Introduction to Computer Vision

Coursework

Submission 2

Your name ___ HAO BAI

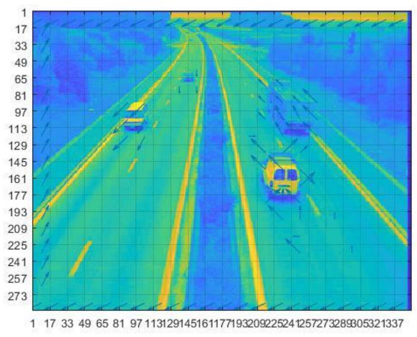
Student number 201218765

Question 4(a)

I_t

I_{t+1}

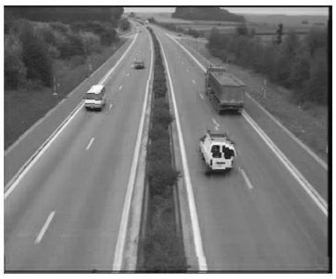
Motion field of I_{t+1}



Question 4(b)

I_{t+1}

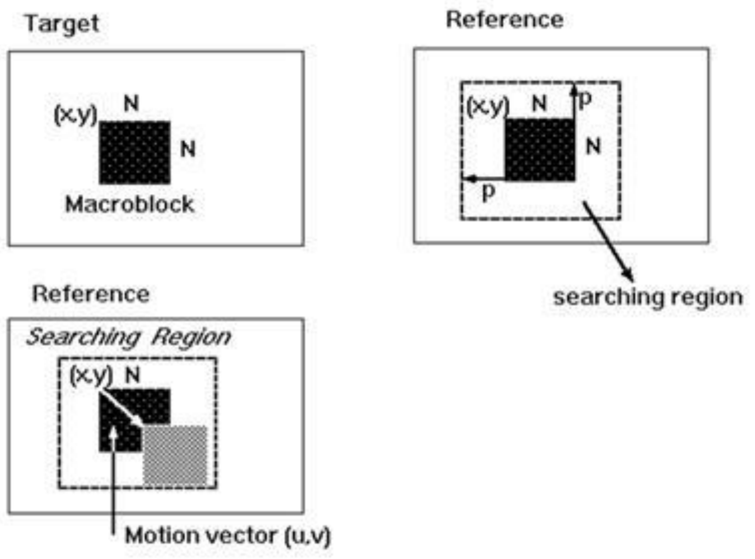
P_{t+1}



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 I_t and I_{t+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 I_t and I_{t+1} images and *block size* and *search windows size*.
4. To predicte the I_{t+1} images, the ICV_predictionOfFrame () can predicte the I_{t+1} images as P_{t+1}. The function required enter a the I_t and I_{t+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 I_{t+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.

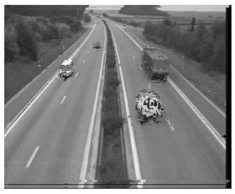
ICV_predictionOfFrame

1. This function when doing the block matching and motion estimate is same as the *ICV_blockMatch* function.
2. To predicte the next frame, the block will move to the estimated location.
3. If the location is out of bound, the function will cut the extra block.
4. If the pointer location is not equals to 0, the image block will over-write the pointer location block.

Question 4(c)

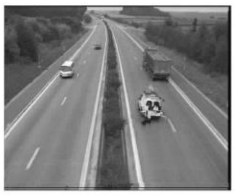
P$_{t+1}$

Block size = 4x4



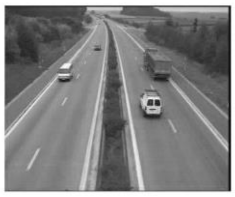
P$_{t+1}$

Block size = 8x8



P$_{t+1}$

Block size = 16x16



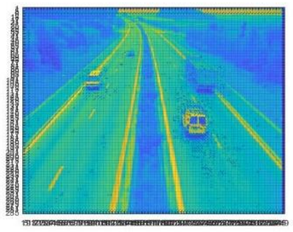
Your comments:

How to use this function:

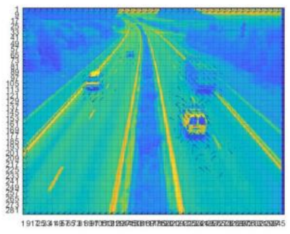
Run *cell 4 - c : 16x16 search window, block_Size 4x4* and *4 - c : 16x16 search window, block_Size 88* and *4 - d : 88 search window, block_Size 88* in *ICV_Assignment2_main_file.m* file.

