Queen Mary
University of London

# Introduction to Computer Vision

## Coursework

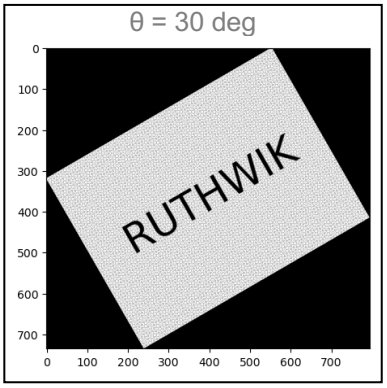
## Submission

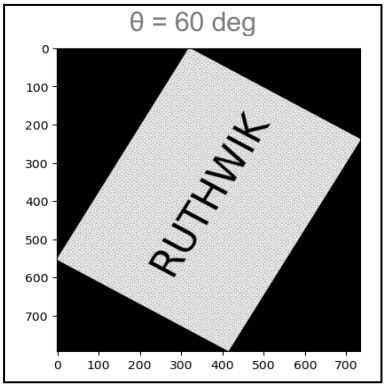
**Ruthwik Ganesh**

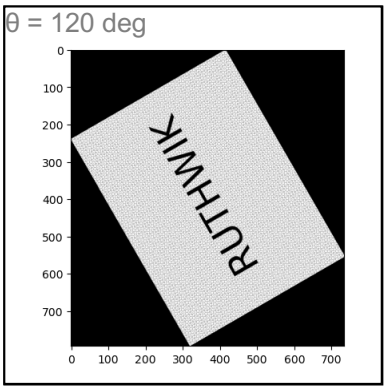
**230702930**

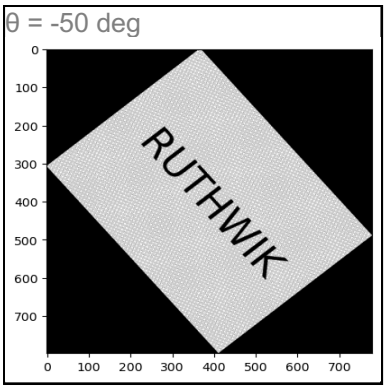
**Transformations**
**Question 1(b):**


**Rotated images:**

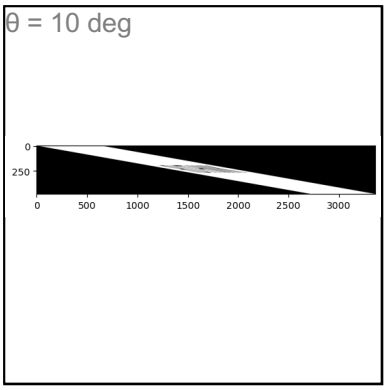


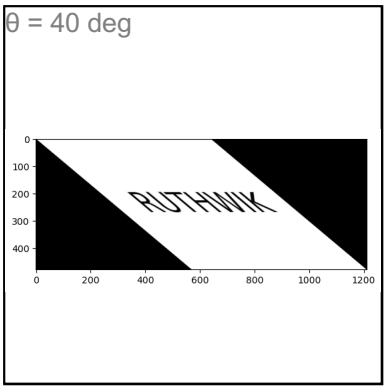



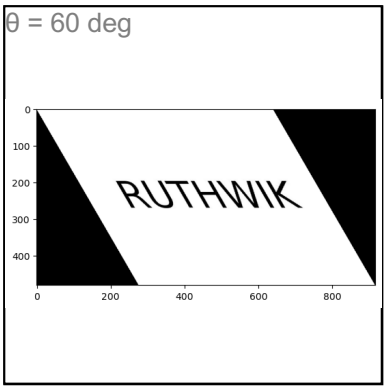



**Skewed images:**




**Your comments:**

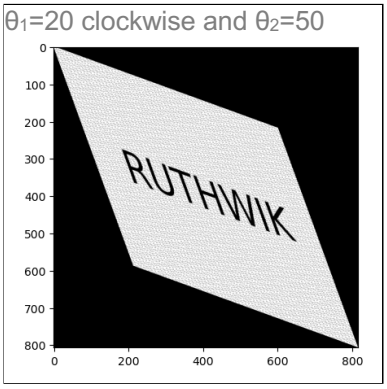
*Advantages*

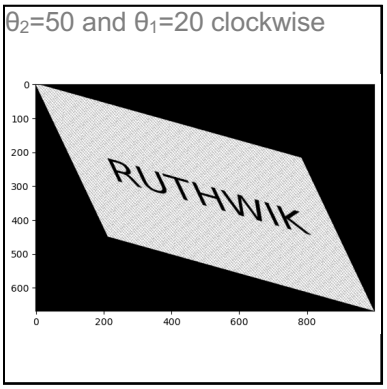
• Image transformations can be easily pipelined, i.e. rotation and skewed matrix's can be multiplied to arrive at a resultant matrix that can be applied to the image at once.
• Easy to debug on due to the modularity in transformations compared to my early approach of having these transformations as a separate function.
• Computationally efficient when compared to the brute-force pixel by pixel transformations

*Disadvantages*

• Separate transformation for rotation and skew can become cumbersome while pipelining.
• Rotating and skewing the image using this method can introduce some noise.

