Queen Mary
University of London

Introduction to Computer Vision

Coursework

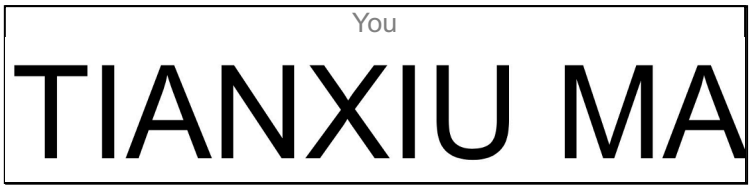
Submission

NAME: TIANXIU MA

230277302

Transformations
Question 1(b):

You

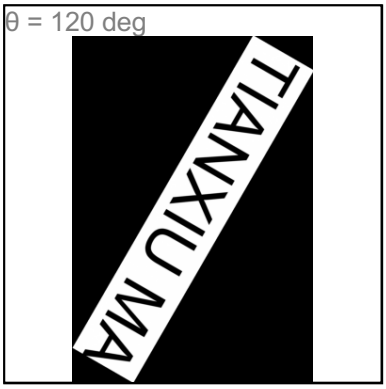


Rotated images:

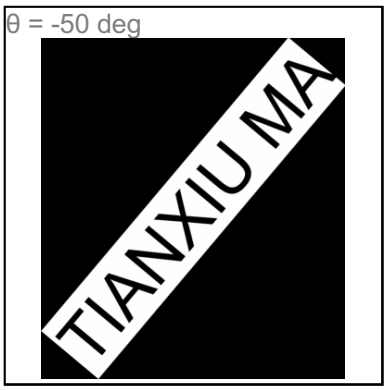
θ = 30 deg

θ = 60 deg

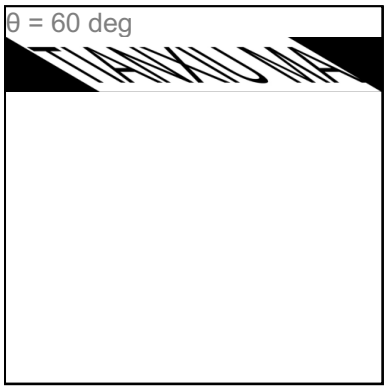
θ = 120 deg



θ = -50 deg



Skewed images:



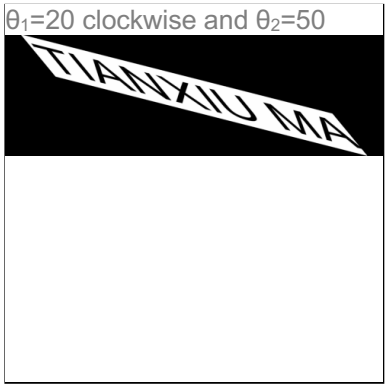
Your comments:

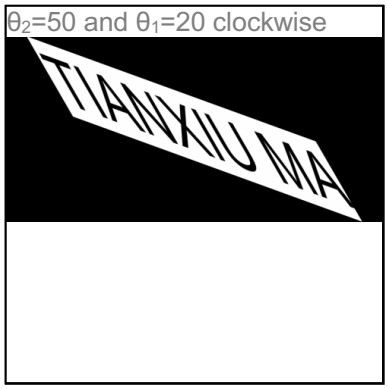
the method described in the **ICV_rotate_image** function uses backward mapping for image rotation. Backward mapping involves mapping each pixel in the destination image back to its corresponding position in the source image. When computing the size of the new image, we need to pay attention to the trigonometric functions of degrees in case the results are negative.

the advantages and disadvantages of different approaches：

1. I have tried bilinear interpolation and Nearest Neighbour interpolation to calculate the pixel values of the transformed image. But bilinear interpolation results in a smoother and more visually pleasing output compared to using Nearest Neighbor interpolation.
2. The presence of "black points" in rotated images, in contrast to skewed images, can be attributed to the nature of rotation. Black areas may appear in an image after it has been rotated because some of the resulting pixels may not have well-defined values. This happens as a result of the rotation process adding new positions that might not match the original locations of the pixels. On the other hand, when an image is skewed, the coordinate axis is distorted rather than rotated. Skewing retains all original pixel positions, and bilinear interpolation can effectively fill in the values for the entire skewed image, without introducing undefined regions. It's important to remember that although bilinear interpolation works well at smoothing transitions during transformations, noise in the image may still remain. Noise reduction is not a feature of bilinear interpolation; rather, it is a weighted average of adjacent pixel values. To remove any noise from the original or altered images, different noise reduction techniques would need to be used.

