Queen Mary
University of London

Introduction to Computer Vision

Coursework

Submission 2

Your name Wei-Shiang Lian

Student number 170509682

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

There are several barriers to achieve the goal of this task. Firstly, according to the block size, a frame from a video sequence is divided into different blocks, and each block has its search window for finding the similar part between two frames. However, there is a situation that when search window size is bigger than block size and the blocks are on the edge of the frame; the search window may be located beyond the boundary of the frame so that it will catch an error when the program tries to find the comparable sector. Therefore, my solution to solve this problem is that using conditional statements for every block when deciding the start and end points. Specifically, I find the minimum value between the search range and the left-top pixel position, and likewise, find the minimum value between the search range and the right-bottom pixel position to prevent the search pixel exceeding the boundary. Secondly, based on the motion vector found from two frames by block matching method, it can predict the next frame, but there are some questions to solve, one of them is the remained blank blocks when the predicted objects move toward their direction from estimated motion vector. Nevertheless, it can be fixed by filling the blocks with the same position from origin frame, or it can be filled the blocks which are handled by motion vector as well. Therefore, I choose the latter to resolve this problem because of it is the better one.

Question 4(c)

P_{t+1}

Block size = 4x4

P_{t+1}

Block size = 8x8

P_{t+1}

Block size = 16x16

Your comments:

According to the above results, it is obviously to see that lots of noisy blocks in the predicted frame when the block size is 4x4. The reason is that block is too small so that its feature is not clear so that it can be matched with some blocks not the predicted one. Next, the frame predicted by 8x8 block size is less noisy blocks than previous one but it still has some noises on the frame. Back to the motion vectors estimated before, it shows some blocks which are static but still detected between different frames and this situation becomes more serious when he block size is smaller. In my opinion, when the block size is bigger and it contains more information so that it can be matched accurately. However, it is not the bigger block size gets the better result. It should be sure the block size contains enough information first, and then adjust block smaller and it can get the more accurate movement to the motion vector. Meanwhile, this means more workloads for finding the similarity, in short, the size needs to be set based on the circumstance.

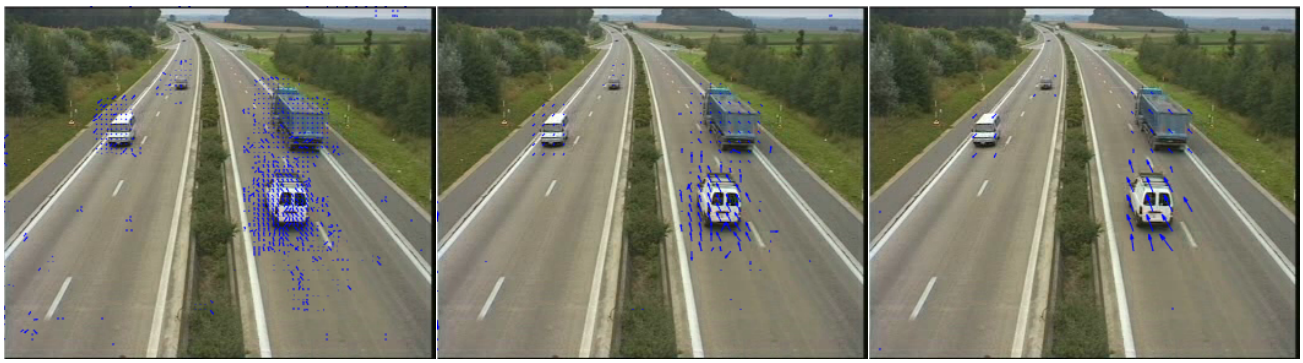


Fig 1. Motion fields for Pt+1 with block size 4x4, 8x8 and 16x16

Question 4(d)

P_{t+1}

Window size = 8x8

P_{t+1}

Window size = 16x16

P_{t+1}

Window size = 32x32

Your comments:

These effective situations by modifying the search window size are like the results in previous question part, and the reason is also similar. In the case of search window size is 8x8, which is smaller than the real movement it should be, the block can't move to the right position due to the small search window. It is better when window size increases, the reason is that there is more possibility that block can get the right position. However, it's not always the bigger the better, when windows size is added to 32x32, it shows some fragment on the frame, the reason is that there is more opportunity to find the wrong matched block. Furthermore, larger search window size means more workloads to execute loops to search every possible block. Finally, it depends on the environment to adjust the search window, one of them is the speed of cars can be a reference to find the best search window size.