According to the result:

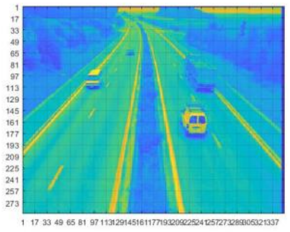
block 4*4:



block 8*8:



block 16*16:



The smaller block size will cause accurate moving object location calculate. But the object will be breaked into pieces. That because more pointer will be included in the motion estimating. The block size that same as the search windows size will cause can not making the motion estimating. So the predition frame will same as the previous frame.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 4(d)

Pt+1

Window size = 8x8

Pt+1

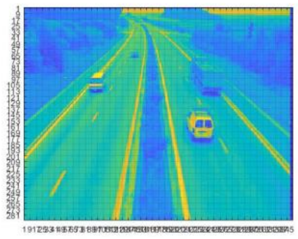
Window size = 16x16

Pt+1

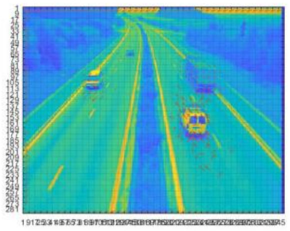
Window size = 32x32

Your comments:

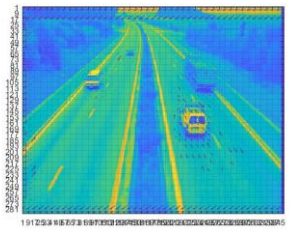
Windows size 8*8



Windows size 16*16



Windows size 32*32

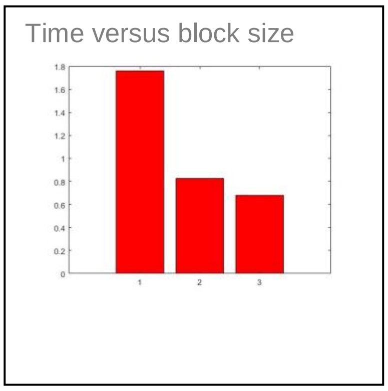


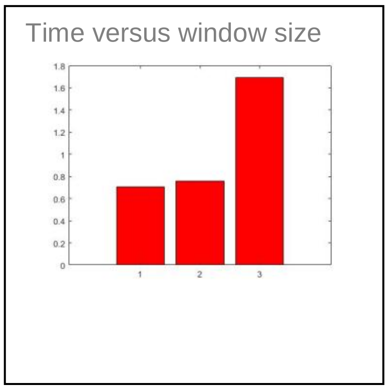
The block size that same as the search windows size will cause can not making the motion estimating. So the predition frame will same as the previous frame. The biger size searching windows will generate accurate result than the smaller sized searching windows. More accurate pointer generated.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 4(e)

Plot graphs:





Your comments:

Block size	**Search windows size**	**Time**
4*4	**16*16**	**1.761944s**
8*8	**16*16**	**0.824722s**
16*16	**16*16**	**0.679418s**

Block size	**Search windows size**	**Time**
8*8	**8*8**	**0.705818s**
8*8	**16*16**	**0.759334s**
8*8	**32*32**	**1.690244s**

As the graph shows that, the smaller block size the more time needed. The biger search windows size the more time needed. That because the smaller size block windows and biger size search window need more loop to calculate the pointers, and amount of the pointers are more.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 5(a)

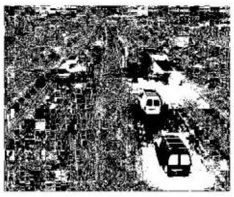
Original frames:

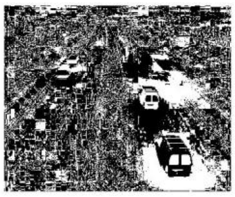
Reference frame

Selected frame 1

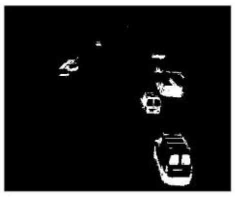
Selected frame 2

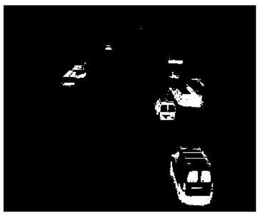
Frame differencing:





Threshold results: Threshold = 40





Your comments:

How to use this function:

Run *cell 5-a* and *cell 5-b* in *ICV_Assignment2_main_file.m* file.

The algorithm in this cell is that:

To calculate the frame differencing and apply the classification, function *ICV_captureMovingObjectFirstFrame* taken to be used. This function required a video path and the threshold.

