Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. **Background substraction using MOG**
2. **MOG2**
3. **GMG**
4. **KNN**

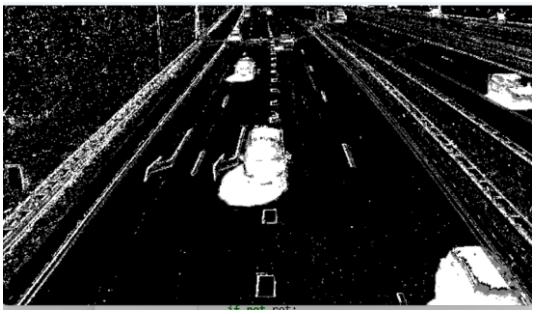
✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)

1) ***Background substraction using MOG***

```
import cv2
cap = cv2.VideoCapture(r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
bg_subtractor = cv2.createBackgroundSubtractorMOG2()

while True:
    ret, frame = cap.read()
    if not ret:
        break
    fg_mask = bg_subtractor.apply(frame)
    cv2.imshow('Original', frame)
    cv2.imshow('Foreground Mask', fg_mask)
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



2) ***Background substraction using MOG2***

```
import cv2
video_path = (r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
bg_subtractor = cv2.createBackgroundSubtractorMOG2()
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()
while True:
    ret, frame = cap.read()
    if not ret:
        break
    fg_mask = bg_subtractor.apply(frame)
    _, binary_mask = cv2.threshold(fg_mask, 127, 255, cv2.THRESH_BINARY)
    cv2.imshow('Original Frame', frame)
    cv2.imshow('Foreground Mask', fg_mask)
    cv2.imshow('Binary Mask', binary_mask)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

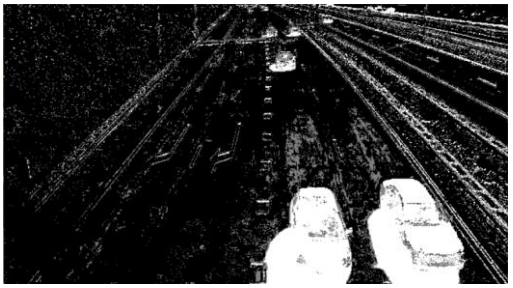
✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

3) ***Background substraction using GMG***

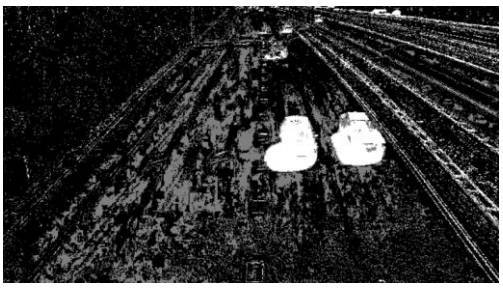
```
import cv2
video_path = (r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
bg_subtractor = cv2.createBackgroundSubtractorMOG2()
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()
while True:
    ret, frame = cap.read()
    if not ret:
        break
    fg_mask = bg_subtractor.apply(frame)
    cv2.imshow('Original Frame', frame)
    cv2.imshow('GMG', fg_mask)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()

cv2.destroyAllWindows()
```