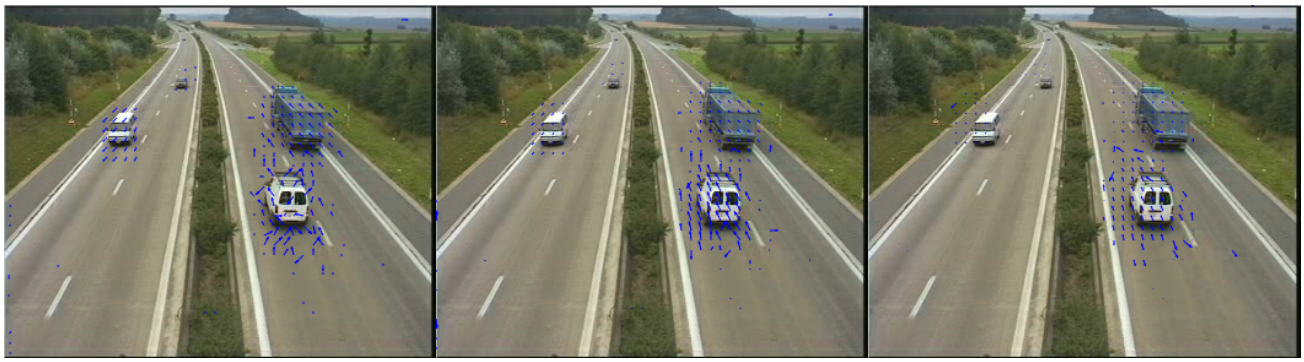
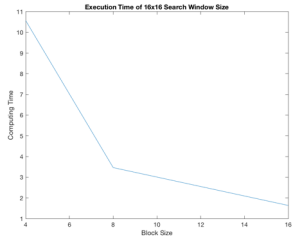


Fig2. Motion fields for Pt+1 with search window size 8x8, 16x16 and 32x32

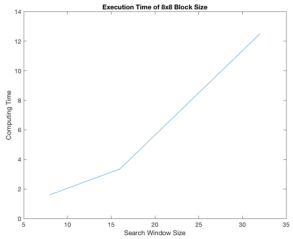
Question 4(e)

Plot graphs:

Time versus block size



Time versus window size



Your comments:

The above plots show that the execution time increases with the decrease of the of block size and the increase of the search window size. The results are reasonable because how many times the program need to execute, larger search window size with smaller block size causes great amount searching actions that computers should handle. It is not an easy way to find the balance between block matching with decided size and execution time.

Question 5(a)

Original frames:

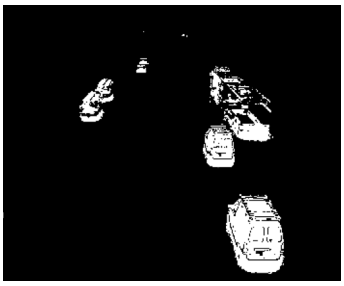
Reference frame

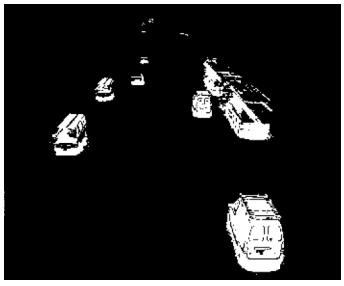
Selected frame 1

Selected frame 2

Frame differencing:

Threshold results:





Your comments:

It is obviously to see that the frame comparison by pixel-by-pixel, most of the pixels with same values compared to the reference frame are disappeared. However, the frame differences show that both cars remain on the result. It is easy to find the reason why they all stay because these objects area has different pixels' intensities. When it comes to the threshold of the image, the threshold determines the quality of objects detection because the higher threshold can reduce the noise but may cause the shape of objects is not apparent, and the lower threshold may cause many noises so that it can be distinguished. So, how to find the most suitable one is hard, but I find a way called "Otsu's method" to find the better threshold by calculating the model from image histogram. The define of this formula to find the best threshold is that dividing the image histogram into two groups and find the minimum variance position between these two groups by executing a loop from 1 to 256. Therefore, its index can correspond a number between 1 to 256, and that is the best threshold in this image.