**Question 1(c):**



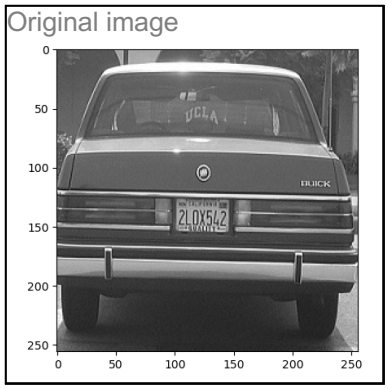



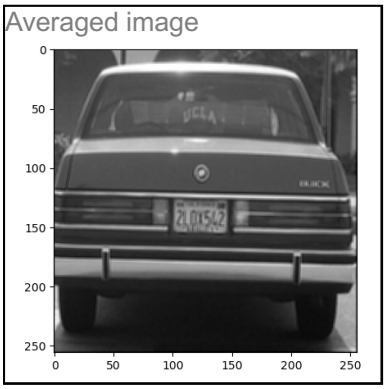

**Your comments:**

Swapping the transformations, we do not obtain same results as seen in the above images, this due to the non-commutative property of matrix multiplication. When the rotation (R) and skew (S) matrices are multiplied the order of these matrices matter i.e. (R) x (S) ! = (S) x (R)

**Convolution**
**Question 2(b)**:

**Designed kernel:** kernel_avg = np.ones((3, 3)) / 9.0

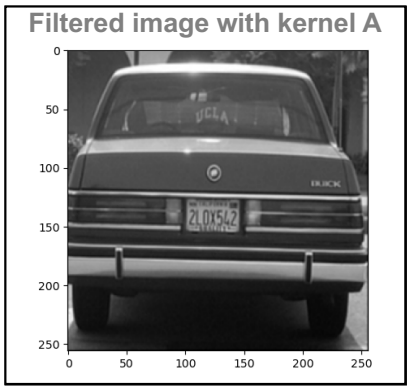


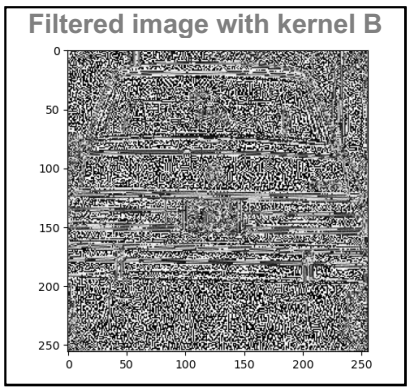




**Your comments:**

The average kernel smoothen/blurring the pixel intensities by a constant factor (here 1/9). The average intensity kernel is also known as a box filter or a mean filter. It is a simple kernel that takes the average of pixel values in the neighbourhood of each pixel. The kernel is typically a square matrix with all elements equal to 1/N, where N is the total number of elements in the kernel. This ensures that the sum of the kernel elements is equal to 1.

**Question 2(c):**



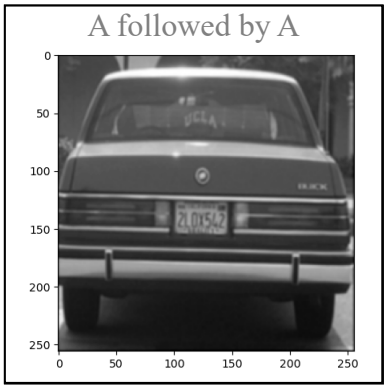



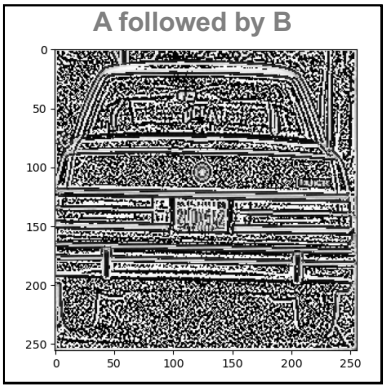

**Your comments:**

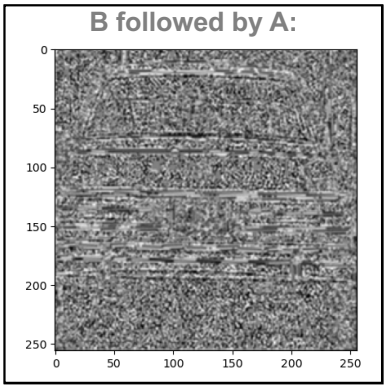
*Kernel A:*
A Gaussian filter is a type of convolution kernel that is often used for blurring or smoothing images. It derives its name from the fact that it follows a Gaussian distribution. The Gaussian filter is designed to give more weight to the central pixel and less weight to the surrounding pixels, creating a smoother effect compared to the average intensity filter.

*Kernel B:*
The Laplacian filter is a convolution kernel used in image processing for edge detection. It highlights regions of an image where there is a rapid change in intensity, helping to identify edges and fine details. The Laplacian filter is particularly useful for detecting features that are not aligned with the image axes. One challenge with the Laplacian filter is that it responds to both positive and negative edges, and it can produce responses around zero intensity regions. To detect edges more effectively, zero-crossings of the Laplacian response are often considered. These are points in the image where the intensity changes sign, indicating the presence of an edge.

**Question 2(d):**




**Your comments:**

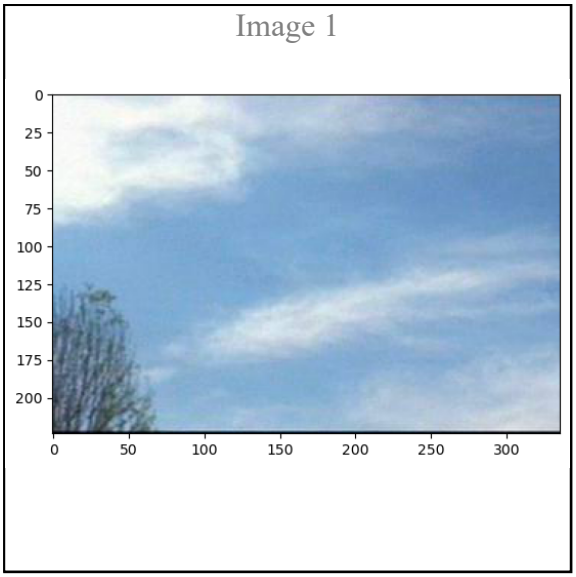
*A followed by A:*
Here the gaussian filter is applied twice which leads to double-smoothening of the image leading to a blurry image.

