UPES

UNIVERSITY WITH A PURPOSE

Proposal On

AUTOMATED NUMBER PLATE DETECTION AND RECOGNITION

UPES
UNIVERSITY WITH A PURPOSE

Team Members

ROLL NO.	BRANCH	NAME
R103216079	CSE-BAO	Rajeshwarpreet S Boparai
R103216098	CSE-BAO	Smarth Galhotra
R100216126	CSE-BAO	Anisha Gera

UPES
UNIVERSITY WITH A PURPOSE

PROBLEM STATEMENT

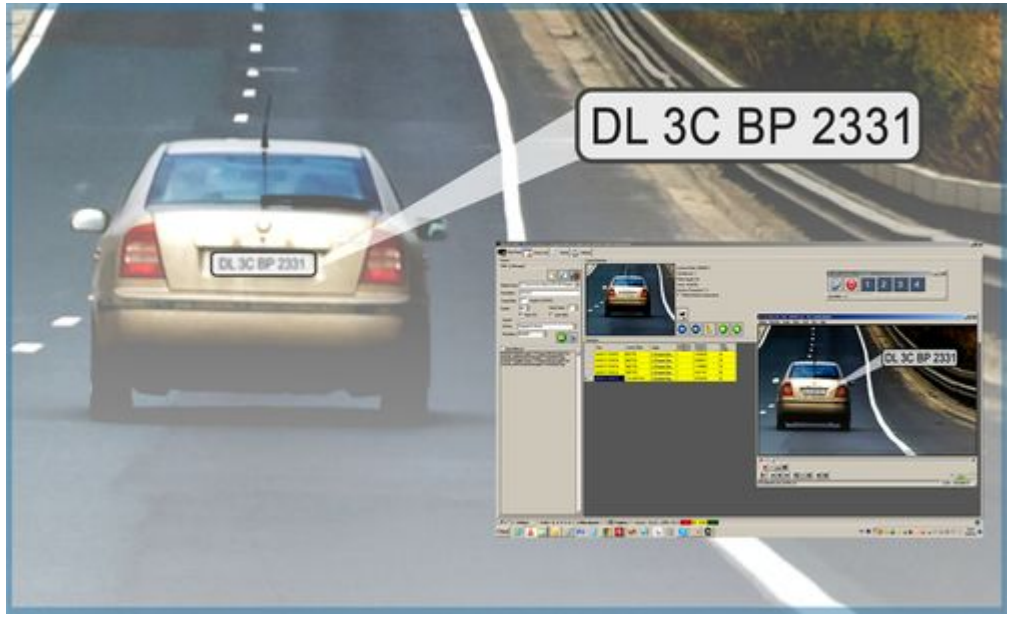
Available solutions work only in restricted conditions such as:

- Fixed Illumination
- Limited Vehicle Speed
- Stationary Backgrounds

UPES
UNIVERSITY WITH A PURPOSE

ABSTRACT

Automate detection and recognition of number plates in dynamic conditions



UPES
UNIVERSITY WITH A PURPOSE

INTRODUCTION

- Recognize Number Plates
- Remove current restrictions:
  - Fixed illumination
  - Limited vehicle speed
  - Stationary backgrounds
- Contribute to build Intelligent Transportation Systems (ITS)