The algorithm of the using functions:

ICV_captureMovingObjectFirstFrame

1. Loading the video and get the amount of the frames
2. Looping from the first frame to the last-1 frame.
3. Compare each pixel of the two frames, in this function compare the first frame and other frames.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

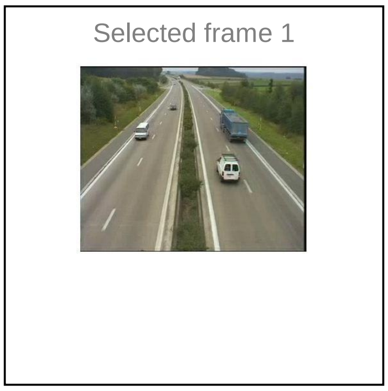
4. If in same location, two frames are different. And the different number is less than the *classification threshold* number. This location of pixel is the moving object. And set the color of the pixel equals to white (255). If the location of the two frame is same or less than the *classification threshold* the color of the pixel is set to black (0).
5. Show the calculated images.

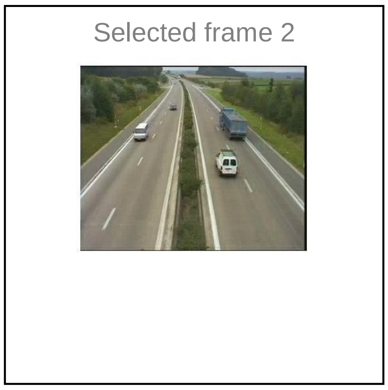
Observed
The higher *classification threshold,* the smaller and gatherd the moving objects. Both of the 1st frame cars and the ith frame cars will shown on the resulting image.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

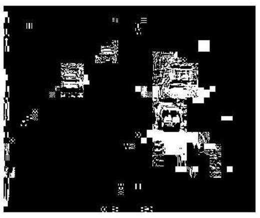
Question 5(b)

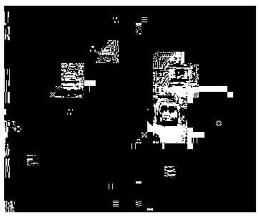
Original frame:



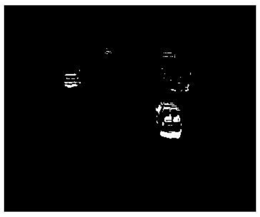


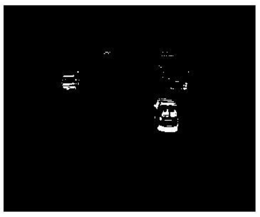
Frame differencing:





Threshold results:





Your comments for 5a,5b:

How to use this function:
Run *cell 5-a* and *cell 5-b* in *ICV_Assignment2_main_file.m* file.

The algorithm in this cell is that:
To calculate the frame differencing and apply the classification, function *ICV_captureMovingObjectPreviousFrame* taken to be used. This function required a video path and the threshold.

The algorithm of the using functions:
ICV_captureMovingObjectPreviousFrame
1. Loading the video and get the amount of the frames

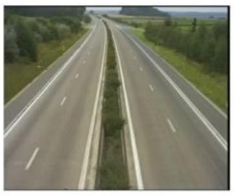
2. Looping from the first frame to the last-1 frame.
3. Compare each pixel of the two frames, in this function compare the i frame and i+1 frames.
4. If in same location, two frames are different. And the different number is less than the classification threshold number. This location of pixel is the moving object. And set the color of the pixel equals to white (255). If the location of the two frame is same or less than the classification threshold the color of the pixel is set to black (0).

Observed
Both frame cars have shown on the resulting image, the resulting image cars are are both the I frame car and I+1 frame car.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 5(c)

Generated background



Your comments:

How to use this function:
Run *cell 5-c* in *ICV_Assignment2_main_file.m* file.

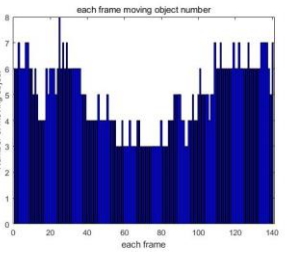
The algorithm in this cell is that:
Using the function *ICV_generateBackground(video_path)* to calculate the background image. This function required a video path.

The algorithm of the using functions:
ICV_generateBackground(video_path)
1. Load the video and get the frame number.
2. Convert the video each frame from uint8 to double and add each location pixel together.
3. Taking the average number of each frames.
4. Show the background.