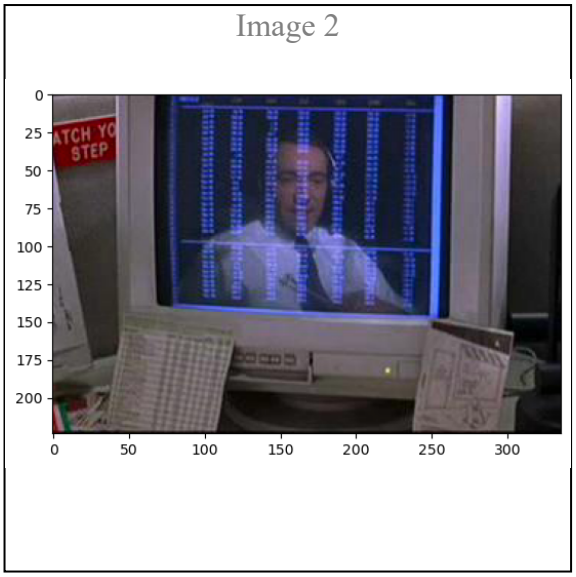
*A followed by B:*
The Gaussian of Laplacian (LoG) filter is a two-step process that involves smoothing the image with a Gaussian filter followed by applying the Laplacian filter. This combined operation is used for edge detection and is particularly effective for detecting edges at different scales. The LoG filter is useful for edge detection because the initial Gaussian smoothing helps reduce noise, and the subsequent Laplacian operation enhances the edges. This combination is effective in detecting edges at different scales, making the LoG filter suitable for detecting both fine and coarse details in an image.

*B followed by A:*
Applying the Laplacian filter and then applying the gaussian will lead to smoothening of edges .

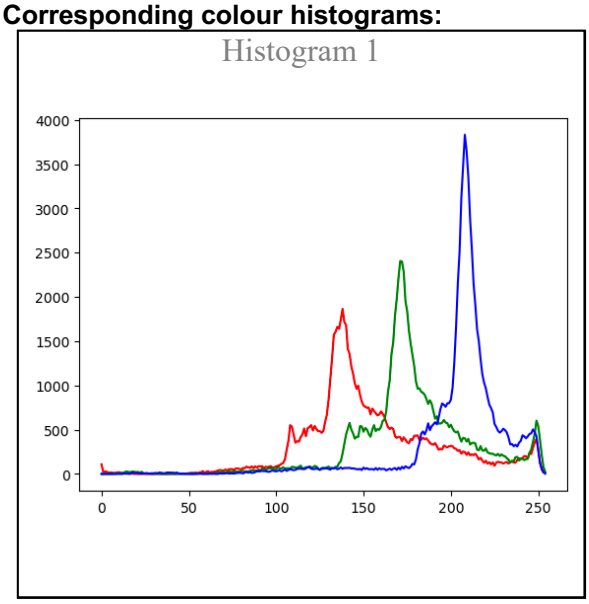
**Histograms**
**Question 3(a):**

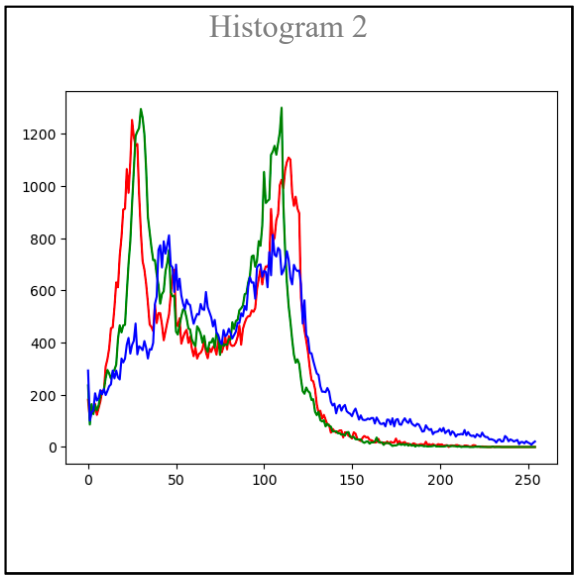
**Two non-consecutive frames:**




**Corresponding colour histograms:**



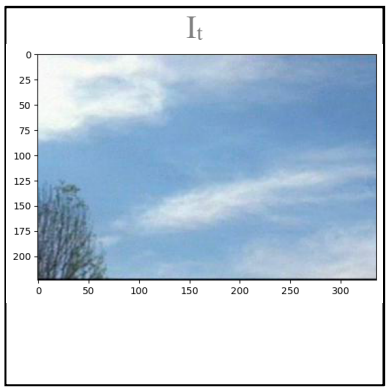



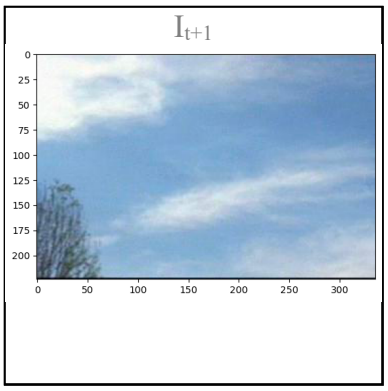

**Your comments:**

The colour intensities on the histogram aptly depicts the colour distribution in images, we notice high counts for blue in the histogram of first image which corresponds to large number of pixels capturing the blue sky.