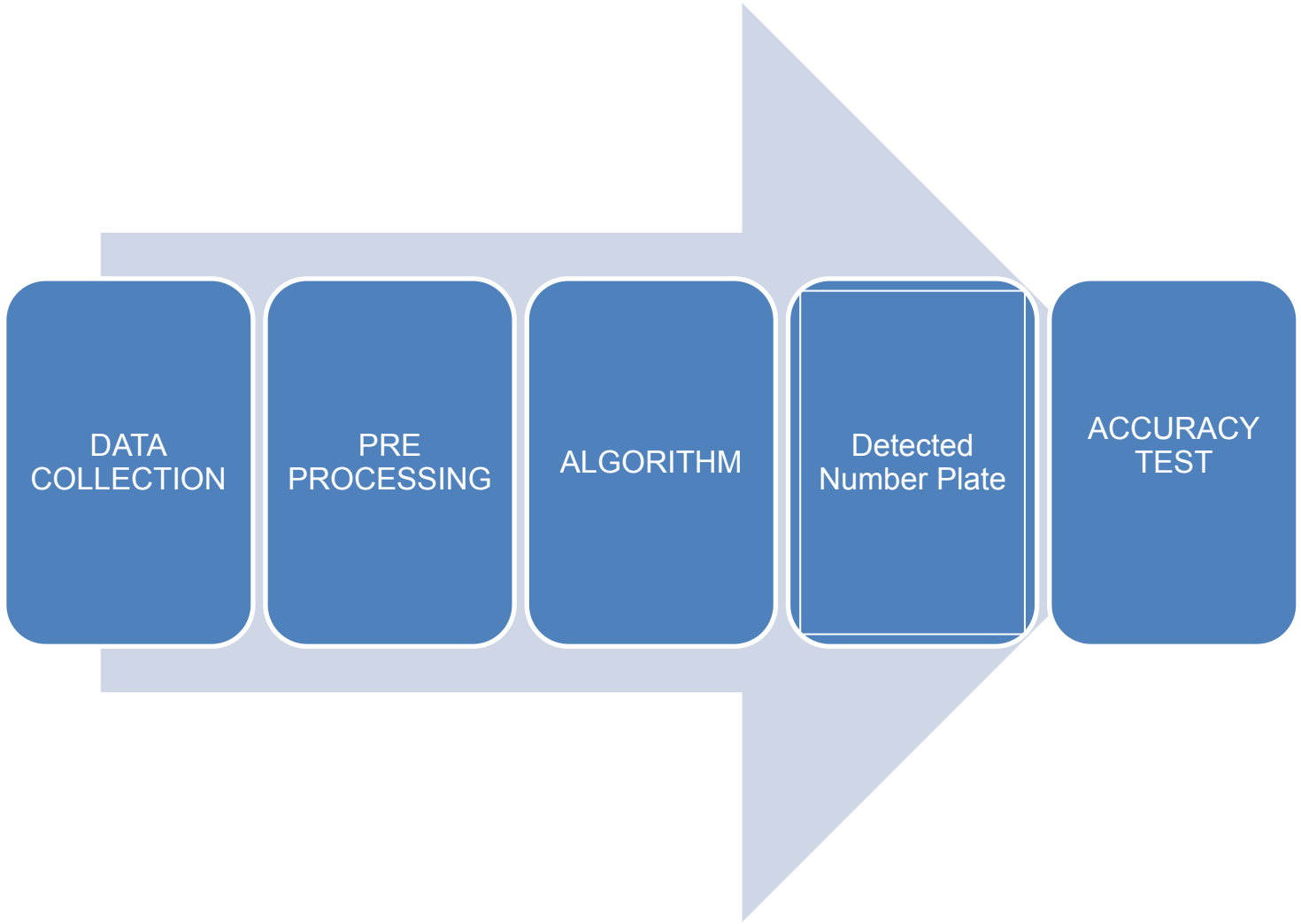
OBJECTIVE

Recognize License Plates efficiently in dynamic conditions.