Question 1(c):





Your comments:

The images resulting from the combined rotation and skew transformations clearly exhibit significant differences. The sequence in which transformations are applied plays a crucial role in determining the final outcome.

The implementation of combining rotation and skew transformations is straightforward. By chaining these transformations with different orders, the desired results can be achieved effortlessly.

A direct comparison of the two images emphasizes the substantial differences resulting from the order in which transformations are applied. The non-commutative nature of matrix multiplication contributes to this impact, highlighting the significance of the transformation sequence.

Convolution
Question 2(b):

Designed kernel:

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Original image

Averaged image

Your comments:

Firstly, I changed the original image into a grayscale format, providing a single-channel representation for subsequent operations.

Secondly, To prevent pixel values from exceeding the standard 0-255 range, the kernel is normalized. Normalization ensures that the convolution operation does not introduce artifacts by maintaining the integrity of pixel values.

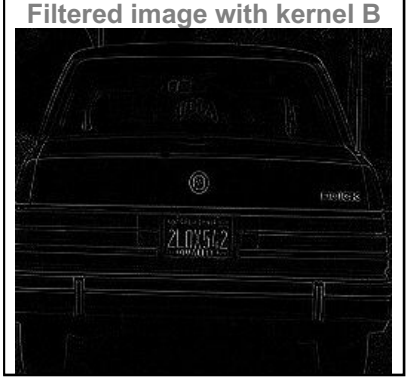
Thirdly, convolving the grayscale image with the normalized kernel.

The application of the averaging filter results in a perceptibly smoother image; however, the drawback is a reduction in the clarity of details such as the license plate number. This implies that while this method effectively mitigates high-frequency noise, it simultaneously leads to a loss of fine-grained information. This outcome is a consequence of each pixel being influenced by its neighboring pixels, resulting in an averaged representation over the kernel size applied to the entire image. And it becomes more challenging to discern due to the filtering's impact on pixel values across the entire image

Question 2(c):

Filtered image with kernel A

Filtered image with kernel B



Your comments:

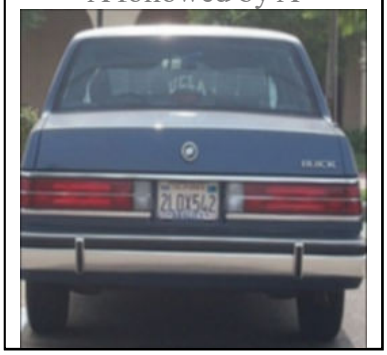
The filtered image with kernel A exhibits enhanced smoothness due to the nature of a Gaussian filter. This type of filter is proficient in noise elimination, leading to a clearer and more refined image. Notably, the effect of kernel A shares similarities with the mean kernel discussed in the last question, as both filters introduce a level of blurriness to the image. However, kernel A, being a Gaussian filter, utilizes weighted averaging. Specifically, pixels closer to the center of the filter receive higher weights, contributing to a more nuanced and visually appealing result.

For kernel B, it functions as a Laplacian filter, designed to accentuate edges in an image. The second image clearly reveals the sharpened delineation of the car's edges. Kernel B is particularly sensitive to both horizontal and vertical edges, making it effective in edge detection. It's worth noting that the Laplacian filter relies on the second derivative of the image. Edge points, representing zeros of the second derivative, contribute to the summation of weights in the Laplacian kernel, which totals to zero.

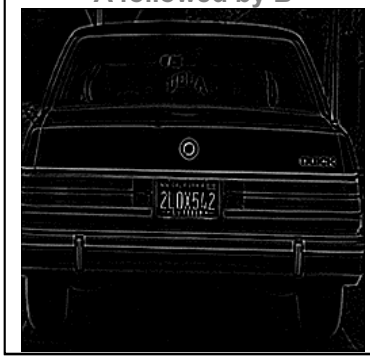
In summary, kernel A, as a Gaussian filter, enhances smoothness with weighted averaging, while kernel B, as a Laplacian filter, excels in edge detection, emphasizing the clarity of object boundaries in the processed image.