Observed
If many cars crossing the road in a very fast speed, this function can not taking a background image correctly.

Question 5(d)

Bar plot



Your comments:

How to use this function:
Run *cell 5-d* in *ICV_Assignment2_main_file.m* file.

The algorithm in this cell is that:
1. Using the *cell 5-c* taking the background
2. Using the function *ICV_countMovingObject* to calculate the moving object. This function required a background image of the video and a video path and a threshold value.

The algorithm of the using functions:
ICV_countMovingObject
1. The function will load the video and get the number of frames.
2. Use a loop to get each of the frame in the video.
3. For each frame if the pixel difference number is higher than the threshold value the pixel will set to white color (255) else the pixel will set to black(0).
4. Using *ICV_filter77* to blur the image 2 times the filter using 7*7 kernal that the pixel neighbor has white color block that pixel will not a black pixel
5. Check each pixel in the after filtered image, if the pixel is not totally black(equals to 0), then set the pixel to white.

6.
7. The scattered white area will be gathered togather.
8. Using *ICV_countHowManyConnectInMatrix* to get the the amount of the connected white area.
9. The function will return a number.

ICV_countHowManyConnectInMatrix
1. Loading the images, and findind white pixels.

2. When finding a pixel, the count number will add one, and using *ICV_findConnect* function to fill the images.
3. The functon will return a new image matrix as the image.

ICV_findConnect

1. This function will find the 4 neighbors of the pixel, if the neighbor is black then set the neighbors pixel value equals to the count value, and then *recall this function* on the neighbors location.

The *ICV_countHowManyConnectInMatrix* works like this:
In this example using 1 as 255, because 255 not easy to format

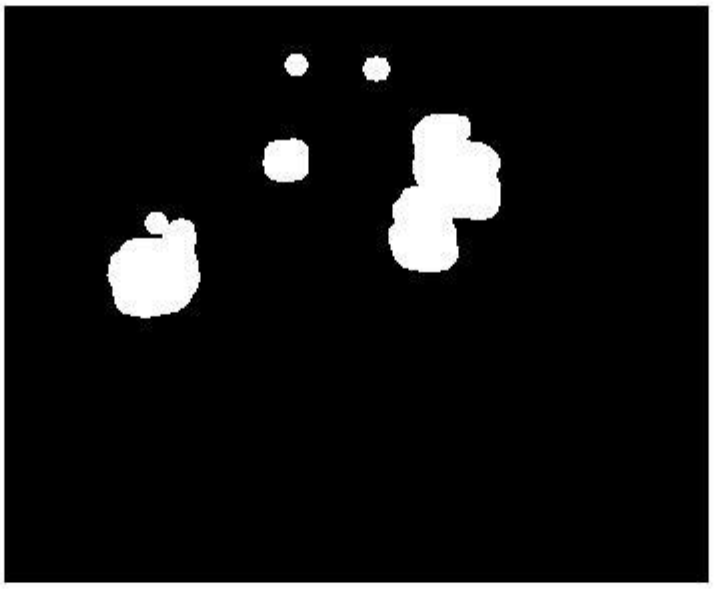
```
0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 1 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

After allpy the function the matrix will be like this, the largest number is the count.