UPES
UNIVERSITY WITH A PURPOSE

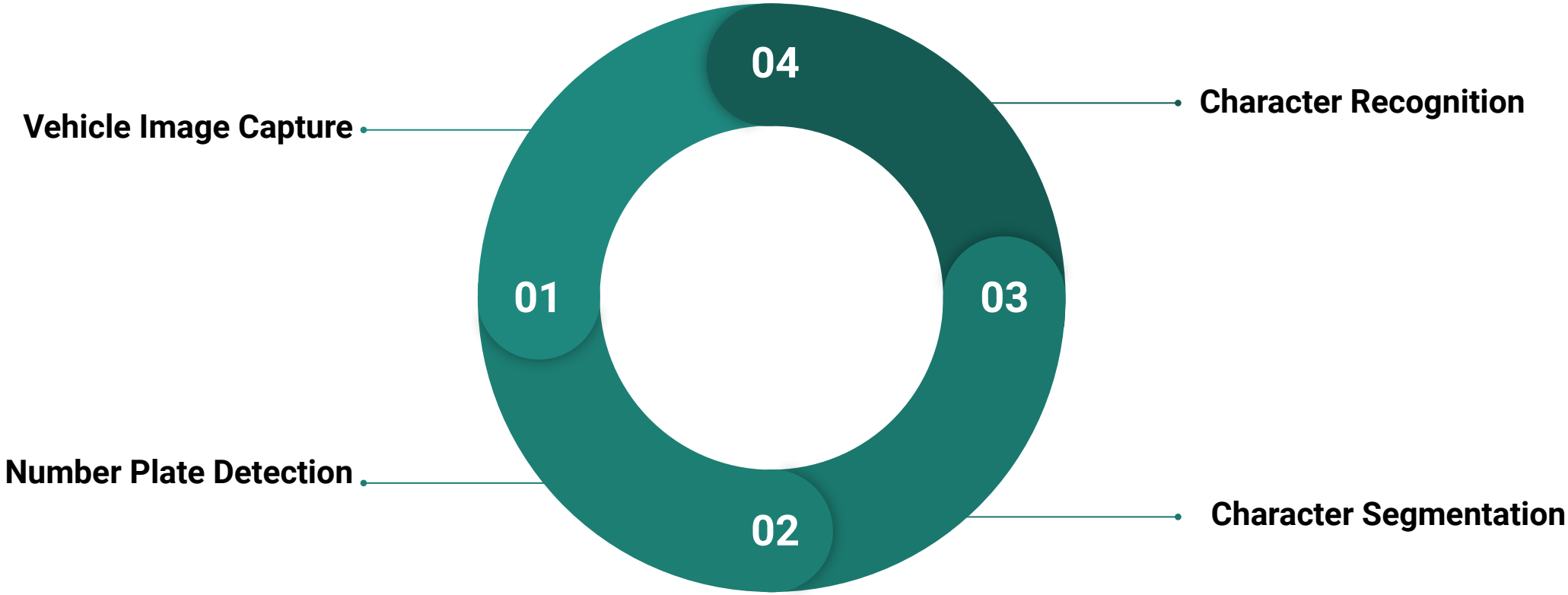
METHODOLOGY

Architecture of Automated detection and recognition of number plate



UPES
UNIVERSITY WITH A PURPOSE

METHODOLOGY



UPES
UNIVERSITY WITH A PURPOSE

METHODOLOGY

- Data Collection-
- Data Preprocessing
- Algorithm- Number Plate Detection
  Character Segmentation
  Character Recognition

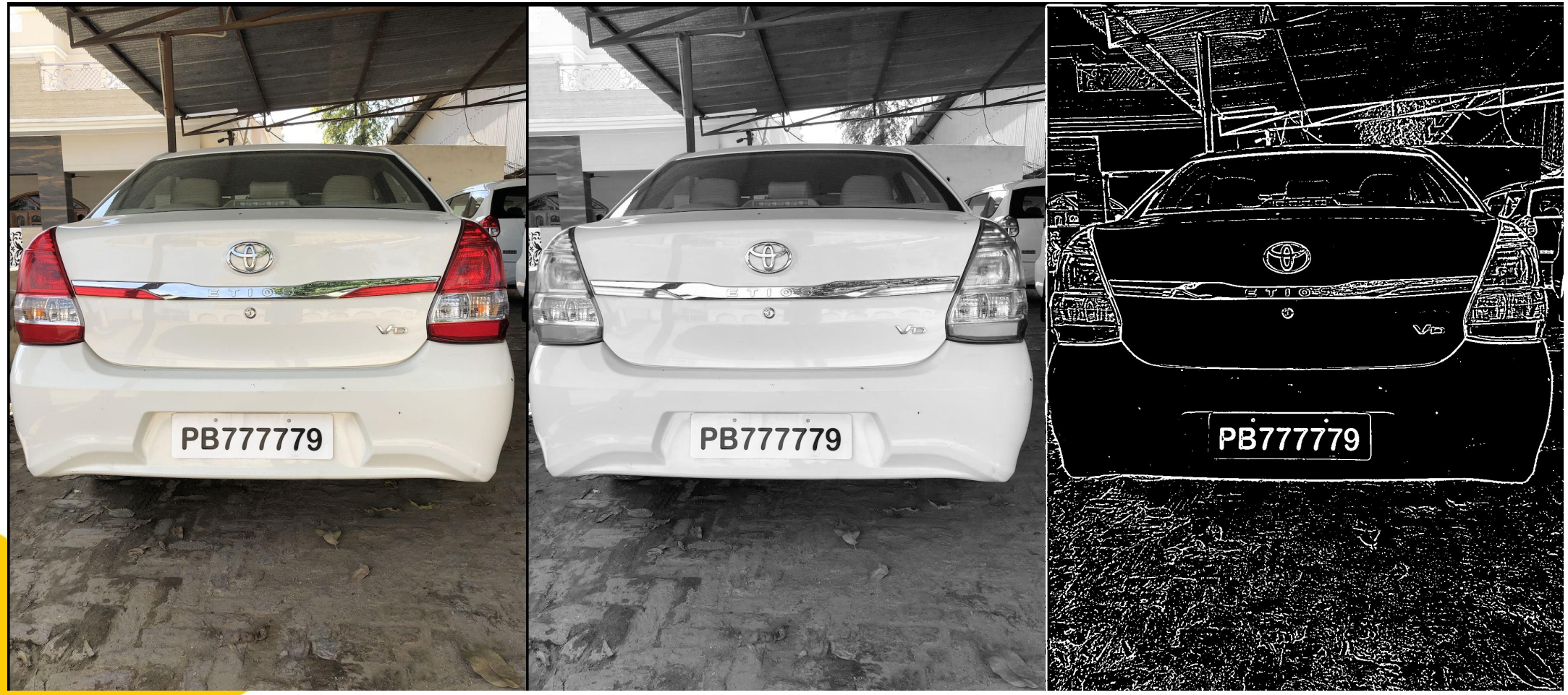
UPES
UNIVERSITY WITH A PURPOSE

METHODOLOGY

1. Find all “possible” Plates
   a. Pre Process
   b. Find contours
2. Look for characters in each possible plate
3. Print the characters found in most feasible plate