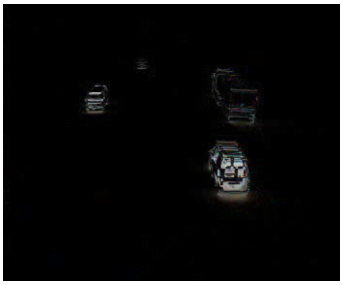
Question 5(b)

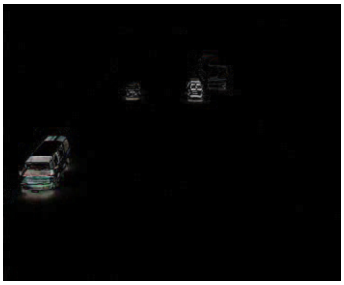
Original frame:

Selected frame 1

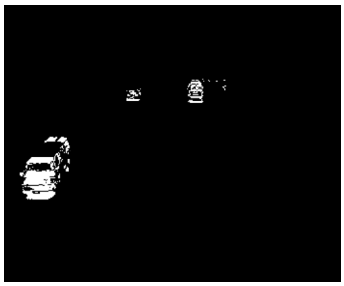
Selected frame 2

Frame differencing:





Threshold results:



Your comments for 5a,5b:

The results are better than the previous results because the objects in frame difference are more clearly. However, it can be observed carefully that the objects are combined with two frames and it can be worse when the objects move fast between consecutive frames. Therefore, it is not a perfect way to detect the objects from video.

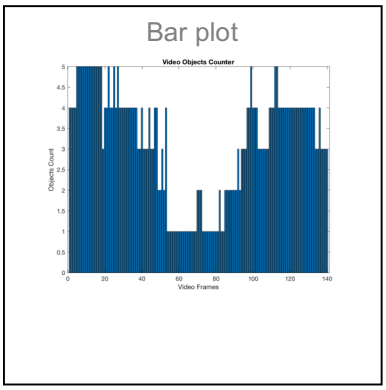
Question 5(c)

Generated background

Your comments:

It is amazing when I implement a weighted temporal averaging algorithm to sum all the pixels' value from the whole frames in the video and average them, and I get a clear and high-quality background. I think this method can be applicable to the videos from fixed position cameras with not too many objects movement places.

Question 5(d)



Your comments:

How to distinguish the pixels over the threshold are the same objects is the most important problem to solve. According to the demonstrator's advice, "Connected Components Labelling" is a way to discriminate the objects from the images by the pixels' distances and assign labels to these pixels. Further, the progress of implementation is that connect the pixels with their neighbours by defining a fixed search range and mark them a label by adding a number which means they are a group with order and use the same way to each pixel for counting the whole objects. However, there are many noisy dots after labelling, and I use a simple method to filter these labelling groups which are the real objects or the noise by set an amount threshold to keep the labelling groups which are over threshold and delete those are under the threshold. And it works pretty well for me, but how to determine a best object counting threshold for all the frames from the video is the room can be improved.

Question 6(a)

Three non-consecutive windows

W1

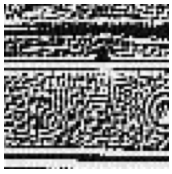
W2

W3

LBP of windows

LBP1

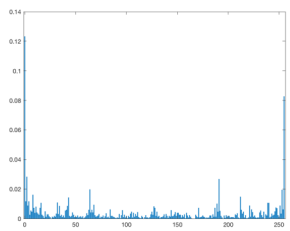
LBP2



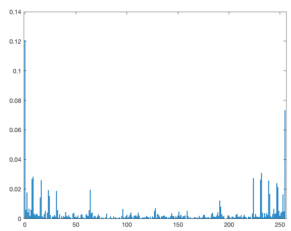
LBP3

Histograms of LBPs

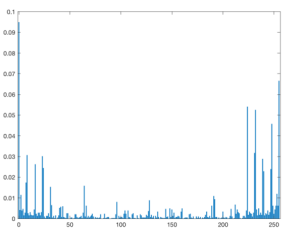
H1



H2

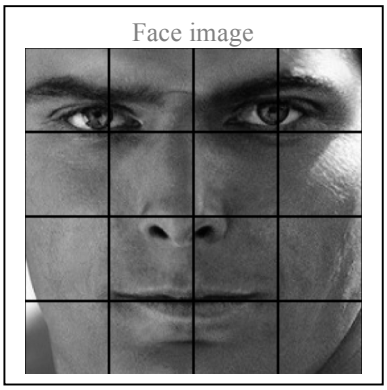


H3

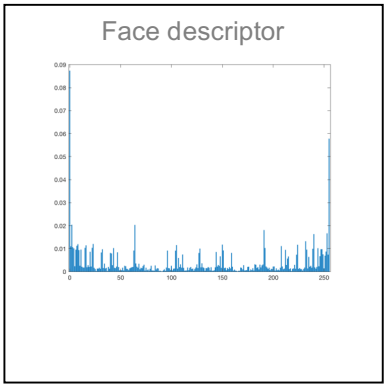


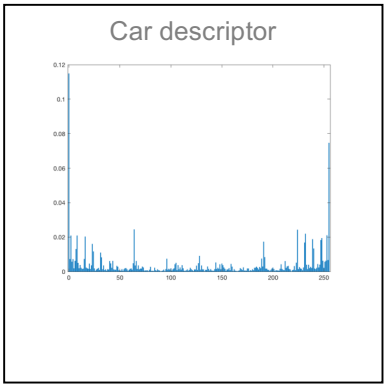
Question 6(b)