```
0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 2 0 0 0 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0 3 0 0 0
0 0 0 0 2 0 0 0 0 0 3 0 0 0
0 0 0 0 0 0 0 0 0 0 3 0 0 4
0 0 0 0 0 0 0 0 3 3 3 0 0 4
0 0 0 0 0 0 0 0 0 0 0 0 0 4
```

Observed

The algorithm works well but including some errors, that of two car locations too close, the counting will count the two object as one, because the two car in the after filtering image is connected. The calculating takes long times.



Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 6(a)

Three non-consecutive windows

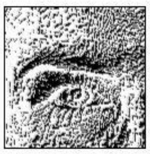
W1

W2

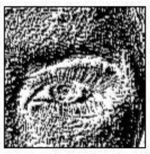
W3

LBP of windows

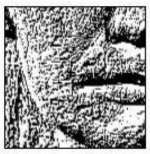
LBP1



LBP2

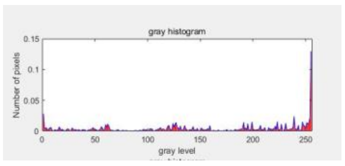


LBP3

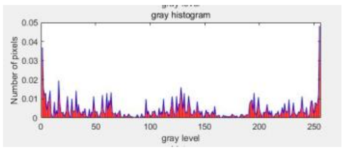


Histograms of LBPs

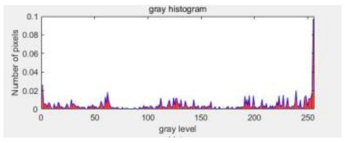
H1



H2



H3



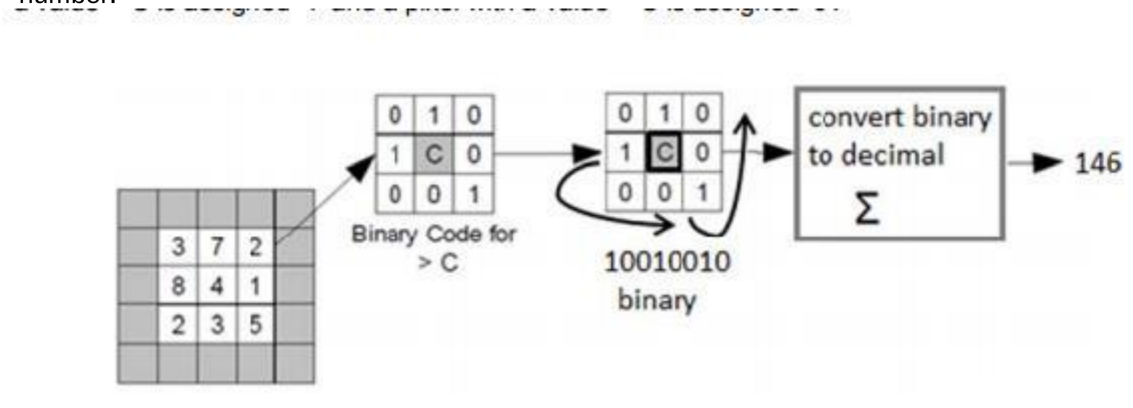
Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

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.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 6(b)

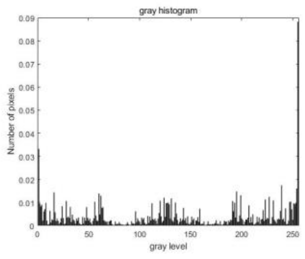
Two example images:

Face image

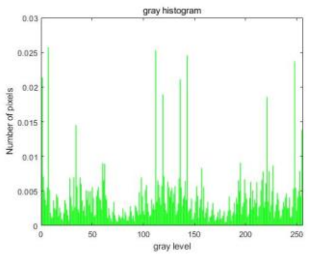
Car image

Descriptors:

Face descriptor



Car 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:

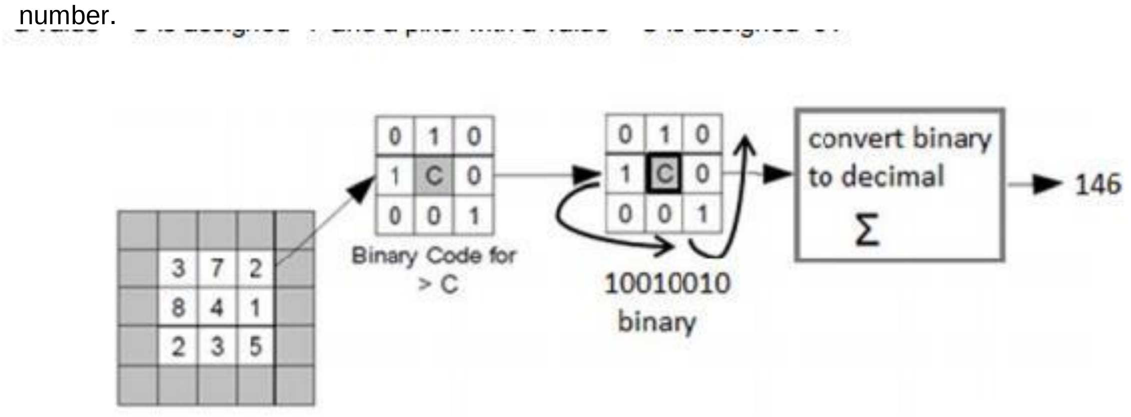
6. This function will divide the big image into small piece of the non-overlapping images as the return value accordong to the block size needed.
7. The function will get to know the length and width of the image

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

8. sometimes the image cannot cut into equally sized non-overlapping block as the asked size, the function will cut extra length or width.
