Document Image Processing

Mini Project

ADVISOR: PROF. SAMIT
BISWAS

Team Members:

1. **PENUGONDA S M S N RAJESH (2020CSB064)**
2. **ASTIK GORAI (2020CSB038)**
3. **GOURAV KUMAR SHAW (2020CSB010)**
4. **BABBADI GOVARDHAN RAO NAIDU (2020CSB060)**
5. **TAMOGHNA ROY (2020CSB011)**
6. **ANKIT RAJPUT (2020CSB081)**

Contents:

Document Image Processing

Mini Project

Abstract

Computer Vision is a field of Artificial Intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. Until recently, computer vision only worked in limited capacity. Thanks to advances in artificial intelligence and innovations in deep learning and neural networks, the field has been able to take great leaps in recent years and has been able to surpass humans in some tasks related to detecting and labeling objects.

One of the driving factors behind the growth of computer vision is the amount of data we generate today that is then used to train and make computer vision better.

By 2022, the computer vision and hardware market is expected to reach $48.6 billion

Today, image processing with its wide array of applications such as - enhancing images, extracting text from images, detecting edges in images, applying of cosmetic effects, details-fetching, etc. finds use in myriad of industries including but not limited to the medical field, robotics., neural networking, criminal investigation, communication devices. This project therefore deals with the applications of digital image processing.

Introduction

1.1 DEFINITION

WHAT IS AN IMAGE

An image is defined as a two-dimensional function F (X, Y), where x and y are coordinates. In other words, an image can be defined by a two-dimensional array specially arranged in row and columns. A digital image is composed of a finite number of elements.

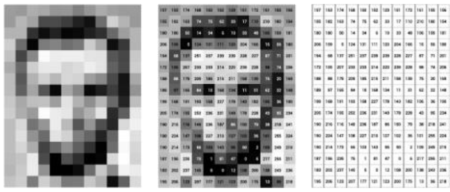
WHAT IS DIGITAL IMAGE PROCESSING?

Digital image processing is the use of a digital computer to process digital images through an algorithm. It is primarily used to enhance images or extract some useful information about it.

Today, it is a rapidly growing technology. It forms core research area within engineering and computer science disciplines too. Image processing has its wide applications in robotics, machine learning, neural networking, signal processing, medical field, graphics and animations and in many other fields.

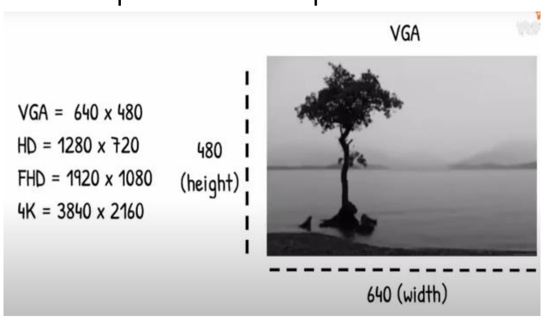
1.2 HISTORY

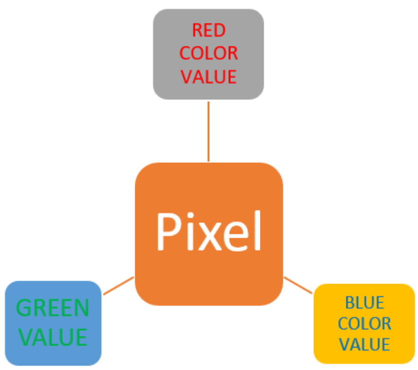
Many techniques of digital image processing were developed at the Jet Propulsion Laboratory, the cost of processing of image was very high. But all that changed in 1970s, when digital image processing. grew rapidly as cheaper computer and hardware became available



Massachusetts Institute of Technology, Bell Laboratories, University of Maryland in the 1960s.

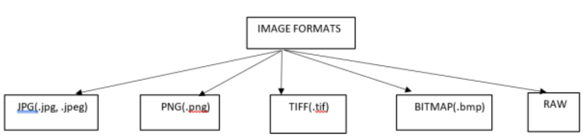
Images then could be processed in real time. With the development of fast computers available in the





2000s, digital image processing has become one of the most common forms of image processing. It is used so abundantly not only because it is the most versatile method, but also the cheapest.

1.3 DIFFERENT IMAGE FORMATS



RAW IMAGE FILES: These are unprocessed images created by a camera or scanner. Many digital cameras can shoot in RAW. These images are the equivalent of a digital negative meaning that they hold a lot of image information. These images need to processed in an editor such as Adobe Photoshop or Light room.

TIFF Tagged Image File Format: This format stores image data without losing any data. It does not perform any compression on images. Quality images are obtained but size of the image is also large, which is good for printing

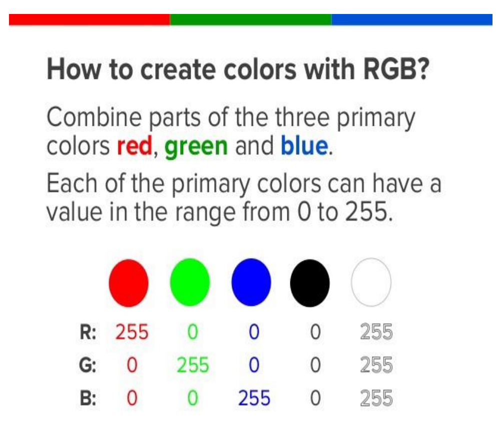
1.4 REPRESENTATION OF IMAGES

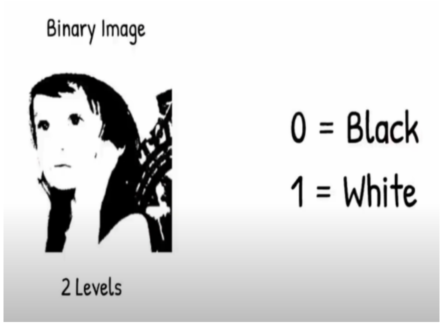
PIXEL: PICTURE + ELEMENT - It is the smallest part of an image. We can think of an image as a combination of millions of pixels. If we look at an image properly, we can see there are multiple small boxes the image contains, that boxes are pixel.

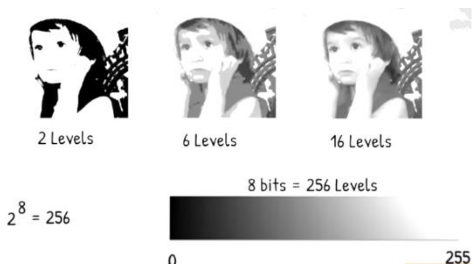
Pixel data diagram. At left, our image of Lincoln; at center, the pixels labeled with numbers from 0–255, representing their brightness; and at right, these numbers by themselves.