Question 2(d):

A followed by A



A followed by B



B followed by A:

Your comments:

The first image appears notably smoother, and this can be attributed to the consecutive application of Gaussian filters. Employing Gaussian filters twice tends to further enhance the smoothness of the image. However, it's important to note that this heightened smoothness comes at the cost of losing some fine details, resulting in a somewhat fuzzier appearance.

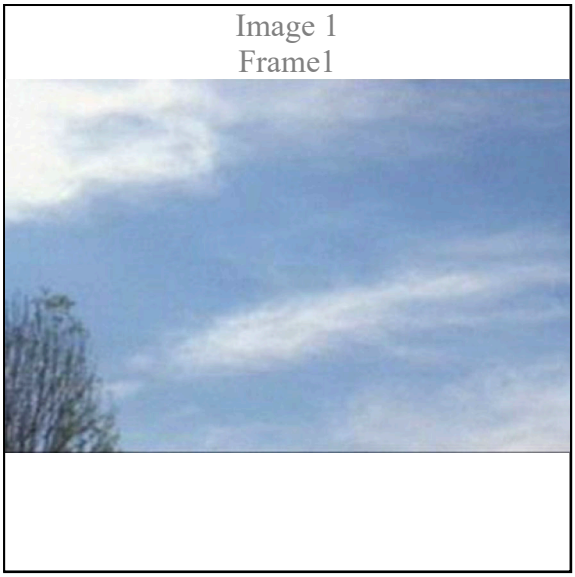
In the second image, a dual-filter approach is adopted. Initially, a Gaussian filter is utilized to effectively denoise the image. Subsequently, a Laplacian filter is applied to emphasize and sharpen the edges. This combination aims to strike a balance between noise reduction and edge enhancement.

In the final image, a reverse order of filters is employed. The Laplacian filter is first utilized to accentuate the details and edges in the image. Following this, a Gaussian filter is applied to mitigate noise. This sequential application is designed to prioritize edge enhancement before addressing noise reduction.

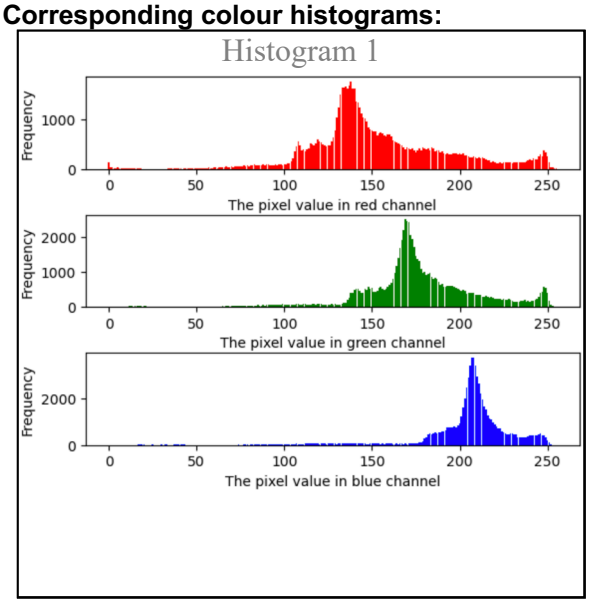
In summary, the choice and order of filters significantly impact the visual characteristics of the images, with each sequence tailored to achieve specific goals such as smoothness, edge emphasis, or noise reduction.

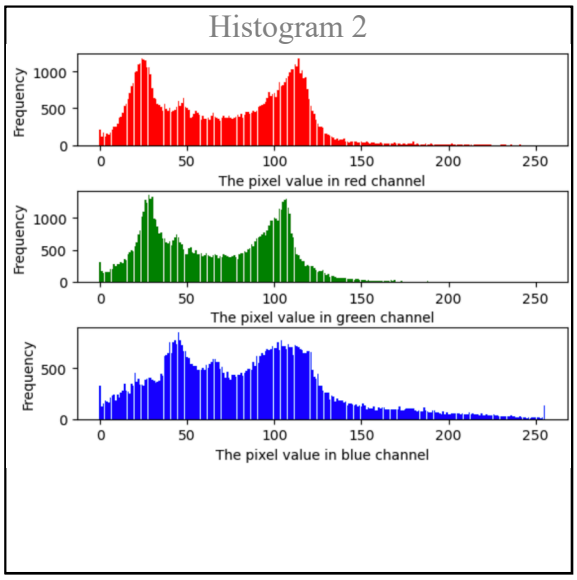
Histograms
Question 3(a):

Two non-consecutive frames:



Corresponding colour histograms:





Your comments:

The color difference between the first and tenth frames is conspicuous, leading to distinct variations in their respective color histograms. Notably, the left histogram exhibits peak values towards the right, while the right histogram displays peak values towards the left. This discrepancy is likely due to the darker nature of the right image, with 0 representing black and 255 representing white.