Two example images:



Descriptors:





Your comments:

Firstly, I reuse a function for question 5: objects to convert a colorful image to a grayscale image and divide the greyscale image into equally sized square non-overlapping windows, and calculate the feature descriptor for each window to "Local Binary Pattern" (LBP) used for texture classification and representation. Further, I compare each pixel in the window to its neighbours from its left-top to right-bottom followed anticlockwise. Next, I compute the histogram over the window about the frequency of each "number" occurring and normalize the histogram by dividing the number of pixels. Secondly, I combine all the local descriptors to a global descriptor to represent the whole image as consisting of multiple windows. However, there are still some black borders remained because the global descriptor is patched by several local descriptors and the block borders are generated because the pixels on the edge of local windows can't generate LBP due to lose information around it. Thirdly, I try to use a local descriptor with a whole image size (256x256) to generate the result and compare it to the result generated by global descriptor. As I have expected, the results between entire image size local descriptor and global descriptor are all the same.

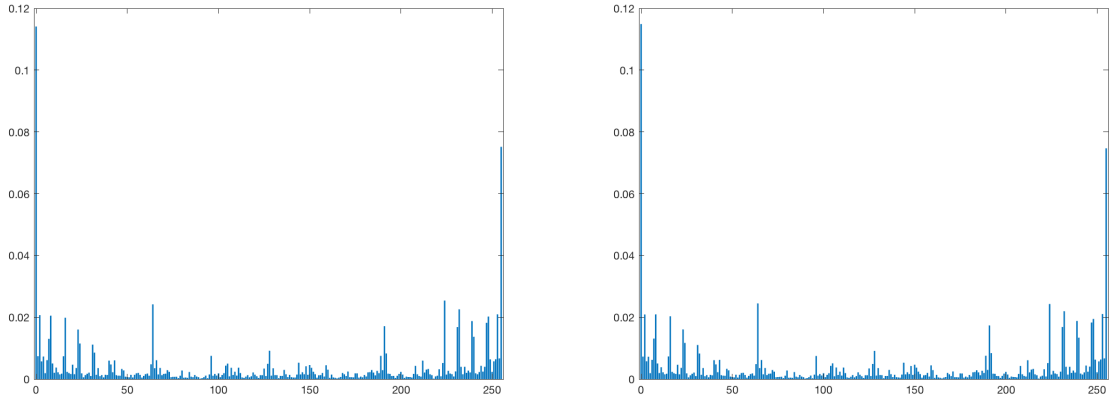
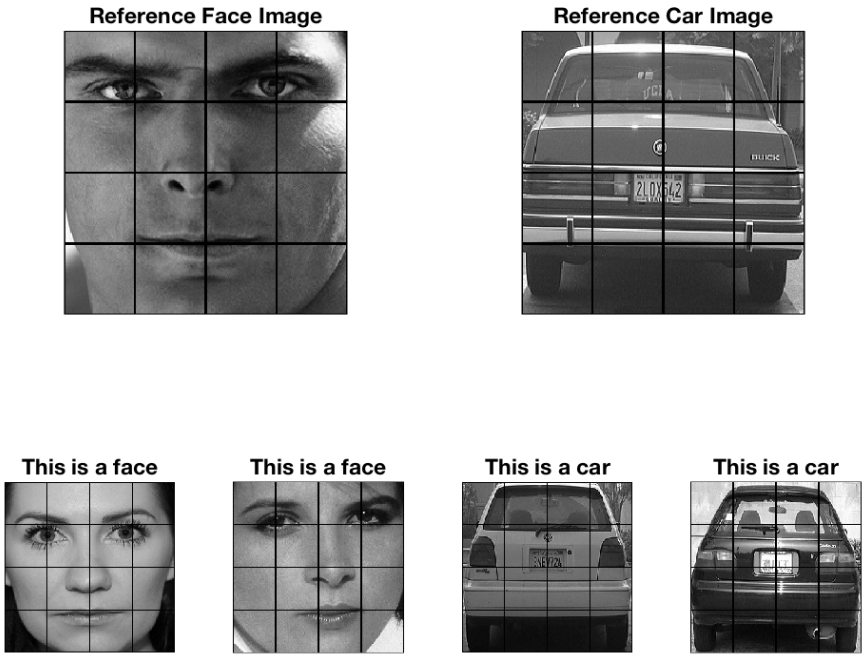


Fig 3. Comparison between global descriptor and local descriptor with whole image size

Question 6(c)

Block diagram of classification process



Your comments:

Firstly, I implement a classification process to classify the images from the reference images into two categories which are the face and car images. Secondly, the process is that I create two global descriptors from the reference images (face-1 and car-1) first, and create the global descriptors for input images. And I use simple histogram similarities to distinguish which image are belong to face or car category by finding the minimum sum subtraction of two image histograms with the reference histograms. Finally, it shows that the input images are discriminated into two categories precisely.

