimes the image cannot cut into equally sized non-overlapping block as the asked size, the function will cut extra length or width.
4. The function will divide the image into small block, the step size is the block size. Each block will have stored in the return value.
5. The block size is 128 pixels, so the return value will be a 128\*128\*4 matrix.

*ICV\_LBPfunction*

1. This function compares 8 of each pixel neighbors with the pixel self. If the number larger or equals to the pixel, then the neighbors number equals to 1. Else, the neighbors number will equals to 0.
2. And then each pixels equals to the numbers that converted counter-clockwise neighbors binary to decimal number.



*ICV\_LBPfunction*

1. The function will count the of the gray level and the total number of the pixels.
2. The function will calculate the frequency of each numbers and using bar chart to show.
3. Normalize the histogram and show in the same sub-graph.

**Question 6(b)**

**Two example images:**

Car image

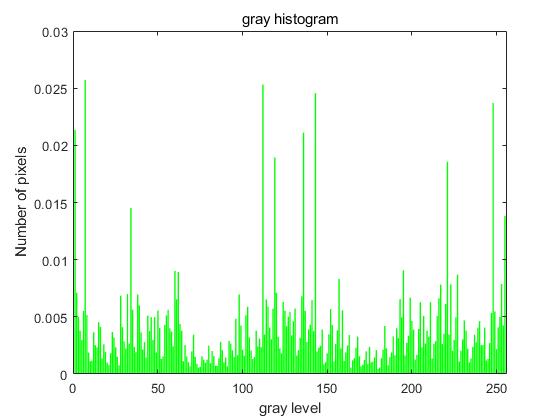


Face image

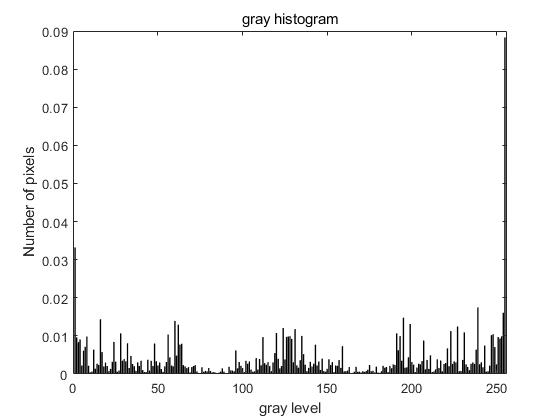


**Descriptors:**

Car descriptor



Face descriptor



**Your comments:**

How to show the result:

Run the ***cell 6- b*** in ***ICV\_Assignment2\_main\_file.m*** file.

The algorithm in this cell is that

1. Load the face and car image (*face-2.jpg and car-1.jpg*) and convert to grayscale mode using *rgbgray* function
2. Set the *Block size* = 128, that means each block windows is 128 pixels \* 128 pixels. That divided the face image into 4 small windows.
3. Using *ICV\_divideIntoNonOverlapping* functions. That required input an *image* and the *block size.* And return a group of non-overlapping images that divided from the images.
4. Using *size ()* function to get how many blocks, the image has been divided.
5. The loop is to read each blocks and handle.
6. The *ICV\_LBPfunction* is a function that required input a image and return a LBP image.
7. Then function *ICV\_hisgram* will print the histogram of the frequency of the numbers, the required input is the block images and number of the all block images and the current count number. That will help the function draw the bar chart together.
8. Each of the histogram will as the feature descriptor of each block.
9. Combine each of the sub-histogram in to one histogram that is the descriptor of the whole image.
10. To combine the frequency of each of the number occur, the function will add all of the frequency first and divide by the number of the blocks.

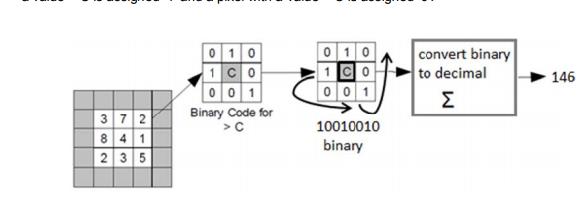
The algorithm of the using functions (same as the 6 -a):

*ICV\_divideIntoNonOverlapping:*

1. This function will divide the big image into small piece of the non-overlapping images as the return value accordong to the block size needed.
2. The function will get to know the length and width of the image
3. sometimes the image cannot cut into equally sized non-overlapping block as the asked size, the function will cut extra length or width.
4. The function will divide the image into small block, the step size is the block size. Each block will have stored in the return value.
5. The block size is 128 pixels, so the return value will be a 128\*128\*4 matrix.

*ICV\_LBPfunction*

1. This function compares 8 of each pixel neighbors with the pixel self. If the number larger or equals to the pixel, then the neighbors number equals to 1. Else, the neighbors number will equals to 0.
2. And then each pixel equals to the numbers that converted counter-clockwise neighbors binary to decimal number.



