1. **Introduce Tessseract OCR**

Tesseract is an open-source OCR engine developed by Google and is widely considered one of the most accurate OCR engines available.

Pytesseract is a useful Python library that provides an interface to the Tesseract OCR engine. It pre-processes the input image first in order to improve its quality. After that, it examines the page’s arrangement/orientation to determine text blocks, paragraphs and characters. By matching patterns in the segmented areas, Tesseract recognizes individual characters through a combination of [machine learning](https://builtin.com/machine-learning) and conventional [image processing](https://builtin.com/software-engineering-perspectives/image-processing-python) approaches. In order to increase accuracy and handle many languages, it uses language models. Following identification, post-processing operations like spell checking and error correction are used to improve the outcomes.

To effectively recognize text, Tesseract, the OCR engine underlying pytesseract, is trained on language-specific data sets. It offers support for several languages and comes with training data sets specific to each language.

1. **How Tesseract Works**

Tesseract works in several steps to extract text from images. First, it performs preprocessing of the image to optimize its quality for text recognition. Then Tesseract OCR segments the image into text blocks, lines and words and analyzes the structure of the text.

In the preprocessing phase comes [Leptonica](http://leptonica.org/) into play, also a[Open source library.](https://github.com/DanBloomberg/leptonica)It is responsible for image processing and manipulation. Leptonica optimizes the images by reducing noise, normalizing colors, and adjusting scaling to increase the effectiveness of Tesseract's machine learning. Apart from that, however, Leptonica does not take care of text recognition and extraction, which is the exclusive responsibility of Tesseract.

In the recognition phase, the tool uses machine learning to identify the characters in the images. Since version 4.0, the software has focused on Long Short-Term Memory (LSTM) networks to further improve recognition accuracy. Finally, the tool performs post-processing to correct incorrectly recognized characters and generate the final text result.

**Advantages**

The advantages of OCR are numerous, but namely:

* it increases the efficiency and effectiveness of office work
* The ability to instantly search through content is immensely useful, especially in an office setting that has to deal with high volume scanning or high document inflow.
* OCR is quick ensuring the document’s content remains intact while saving time as well.
* Workflow is increased since employees no longer have to waste time on manual labour and can work quicker and more efficiently.

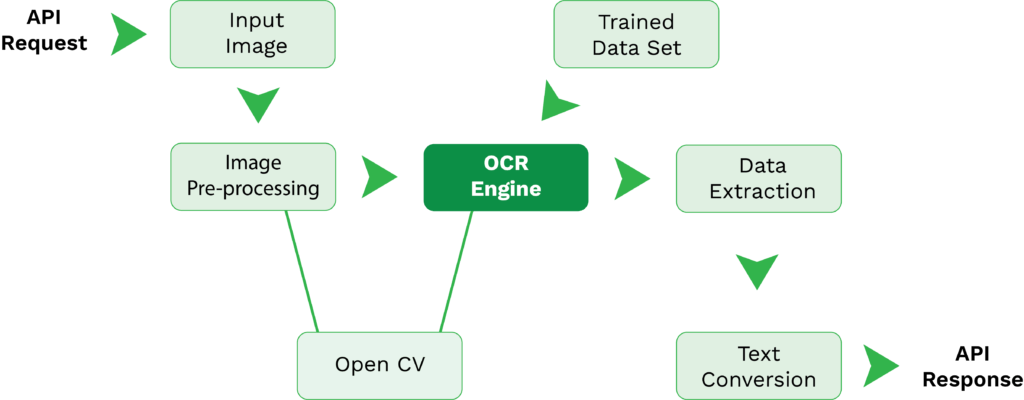
**Disadvantages**

* The OCR is limited to language recognition.
* There is lot of effort that is required to make trainer data of different languages and implement that.
* One also need to do extra work on image processing as it is the most essential part that really matters when it comes to the performance of OCR.
* After doing such a great amount of work, no OCR can offer an accuracy of 100% and even after OCR we have to determine the unrecognized character by neighbouring methods of machine learning or manually correct it.

### **Steps in the Tesseract OCR process**

To help you understand what the Tesseract OCR process normally looks like, we have broken it down into the following steps:

1. **API request** – Tesseract OCR can only be accessed via API integration. Once that connection between your solution and Tesseract is established, you can send API requests from your solution to the Tesseract OCR engine.
2. **Input Image**– With an API request, you can send in your input image for text extraction.
3. **Image pre-processing** – Before [data extraction](https://www.klippa.com/en/blog/information/data-extraction/), the image pre-processing features of the Tesseract OCR engine kick in. This step exists to ensure that the image quality is as high as possible to achieve accurate data extraction results. Often OpenCV is coupled with Tesseract to increase the image quality before data extraction.
4. **Data Extraction**– Together with trained data sets and Leptonica or OpenCV, the Tesseract OCR engine processes the input image and extracts the data.
5. **Text conversion**– As the data (text) has been extracted from the input, it can now be converted into a desired format that Tesseract supports, including PDF, plain text, HTML, TSV, and XML.
6. **API response back**– Once the output is ready, your solution will get an API response back with the finalized output.



To have this OCR flow established, it will require knowledge and time to build all the relevant API connections. Furthermore, you would need to find the relevant components, such as libraries and wrappers, and do extensive coding. This depends mostly on your use case and application of OCR.

As stated earlier, Tesseract is often paired with OpenCV to enhance the input image quality to today’s standards. Let’s have a more detailed look at how this works.