3) ***Background substraction using KNN***

```
import cv2
mog = cv2.createBackgroundSubtractorMOG2()
cap = cv2.VideoCapture(r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
while True:
    ret, frame = cap.read()
    if not ret:
        break
    mask = mog.apply(frame)
    cv2.imshow('Mask', mask)
    if cv2.waitKey(30) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Background subtraction is a fundamental technique in computer vision used to separate foreground objects from the background in a video stream. Different algorithms are employed for this purpose, among which are MOG, MOG2, GMG, and KNN. Here are two points about each of these algorithms:

MOG (Mixture of Gaussians):

Utilizes a probabilistic model to represent the background of a scene.

Decomposes each pixel's intensity value over time into a mixture of Gaussians, where the Gaussian distribution parameters are dynamically updated to adapt to changing background conditions.

MOG2 (Improved Mixture of Gaussians):

An enhanced version of the MOG algorithm designed to improve performance and adaptability.

Employs a more sophisticated algorithm for updating the background model, including learning rates and thresholding mechanisms, resulting in better adaptability to varying lighting conditions and scene changes.

GMG (Gradient-based Motion Detection):

Based on the idea of using gradients to detect motion between consecutive frames.

Employs a Bayesian framework that models each pixel's intensity gradients over time, allowing for the detection of moving objects while suppressing noise and background clutter.

KNN (K-nearest Neighbors):

Utilizes a machine learning approach where each pixel's intensity value is classified based on its similarity to neighboring pixels.

Maintains a set of historical pixel values and classifies each pixel as foreground or background based on its similarity to the k-nearest historical values, making it robust to gradual changes in the scene.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

✱***As applicable according to the experiment. Two sheets per experiment (10-20) to be used.***

Centurion University
Shaping Lives... Empowering Communities...

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement :

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

1. **Histogram difference**
2. **Change detection**

✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Testing Phase: Compilation of Code (error detection)

1) ***Histogram Difference***

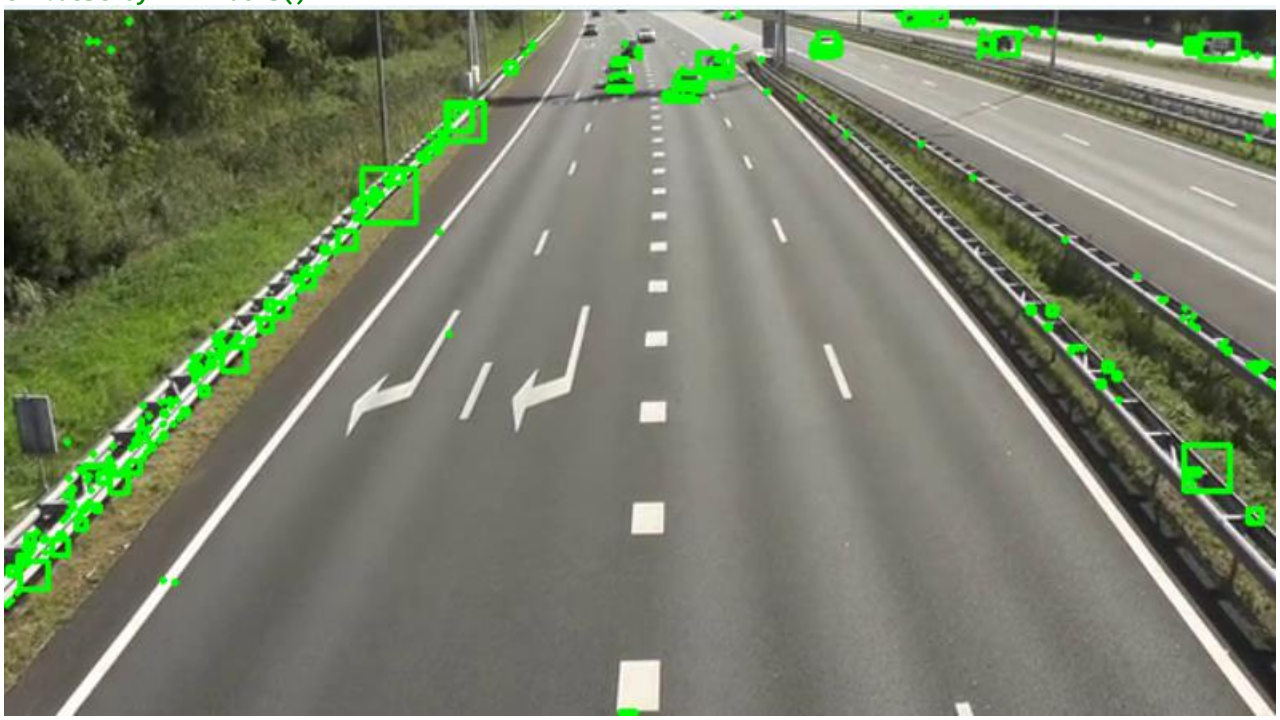
```
import cv2
import numpy as np
cap = cv2.VideoCapture(r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()
ret, prev_frame = cap.read()
if not ret:
    print("Error: Could not read the first frame.")
    exit()
prev_frame_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
prev_hist = cv2.calcHist([prev_frame_gray], [0], None, [256], [0,256])
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    curr_hist = cv2.calcHist([frame_gray], [0], None, [256], [0,256])
    hist_diff = cv2.compareHist(prev_hist, curr_hist, cv2.HISTCMP_CORREL)
    frame_with_hist_diff = np.copy(frame)
    cv2.putText(frame_with_hist_diff, f"Histogram Difference: {hist_diff}", (20, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow('Frame with Histogram Difference', frame_with_hist_diff)
    prev_frame_gray = frame_gray
    prev_hist = curr_hist
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

2) ***Change detection***

```
import cv2
cap = cv2.VideoCapture(r"C:\Users\HP\Downloads\traffic_-_27260 (360p).mp4")
if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()
ret, prev_frame = cap.read()
if not ret:
    print("Error: Could not read the first frame.")
    exit()
prev_frame_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    frame_diff = cv2.absdiff(frame_gray, prev_frame_gray)
    _, thresholded = cv2.threshold(frame_diff, 30, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresholded, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.imshow('Frame with Changes Marked', frame)
    prev_frame_gray = frame_gray
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```



Histogram Difference: Histogram difference measures the dissimilarity between histograms of consecutive frames in a video sequence, providing a quantitative measure of the change in pixel intensity distributions over time.

Change Detection: Change detection involves the process of identifying alterations in a scene by analyzing differences between consecutive frames, with histogram difference being one of the techniques used to detect significant changes such as motion, object appearance, or lighting variations in video streams.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

Campus
Ramchand
rapur, Kaliabali
Chhak, P.O-Chatrapur, Dist.- GanjamOdisha, India, PIN- 761020

✱*As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.