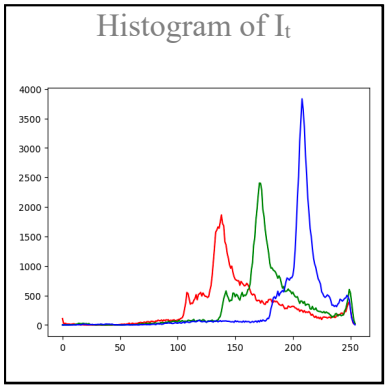
**Question 3(b):**

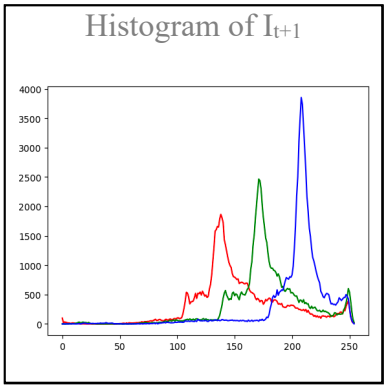
**Two consecutive frames:**




**Histograms:**



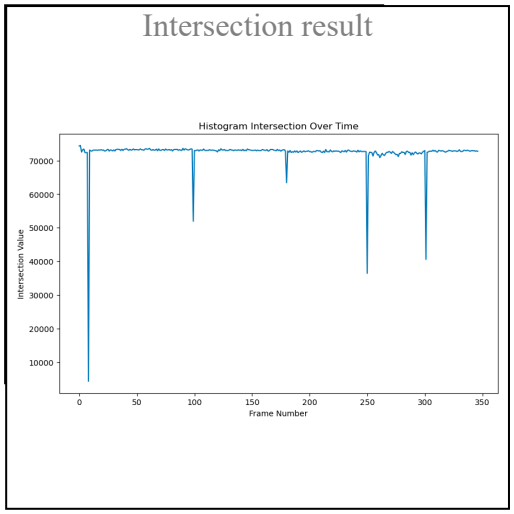



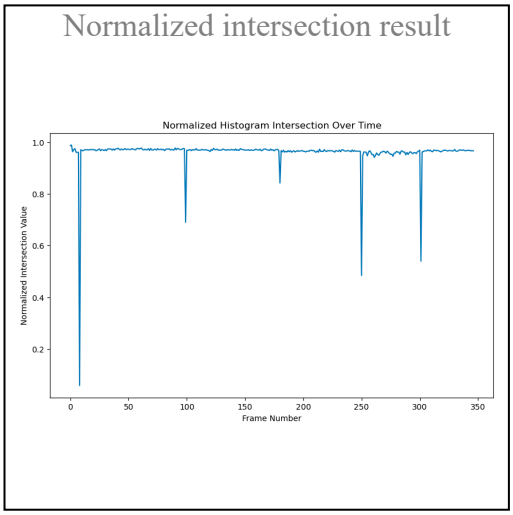

**Your Comments :**

Taking two consecutive frames may will fetch the above result, there is a parity in the minimum pixel value counts. There would have some sharp change in the intersection results if the two select frames for comparison were of different scenes. Unnormalized and Intersection values between 2nd and 3rd frame are 72533, 0.9637143920068028 are respectively.

**Question 3(b):**

**Intersection result for a video sequence:**




**Your Comments :**

**The spikes in the intersection results indicates changes in the frames in a video sequence. It's apparent that there about 5 scene changes in given video sequence.**

**Question 3(c):**
**Comments:**

Histogram intersection is a measure commonly used in image processing for comparing the similarity between two histograms. Its robustness depends on the characteristics of the data and the specific application context. Let's discuss some aspects of the robustness of histogram intersection in image processing:

**Insensitive to Scalar Changes:**
Histogram intersection is relatively robust to scalar changes in intensity values. It compares the distribution of values rather than their absolute magnitudes. Therefore, it can be effective when images have undergone changes in contrast or brightness.

**Invariant to Histogram Scaling:**
Histogram intersection is invariant to histogram scaling. Whether the histogram values are in the range of 0 to 1 or any other range, the intersection value remains unaffected. This makes it robust to changes in the scale of the histogram.

**Robust to Local Changes:**
Histogram intersection can be robust to local changes in the image. It focuses on the overall distribution of pixel values rather than specific pixel values, which can make it less sensitive to small changes in localized regions.

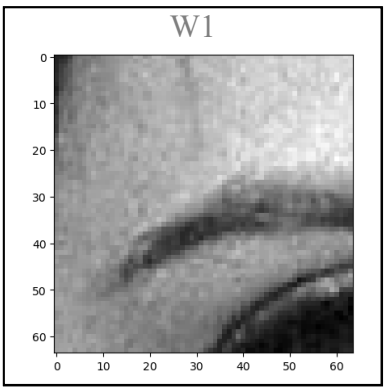
**Insensitive to Histogram Shape:**
Histogram intersection is insensitive to changes in the shape of histograms, as long as the overall distribution remains similar. This makes it suitable for comparing images with different lighting conditions or contrast variations.