UPES
UNIVERSITY WITH A PURPOSE

Original -> GrayScale -> Threshold

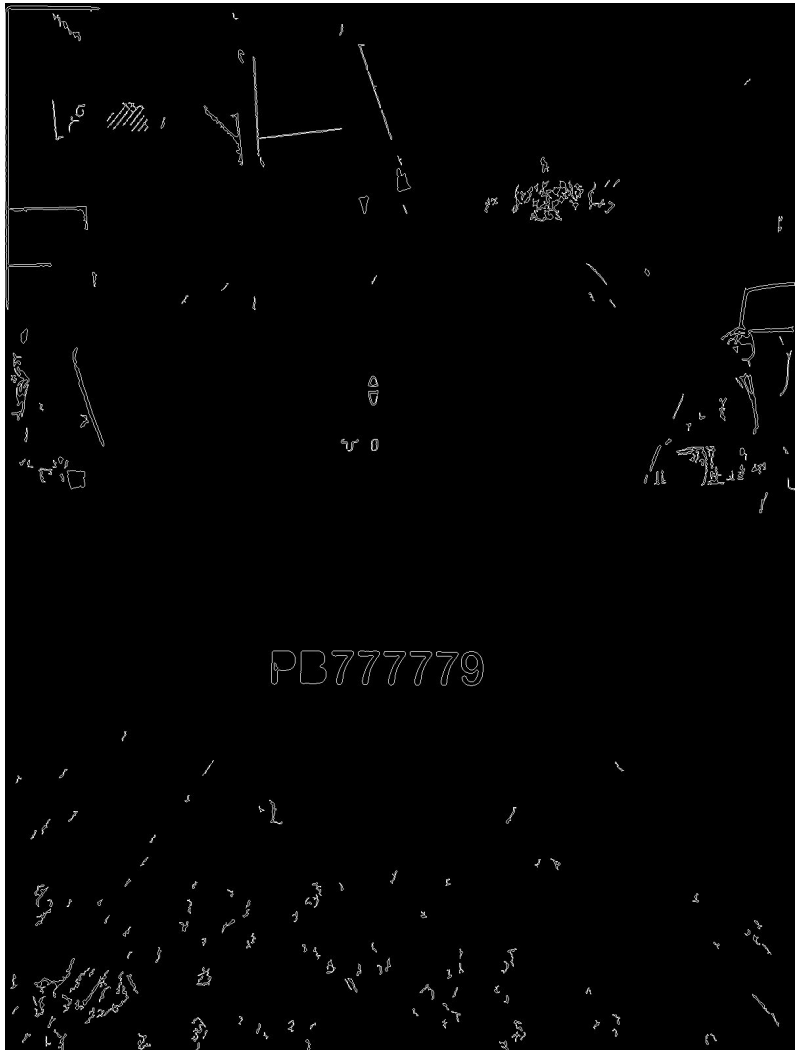


UPES
UNIVERSITY WITH A PURPOSE

All contours in thresholded image

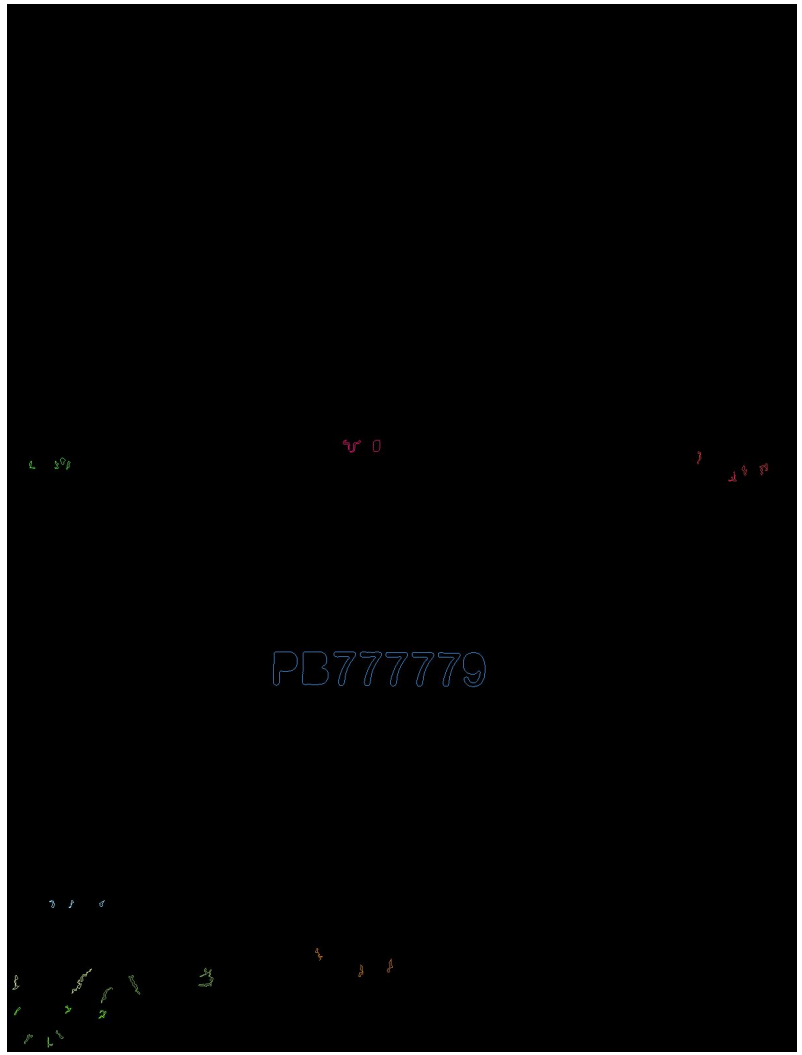
UPES
UNIVERSITY WITH A PURPOSE

Select contours selected on the basis of characteristics like width, height and aspect ratio