1.5 COMPONENTS OF PIXEL

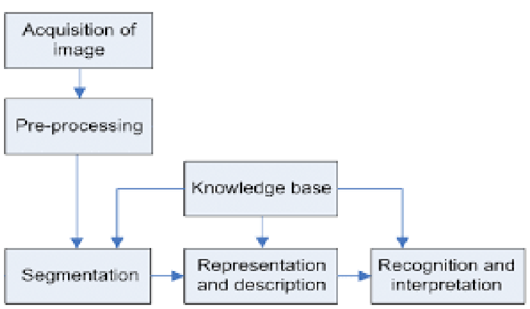
Each of the color takes only one 1byte memory. So, a pixel takes total 3 bytes of memory. Since each component of the pixel takes only 1 byte of memory to represent the color, therefore each component can take 256 values. Thus, total color combinations is 256*256*256 =1,67,77,216.







1.6 STAGES IN IMAGE PROCESSING



1.7 APPLICATIONS

Almost in every field, digital image processing puts a live effect on things and is growing with time to time and with new technologies.

<u>Image sharpening and restoration-</u>

It is the process in which we can modify the image. We can convert the color image to grey image, sharpening, enhancement of the image, detecting edges, and recognition of images.

<u>Medical field-</u>

Now a days if we have brain tumor through the image processing the tumor is detect that where the tumor is. Also, it is used to detect any kind of cancer.

X-ray imaging, medical CT scan, UV imaging depends on the functioning of digital image processing.

<u>Robot -Vision-</u>

There are several robotic machines which work on this technique. Through this technique robots find their ways. Like they can detect the hurdle and line follower robot.

<u>Pattern- recognition-</u>

It involves study of image- processing. It is also combined with the artificial intelligence such that computer-aided diagnosis, handwriting- recognition and images- recognition can be easily implemented.

<u>Video processing-</u>

The collection of frames and pictures are arranged in such a way that movement of pictures become faster. It involves frame rate, motion detection, reduction of noise and color space conversion etc.

1.8 ADVANTAGES AND DISADVANTAGES

<u>Advantage</u>

Processing of images are faster. It requires less time to process the image. There is no need of films and other photographic equipment.

Interactive method for detecting face, recognizing fingerprints, detecting cancer.

It is an eco-friendly process since it does not require chemicals while processing images.

We can change the quality of image. We can compress, enhance, and quality of image produced is good. Image can be made in any required format.

Now a days each and every book is available on the digital stage. The demand and needs of people are changing so having optimized digital book is need of today's generation. So, the digital image processing plays vital role in publishing world.

Errors in images can easily be rectified. It analyses blood cells and their composition in our body.

Through this technique robots can detect their surroundings(vision). It also helps in pattern recognition

Disadvantage

It is more time consuming. It is cost effective. More complex programs are required for implementing digital image processing.

What is An Image

An image is defined as a two-dimensional function F (X, Y), where x and y are co-ordinates. In other words, image can be defined by two-dimensional array specially arranged in row and columns. Digital image is composed of a finite number of elements.

What Is Digital Image Processing?

Digital Image Processing is the use of a digital computer to process digital images through different algorithm. It is used to enhanced the images to get some useful information about it.

Types Of Image Formats

JPEG (.jpeg, .jpg) : JPEG stands for "Joint Photographic Experts Group". It's a standard image format for containing lossy and compressed image data. Despite the huge reduction in file size JPEG images maintain reasonable image quality.

PNG (.png) : PNG is short for Portable Network Graphic, a type of raster image file. It's particularly popular file type with web designers because it can handle graphics with transparent or semi-transparent backgrounds.

TIFF(.tif) : TIFF stands for "Tagged Image File Format". This format stores image data without losing any data. It do not perform any compression on images Quality images are obtained but size of the image is also large. which is good for printing.

BITMAP (.bmp) : Bitmap, method by which a display space (such as a graphics image file) is defined, including the color of each of its pixels (or bits). In effect, a bitmap is an array of binary data representing the values of pixels in an image or display. A GIF is an example of a graphics image file that has a bitmap.

RAW IMAGE FILES: These are unprocessed images created by a camera or scanner. Many digital cameras can shoot in RAW these images are the equivalent of a digital negative meaning that they hold a lot of image information. These images need to processed in an editor such as Adobe Photoshop or Light room.

Representation Of Images

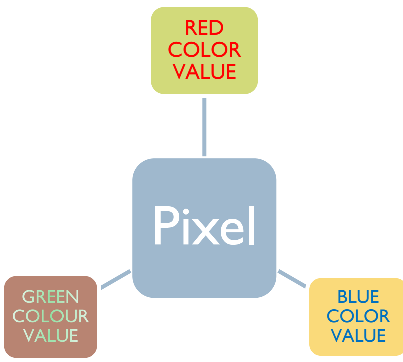
PIXEL: PICTURE + ELEMENT

It is the smallest part of an Image. We can think a image is combination of millions of pixel. If we look an image properly, we can see there are multiple small Boxes the image contains, that boxes are pixel.

Components Of Pixel

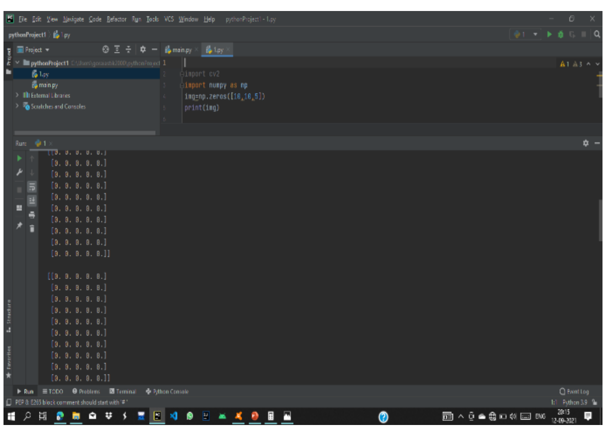
Each of the color takes only one 1byte memory. So a pixel takes total 3 bytes of memory

Since each component of the pixel takes only 1 byte of memory to represent the color. So each component can take 256 values. So total color combination is 256*256*256 =1,67,77,216



How Computer Stores Images?

Well Computers can only understand 0 or 1. So whatever we want to store in computer it stores in 0/1 format Images are 2D so computer store each pixel value in a 2D array. Here we have created a black image And print it. It shows an 2D array with 0 values. If it is a color image then it stores corresponding color values for each pixel in a 2D array.



Basic Python Opencv Function

HOW TO READ AN IMAGE:

To read the images "cv2.imread()" method is used. This method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Syntax: cv2.imread(path, flag)

Parameters:

path: A string representing the path of the image to be read.

flag: It specifies the way in which image should be read. It's default value is cv2.IMREAD_COLOR.