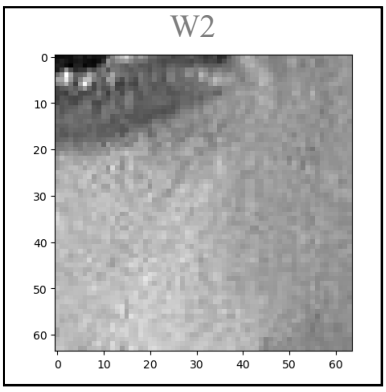
**Limited Discrimination Power:**
While histogram intersection is robust in certain scenarios, it may lack discrimination power when comparing histograms with similar shapes. It treats all bins equally, which might not capture subtle differences between distributions. If frames of a video can have same colour distribution albeit different pixel arrangement is when the histogram comparisons will fail.

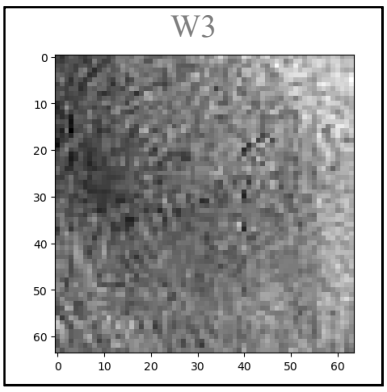
**Affected by Outliers:**
Histogram intersection can be sensitive to outliers in the histograms. Outliers, especially in small frequency bins, can disproportionately impact the intersection value.