Grouping Characters

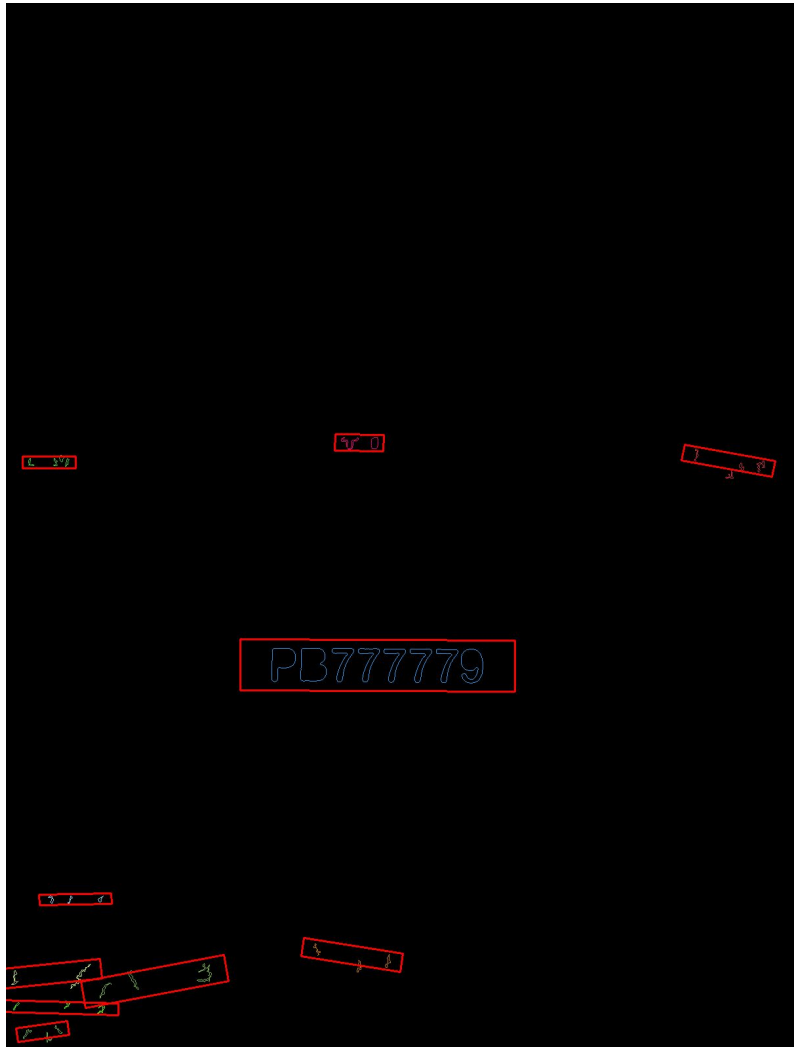
That may end up as License Plate



UPES
UNIVERSITY WITH A PURPOSE

Extracting Possible Plate

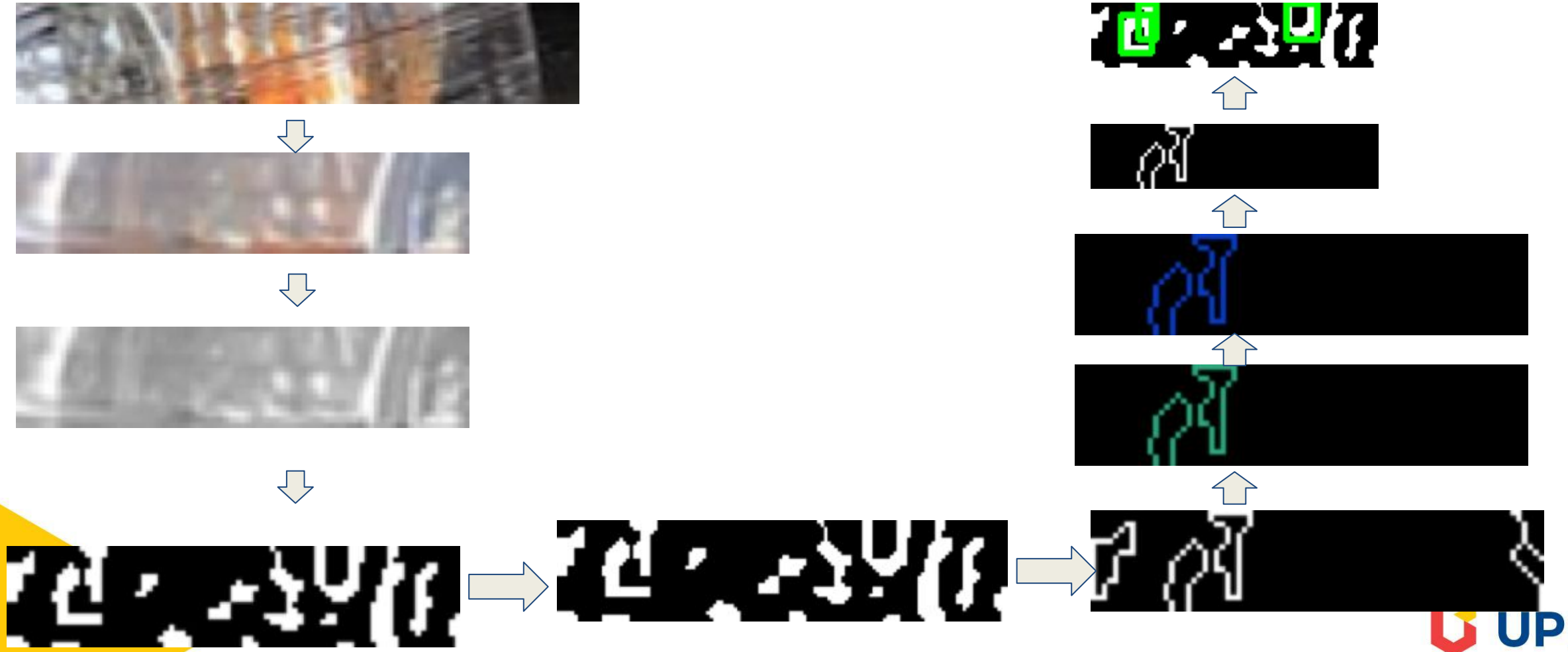
Give group of characters as possible plates



UPES
UNIVERSITY WITH A PURPOSE

Process Each Extracted Plate

Process Each Extracted Plate



Process Each Extracted Plate

```
chars found in plate number 1 = PB777779
```

```
chars found in plate number 2 = UIU
```

```
License plate read from image = PB777779
```

IMPLEMENTATION

Pre Processing

```
def preprocess(imgOriginal):
    imgGrayscale = extractValue(imgOriginal)
    imgMaxContrastGrayscale = maximizeContrast(imgGrayscale)
    height, width = imgGrayscale.shape
    imgBlurred = np.zeros((height, width, 1), np.uint8)
    imgBlurred = cv2.GaussianBlur(imgMaxContrastGrayscale, GAUSSIAN_SMOOTH_FILTER_SIZE, 0)
    imgThresh = cv2.adaptiveThreshold(imgBlurred, 255.0, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, ADAPTIVE_THRESH_BLOCK_SIZE, ADAPTIVE_THRESH_WEIGHT)
    return imgGrayscale, imgThresh
```

UPES
UNIVERSITY WITH A PURPOSE

IMPLEMENTATION

Plate Detection

```
def detectPlatesInScene(imgOriginalScene):
    listOfPossiblePlates = []
    height, width, numChannels = imgOriginalScene.shape
