Computer Vision

T5 Bootcamp by SDAIA

Agenda

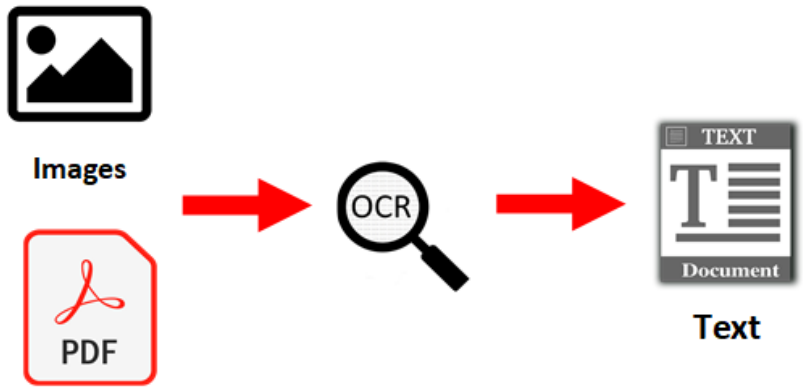
Optical Character Recognition (OCR)

Optical Character Recognition (OCR)

Optical Character Recognition (OCR)

Optical character recognition (OCR) is sometimes referred to as Text Recognition. An OCR program extracts and repurposes data from scanned documents, camera images and image-only pdfs.

- OCR software extracts out letters on the image, puts them into words and then puts the words into sentences, thus enabling access to and editing of the original content. It also eliminates the need for manual data entry.
- OCR systems converts physical, printed or scanned documents into machine-readable text.



Evolution of OCR

The history of optical character recognition.

1970s-1980s: Inception and Accessibility

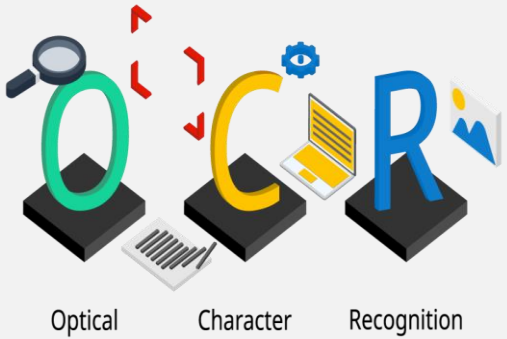
- Kurzweil Computer Products, Inc. pioneers omni-font OCR for text recognition.
- Ray Kurzweil's focus on accessibility leads to the development of a reading machine for the blind.

1980s-1990s: Commercialization and Adoption

- Xerox acquires Kurzweil's company, driving commercialization of OCR technology.
- OCR gains popularity in digitizing historical newspapers, marking significant adoption.

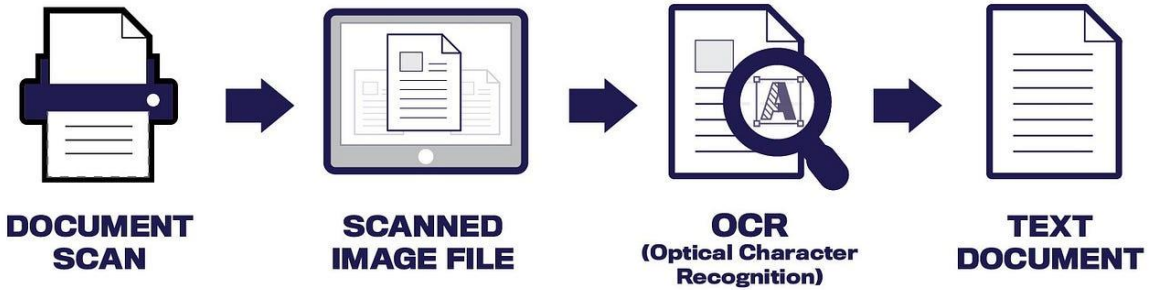
Impact and Applications:

- OCR eliminates manual retyping, transforming document digitization.
- Widely available OCR services, like Google Cloud Vision OCR, empower users to scan and organize documents with ease.