**Texture Descriptors and Classification**
**Question 4(a)**

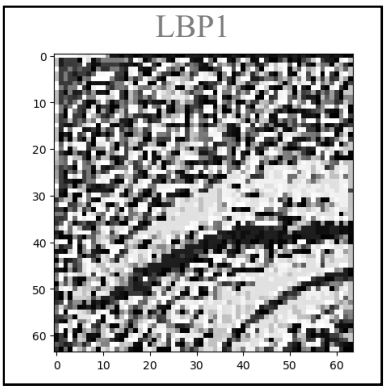
**Three non-consecutive windows**

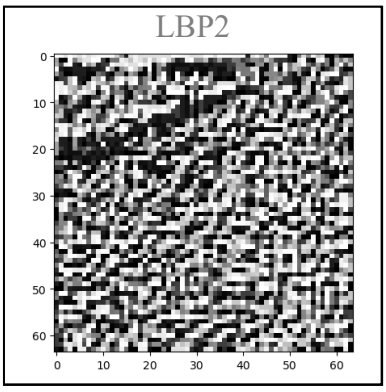


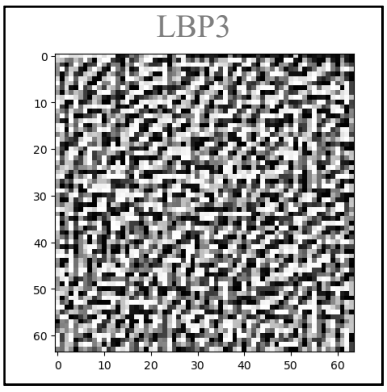



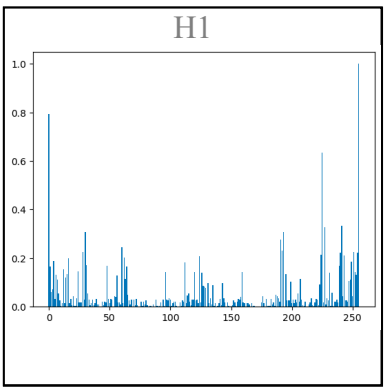

**LBP of windows**

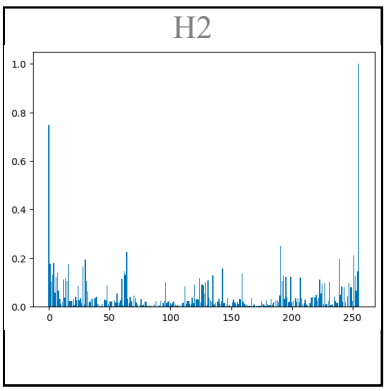


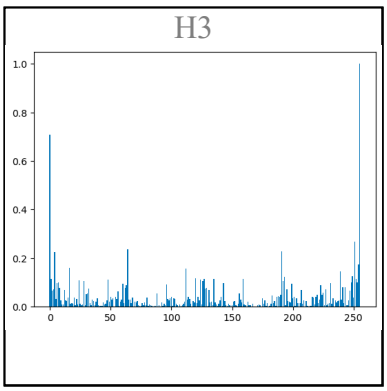




**Histograms of LBPs**

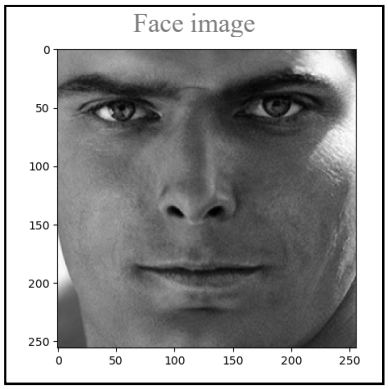


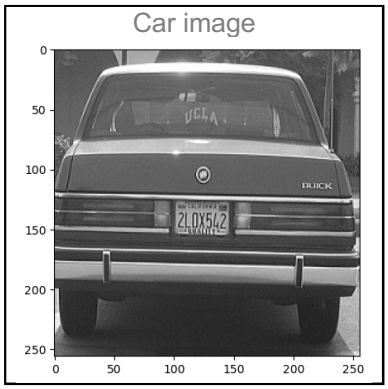



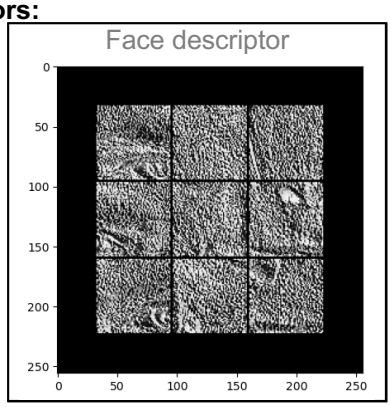
**Question 4(b)**

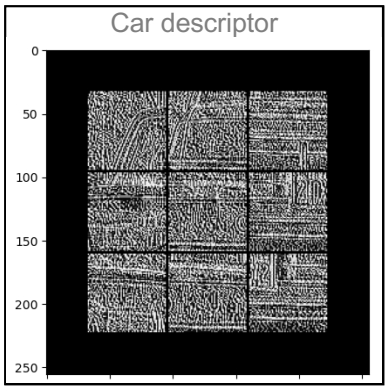
**Two example images:**




**Descriptrors:**




**Your comments:**

Generated 1 global descriptor for each image which can distinguish between car and face by providing a score that was obtained by the sum of its feature descriptors. For cars sum of feature descriptors are in the range of 53-54 and 63-64 for face.

## Limitations

**Sensitivity to Noise:**
LBP is sensitive to noise in images. Noisy pixels or small variations in pixel values can result in different LBP codes, impacting the robustness of the descriptor.

**Limited Discriminative Power:**
LBP may not capture fine-grained details and subtle variations in textures. In cases where intricate details are crucial for classification, LBP might lack the required discriminative power.

**Dependence on Local Patterns:**
LBP focuses on local patterns and may not effectively capture global structures or relationships between distant regions in an image. This limitation can be significant in certain classification tasks.

**Rotation Invariance Limitations:**
LBP is not inherently rotation-invariant. Changes in orientation can affect the resulting LBP codes, making it necessary to apply additional techniques (e.g., rotation-invariant variants) for improved performance.

**Scale Sensitivity:** LBP is sensitive to scale changes in images. When the size of patterns in the image varies, the ability of LBP to capture those patterns might be compromised.

## Possible ways to mitigate the above limitations are given below.

**Noise Reduction Techniques:**
Apply pre-processing techniques to reduce noise in images before extracting LBP features. Common noise reduction methods include Gaussian smoothing or median filtering.

**Local Patterns and Texture Variations:**
Explore variations of LBP, such as uniform LBP or rotation-invariant LBP, which focus on specific types of local patterns and provide better performance in the presence of noise or texture variations.

**Scale-Invariant LBP:**
Incorporate scale-invariant LBP variants to make the descriptor more robust to changes in scale. This can involve analysing patterns at multiple scales or applying scale normalization techniques.

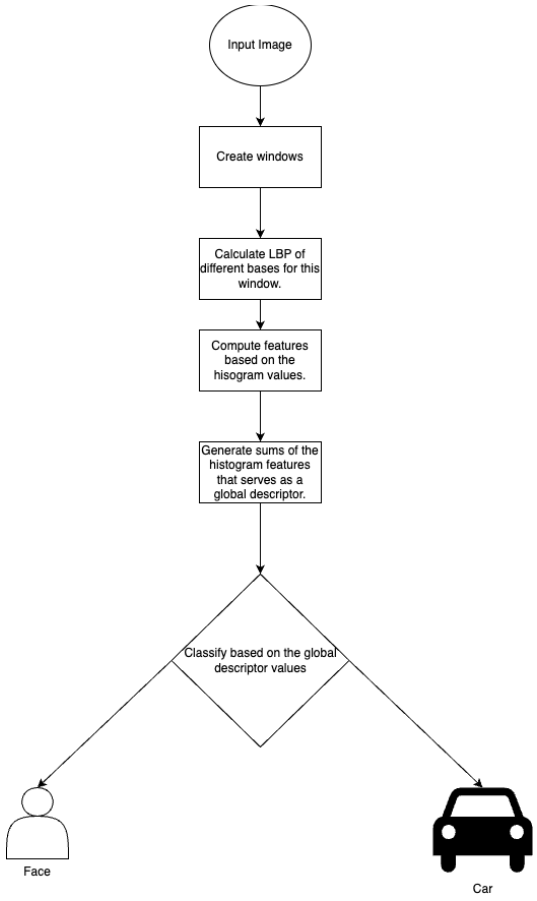
**Rotation-Invariant LBP:**
Use rotation-invariant LBP variants to address sensitivity to orientation changes. These variants modify the LBP computation to ensure that the descriptor is invariant to rotations.

**Combined Feature Descriptors:**
Combine LBP with other feature descriptors, such as Histogram of Oriented Gradients (HOG) or Local Ternary Patterns (LTP), to create more comprehensive representations that capture different aspects of image content.

**Deep Learning Approaches:**
Consider using deep learning methods, such as Convolutional Neural Networks (CNNs), which can automatically learn hierarchical features and may be less sensitive to the limitations of handcrafted descriptors like LBP.

**Question 4(b)**

**Block diagram of classification process**




**Your comments:**

Calculate the LBP values for a given window taking multiple bases this will create a local descriptor for a window of an image, then histograms are calculated to extract features which are then summed for calculating a value, we observe that these values are always within a certain range for a given image and a fixed window size that aids in classification.

| Window size | Face | Car |
|---|---|---|
| 16 | 749 - 750 | 699 - 705 |
| 32 | 250 - 251 | 220 - 223 |
| 64 | 63 - 64 | 54 - 53 |

**Question 4(c) and 4(d)**

**Your comments:**

Smaller window sizes capture local details and fine-grained textures, allowing the classifier to focus on small-scale patterns. Larger window sizes, on the other hand, provide a more global view, considering broader contextual information.