## How can Tesseract OCR be optimized for accurate text recognition from low-quality images?

Tesseract OCR (Optical Character Recognition) is an open-source software tool primarily used for extracting text from images. Developed by Google, it is one of the most accurate and widely used OCR engines available. Tesseract is capable of recognizing various languages and fonts, making it a versatile solution for text recognition tasks.

Here’s a detailed breakdown of Tesseract OCR:

1. **Functionality:** Tesseract OCR works by analyzing the patterns of light and dark pixels in an image to identify characters and words. It processes images in several stages, including preprocessing, text detection, character recognition, and post-processing.
2. **Preprocessing:** Before text recognition can occur, Tesseract preprocesses the image to enhance its quality and improve OCR accuracy. This may involve tasks such as noise reduction, binarization (converting the image to black and white), skew correction (straightening tilted text), and de-speckling.
3. **Text Detection:** Tesseract identifies regions of text within the image using techniques like connected component analysis and contour detection. It locates areas where text is present and extracts them for further processing.
4. **Character Recognition:** Once the text regions are identified, Tesseract segments the image into individual characters and attempts to recognize each one. It utilizes machine learning algorithms, including deep neural networks, to match patterns and classify characters accurately.
5. **Language Support:** Tesseract supports a wide range of languages, from commonly used ones like English, Spanish, and Chinese to less widely spoken languages. It also offers the capability to recognize multiple languages within the same document.
6. **Output:** After processing the image, Tesseract outputs the recognized text in a plain text format, which can be further processed or used for various applications. Additionally, it may provide information about the confidence level of each recognized character or word.
7. **Accuracy:** Tesseract is known for its high accuracy, especially when used with properly preprocessed images and trained language data. However, the accuracy can vary depending on factors such as image quality, font type, and language complexity.

### **Better Image Processing by combining OpenCV & Tesseract**

To understand why OpenCV is often combined with Tesseract OCR, we must explain computer vision. Computer vision is a subfield of Artificial Intelligence (AI) that enables computers and software to see, observe, and make sense of digital images, videos, or other visual inputs. But what does it have to do with OpenCV?

OpenCV is an open-source library of computer vision functions that can enhance data extraction of OCR engines such as Tesseract. To do so, you could use the OpenCV library to integrate the following functions into the OCR solution:

* **Object detection** – enables the solution to detect a variety of objects
* **Deep neural networks (DNN)** – enables the solution to classify images
* [**Image processing**](https://www.klippa.com/en/blog/information/ai-image-processing/) – enables the solution to process input images better with various techniques such as edge detection, pixel manipulation, de-skewing, etc.

Without OpenCV, Tesseract is not as sophisticated as we would expect from today’s OCR solutions, as many of them apply various AI technologies.

1. **Areas of application of Tesseract**

Tesseract OCR finds application in various fields where the conversion of scanned documents, images or [PDFs](https://konfuzio.com/en/pdf-in-google-doc-convert-alternative-to-adobe-and-word/) into editable text is required. Some of the most common use cases are:

* Automation of data entry and [Text extraction](https://konfuzio.com/en/pdf-extraction/)
* [Digitization](https://konfuzio.com/en/artificial-intelligence-in-health-insurance/) of books and archive material
* Recognition of text on business cards and forms
* Automatic recognition of text in images
* Recognition of license plates and traffic signs
* Text recognition with Tesseract on mobile devices and web services

Tesseract is a versatile and powerful OCR engine that can be used by both developers and end users. It provides a solid base for OCR projects and can be customized to meet specific requirements.

## ****Limitations of Tesseract****

Tesseract OCR, initially developed by HP in the 1980s, is considered to be one of the best open-source optical character recognition (OCR) programs out there. However, it does have its limitations. Text recognition accuracy for languages with non-Roman scripts (such as Arabic or Chinese) is still lacking and it's difficult to recognize rotated or skewed text accurately--sometimes even when using an image preprocessor like ImageMagick to rotate the original image. Tesseract isn't great with complex formatting either, so PDFs can be tricky to get right. Despite these challenges, Tesseract has many powerful features that make it a great first stop every time you need a reliable open-source OCR solution.

Some of the limitations of Tesseract OCR include:

1. Tesseract is not very good at handling low-quality images or documents with low resolution. It may have difficulty accurately extracting text from images that are blurry, pixelated, or have poor lighting.  
   ‍
2. Tesseract can struggle with certain types of fonts, such as handwriting or stylized fonts. It may have difficulty accurately recognizing characters in these fonts, particularly if the handwriting is messy or the font is very different from the standard fonts that Tesseract was trained on.  
   ‍
3. Tesseract may have difficulty accurately extracting text from images or documents that contain a lot of noise or distractions, such as watermarks, logos, or other graphics.  
   ‍
4. Tesseract is not designed to handle images or documents that contain multiple languages or scripts. It can only recognize one language at a time, so it may have difficulty accurately extracting text from documents that contain multiple languages or scripts.  
   ‍
5. Tesseract may have difficulty accurately extracting text from documents that contain a large amount of text or that have complex layouts. It may struggle to correctly identify the structure of the document and may produce results that are difficult to parse.

Despite these limitations, Tesseract is still a very useful tool for OCR and can produce accurate results in many cases. To improve the accuracy of Tesseract OCR, it is important to use high-quality images or documents with good resolution and to carefully pre-process the images to remove any noise or distractions.