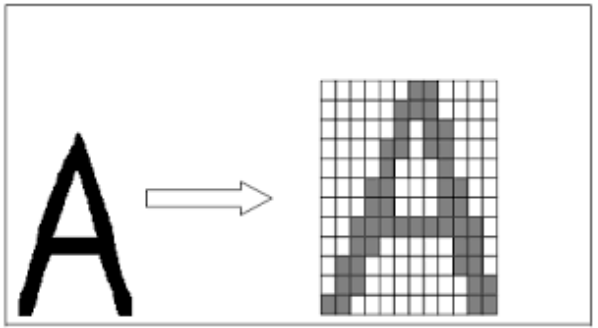
How OCR works?

1. **Scanning the Document**
   - Utilize a scanner to convert the physical document into a digital image.
2. **Image Conversion**
   - OCR software transforms the scanned image into a two-color (black and white) version.
   - Analyzes light and dark areas, distinguishing characters from the background.
3. **Character Recognition**
   - Dark areas are processed to identify alphabetic letters or numeric digits.
   - OCR programs employ pattern recognition or feature detection algorithms.



How OCR works? *(cont'd)*

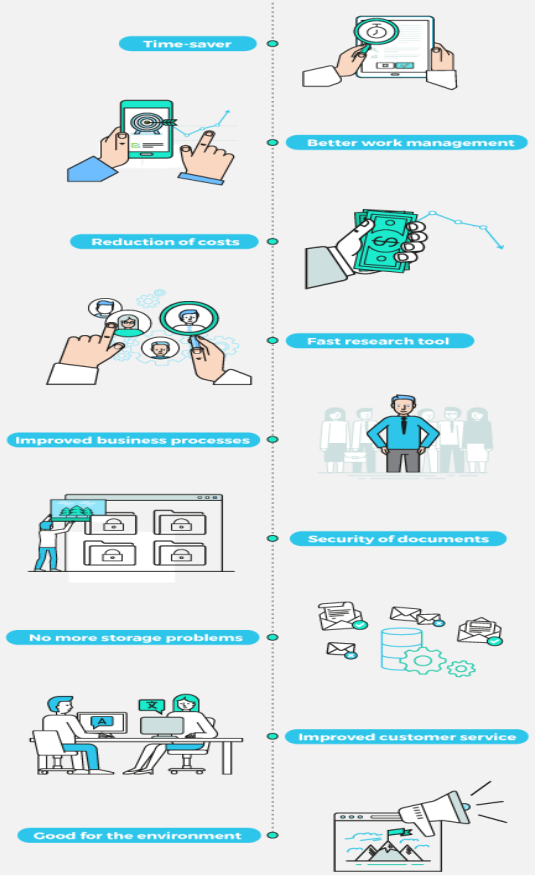
4. **Pattern Recognition**
    - Compares characters in the scanned document with examples of text in various fonts and formats.
5. **Feature Detection**
    - Applies rules regarding specific features of characters to recognize them in the scanned document.
    - Features may include the number of lines, curves, or intersections in a character.
6. **Conversion and Correction**
    - Identified characters are converted into ASCII codes for computer processing.
    - Users may correct errors and proofread the document before saving for future use.



Benefits of OCR

The main advantages of OCR technology are the following:

- Saves time
- Decreases errors
- Minimizes effort
- Enables actions that are not possible with physical copies, such as compressing into ZIP files, highlighting keywords, incorporating into a website and attaching to an email.



Limitations of OCR

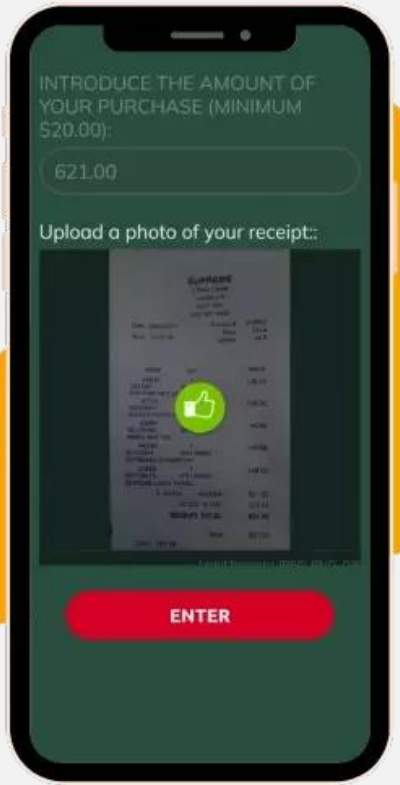
Despite all the benefits, there are limitations to OCR:

1. **Requires Manual Checks for Accuracy**

   - Accuracy is both a benefit and one of the limitations of using OCR. While OCR technology offers better accuracy than entering invoices manually, it still needs to go through a validation process, where you can detect and correct errors.