Thus, window size selection is highly context dependent i.e. either to extract features or to generalize the features.

Smaller window sizes generally lead to faster computations as they involve fewer pixels. Larger window sizes require processing a larger number of pixels and may increase computational complexity.

Smaller windows may not be robust to scale changes in the image, as patterns may vary in size. Larger windows can improve scale invariance but may lose sensitivity to fine-scale details.

**Question 4(e)**

**Your comments:**

**Flow-Based LBP:**

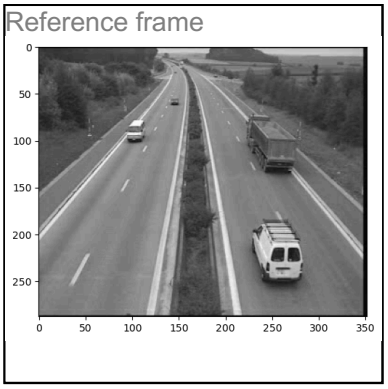
**Idea:** Utilize optical flow information to enhance LBP for dynamic texture analysis.

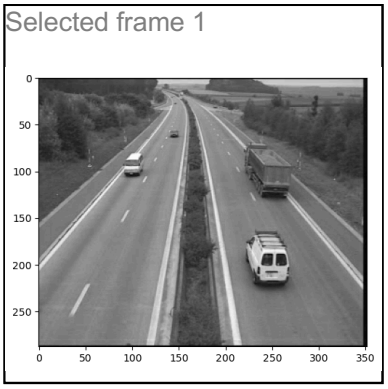
**Implementation:** Compute optical flow between consecutive frames and incorporate the flow information into the LBP computation.

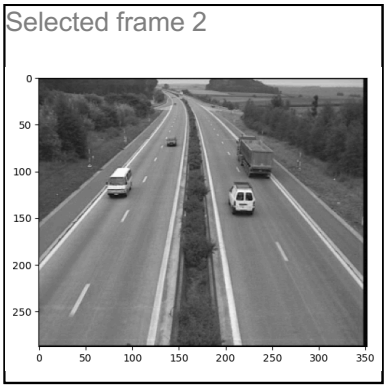
**Advantages:** Accounts for motion information and provides a more comprehensive representation.

**Object Segmentation and Counting**
**Question 5(a)**

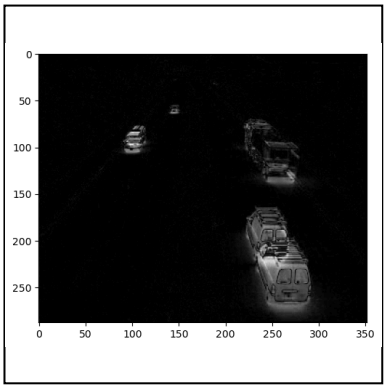
**Original frames:**

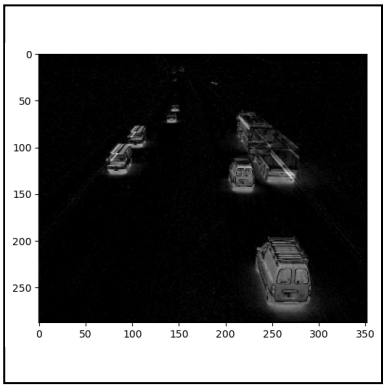


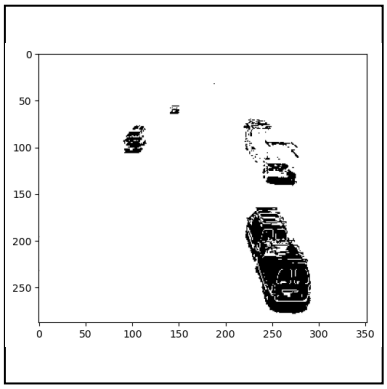



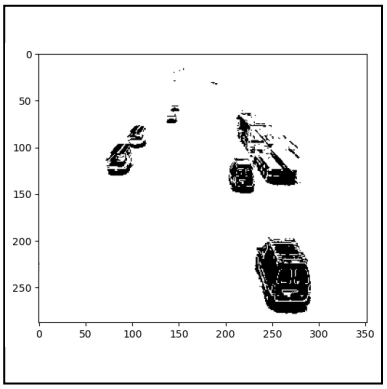

**Frame differencing:**





**Threshold results:**

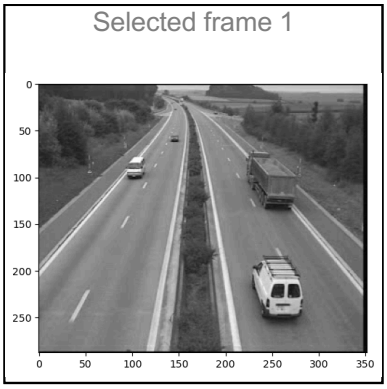


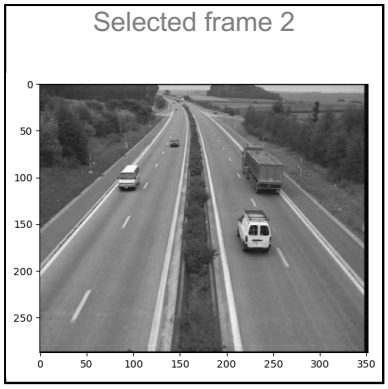



Fill the available spaces for your answers.

