ROADMAP...

01

What is digital image processing? (Basic Introduction)

02

How text detection is performed? (Steps and algorithm involved)

03

What are the various Functions involved?

04

Where do we see its Applications?

INTRODUCTION

- Automatically Detect and Recognize Text in Natural Images This example shows how to detect regions containing text in an image. It is a common task performed on unstructured scenes, for example when capturing video from a moving vehicle for the purpose of alerting a driver about a road sign. Segmenting out the text from a cluttered scene greatly helps with additional tasks such as optical character recognition (OCR).
- The automated text detection algorithm in this example starts with a large number of text region candidates and progressively removes those less likely to contain text. To highlight this algorithm's flexibility, it is applied to images containing a road sign, a poster and a set of license plates.

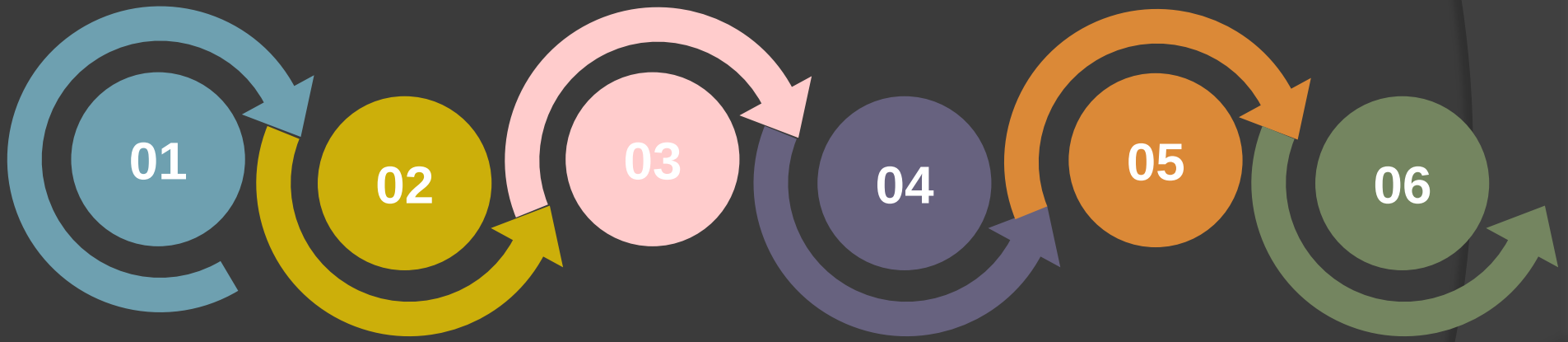
METHODOLOGY

- The software model uses image processing technology. The programs are implemented in MATLAB.

- The algorithm is divided into the following parts:
  - Capture image,
  - Edge Enhanced MSER
  - Geometric Filtering
  - Stroke Width Computation
  - Letter Pairing
  - Text Line Formation.
  - Word Separation

- The flow chart of various stages involved in this text recognition system implementation is shown in the following figure.