2. **Format Constraints**

   - OCR technology performs well when reading and extracting data from standard documents, it may not be able to extract data from small or handwritten text, blurred or unclear text, illegible fonts, individual line items, or data from tables.



Top Open-Source OCR Libraries in Python

Since OCR is a popular ongoing problem, many open-source libraries try to solve it. Below, we will cover the ones that gained the most popularity due to their high performance and accuracy..

Package Name	Advantage	Disadvantages
Tesseract (pytesseract)	Mature, widely used, extensive support	Slower, lower accuracy on complex layouts
EasyOCR	Simple to use, multiple models	Lower accuracy, limited customization
Keras-OCR	Higher accuracy, customizable	Requires GPU, steeper learning curve

Tesseract

Tesseract OCR is an open-source optical character recognition engine that is the most popular among developers. Tesseract can take images of text and convert them into editable machine-readable text.

- Tesseract runs on Windows, macOS and Linux platforms. It supports Unicode (UTF-8) and more than 100 languages.

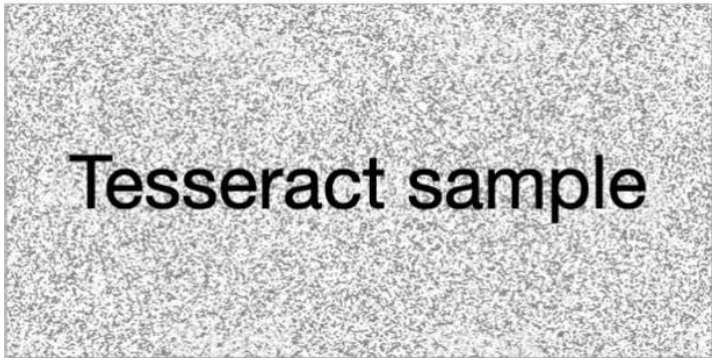
Installing Tesseract

```
# getting pytesseract installed
! pip install pytesseract
! pip install tesseract
! pip install tesseract-ocr

# import the Tesseract & dependencies
from PIL import Image
import pytesseract
import numpy as np
```

Python OCR Implementation

Tesseract are good enough for simple images. However, in the real world it is difficult to find images that are really simple, Tesseract could also read images with noise as below.

```
#  load this image and convert it to text
filename = 'ocr_image.png'
img1 = np.array(Image.open(filename))
text = pytesseract.image_to_string(img1)

# See the results
print(text)
Tesseract Sample
```



Text Localization and Detection

Tesseract can also do text localization and detection from images. First, load image and extract the data But, we will not immediately change the image into a string as previous example. In this example, we'll convert the image into a dictionary.

```
filename = 'image_01.png'
image = cv2.imread(filename)

results = pytesseract.image_to_data(image,
output_type=Output.DICT)
```

Text Localization and Detection

The following results are the contents of the dictionary. The **left**, **top**, **width** and **height** are used to draw a bounding box around the text along with the text itself.

In addition, we will need a conf key to determine the boundary of the detected text.

```
{
'level': [1, 2, 3, 4, 5, 5, 5],
'page_num': [1, 1, 1, 1, 1, 1, 1],
'block_num': [0, 1, 1, 1, 1, 1, 1],
'par_num': [0, 0, 1, 1, 1, 1, 1],
'line_num': [0, 0, 0, 1, 1, 1, 1],
'word_num': [0, 0, 0, 0, 1, 2, 3],
'left': [0, 26, 26, 26, 26, 110, 216],
'top': [0, 63, 63, 63, 63, 63, 63],
'width': [300, 249, 249, 249, 77, 100, 59],
'height': [150, 25, 25, 25, 25, 19, 19],
'conf': ['-1', '-1', '-1', '-1', 97, 96, 96],
'text': ['', '', '', '', 'Testing', 'Tesseract', 'OCR']}
```