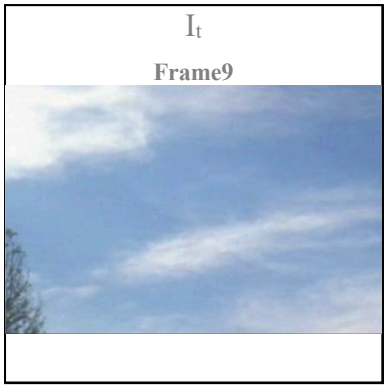
Obtaining a color histogram involves an algorithm that maps quantization levels to the frequency of each quantization level in the image. In simpler terms, the color histogram is derived by tallying how often each pixel value occurs for the corresponding color channel in the image.

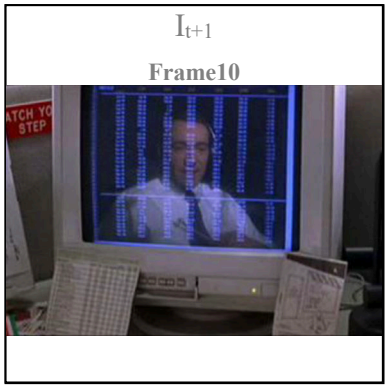
Histogram 1 highlights the potential dissimilarity in the distributions of pixel values across different channels. On the other hand, Histogram 2 illustrates that the distributions of pixel values in different channels can also exhibit similarity.

Histogram 2 reveals multiple peaks in each channel, indicating that Image 2 can be segmented into distinct regions based on color distribution. This suggests a more pronounced color partition in Image 2.

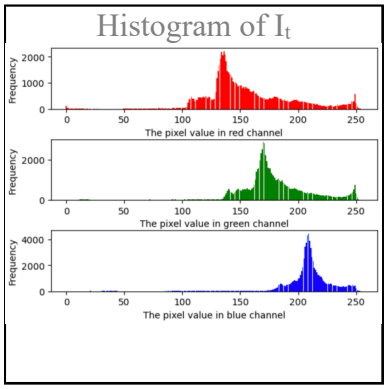
Question 3(b):

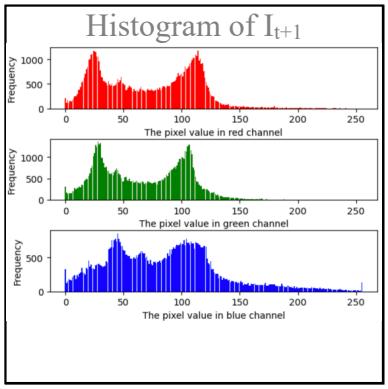
Two consecutive frames:

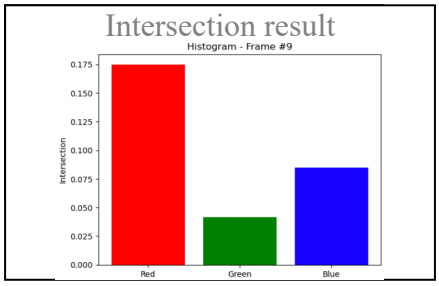




Histograms:

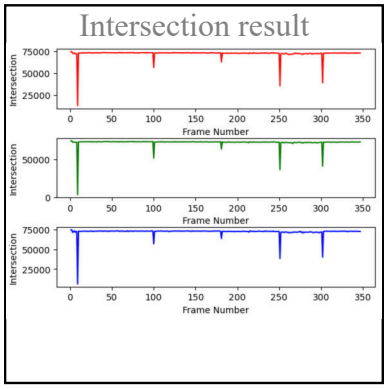


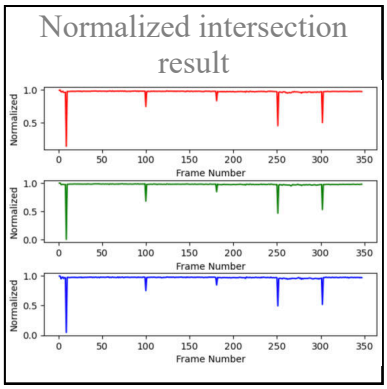




Question 3(b):

Intersection result for a video sequence:





Your Comments:

The intersection results can be effectively used for image classification, as similar consecutive images exhibit comparable intersection results, while different images yield distinct intersection values.