FLOW CHART



Step 01
Image Loading

Step 02
MSER Region Detection

Step 03
Canny Edges Detection

Step 04
Character Candidate Filteration

Step 05
Text Region Boundary Determination

Step 06
OCR performance on text region

Flow Work Process

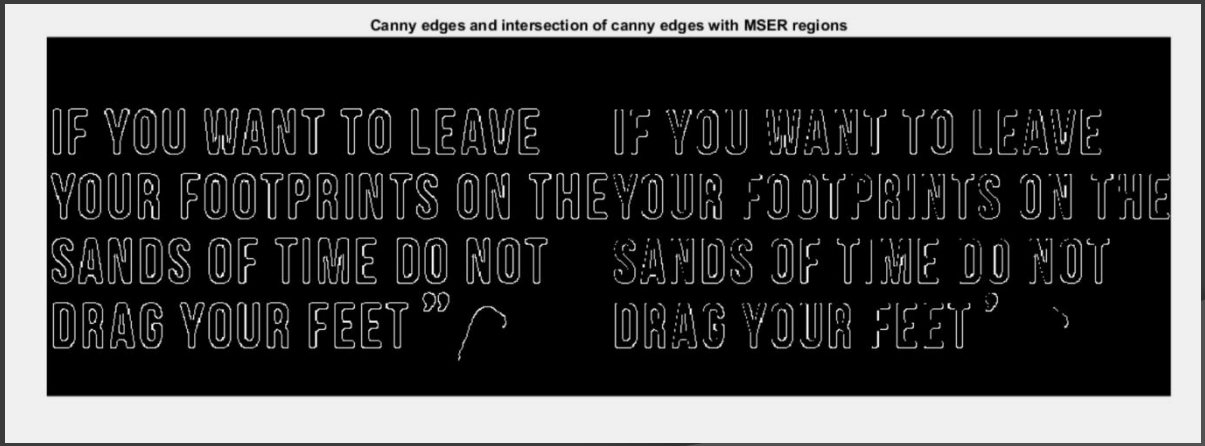
- Image Loading

MSER Regions Detection

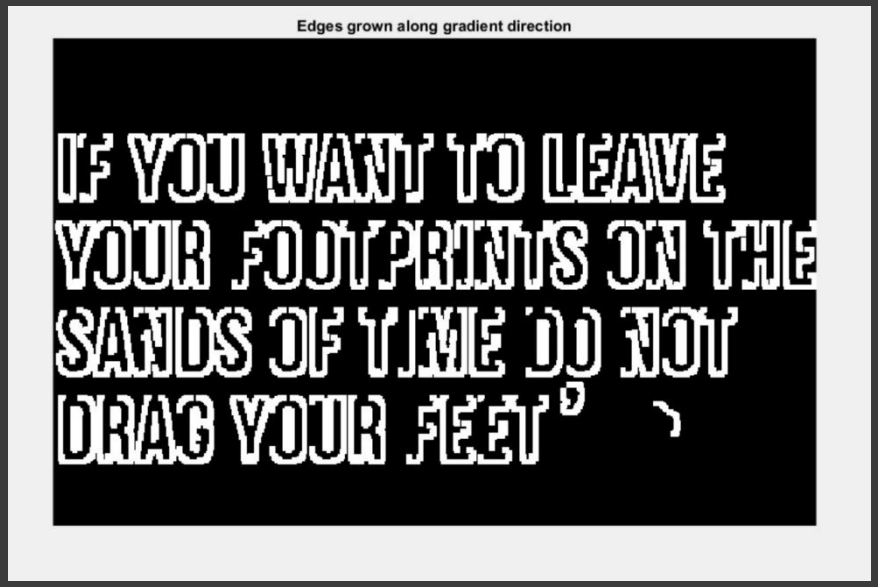
- Since text characters usually have consistent color, we begin by finding regions of similar intensities in the image using the MSER region detector
- By function detect MSER Features (I ,'Region Area Range', [limits]) which returns an MSER Regions containing information about MSER features detected in the 2-D grayscale input image, I. This object uses Maximally Stable Extremal Regions (MSER) algorithm to find regions.

- Canny edges Detectionand intersection of canny edges with MSER regions

a. Use Canny Edge Detector to Further Segment the Text Since written text is typically placed on clear background, it tends to produce high response to edge detection. Further more, an intersection of MSER regions with the edges is going to produce regions that are even more likely to belong to text. then Find intersection between edges and MSER regions.



- Note that the original MSER regions in |mserMask| still contain pixels that are not part of the text. We can use the edge mask together with edge gradients to eliminate those regions. Grow the edges outward by using image gradients around edge locations.



- This mask can now be used to remove pixels that are within the MSER regions but are likely not part of text. Remove gradient grown edge pixels

In this image, letters have been further separated from the background and many of the non-text regions have been separated from text.

Filter Character Candidates Using Connected Component Analysis

- Some of the remaining connected components can now be removed by using their region properties. The thresholds used below may vary for different fonts, image sizes, or languages.
- We find the regions properties for connected components (like area, eccentricity, solidity)
- Compare those property values with standard measurements.Like area should lie b/w specific range, and so on .
- Eliminate regions that do not follow common text measurements and visualizing result of filtering.

Character Candidates Using the Stroke Width Image.

- Another useful discriminator for text in images is the variation in stroke width within each text candidate. Characters in most languages have a similar stroke width or thickness throughout. It is therefore useful to remove regions where the stroke width exhibits too much variation .
- First compute distance b/w each pixels with the nearest non zero pixel [bwdist ()]
- And then compute the stroke width image, by using helperstrokewidth(distanceimage);

- **Note that most non-text regions show a large variation in stroke width. These can now be filtered using the coefficient of stroke width variation**
- So we find out remaining components by compute normalized stroke width variation and compare to common value and remove from text candidates.