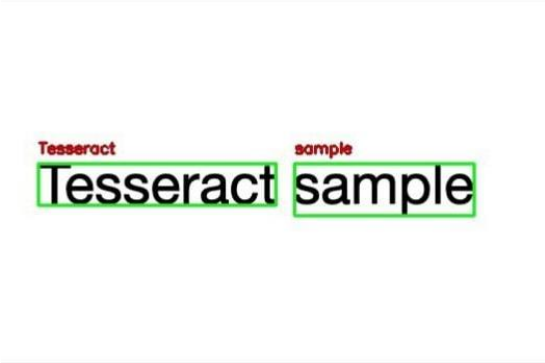


Image Preprocessing techniques for OCR

Proper preprocessing is crucial for enhancing OCR accuracy. Various techniques can optimize image quality before text recognition.

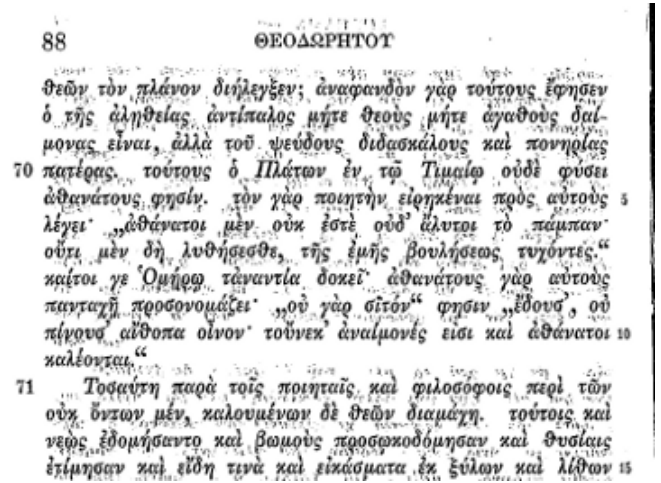
1. Grayscale Conversion:
    - Convert colored images to grayscale to remove color variations that may confuse OCR.
    - Use OpenCV's `cv2.cvtColor` function with `COLOR_BGR2GRAY` conversion.

Image Preprocessing techniques for OCR *(cont'd)*

2. Noise Reduction:

   - Apply denoising filters like median blur to reduce artifacts from scanned or aged documents.

```python
def denoise(image):
    """
    Reduces noise in the image using a median blur filter.

    image: The input grayscale image.

    Returns:The denoised image.
    """
    # Adjust kernel size as needed
    return cv2.medianBlur(image, 5)
```



88 ΘΕΟΔΩΡΗΤΟΥ

θεῶν τὸν πλάνον διήλεγξεν; ἀναφανδὸν γὰρ τούτους ἔφησεν
ὁ τῆς ἀληθείας ἀντίπαλος μήτε θεοὺς μήτε ἀγαθοὺς δαί-
μονας εἶναι, ἀλλὰ τοῦ ψεύδους διδασκάλους καὶ πονηρίας
70 πατέρας. τούτους ὁ Πλάτων ἐν τῷ Τιμαίῳ οὐδὲ φύσει
ἀθανάτους φησίν. τὸν γὰρ ποιητὴν εἰρηκέναι πρὸς αὐτοὺς 5
λέγει· „ἀθάνατοι μὲν οὐκ ἐστὲ οὐδ' ἄλυτοι τὸ παμπαν·
οὔτι μὲν δὴ λυθήσεσθε, τῆς ἐμῆς βουλήσεως τυχόντες.“
καίτοι γε Ὁμήρῳ τἀναντία δοκεῖ· ἀθανάτους γὰρ αὐτοὺς
πανταχῇ προσονομάζει· „οὐ γὰρ σῖτον“ φησιν „ἔδουσ', οὐ
πίνουσ' αἶθοπα οἶνον· τοὔνεκ' ἀναίμονές εἰσι καὶ ἀθάνατοι 10
καλέονται.“