After normalization, the Y-axis label indicates the weighted value of a specific pixel value in a channel of the image. Without normalization, the Y-axis label represents the number of pixels that overlap in the two corresponding frames. With normalization, the Y-axis label represents the similarity of the two frames.

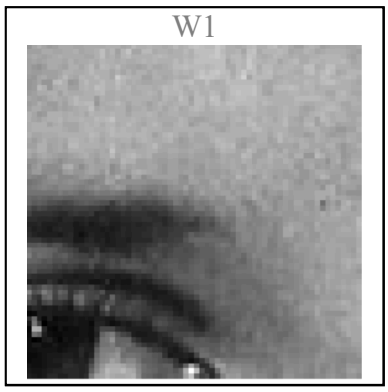
By applying the normalization method, for two highly related histograms (images), the intersection value will be close to 1 at each channel; for two highly different histograms (images), the intersection value will be closer to 0 at each channel. In brief, the higher the intersection value is, the more similar the two images are in terms of pixel values.

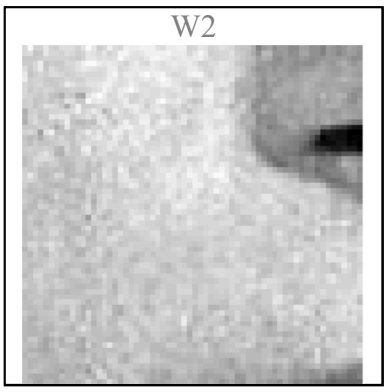
Question 3(c):
Comments:

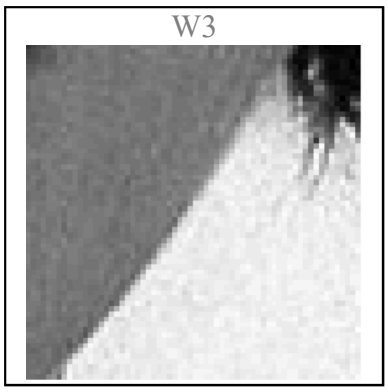
1. The intersection value serves as a metric for the similarity between two consecutive frames in a given video.
2. For scenes with similarities, the intersection value approaches 1, and the variation in consecutive intersection values is minimal during video playback. Conversely, when the scene changes, the intersection value experiences a significant drop, potentially approaching zero. Therefore, a threshold for changes in intersection values can be established to detect scene transitions in a video.
3. This technique exhibits robustness to various geometric transformations like rotation, scaling, mirroring, shear, etc. However, it is not robust to factors such as texture.
4. Histogram intersection may face challenges. When the histograms of consecutive images are the same, it will be difficult to identify changes among consecutive frames in a video through observation of the intersection histogram.

Texture Descriptors and Classification
Question 4(a)

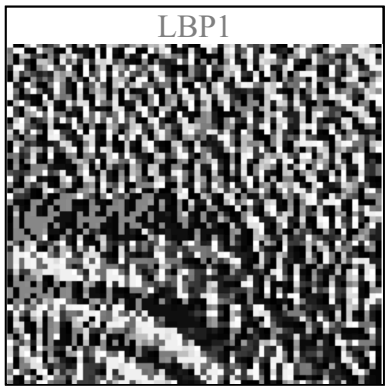
Three non-consecutive windows

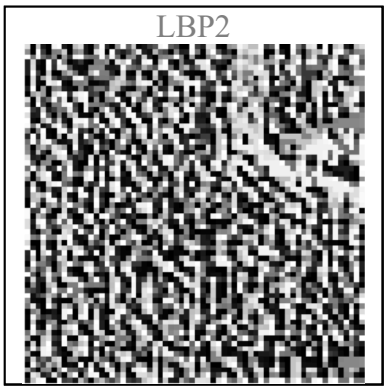


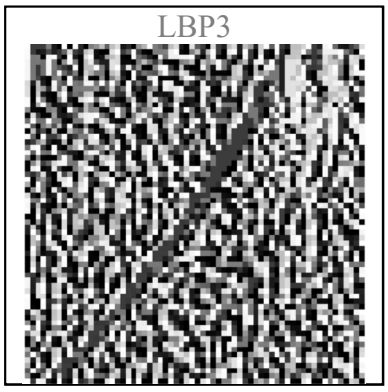




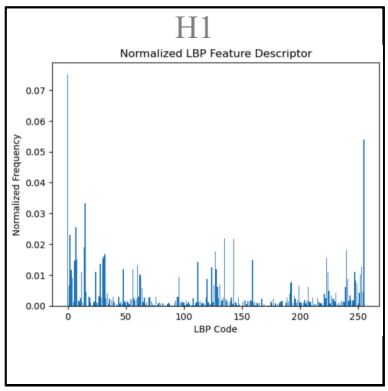
LBP of windows

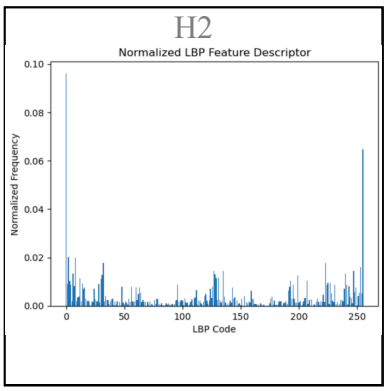


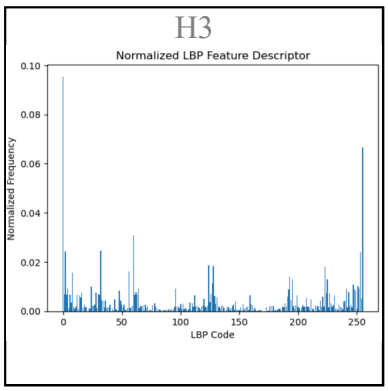




Histograms of LBPs



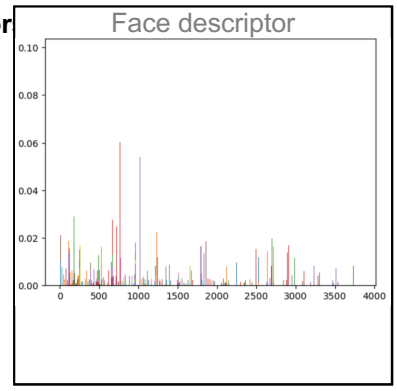


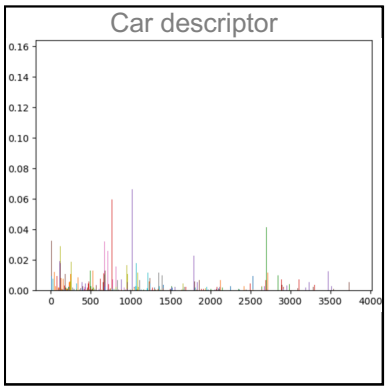


Question 4(b)

Two example images:

Descriptor





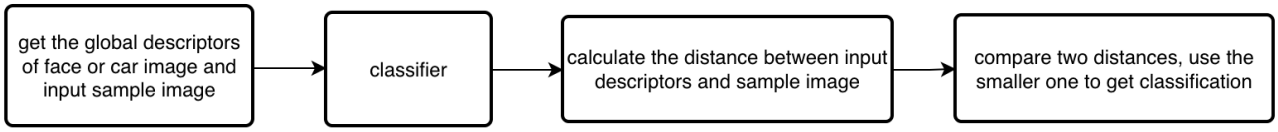
Your comments:

Initially, Ipartitioned the input image into 16 equally sized windows. Subsequently, the focus was on acquiring the Local Binary Pattern (LBP) image for each window. To address border issues, I employed the mirror padding technique for each window and applied the LBP operation. Afterward, by processing these LBP images, I derived corresponding feature descriptors (histograms). Finally, all these local descriptors were amalgamated into a global descriptor.

When comparing the face descriptor with the car descriptor, notable differences emerge in terms of peaks and intensity. It is evident that distinct types of images yield significantly different descriptors.

Question 4(b)

Block diagram of classification process



Your comments:

The predictions from the classifier exhibit remarkable accuracy. The classifier's predictions align perfectly with the known ground truth. All confidences surpass 0.5, indicating a high level of confidence in the accuracy of our prediction results.

There have some issues arise from the potential impact of feature positions on classification outcomes. For instance, face-1 exhibits a larger forehead area compared to other facial features. Consequently, differences in feature distribution may affect the classification outcome. Therefore, preprocessing of the data becomes essential when employing this technique.

Employing a global descriptor derived from the mean of images within the same class proves effective in accurately classifying images. However, relying solely on mean square error (MSE) may not be highly significant due to the small differences between car and face MSE. To address this, incorporating the local descriptor becomes crucial. The local descriptor considers similarities based on different blocks, thereby enhancing the classification robustness.

Question 4(c)

Your comments:

For testing purposes, I designated 'car-1.jpg' and 'face-1.jpg' as the classes for creating the classifier, while the remaining images from the provided dataset were used for testing.

The classifier consistently achieves a prediction accuracy of 1, irrespective of changes in the window size. This indicates that the classifier accurately classifies test samples across various window sizes, demonstrating robust and consistent performance.

As the window size decreases (and the window number increases), the local descriptor may become inadequate to represent the features of the image. For example, in the case of the 'Face' image, when the window size is too small, the windows may fail to capture the complete features of, for instance, the eyes.

Question 4(d)

Your comments:

Conversely, as the window size increases (and the window number decreases), the classification quality improves. Larger windows tend to provide a more comprehensive representation of image features.

In general, a larger window size contributes to more representative image features. However, caution is required not to choose an excessively large window, as it may lead to the neglect of fine-grained feature details. Additionally, selecting a very large window increases the risk of different images having similar local pattern descriptors. In summary, it is essential to choose a suitable window size that balances capturing representative features without sacrificing crucial details.

Question 4(e)

Your comments:

To analyze dynamic texture, a combination of block matching technology and Local Binary Pattern (LBP) can be employed to extract both motion and appearance features. However, this approach has certain limitations. Firstly, relying on a single LBP Operator for calculation may result in the loss of information related to changes in texture intensity, leading to incomplete texture feature extraction. Secondly, in motion feature extraction, the use of LBP as a matching criterion can result in multiple matching points, thereby impacting the accuracy of the matching process.

Object Segmentation and Counting
Question 5(a)

Original frames:

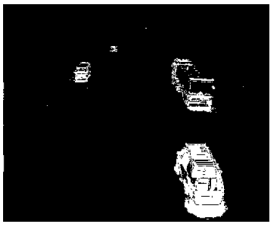
Reference frame

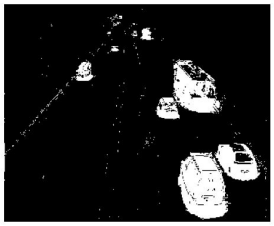
Selected frame 1

Selected frame 2

Frame differencing:

Threshold results:





Question 5(b)

Original frame:

Selected frame 1

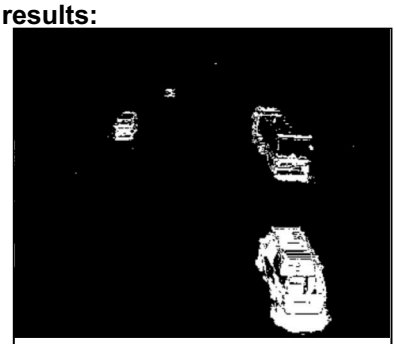
Selected frame 2

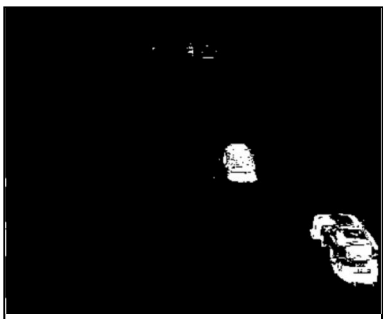
Frame differencing:

Difference between Frame1 and Frame2

Difference between Frame99 and Frame100

Threshold results:





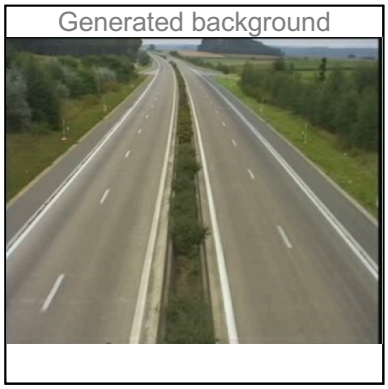
Your comments for 5a,5b:

I chose the 2nd and 100th frames as test samples, with their corresponding reference frames being the 1st and 99th frames, respectively. Additionally, the threshold for comparison remains set at 20.

In the case of two consecutive frames, where the background remains more stable, indicating smaller changes in the image, there is a reduction in the number of recognized cars.

Regarding the results with the adjusted threshold, it is evident that there are fewer vehicles compared to those in 5a. This is because the absolute result is calculated between the two frames, reflecting the differences between the two images. For instance, the car identified in the threshold results in 5b might appear stretched, and the increased number of cars in 5a may be attributed to variations from the original image.

Question 5(c)

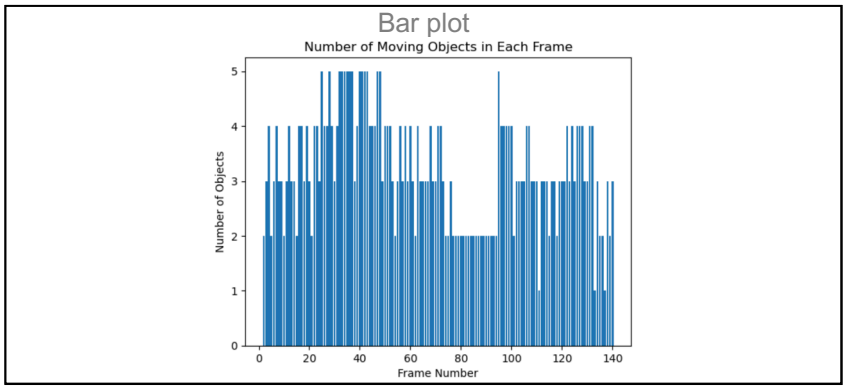


Your comments:

The process of obtaining the reference frame (background) involves the following steps. Initially, repeat the method described in 5b to obtain the threshold results for all consecutive frames. Subsequently, assess whether each pixel value in these threshold results is 0. If it is, count and accumulate each pixel value across all frames. No action is taken otherwise (a pixel value of 0 signifies an unchanged area, i.e., background, while any other value indicates a changed area, i.e., moving objects). Finally, for each pixel, divide the accumulated value by the corresponding count to obtain the reference frame.

The resulting effect is highly satisfactory, with no vehicles (moving objects) detected on the roads.

Question 5(d)



Your comments:

implemented solution:

1. Input Parameters:

a. videoFrames: List of video frames.

b. threshold: Threshold value used in the temporal difference function.

2. Function Steps:

a. Convert the first frame of the video (videoFrames[0]) to grayscale using ICV_rgb2grayscale and use it as the background.

b. Iterate through the remaining frames (starting from the second frame) and apply temporal difference (ICV_TemporalDifference) between the background and the current frame.

c. Use the resulting binary difference image to find contours of connected components (ICV_countObjects).

d. Count the number of objects with an area greater than a specified threshold in each frame.

e. Store the counts for each frame in the numberOfObjects list.

3. Output:

numberOfObjects: A list containing the number of objects in each frame.

We can adjust the parameters, such as the threshold, based on the characteristics of your video and the desired level of sensitivity in detecting moving objects.

Advantages:

1. Simplicity:

The function provides a relatively simple approach to detect and count moving objects in a video.

2. Custom Contour Detection:

The custom contour detection (find_contours) allows for more flexibility and understanding of the underlying process compared to using built-in OpenCV methods.

3. Thresholding:

The function uses thresholding to create a binary difference image, which simplifies the detection of moving objects.

4. Visualization:

The use of Matplotlib for visualization makes it easy to understand and analyze the results.

Disadvantages:

1. Sensitivity to Threshold:
The performance of the function is highly dependent on the threshold value used. Selecting an appropriate threshold can be challenging and may require experimentation.

2. Limited Robustness:
The function may not be robust to variations in lighting conditions, noise, and other factors that can affect the effectiveness of the temporal difference method.

3. Recursive Contour Detection:
The custom contour detection method (find_contours) uses a recursive depth-first search (DFS) approach, which may lead to recursion depth errors for large or complex contours. It can also be computationally expensive.

4. Manual Parameter Tuning:
The function requires manual tuning of parameters, such as the threshold and minimum contour area, which might be less convenient than using automated methods.

5. Noisy Environments:
In noisy environments, false positives (detecting non-existent objects) and false negatives (missing real objects) may occur.