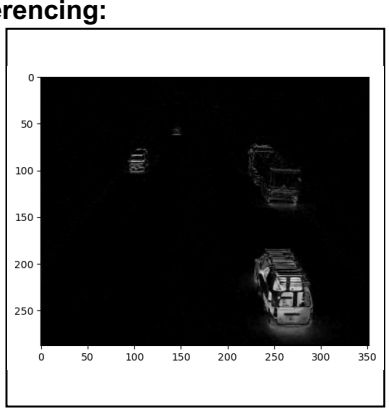
**Question 5(b)**

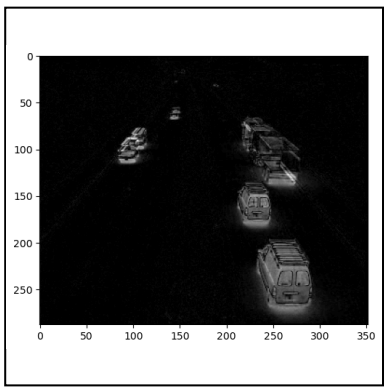
**Original frame:**



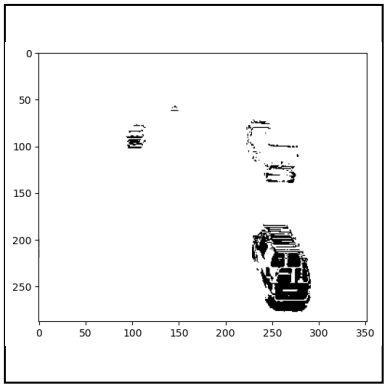



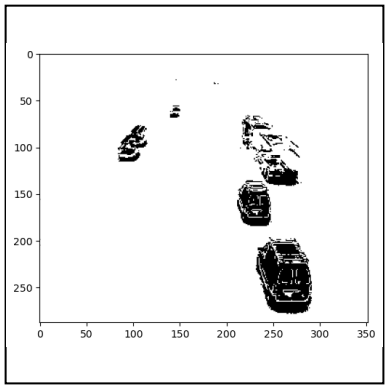

**Frame differencing:**





**Threshold results:**



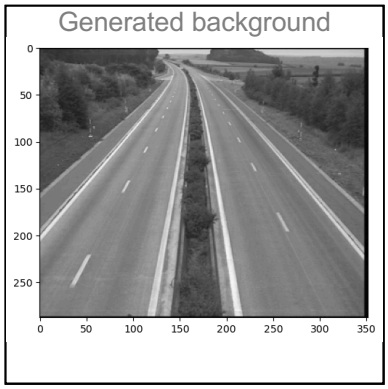


**Your comments for 5a,5b:**

5a)
1. Tested absolute difference of the image with the reference frame being the first frame.
2. In this case we see a trail of objects in the direction of motion on threshold difference.
3. The object in the first frame will remain in the foreground of the frame in the following frames as the Frist frame was considered as a reference frame, which will still impede the progress with object counting.

5b)
1. Here we no longer see the trail of objects in the direction of motion on threshold difference.
2. Although the object in the first frame will not remain in the foreground of the frame this approach still cannot be used for object counting as the background still has objects. The goal would be to obtain a background image without any objects.

**Question 5(c)**




**Your comments:**

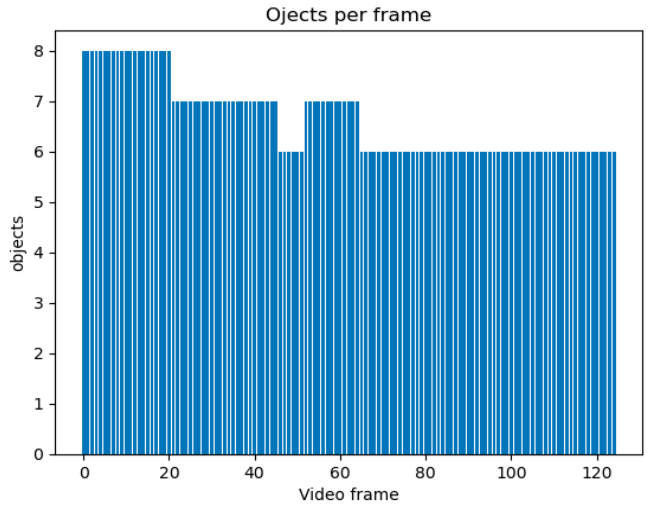
**Weighted Temporal Averaging:**
**Idea:** Weighted temporal averaging is a technique to smooth the variations in pixel values across consecutive frames, assigning different weights to each frame.

**Algorithm:**
1. Capture frames f1, f2, ….., ft over a certain time window.
2. Assign weights w1, w2, …….,wt to the frames based on their temporal position (e.g., linearly increasing weights).
3. Compute the weighted average frame: I=f1·w1+f2·w2+…+ ft·wt.

**Use Cases:**
Temporal smoothing: Reducing noise and temporal fluctuations in video sequences.
Background modelling: Creating a background model that adapts to gradual changes over time.

**Question 5(d)**

Bar plot




**Your comments:**

1. For each frame, it converts the frame to grayscale.
2. Computes the absolute difference between the global background image and the frame.
3. Thresholds the difference to create a binary mask where pixel values above 100 become 255, and others become 0.
4. Counts the number of non-zero pixels in the binary mask (potential objects) and normalizes the count by dividing by 8000.

Advantages:

1. This approach works for a given video sequence.

Disadvantages

1. This approach cannot be generalized any video sequences.
2. This approach computationally intensive alternate localised approaches to find objects will give better results.
3. Finding the threshold was a manual process, this may need human intervention for any other video sequence.