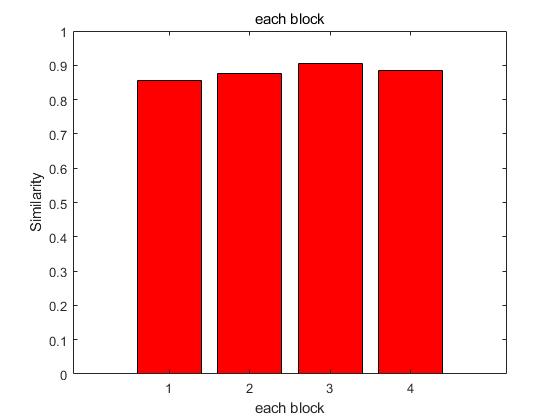
*ICV\_LBPfunction*

1. The function will count the of the gray level and the total number of the pixels.
2. The function will calculate the frequency of each numbers and using bar chart to show.
3. Normalize the histogram and show in the same sub-graph.

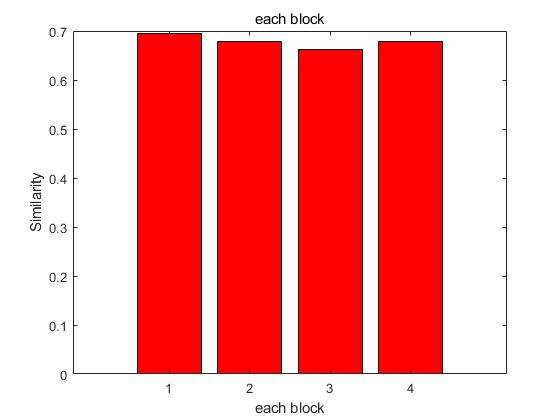
**Question 6(c)**

**Block diagram of classification process**

**Face image each block descriptor and the whole image descriptor intersection value.**

****

**Car image each block descriptor and the face whole image descriptor intersection value.**

****

**Your comments:**

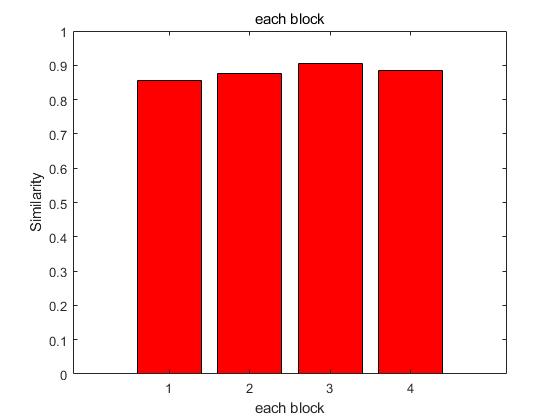
To classify each of the block, the script will calculate the intersction of the two histograms (*the his tograms generated from the pervious cell 6- b.)* of the car image and the face image (*face-2.jpg and car-1.jpg*) first. The intersection value is the sum of each bars value (The two histograms of each image is the probbility.), the two-image intersection value is 0.6854.

The intersection value is calculate by:

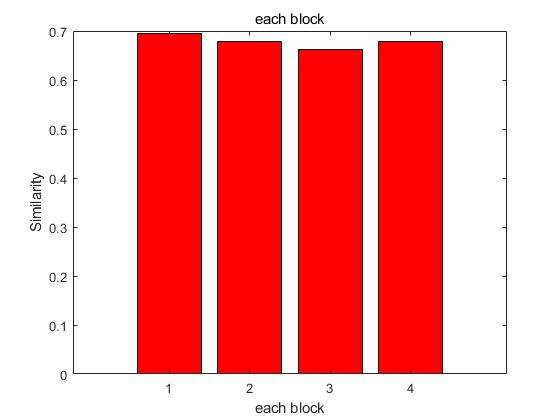


Then calculate two different image the block histograms and the whole face histogram intersection value. According to the graph shows that:

1. ***Face image each block descriptor and the whole image descriptor intersection value.***

****

1. ***Car image each block descriptor and the face whole image descriptor intersection value.***

****

The graph shows that the same image different block intersection value is higher than 0.8. If using the different image blocks, the intersection value will be lower than 0.7. So that the block classify can be done by calculating the intersection value of each block and the whole image.

**Question 6(d)**

**Your comments:**

When decreasing the windows size, the size decreases to 64 pixels \* 64 pixels block.

****

The two-image (*face-2.jpg and car-1.jpg*) intersection value is not changed (0.6854). How ever the intersection value of the small block (*64 pixels \* 64 pixels block*) decrease:

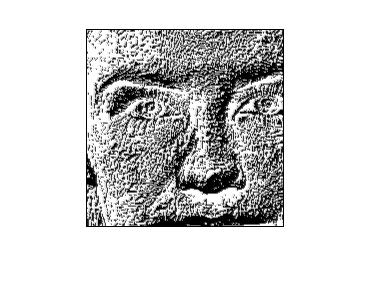
1. *The intersection value of the face image histogram and the first block of the face image is: 0.6789*
2. *The intersection value of the face image histogram and the first block of the car image is: 0.4843*

That shows that the small windows size will only decrease the intersection value of the image and the block histogram. The number of the intersection values can be used to classify the images.

**Question 6(e)**

**Your comments:**

When increase the windows size, the size increase to 200 pixels \* 200 pixels block.

****

1. The intersection value of the face block and face image increase is 1.0
2. The intersection value of the car block and face image increase is 0.6897

When the block size is 128\*128 pixels the intersection value of the face block and face image is larger than 0.8, and the intersection value of the car block and the face image is always lower than 0.7. It shows that as the block size increase the intersection value of the image and blocks increases and the accuracy of the classify increase.

**Question 6(f)**

**Your comments**