Return Value: This method returns an image that is loaded from the specified file

How To Show An Image?

cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.

Syntax: cv2.imshow(window name, image)

Parameters:

window name: A string representing the name of the window in which image to be displayed.

image: It is the image that is to be displayed.

Return Value: It doesn't return anything

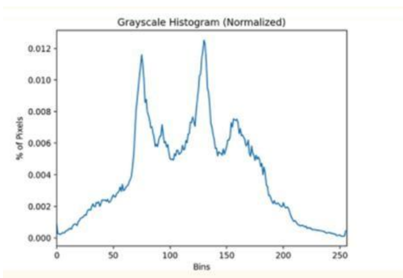
Histogram

Definition:

A diagram consisting of rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval.

So what is for image?

The histogram of an image is a function that maps each gray level of an image to the number of times it occurs in the image.



Characteristics Of Histogram

Total Number of Pixels: The total number of pixels constituting the image can be obtained by adding up the number of pixels corresponding to each gray level.

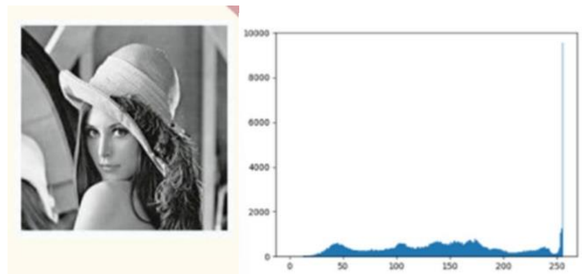
Image Brightness: You can get a general idea of the brightness of an image by looking at the histogram and observing the spatial distribution of the values. If the histogram values are concentrated toward the left, the image is darker. If they are concentrated toward the right, the image is lighter.

Contrast of an image: A histogram in which the pixel counts evenly cover a broad range of grayscale levels indicates an image with good contrast. Pixel counts that are restricted to a smaller range indicate low contrast

Algorithm

First, we have to create a 2D array of size equal to the dimensions of the image with all the zeros in it. We need to traverse through the whole image and for every pixel, increase the value of corresponding gray level in the 2D array.

After we have to plot the histogram having intensity level on X-axis and number of pixels in Y- axis. We can use Matplotlib for plotting the histogram.



Histogram By Library Function and Comparison

By Library Function:

We use OpenCV library in python.

```
img = cv.imread('sample.png', 0)
cv.imshow("Image", img)
plt.hist(img.ravel(), 256, [0, 256])
plt.show()
```

Histogram Equalization

Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast

Histogram equalization is commonly used in order to enhance the contrast of the image. By this technique we spread the histogram throughout the gray levels as much as possible

This technique can't guarantee to always improve the quality of the image

By spreading pixel values to all gray levels, the quality of the image sometime increases

Contrast Stretching

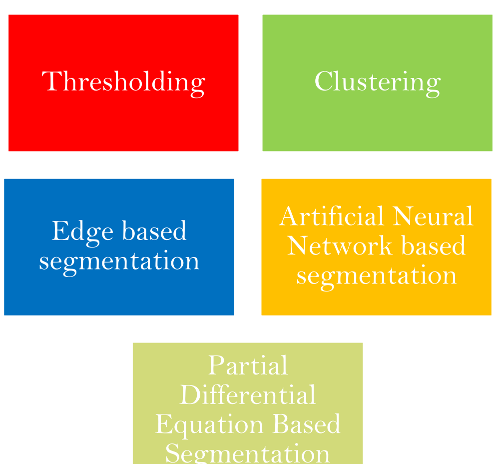
Contrast stretching or Normalization is one operation applied on images, which improves the contrast of the image so that the details present in the image can be clearly seen. Contrast stretching is applied directly on the image by modifying each pixel present in the image- that is by applying point operations rather than using a kernel

By not using a kernel means each pixel value is not determined using the values of a neighborhood pixels. The intensity value of the pixel is modified using a normalization formula.

Image Segmentation:

Image segmentation is the process of the partition the image into different set of pixels.

NOW WE HAVE TO CHOOSE A THRESHOLD VALUE. HOW CAN WE TAKE THE THRESHOLD VALUE?



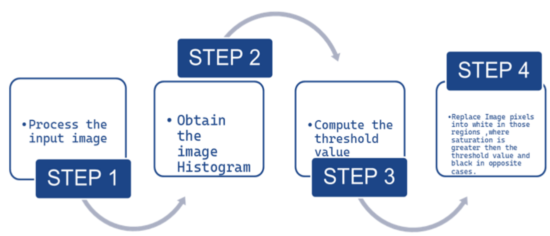
Frequently we have done many layers of processing steps and want either to make a final decision about the pixels in an image or to categorically reject those pixels below or above some value while keeping the others. Th e OpenCV function cv2.threshold() accomplishes these tasks. The basic idea is that an array is given, along with a threshold, and then something happens to every element of the array depending on whether it is below or above the threshold.

Definition: Thresholding is a very popular segmentation technique used for separating objects from its background The process involves comparing each pixel of the image to a given pixel called threshold. Depending upon the threshold value we divide the image into two types:

-pixel having value less then threshold value

-pixel having higher value then threshold

ALGORITHM FOR THRESHOLD TECHNIQUE



CHOOSING THE THRESHOLD VALUE

Depending on the process of choosing threshold value we can divide the threshold technique into mainly two types:

→ Simple Thresholding

→Adaptive Thresholding

SIMPLE THRESHOLDING:

In simple thresholding process we take a predefined pixel value as threshold value. It is for the whole image, that is, the given threshold value acts as a global threshold value.

Based on this threshold value we divide the images into two regions - one group having pixel values higher than the predefined threshold value and another having lower than the threshold value.

Application example: Separate out regions of an image corresponding to objects which we want to analyse. This separation is based on the variation of intensity between the object pixels and the background pixels.