    imgGrayscaleScene = np.zeros((height, width, 1), np.uint8)
    imgThreshScene = np.zeros((height, width, 1), np.uint8)
    imgContours = np.zeros((height, width, 3), np.uint8)
    imgGrayscaleScene, imgThreshScene = Preprocess.preprocess(imgOriginalScene)
    listOfPossibleCharsInScene = findPossibleCharsInScene(imgThreshScene)
    listOfListsOfMatchingCharsInScene = DetectChars.findListOfListsOfMatchingChars(listOfPossibleCharsInScene)
    for listOfMatchingChars in listOfListsOfMatchingCharsInScene:
        possiblePlate = extractPlate(imgOriginalScene, listOfMatchingChars)
        if possiblePlate.imgPlate is not None:
            listOfPossiblePlates.append(possiblePlate)
        for i in range(0, len(listOfPossiblePlates)):
            p2fRectPoints = cv2.boxPoints(listOfPossiblePlates[i].rrLocationOfPlateInScene)
            cv2.line(imgContours, tuple(p2fRectPoints[0]), tuple(p2fRectPoints[1]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[1]), tuple(p2fRectPoints[2]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[2]), tuple(p2fRectPoints[3]), Main.SCALAR_RED, 2)
            cv2.line(imgContours, tuple(p2fRectPoints[3]), tuple(p2fRectPoints[0]), Main.SCALAR_RED, 2)
    return listOfPossiblePlates
```