Question 6(d)

Your comments:

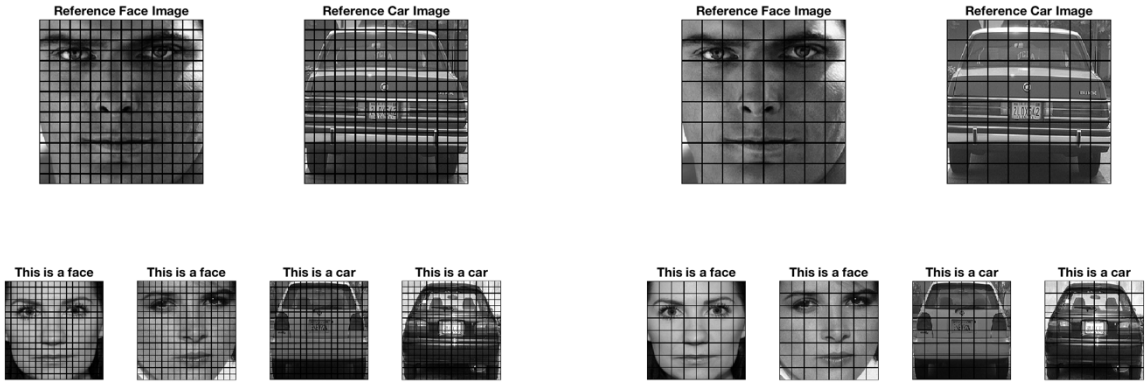


Fig 4. Decreasing the window size to 16x16 and 32x32

The images are divided into more fragments by decreasing the window size, and the results of classification can be predicted exactly as well.

Question 6(e)

Your comments:

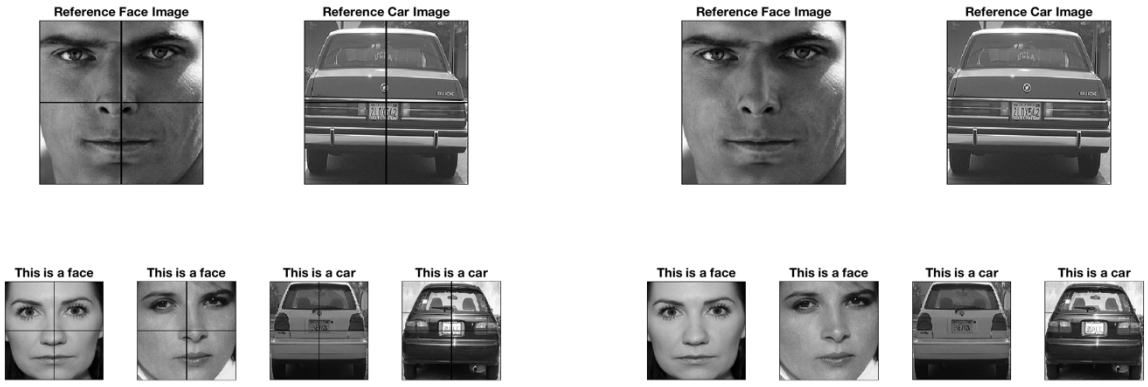


Fig 4. Increasing the window size to 128x128 and 256x256

The images are divided into less fragment(s) by decreasing the window size, and the results of classification can be predicted exactly as well.

Question 6(f)

Your comments

According to the previous results, the textures are modelled with concatenated local binary patterns and it is more powerful than it looks like. People can distinguish the objects by their shape and computers can discriminate the objects by LBPs of those pixels and tell us whether a block of pixels belongs to a car or a face. I think this method could be used in the video distinction. For instance, it can be used to detect the frame changes in videos, and once the frame components changed, the texture changes dramatically, and this could be detected and use it to handle more image processing. Moreover, the cutting-edge technology called "Face ID" may base on this theory and do some extension from it. The way to recognize people face may have lots factors to consider, even different people have different faces outline or skin colours, it may have some errors or mistakes to discriminate who is who and I think there are several ways to solve these problems or reduce the error times occurrence.