THIS IS A PREDEFINED FUNCTION IN OPENCV. WE CAN USE IT TO TRY THESE VARIOUS TYPES OF THRESHOLD TECHNIQUE

DIFFERENT TYPES OF SIMPLE THRESOLDING

We can effectuate 5 types of Thresholding operations with this function:

BINARY THRESOLD

INVERTED BINARY THRESOLD

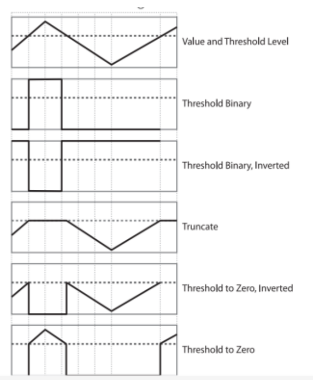
```
_,th1= cv2.threshold(src,threshold valu,max value of grey level, threshold types)
```

TRUNCATE THRESOLDING

THRESOLD TO ZERO

THRESOLD TO ZERO INVERSE

```
double cvThreshold( CvArr* src,
CvArr* dst,
double threshold,
double max_value,
int threshold_type );
```



Results of varying the threshold type in cv2.threshold(). The horizontal line through each chart represents a particular threshold level applied to the top chart and its effect for each of the five types of threshold operations below

Threshold type	Operation
CV_THRESH_BINARY	$dst_i = (src_i > T) \, ? \, M : 0$
CV_THRESH_BINARY_INV	$dst_i = (src_i > T) \, ? \, 0 : M$
CV_THRESH_TRUNC	$dst_i = (src_i > T) \, ? \, M : src_i$
CV_THRESH_TOZERO_INV	$dst_i = (src_i > T) \, ? \, 0 : src_i$
CV_THRESH_TOZERO	$dst_i = (src_i > T) \, ? \, src_i : 0$

OTSU THRESOLDING

The algorithm returns a single intensity threshold that it separates the pixels into two classes foreground and background

It involves iterating through all possible threshold values and calculating a measure of spreads for the pixel levels each side of the threshold.

The aim is to find out the threshold value where the sum of foreground and background spreads as minimum.

(In simple thresholding, we used an arbitrary chosen value as a threshold. In contrast, Otsu's method avoids having to choose a value and determines it automatically.)

Adaptive Thresholding

Thresholding is the simplest way to segment objects from a background. If that background is relatively uniform, then we can use a global threshold value to binarize the image by pixel-intensity. If there's large variation in the background intensity, however, adaptive thresholding also known as local or dynamic thresholding may produce better results.

Difference Between Simple and Adaptive thresholding

Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions.

Difference between simple thresholding and adaptive thresholding

In **simple thresholding**, the threshold value is global, i.e., it is same for all the pixels in the image. **Adaptive thresholding** is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions.

In OpenCV, we can perform Adaptive thresold operation on an image using the method **adaptiveThreshold()** of the **Imgproc** class. Following is the syntax of this method

adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)

This method accepts the following parameters –
src − An object of the class **Mat** representing the source (input) image.

dst − An object of the class **Mat** representing the destination (output) image.

maxValue − A variable of double type representing the value that is to be given if pixel value is more than the threshold value.

OPENCV FUNCTIONS USED

Opencv function used for adaptive thresholding are given by two functions that is..

Adaptive_thresh_mean_c

The threshold value is the mean of neighborhood area.

Adaptive_thresh_gaussian_c

The threshold value is the weighted sum of neighborhood values where weights area gaussian window

thresholdType − A variable of integer type representing

the type of threshold to be used.

blockSize − A variable of the integer type representing size of the pixel neighborhood used to calculate the threshold value.

C − A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean)

CODE TO SHOW ADAPTIVE THRESHOLDING

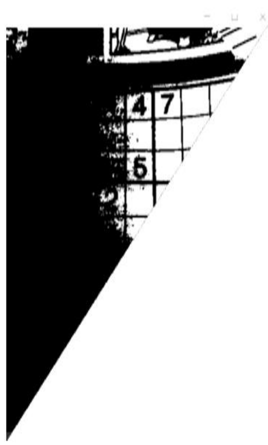
adapting_thresolding.py

```
import cv2 as cv
import numpy as np

img = cv.imread('sudoku.png',0)

_, th1=cv.threshold(img,127,255,cv.THRESH_BINARY)



th2 =cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,cv.THRESH_BINARY,11,2);
th3 =cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,cv.THRESH_BINARY,11,2);


cv.imshow("Image",img)
cv.imshow("th1",th1)
cv.imshow("th2",th2)
cv.imshow("th3",th3)

cv.waitKey(0)
cv.destroyAllWindows()
```

OUTPUT OF OUR CODE

Original image

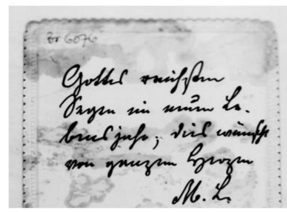
Simple thresholding

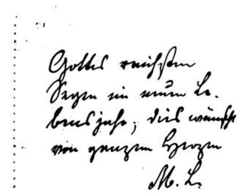


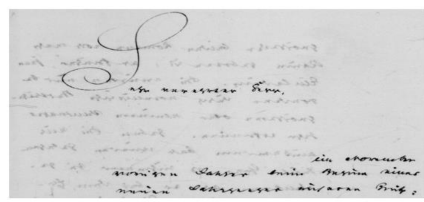
Adaptive thresholding

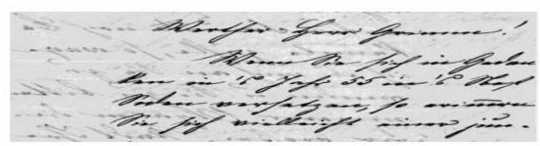
Gaussian thresholding

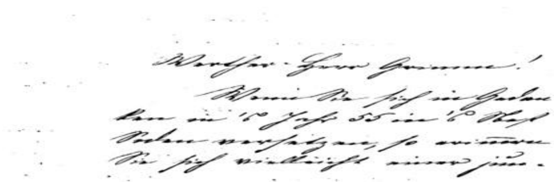
RESULTS OF SIMPLE THRESHOLDING

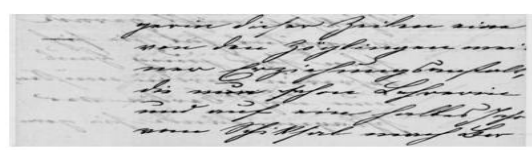


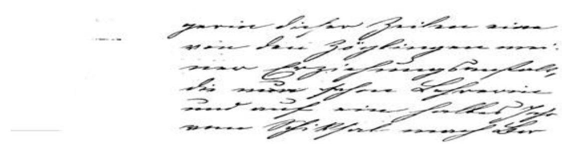


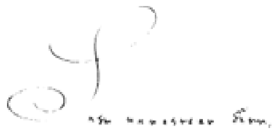


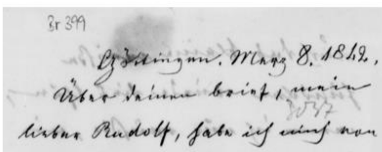


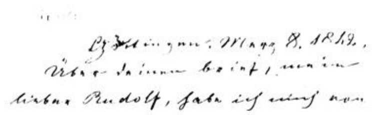


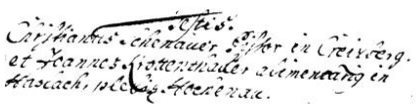


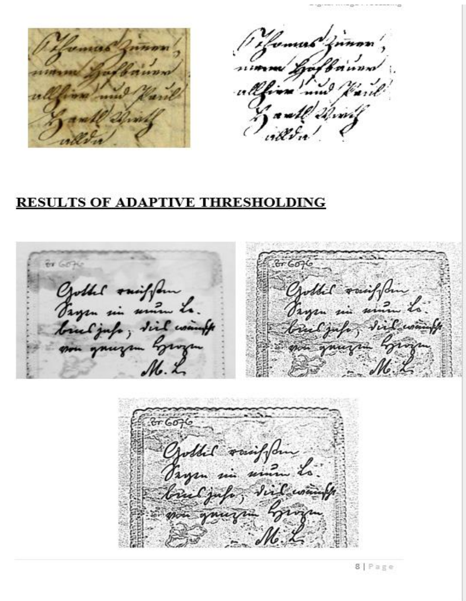






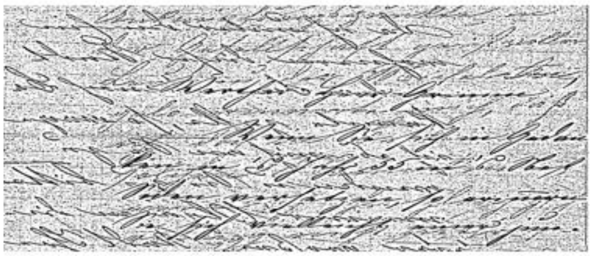


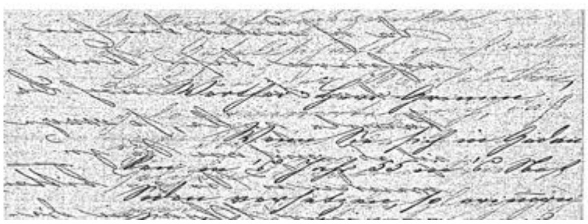


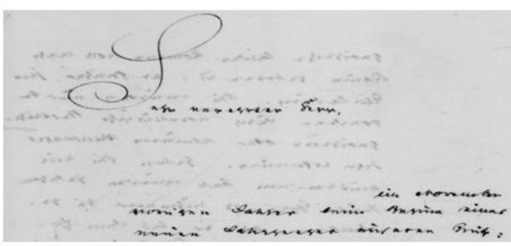


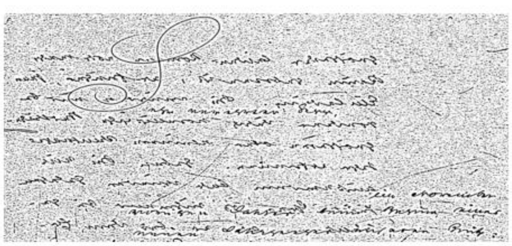
RESULTS OF ADAPTIVE THRESHOLDING

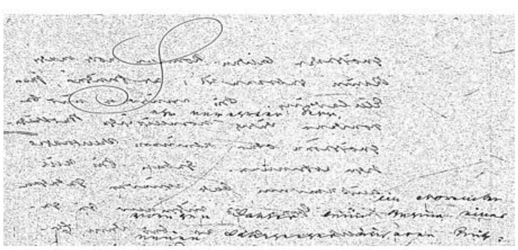
8 | Page



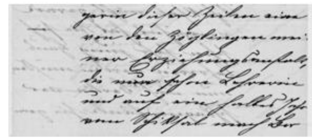


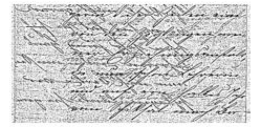


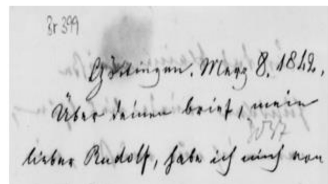


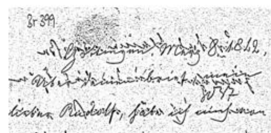


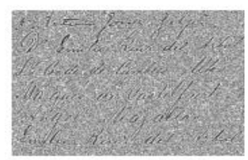
Digital Image Processing

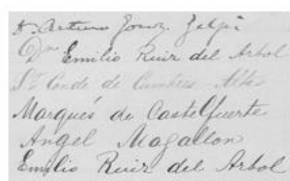


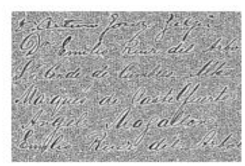


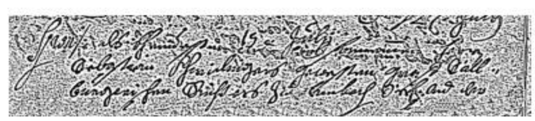


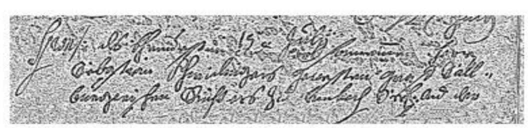




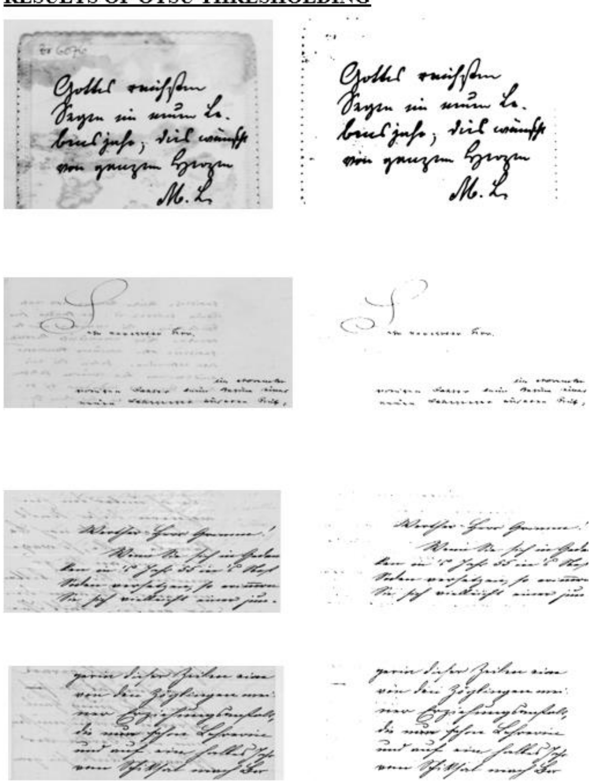


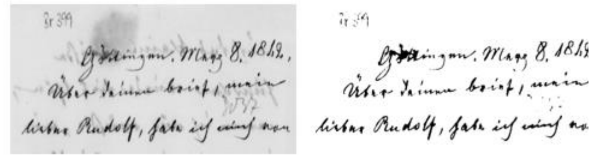


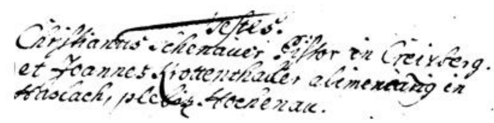




RESULTS OF OTSU THRESHOLDING







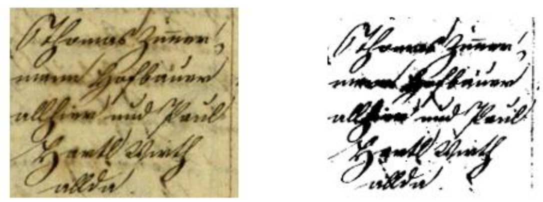


IMAGE FILTERING

Image filtering is the process of modifying **an image by changing the shades or color of the pixel**. It is also used to increase brightness and contrast.

Types are as follows

Image averaging is a digital image processing technique that is often employed to enhance video images that have been corrupted by random noise. The algorithm operates by computing an average or arithmetic mean of the intensity values for each pixel position in a set of captured images from the same scene or view field.

Median filter is also known as salt paper filter

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. It is particularly effective at removing ‘salt and pepper’ type noise.

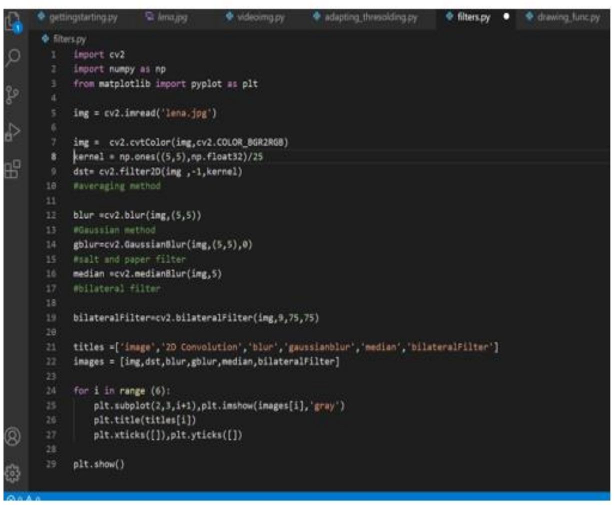
Gaussian filter

Gaussian filtering is more effective at smoothing images. It has its basis in the human visual perception system. It has been found that neurons create a similar filter when processing visual images.

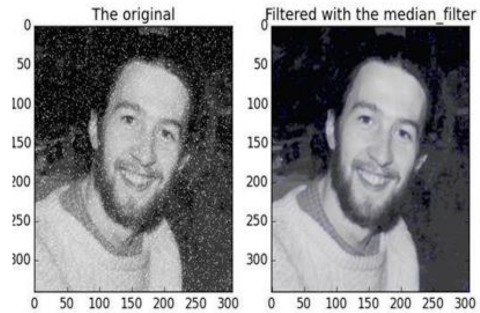
Bilateral filter

A bilateral filter is a **non-linear, edge-preserving, and noise-reducing smoothing filter for images**. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.

CODE TO SHOW FILTERING




```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('lena.jpg')

img =  cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
kernel = np.ones((5,5),np.float32)/25
dst= cv2.filter2D(img ,-1,kernel)
#averaging method

blur =cv2.blur(img,(5,5))
#Gaussian method
gblur=cv2.GaussianBlur(img,(5,5),0)
#salt and paper filter
median =cv2.medianBlur(img,5)
#bilateral filter

bilateralFilter=cv2.bilateralFilter(img,9,75,75)

titles =['image','2D Convolution','blur','gaussianblur','median','bilateralFilter']
images = [img,dst,blur,gblur,median,bilateralFilter]

for i in range (6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```



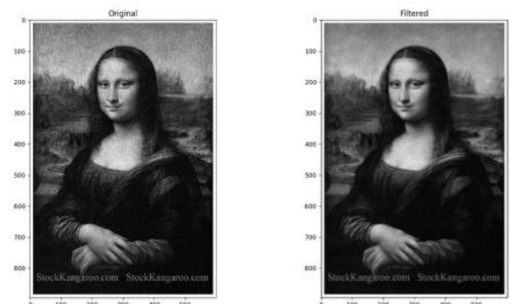
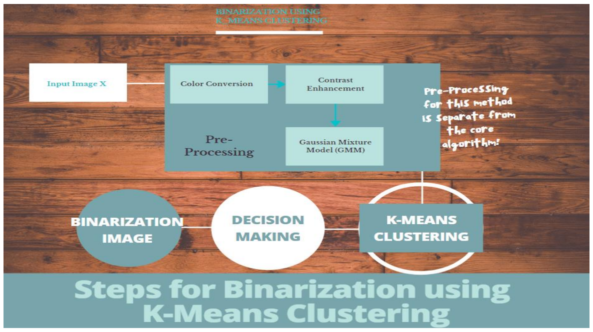


Image Binarization

Binarization is the process of converting a pixel image to a binary image. Image binarization is the process of **taking a grayscale image and converting it to black-and-white**, essentially reducing the information contained within the image from 256 shades of gray to 2: black and white, a binary image

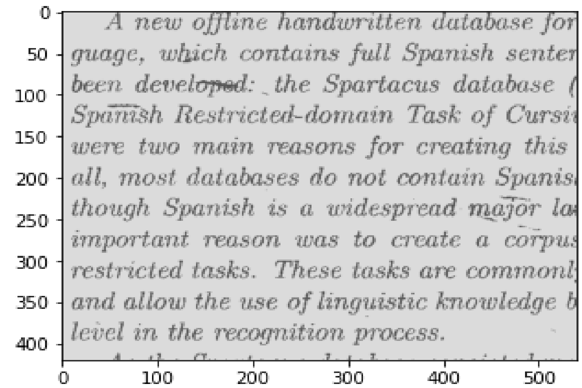
K- Means Clustering

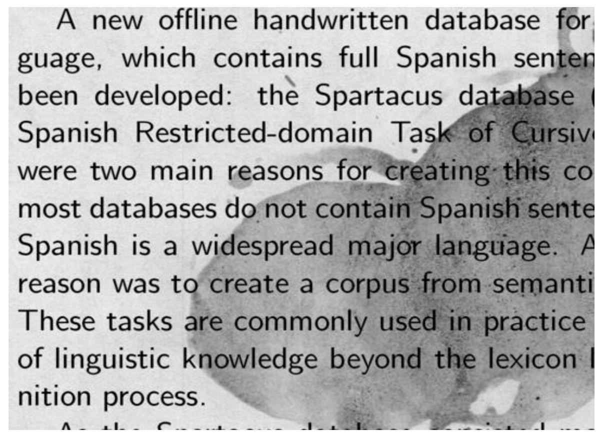
We will consider the image binarization problem as a binary classification problem where each of the pixels in the image have to be classified as one of those 2 classes only. The classes are the “Background” & the “Foreground”. This type of classification problem can be tackled by seeing these 2 classes as clusters in an image where the cluster will be classified as one of the two classes.



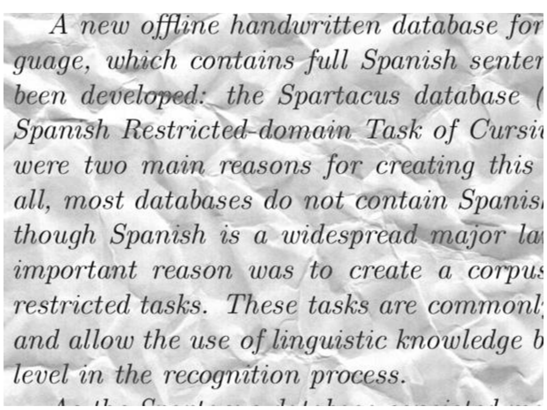
Colour Conversion: This step is needed to convert the color space to RGB if you are using any other color space and we also normalize the pixel values so, that all the pixel values are in the range 0-1.

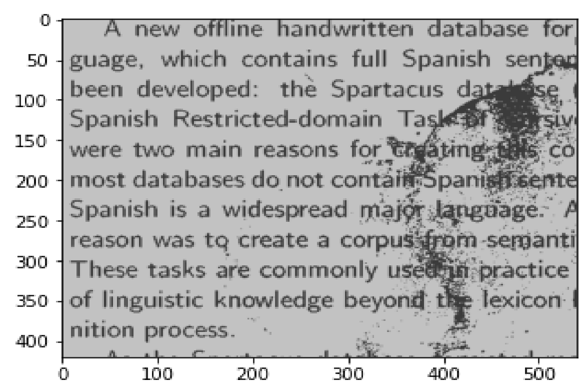
Contrast Enhancement: This step is needed to overcome the problems like uneven illumination and gradients in the image. We use modified histogram equalization for contrast enhancement for this step. This minimizes the problems of over enhancement, saturation artifacts and change in mean brightness.

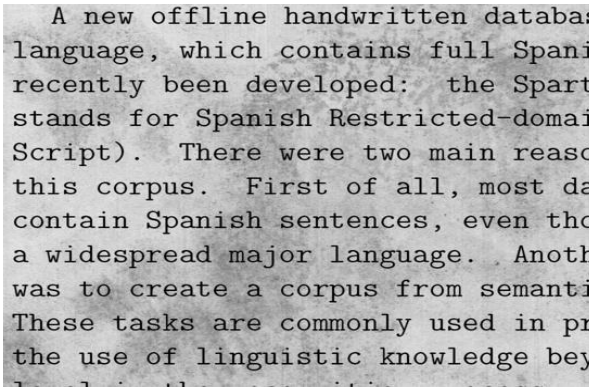

A new offline handwritten database for
guage, which contains full Spanish senter
been developed: the Spartacus database (
Spanish Restricted-domain Task of Cursi
were two main reasons for creating this
all, most databases do not contain Spanis
though Spanish is a widespread major la
important reason was to create a corpus
restricted tasks. These tasks are commonl
and allow the use of linguistic knowledge b
level in the recognition process.


A new offline handwritten database for
guage, which contains full Spanish senten
been developed: the Spartacus database (
Spanish Restricted-domain Task of Cursiv
were two main reasons for creating this co
most databases do not contain Spanish sente
Spanish is a widespread major language. A
reason was to create a corpus from semanti
These tasks are commonly used in practice
of linguistic knowledge beyond the lexicon l
nition process.

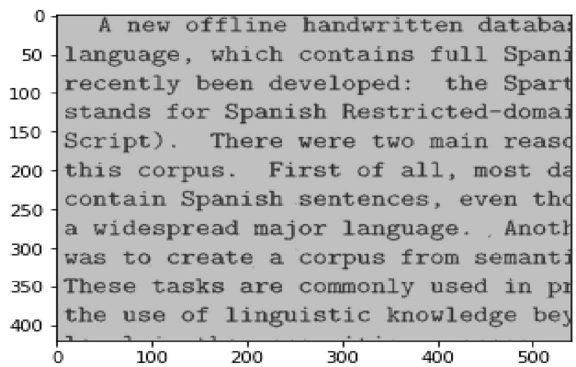
Results


A new offline handwritten database for
guage, which contains full Spanish senter
been developed: the Spartacus database (
Spanish Restricted-domain Task of Cursi
were two main reasons for creating this
all, most databases do not contain Spanis
though Spanish is a widespread major la
important reason was to create a corpus
restricted tasks. These tasks are commonl
and allow the use of linguistic knowledge b
level in the recognition process.


A new offline handwritten database for
guage, which contains full Spanish senten
been developed: the Spartacus database (
Spanish Restricted-domain Task of Cursiv
were two main reasons for creating this co
most databases do not contain Spanish sente
Spanish is a widespread major language. A
reason was to create a corpus from semanti
These tasks are commonly used in practice
of linguistic knowledge beyond the lexicon l
nition process.



A new offline handwritten databas
language, which contains full Spani
recently been developed: the Spart
stands for Spanish Restricted-domai
Script). There were two main reasc
this corpus. First of all, most da
contain Spanish sentences, even thc
a widespread major language. Anoth
was to create a corpus from semanti
These tasks are commonly used in pr
the use of linguistic knowledge bey



A new offline handwritten databas
language, which contains full Spani
recently been developed: the Spart
stands for Spanish Restricted-domai
Script). There were two main reasc
this corpus. First of all, most da
contain Spanish sentences, even thc
a widespread major language. Anoth
was to create a corpus from semanti
These tasks are commonly used in pr
the use of linguistic knowledge bey

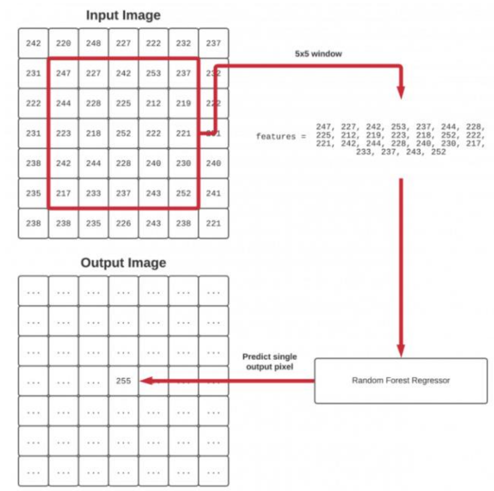
Denoising Images using ML Technique