UPES
UNIVERSITY WITH A PURPOSE

IMPLEMENTATION

Character Detection and Recognition in Plates

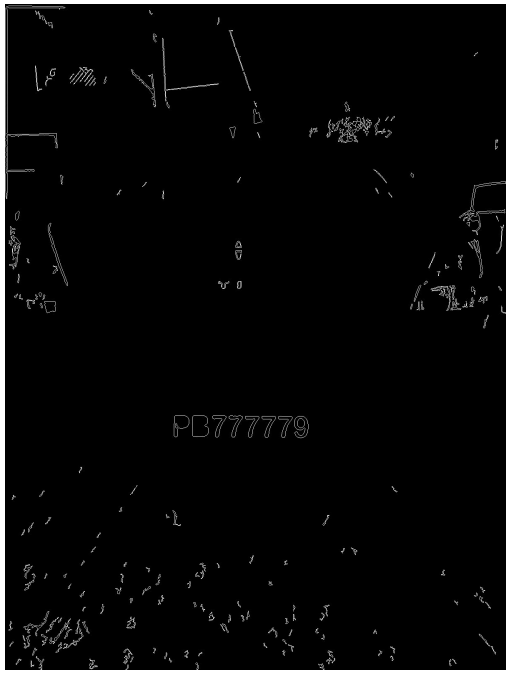
```
def detectCharsInPlates(listOfPossiblePlates):
    intPlateCounter = 0
    imgContours = None
    contours = []
    if len(listOfPossiblePlates) == 0:
        return listOfPossiblePlates
    for possiblePlate in listOfPossiblePlates:

        possiblePlate.imgGrayscale, possiblePlate.imgThresh = Preprocess.preprocess(possiblePlate.imgPlate)
        possiblePlate.imgThresh = cv2.resize(possiblePlate.imgThresh, (0, 0), fx = 1.6, fy = 1.6)
        thresholdValue, possiblePlate.imgThresh = cv2.threshold(possiblePlate.imgThresh, 0.0, 255.0, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
        listOfPossibleCharsInPlate = findPossibleCharsInPlate(possiblePlate.imgGrayscale, possiblePlate.imgThresh)
            for possibleChar in listOfPossibleCharsInPlate:
                contours.append(possibleChar.contour)
            cv2.drawContours(imgContours, contours, -1, Main.SCALAR_WHITE)
        listOfListsOfMatchingCharsInPlate = findListOfListsOfMatchingChars(listOfPossibleCharsInPlate)
        if (len(listOfListsOfMatchingCharsInPlate) == 0):
            possiblePlate.strChars = ""
            continue
        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):
            listOfListsOfMatchingCharsInPlate[i].sort(key = lambda matchingChar: matchingChar.intCenterX)
            listOfListsOfMatchingCharsInPlate[i] = removeInnerOverlappingChars(listOfListsOfMatchingCharsInPlate[i])
        intLenOfLongestListOfChars = 0
        intIndexOfLongestListOfChars = 0
        for i in range(0, len(listOfListsOfMatchingCharsInPlate)):
            if len(listOfListsOfMatchingCharsInPlate[i]) > intLenOfLongestListOfChars:
                intLenOfLongestListOfChars = len(listOfListsOfMatchingCharsInPlate[i])
                intIndexOfLongestListOfChars = i
        longestListOfMatchingCharsInPlate = listOfListsOfMatchingCharsInPlate[intIndexOfLongestListOfChars]
        possiblePlate.strChars = recognizeCharsInPlate(possiblePlate.imgThresh, longestListOfMatchingCharsInPlate)
    return listOfPossiblePlates
```