Determine Bounding Boxes Enclosing Text Regions.

- To compute a bounding box of the text region, we will first merge the individual characters into a single connected component. This can be accomplished using morphological closing (tends to close gaps in the image.) followed by opening to clean up any outliers.
- Then finding bounding boxes of large regions (where major text region)

Perform Optical Character Recognition on Text Region

- The segmentation of text from a cluttered scene can greatly improve OCR results. Since our algorithm already produced a well segmented text region, we can use the binary text mask to improve the accuracy of the recognition results.
- Ocrtext = Ocr(afterstrokewidthtextmask ,boxes) ;
- Ocr.text; % to get the text output

FUNCTIONS USED

- detectMSERFeatures(I): returns an MSER Regions object, regions, containing information about MSER features detected in the 2-D grayscale input image, I. This object uses Maximally Stable Extremal Regions (MSER) algorithm to find regions.

- **vertcat(A1,A2,…,An):** concatenates A1, A2, … , An vertically. vertcat is equivalent to using square brackets for vertically concatenating arrays. For example, [A; B] is equal to vertcat(A,B) when A and B are compatible arrays.

- sub2ind(sz,row,col): returns the linear indices ind corresponding to the row and column subscripts in row and col for a matrix of size sz. Here sz is a vector with two elements, where sz(1) specifies the number of rows and sz(2) specifies the number of columns.

- edge(I,method) :returns a binary image containing 1s where the function finds edges in the grayscale or binary image I and 0s elsewhere. By default, edge uses the Sobel edge detection method(like ‘canny’).

- imgradient(I)=[Gmag,GDir] : returns the gradient magnitude, Gmag, and the gradient direction, Gdir, of the 2-D grayscale or binary image I.

- imshowpair(A,B, method) creates a composite RGB image showing A and B overlaid in different color bands. To choose another type of visualization of the two images, use the method argument. If A and B are different sizes, imshowpair pads the smaller dimensions with zeros on the bottom and right edges so that the two images are the same size. By default, imshowpair scales the intensity values of A and B independently from each other. imshowpair returns obj, an image object.

- bwconncomp(Binaryimage)=CC: finds and counts the connected components (CC) in the binary image .
  The CC output structure contains the total number of connected components, such as regions of interest (ROIs), in the image and the pixel indices assigned to each component.

- regionprops(Binaryimage,properties)=stats: returns measurements for the set of properties for each 8-connected component (object) in the binary image. You can use regionprops on contiguous regions and discontiguous regions .

  regionprops returns the "Area", "Centroid", and "BoundingBox" measurements.

- bwdist(Binaryimage)= D : computes the Euclidean distance transform of the binary image . For each pixel in Binary image, the distance transform assigns a number that is the distance between that pixel and the nearest nonzero pixel of Binary image.

- std(A) : returns the standard deviation of the elements of A along the first array dimension whose size does not equal 1. By default, the standard deviation is normalized by N-1, where N is the number of observations.

- strel("disk",r)=SE: creates a disk-shaped structuring element, where r specifies the radius.
- imclose(I,SE) = J : performs morphological closing on the grayscale or binary image I, using the structuring element SE. The morphological close operation is a dilation followed by an erosion, using the same structuring element for both operations.

- imclose(I,SE) = J : performs morphological closing on the grayscale or binary image I, using the structuring element SE. The morphological close operation is a dilation followed by an erosion, using the same structuring element for both operations.

- imopen(I,SE)= J: performs morphological opening on the grayscale or binary image I using the structuring element SE. The morphological opening operation is an erosion followed by a dilation, using the same structuring element for both operations.

APPLICATIONS

- Preprocessing and Extraction of fileds of bank check.
- Used as a signal reader in auto pilot driving cars.
- Makes our life easy by optimising the search on browser

CONCLUSION

- We have implemented text recognition using OCR. Our algorithm successfully detects the text region from the image, which consists of meaningful information & then recognizes the characters.
- We have applied our algorithm on many images and found that it successfully recognizes them.
- This project successfully detects the embedded text to a great extent, although it has its own limitations.