The objective of Document Image Analysis is to recognize the text & graphics components in image of documents & to extract intended information from them. OCR (optical character recognition) is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as a scanned paper document. The basic process of OCR involves examining the text of a document and translating the characters into code that can be used for data processing. OCR is sometimes also referred to as text recognition. When working with documents generated by a computer, screenshots, or essentially any piece of text that has never touched a printer and then scanned, OCR becomes far easier. That all changes once a piece of text is printed and scanned. From there, OCR becomes much more challenging. To overcome this issue, we need to denoise images. We are going to implement denoising of images using Machine Learning. That will increase the quality of document images and can be helpful for text recognition and for many applications.

ALGORITHM

Our denoising algorithm hinges on training an RFR to accept a noisy image and automatically predict the output pixel values. These algorithms work by applying a 5 x 5 window that slides from left-to-right and top-to-bottom, one pixel at a time across both the noisy image (i.e., the image we want to pre-process automatically and clean-up) and the target output image. At each sliding window stop, we extract the 5 x 5region of the noisy input image. We then flatten the 5 x 5 region into a 25-d list and treat it like a feature vector. The same 5 x 5 region of the cleaned image, but this time we only take the center (x, y)- coordinate, denoted by the location (2, 2). Given the 25-d (dimensional) feature vector from the noisy input image, this single pixel value is what we want our RFR to predict.

Figure: Example feature vector



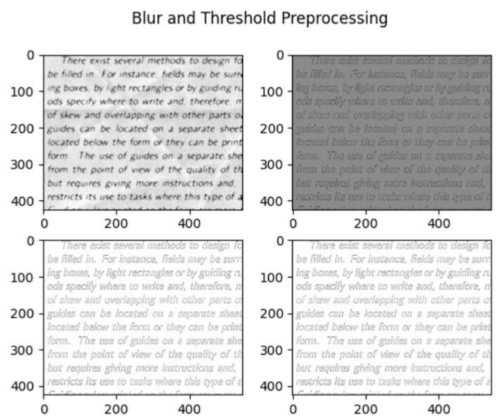
Given our trainX variable (our raw pixel intensities), we want to predict the corresponding cleaned/denoised pixel value in trainY. We will train our RFR in this manner, ultimately leading to a model that can accept a noisy document input and automatically denoise it by examining local 5 x 5regions and then predicting the centre(cleaned) pixel value.

Ex: trainX = [[247 227 242 253 237 244 228 225 212 219 223 218 252 222 221 242 244 228 240 230 217 233 237 243 252]] trainY = [[1]]

Helper function for Pre-processing

To help our RFR predict background (i.e., noisy) from foreground (i.e., text) pixels, we need to define a helper function that will pre-process our images before we train the model and make predictions with it. First, we take our input image, blur it (top-left), and then subtract the blurred image from the input image (top right). We do this step to approximate the foreground of the image since, by nature, blurring will blur focused features and reveal more of the "structural" components of the image.

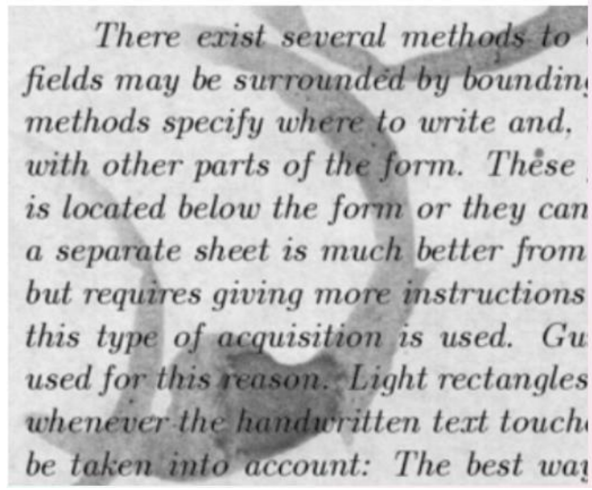
Figure: Pre-processing

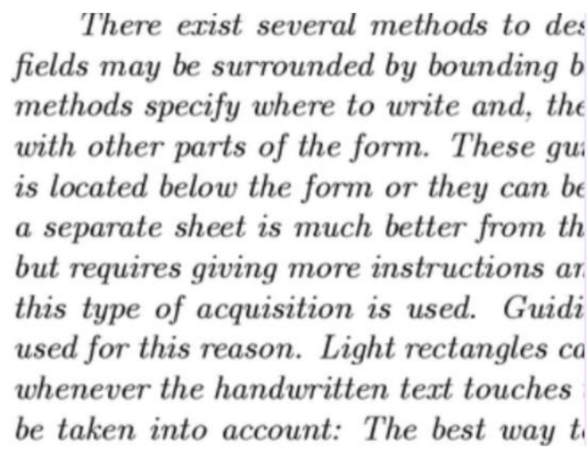


Creating Training Data Set

In this step, we'll create our 5 x5 - 2d feature vectors from the noisy image and then extracting the target (i.e., cleaned) pixel value from the corresponding gold standard image. We'll save these features to disk in CSV format and then train a Random Forest Regression model on them. We start sliding a 5 x 5 window from left-to-right and top-to-bottom across the trainImage and cleanImage. At each sliding window stop, we extract the 5 x 5 ROI of the training image and clean image. We grab the width and height of the trainROI, and if either the width or height is not five pixels (due to us being on the borders of the image), we throw out the ROI (because we are only concerned with 5 x 5 regions). Next, we construct our feature vectors and save the row to our CSV file.

Figure: Sample data image


There exist several methods to
fields may be surrounded by boundin
methods specify where to write and,
with other parts of the form. These
is located below the form or they can
a separate sheet is much better from
but requires giving more instructions
this type of acquisition is used. Gu
used for this reason. Light rectangles
whenever the handwritten text touch
be taken into account: The best wa


There exist several methods to des
fields may be surrounded by bounding b
methods specify where to write and, the
with other parts of the form. These gu
is located below the form or they can be
a separate sheet is much better from th
but requires giving more instructions ar
this type of acquisition is used. Guidi
used for this reason. Light rectangles ca
whenever the handwritten text touches
be taken into account: The best way t

Training the Model

Now we have our features.csv file, we can move on to the training the model. First, we load our features.csv file then start training the model. This is the crucial part in the process. We imported RandomForestRegression class from sklearn package and created an object of it and gave the training data to it. Finally, we have our RFR model that will accept a 5 x 5 region of a noisy image and then predict the cleaned center pixel value. After training is complete, we compute the root-mean-square error (RMSE) to measure how good a job we've done at predicting cleaned, denoised images. The lower the error value, the better the job we've done.

Figure: Test results