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

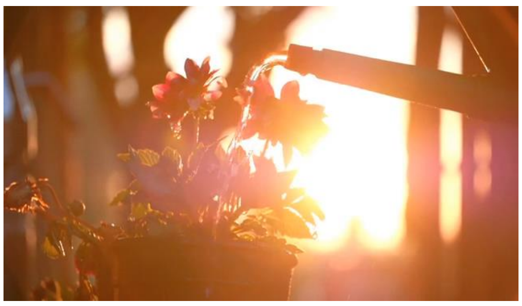
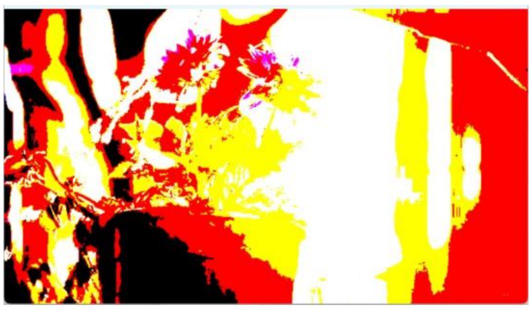
1. **Simple Thresholding**
2. **Binary Thresholding**
3. **Otsu Thresholding**
4. **Adaptive Thresholding**
5. **Binary Inverse Thresholding**

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

* Testing Phase: Compilation of Code (error detection)

1) *Simple Thresholding*

```
import numpy as np
import cv2

cap = cv2.VideoCapture('sample-5.mp4')

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:

        retval,threshed = cv2.threshold(frame,50,255,cv2.THRESH_BINARY)
        newFrame = np.array(threshed)

        cv2.imshow('newFrame',newFrame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
#out.release()
cv2.destroyAllWindows()
```

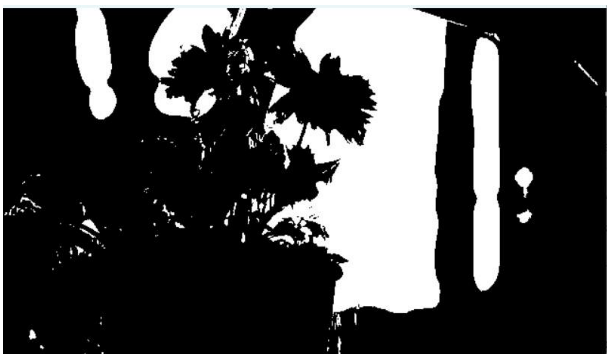



2) Binary Thresholding

```
import numpy as np
import cv2

cap = cv2.VideoCapture('sample-5.mp4')

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:

        # Convert frame to grayscale
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Apply binary thresholding
        retval, threshed = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY)

        # Display the thresholded frame
        cv2.imshow('Thresholded Frame', threshed)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release everything if job is finished
cap.release()
cv2.destroyAllWindows()
```

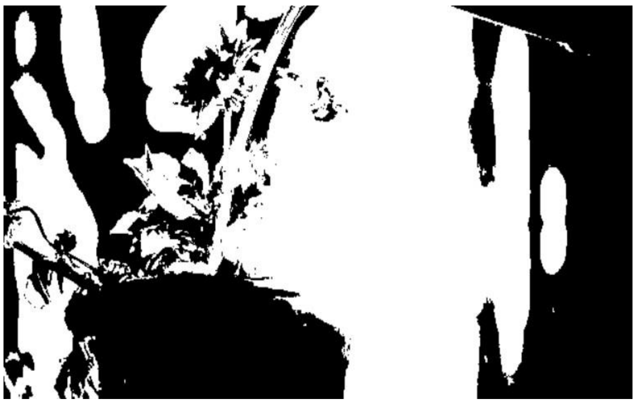
✱As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)



3) ***Binary Inverse Thresholding***

```
import numpy as np
import cv2

cap = cv2.VideoCapture('sample-5.mp4')

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:

        # Convert frame to grayscale
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Apply inverse binary thresholding
        _, threshed = cv2.threshold(gray, 50, 255, cv2.THRESH_BINARY_INV)

        # Display the thresholded frame
        cv2.imshow('Inverse Thresholded Frame', threshed)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release everything if job is finished
cap.release()
cv2.destroyAllWindows()
```

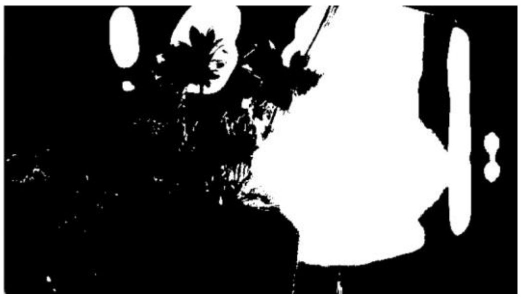


*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

Testing Phase: Compilation of Code (error detection

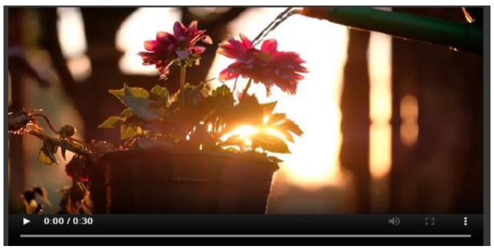
4) ***Otsu Thresholding***

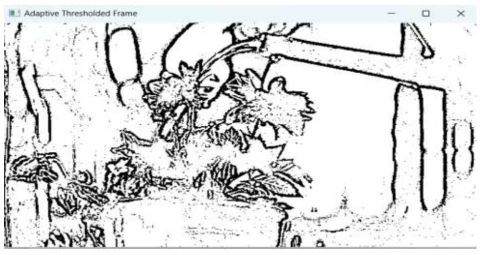
```
import numpy as np
import cv2

cap = cv2.VideoCapture('sample-5.mp4')

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:

        # Convert frame to grayscale
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Apply Otsu's thresholding
        _, threshed = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

        # Display the thresholded frame
        cv2.imshow('Otsu Thresholded Frame', threshed)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release everything if job is finished
cap.release()
cv2.destroyAllWindows()
```



4) Playing a video

```
import numpy as np
import cv2
cap = cv2.VideoCapture('sample-5.mp4')
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        threshed = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 
2)
        cv2.imshow('Adaptive Thresholded Frame', threshed)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
         break
cap.release()
```





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

From this experiment we learnt about different thresholding techniques which are :

Simple Thresholding:

Simple thresholding is a basic image processing technique used to segment an image into two regions based on a fixed threshold value.

Binary Thresholding:

Binary thresholding is a specific type of simple thresholding where pixels with intensity values above the threshold are set to a maximum value (usually 255), while pixels with intensity values below the threshold are set to zero.

Otsu Thresholding:

It calculates the threshold that minimizes the intra-class variance of pixel intensities in the resulting binary image, effectively separating the foreground from the background.

Adaptive Thresholding:

This technique is particularly useful for images with varying lighting conditions, as it adapts the threshold dynamically to different parts of the image.

Binary Inverse Thresholding:

Binary inverse thresholding is similar to binary thresholding, but it inverses the resulting binary image. Pixels with intensity values above the threshold are set to zero, while pixels with intensity values below the threshold are set to a maximum value (usually 255). This results in a binary image with inverted intensity values compared to binary thresholding.

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.