9. 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.
10. The block size is 128 pixels, so the return value will be a 128*128*4 matrix.

ICV_LBPfunction

3. 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.
4. And then each pixel equals to the numbers that converted counter-clockwise neighbors binary to decimal number.



ICV_LBPfunction

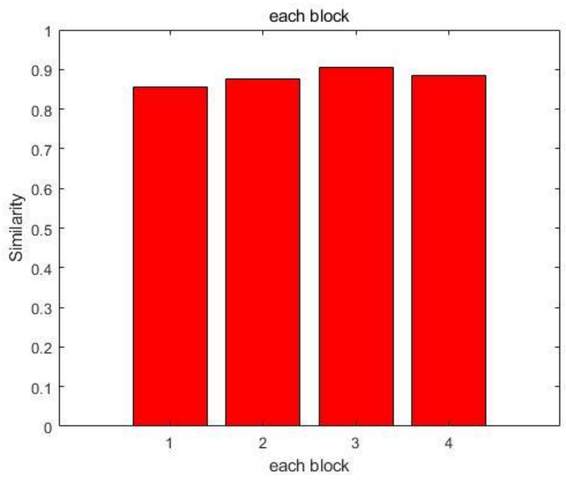
4. The function will count the of the gray level and the total number of the pixels.
5. The function will calculate the frequency of each numbers and using bar chart to show.
6. Normalize the histogram and show in the same sub-graph.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

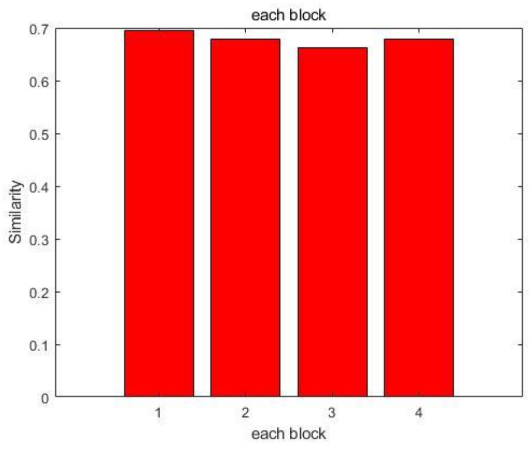
Question 6(c)

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.

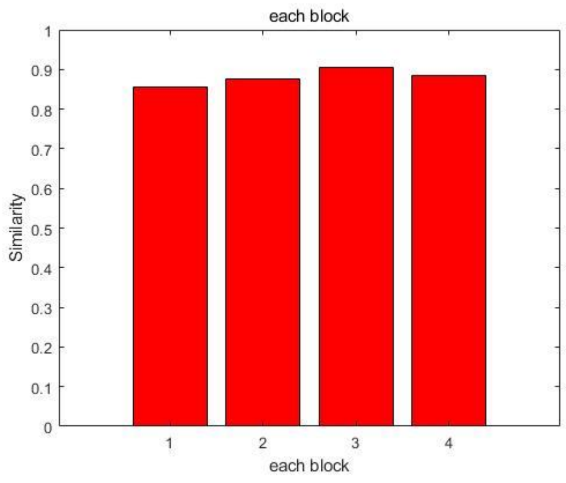


Your comments:

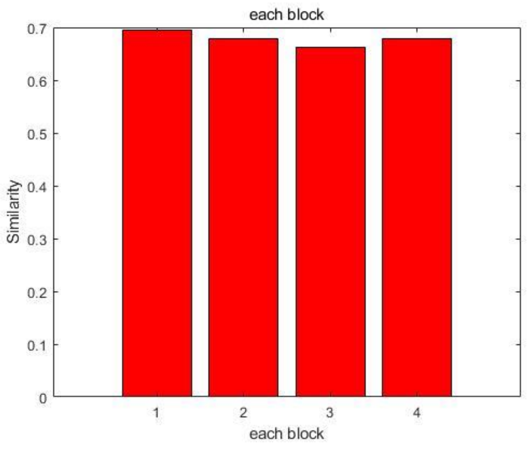
To classify each of the block, the script will calculate the intersction of the two histograms (*the his tograms generated from the pervious cell 6- b.)* of the car image and the face image (*face-2.jpg and car-1.jpg*) first. The

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:
$\sum_{j=1}^{n} \min(I_j, M_j)$
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.



2. 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.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file

Question 6(f)

Your comments

The LBP function using for dynamic texture analysis, can be used in the face identification. That the LBP can store a sequence of the moving face texture histogram to identificate if a face is the correct face.

Fill the available spaces for your answers. Do not extend the spaces and do not change the formatting of the pages. Use the same font size and the same number of lines as in the original file