UPES
UNIVERSITY WITH A PURPOSE

Result

UPES
UNIVERSITY WITH A PURPOSE

Result

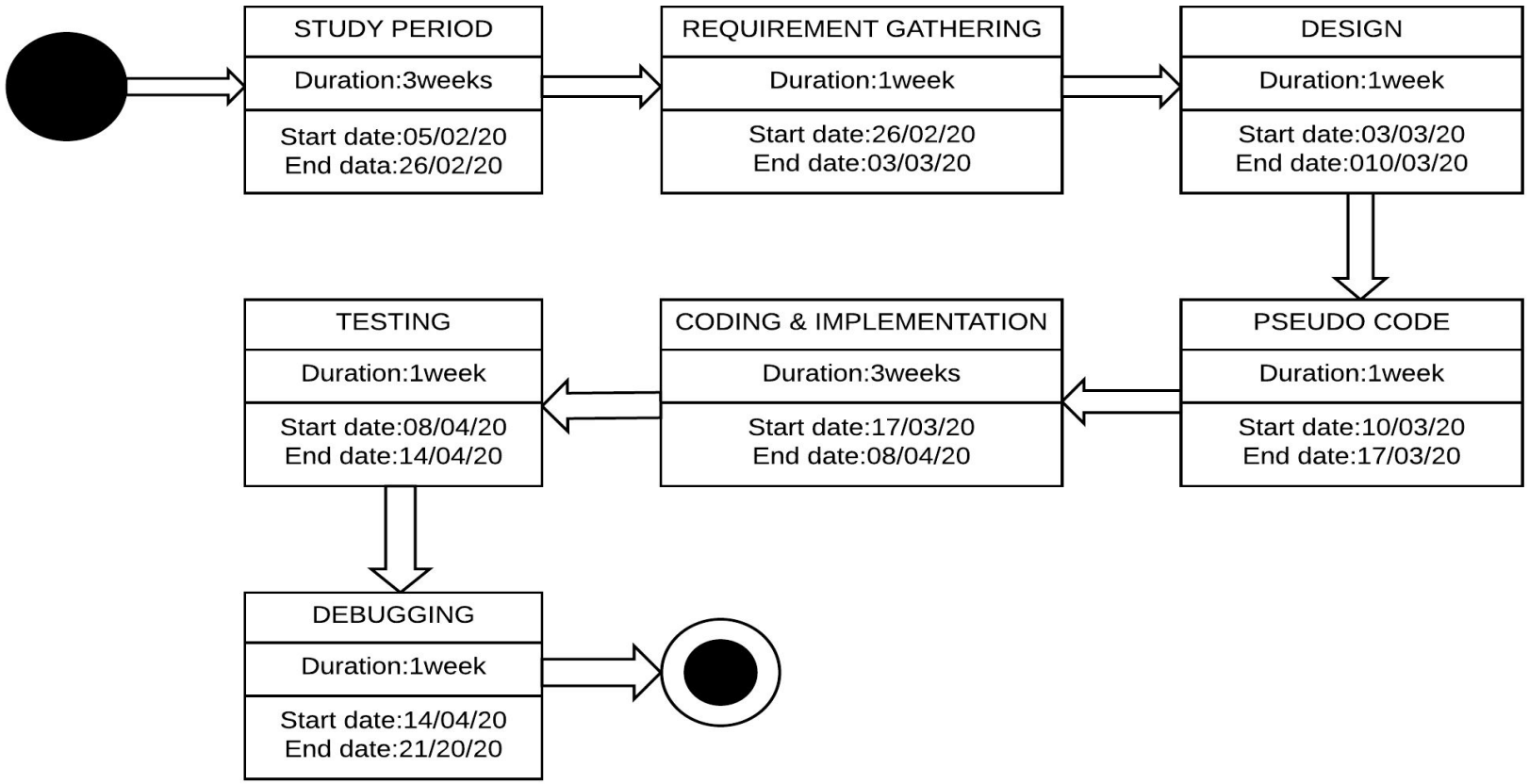


UPES
UNIVERSITY WITH A PURPOSE

Result

UPES
UNIVERSITY WITH A PURPOSE

SCHEDULE


STUDY PERIOD
Duration:3weeks
Start date:05/02/20
End data:26/02/20
REQUIREMENT GATHERING
Duration:1week
Start date:26/02/20
End date:03/03/20
DESIGN
Duration:1week
Start date:03/03/20
End date:010/03/20
PSEUDO CODE
Duration:1week
Start date:10/03/20
End date:17/03/20
CODING & IMPLEMENTATION
Duration:3weeks
Start date:17/03/20
End date:08/04/20
TESTING
Duration:1week
Start date:08/04/20
End date:14/04/20
DEBUGGING
Duration:1week
Start date:14/04/20
End date:21/20/20

References

[1] “Automatic Number Plate Recognition System” by Amr Badr etal Mohamed M. Abdelwahab.

[2] “ Automated Car Number Plate Detection System to detect far number plates” by Anisha goyal OSR Journal of Computer Engineering (IOSR-JCE)

[3] Blog article ”Automatic License Plate Detection & Recognition using deep learning” by Achraf Kharzi

[4]Automatic Number Plate Recognition System(ANPR): A Survey by Chirag Patel etal Dipti Shah International Journal of Computer Applications (0975 – 8887)

[5]Medium blog article “YOLO — You only look once, real time object detection explained” by Manish Chablani

UPES
UNIVERSITY WITH A PURPOSE

THANK YOU