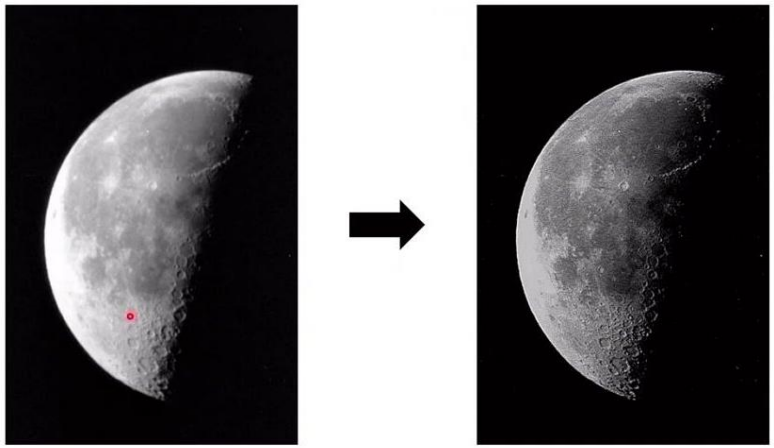
71 Τοσαύτη παρὰ τοῖς ποιηταῖς καὶ φιλοσόφοις περὶ τῶν
οὐκ ὄντων μέν, καλουμένων δὲ θεῶν διαμάχη. τούτοις καὶ
νεὼς ἐδομήσαντο καὶ βωμοὺς προσῳκοδόμησαν καὶ θυσίαις
ἐτίμησαν καὶ εἴδη τινὰ καὶ εἰκάσματα ἐκ ξύλων καὶ λίθων 15

Image Preprocessing techniques for OCR *(cont'd)*

3. Sharpening:

   - Enhance edges and improve character recognition, especially for blurry or low-resolution images.

```python
def sharpen(image):
    """
    Sharpens the image using a Laplacian filter.

    image: The input grayscale image.

    Returns: The sharpened image
    """
    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
    return cv2.filter2D(image, -1, kernel)
```



Original Image

Sharpen Image using

Image Preprocessing techniques for OCR *(cont'd)*

4. Binarization:

   - Convert the image to black and white for better text separation from the background..

```python
def binarize(image):
    """
    Binarizes the image using adaptive thresholding.

    image: The input grayscale image.

    Returns:The binary image.
    """
    thresh = cv2.adaptiveThreshold( image, 255,
             cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,
             11, 2)
    return thresh
```

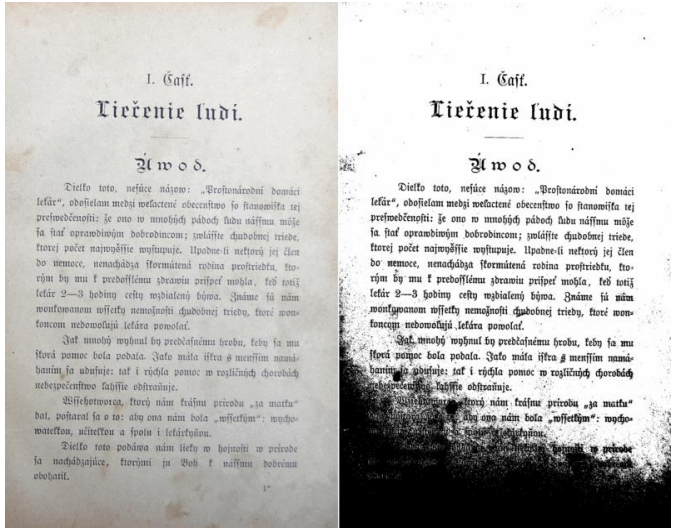


Image Preprocessing techniques for OCR *(cont'd)*

5. Other techniques:

   - **Dilation**: making small and thin fonts bolder to improve recognition
   - **Erosion**: eroding bold text to improve accuracy. Common in historic documents with thick fonts.
   - **Skew Correction**: correcting the tilt (skewness) of text lines. Common in incorrectly scanned documents.

Exercise

Transitioning to Google-Colab for hands-on coding practice.

Notebook:

- TBD

Thank You

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority