



**FINAL REPORT**

**INTELLIGENCE SYSTEM APPLICATION DEVELOPMENT**

**DEVELOPING IMAGE-TO-TEXT-TO-SPEECH PROGRAM USING OPTICAL  
CHARACTER RECOGNITION (OCR) AND NATURAL LANGUAGE PROCESSING  
(NLP)**

**Submitted By:**

**Jeremy Gerald Prawira (001202000003)**

**Angelina Yulisa Waney (001202000076)**

**Mulya Fajar Ningsih Alwi (001202000101)**

**INFORMATION TECHNOLOGY STUDY PROGRAM**

**FACULTY OF COMPUTING**

**PRESIDENT UNIVERSITY**

**2022**

## ABSTRACT

This project or program was conducted with the aim of helping people who cannot read or even blind people to find out the contents of a text containing information but in the form of images by hearing the text audio, with the help of image-to-text-to-speech. In addition, our project can also help other people to translate an image containing text into various languages and can also listen to the text audio.

This program was developed using the python programming language with the help of additional python libraries such as OpenCV, EasyOCR, GoogleTrans, gTTS, PIL, Matplotlib, and several methods. To use this program, the user must input an image file that contains text in it, it could be in any image file format (JPG, JPEG, PNG, etc.), then the program will read and convert the image-to-text form by using OCR (Optical Character Recognition) and after that, the image that has become text form will be converted to audio (text-to-speech) by using NLP (Natural Language Processing), so that the user can listen to the text that contained in the inputted image and the user can also translate the text into various other languages and listen to the text audio in various languages along with the accents.

## ABSTRAK

Proyek yang kami kembangkan ini mempunyai tujuan untuk membantu individu-individu yang mempunyai kekurangan dalam kemampuan untuk membaca atau bahkan individu yang buta untuk mengetahui isi dari sebuah teks yang berisikan informasi. Informasi ini awalnya terbentuk dalam bentuk foto, yang kemudian dikonversi menjadi sebuah audio teks dengan bantuan teknologi *image-to-speech*. Sebagai tambahan, lewat proyek ini kami juga ingin membantu individu yang ingin menerjemahkan sebuah foto yang mempunyai isi teks ke berbagai jenis bahasa dan nantinya akan dapat didengarkan dalam bentuk audio teks.

Program yang kami kembangkan ini menggunakan bahasa pemrograman *python* dengan bantuan berbagai *library* seperti *OpenCV*, *EasyOCR*, *GoogleTrans*, *gTTS*, *PIL*, *Matplotlib*, dan berbagai metode lainnya. Untuk menggunakan program ini, pengguna diminta memasukkan sebuah file foto yang mengandung teks di dalam file foto tersebut. Pengguna dapat menggunakan atau memasukkan foto dengan berbagai format file foto (JPG, JPEG, PNG, dll.). Setelah itu,

program akan membaca dan mengkonversi bentuk file *image-to-text* menggunakan *OCR (Optical Character Recognition)*. Setelah itu, foto yang sudah menjadi teks dapat dikonversi menjadi audio, sehingga pengguna dapat mendengar teks audio dari foto yang dimasukkan. Pengguna juga dapat menerjemahkan teks ke berbagai bahasa lain dan mendengarkan teks audio dalam berbagai bahasa beserta aksennya.

## TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>I</b>
<b>TABLE OF CONTENTS .....</b>	<b>III</b>
<b>CHAPTER 1 – INTRODUCTION .....</b>	<b>1</b>
<b>CHAPTER 2 – LITERATURE STUDY .....</b>	<b>2</b>
2.1 IMAGE RECOGNITION .....	2
2.2 OPTICAL CHARACTER RECOGNITION (OCR) .....	2
2.3 TEXT-TO-SPEECH.....	3
2.4 RELATED WORK .....	3
2.4.1 TEXT SCANNER .....	3
2.4.2 GOOGLE TRANSLATE .....	4
2.4.3 CAM FIND .....	4
2.5 COMPARISON OVERVIEW .....	5
<b>CHAPTER 3 – METHODS.....</b>	<b>6</b>
3.1 IMAGE-TO-TEXT .....	6
3.2 TEXT-TO-SPEECH.....	9
<b>CHAPTER 4 – RESULTS.....</b>	<b>11</b>
<b>CHAPTER 5 – CONCLUSION.....</b>	<b>16</b>
<b>REFERENCES.....</b>	<b>17</b>
<b>ATTACHMENTS .....</b>	<b>19</b>

## CHAPTER 1

### INTRODUCTION

We have grown in a situation where technology has been a massive help when it comes to numerous problems that humans face on a daily basis. If people could assume that it only helps at a large scale, then they might be wrong. The current technology even allows humans to actually help even at a smaller situation such as cleaning the house. Humans now can no longer give some borders of limitation to technology as its potential is limitless. In Indonesia itself, people also gain some huge help from technology, including a big help from recognizing text.

Indonesia has been such a danger in the area of reading, which is actually could be called as in literacy. Literacy comes from the base word of literate. Literate itself comes from the word from Latin America which means as *litteratus* which means as learned or lettered. [1] Based on another source, literacy itself, based on UNESCO, is the ability to identify, understand, interpret, create, communicate and compute, using printed and written materials associated with varying context. UNESCO, then added that literacy is a continuation of learning and should be helping to achieve their goals and growing their skills in order to contribute to the community globally. [2]

According to survey held by the Program for International Student Assessment (PISA) that is posted on the Organization for Economic Co-operation and Development (OECD) via The Coordinating Ministry for Human Development and Cultural Affairs (Kemendiknas), Indonesia is unfortunately ranked 62 out of 70 countries on the rate of literacy. We could conclude that Indonesia is on the bottom 10% of its literacy rate. The number says that Indonesians need a big help in terms of literacy critically. [3]

From this, the idea is to make a contribution in developing a technology that will help in the understanding of literacy. The proposed idea would be adopting deep learning, machine learning and also artificial intelligence in order to give the machine the set of sentences as the current main problem to solve, which would later be recognized by the machine. After the machine already recognized all of its sentences, the machine later would speak it out to the individuals.

## CHAPTER 2

### LITERATURE STUDY

#### 2.1 Image Recognition

Image recognition is used to perform many machine-based visual tasks, such as labeling the content of images with meta-tags, performing image content search and guiding autonomous robots, self-driving cars and accident-avoidance systems. [4] A common example of image recognition is optical character recognition (OCR). A scanner can identify the characters in the image to convert the texts in an image to a text file. With the same process, OCR can be applied to recognize the text of a license plate in an image. [5]

#### 2.2 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is an electronic conversion of the typed, handwritten or printed text images into machine-encoded text. With OCR a huge number of paper-based documents, across multiple languages and formats can be digitized into machine-readable text that not only makes storage easier but also makes previously inaccessible data available to anyone at a click. [6] Optical character recognition works by dividing up the image of a text character into sections and distinguishing between empty and non-empty regions. Depending on the font or script used for the letter, the checksum of the resulting matrix is subsequently labeled (initially, by a person) as corresponding to the character in the image. The OCR flow diagram could be illustrated below. [7]

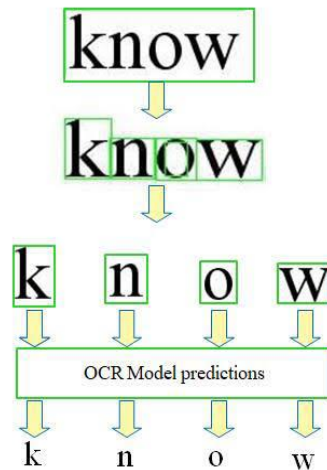


Fig. 1 OCR Flow Diagram

## 2.3 Text-to-Speech

Text-to-Speech (TTS) is a type of assistive technology that reads digital text aloud. It's sometimes called “read aloud” technology. With a click of a button or the touch of a finger, TTS can take words on a computer or other digital device and convert them into audio. The structure of the Text-to-Speech synthesizer can be broken down into major modules:

- **Natural Language Processing (NLP) module:** It produces a phonetic transcription of the text read, together with prosody.
- **Digital Signal Processing (DSP) module:** It transforms the symbolic information it receives from NLP into audible and intelligible speech.

## 2.4 Related Work

### 2.4.1 Text Scanner

Text Scanner is app to recognize any text from an image with 98% to 100% accuracy and works with several languages. This app can scan/extract text from images/photos/pictures by using phone's camera. OCR (Optical Character Recognition) technology is used to recognize text on image.

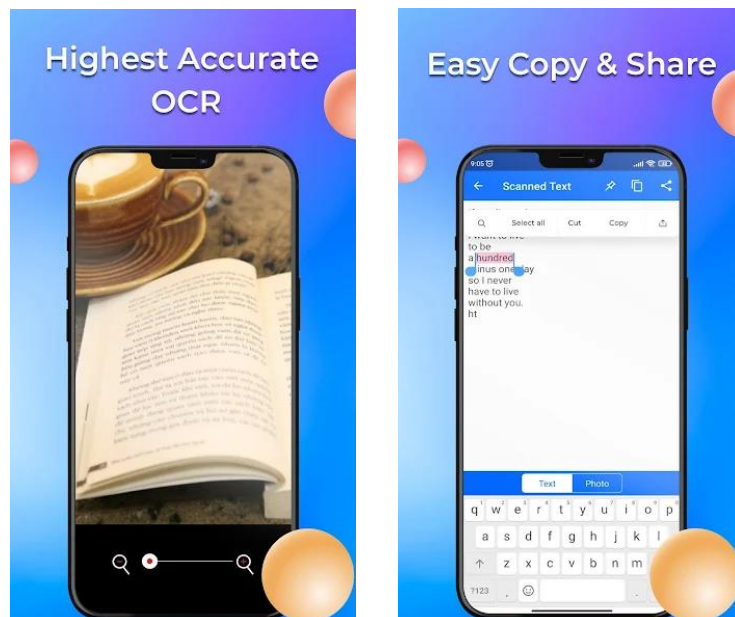


Fig. 2 Text Scanner

### 2.4.2 Google Translate

Google Translate is a translator application by Google that help the user translate sentences or words into several languages. Google Translate also can translate a text from image.

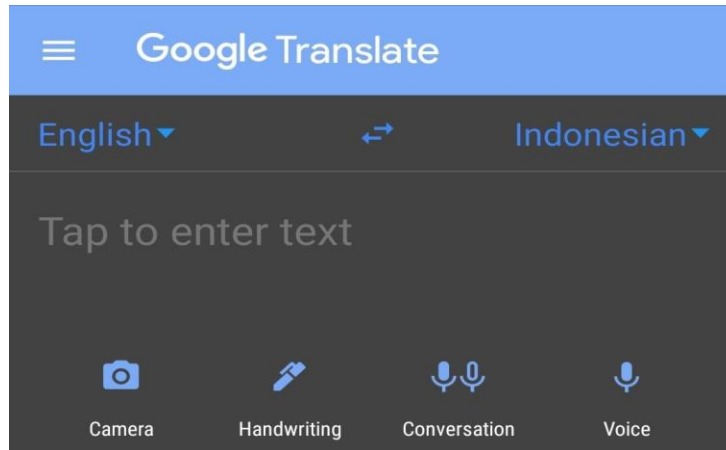


Fig. 3 Google Translate

### 2.4.3 Cam Find

Cam Find is an app that identifies objects by picture for you. The app's most intuitive feature is the visual search engine through which the user can search the physical world. This picture recognition app's simple interface helps you take pictures of an object. The visual search engine behind the app will tell you what the object is.

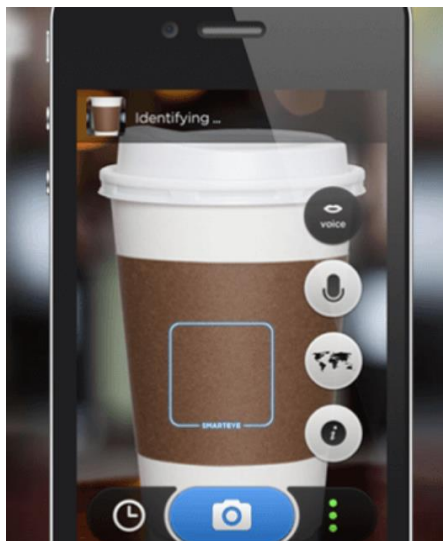


Fig. 4 Cam Find



## 2.5 Comparison Overview

Listed in Table 2.5.1 below are the several features that compared from related work with Image-to-Text-to-Speech Program.

**Table 2.5.1 Design and Development of Image-to-Text-to-Speech Program Comparison Overview**

Features	Text Scanner	Google Translate	Cam Find	Image-to-Text-to-Speech Program
Recognizing Inputted Image and Converting it to Text (Image-to-Text)	Yes	Yes	Yes	Yes
Translating the Detected Text into Another Language	No	Yes	No	Yes
Reading and Playing the Text Audio File into Several Languages (Text-to-Speech)	No	Yes	No	Yes
Saving the Text Audio File with Several Languages	No	No	No	Yes

## CHAPTER 3

### METHODS

#### 3.1 Image-to-Text

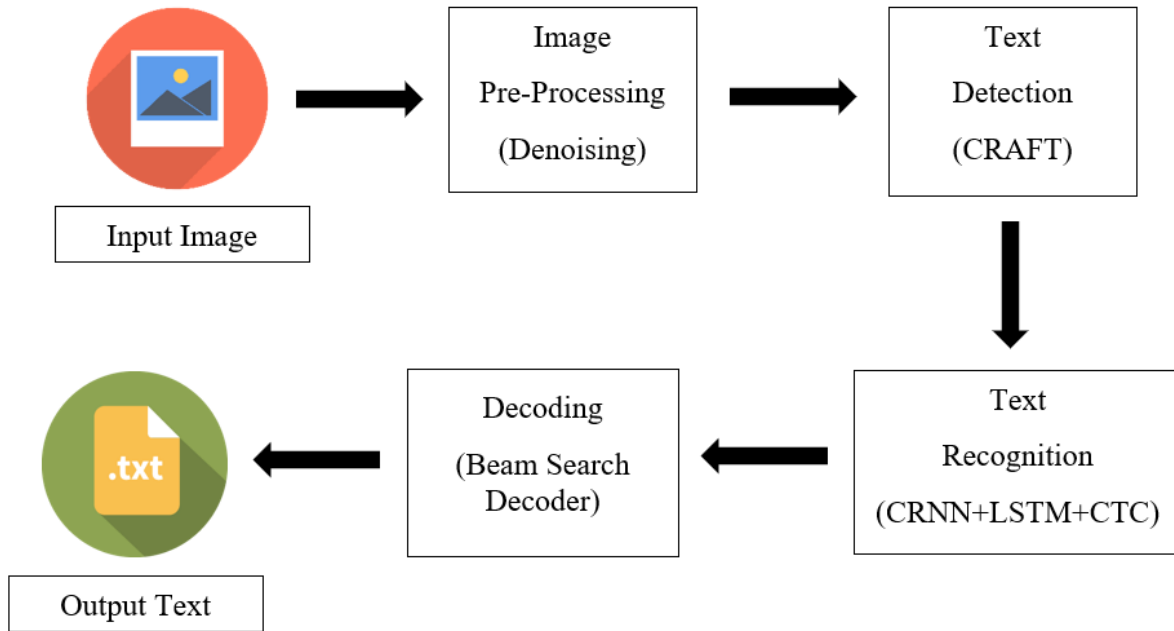


Fig. 5 Image-to-Text Flow Diagram

To convert this image-to-text, we use the OCR (Optical Character Recognition) method with several steps. OCR is a technology that recognizes text within a digital image. It is commonly used to recognize text in scanned documents and images. OCR can be used to convert a physical paper document or an image into an accessible electronic version with text. The OCR will recognize the text and convert the document to an editable text file. OCR processes a digital image by locating and recognizing characters, such as letters, numbers, and symbols. Advanced OCR can export the size and formatting of the text as well as the layout of the text found on a page. Below is the process or step-by-step on how to convert image-to-text using OCR:

##### 1. Image Pre-Processing (Denoising)

This first step is image pre-processing it means we process the image before we use it and make the image more suitable for the next step. For example,

the input image may be damaged due to noise or other things that might cause errors in the results. Therefore, the image should always be pre-processed to remove the deformity. Image denoising is to remove noise from a noisy image, so as to restore the true image and be more high-quality. Denoising the image can help to get better or to improve the OCR accuracy.

The method we use:

```
cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
```

## 2. Text Detection (CRAFT)

Text detection, as clear from the name, simply means finding the regions in the image where text can be present. This is clearly illustrated by the bounding boxes that are drawn around the detected text regions. It also applies the CRAFT (Character Region Awareness for Text Detection) algorithm to detect the text. CRAFT is a scene text detection method to effectively detect the text areas by exploring each character and affinity between the characters. There are several parameters in Text Detection from CRAFT:

- **text\_threshold** (float, default = 0.7) - Text confidence threshold
- **low\_text** (float, default = 0.4) - Text low-bound score
- **link\_threshold** (float, default = 0.4) - Link confidence threshold
- **canvas\_size** (int, default = 2560) - Maximum image size. Image bigger than this value will be resized down
- **mag\_ratio** (float, default = 1) - Image magnification ratio [8]

The method we use:

```
ImageDraw.Draw(image)
```

## 3. Text Recognition (CRNN+LSTM+CTC)

In the previous step, we segmented out the text regions. Now, we will recognize what text is present in those segments. This is known as Text Recognition. So, in this step, we pass each segment one-by-one to our text recognition model that will output the recognized text. Also, we keep a track of

each segment bounding box coordinates. In general, this step outputs a text file that contains each segment's bounding box coordinates along with the recognized text and the model confidence level. The recognition model uses CRNN (Convolutional Recurrent Neural Network). The sequencing labelling is performed by LSTM (Long Short-Term Memory) and CTC (Connectionist Temporal Classification), here the CTC is meant for labelling the unsegmented sequence data with RNN (Recurrent Neural Network). There are several parameters in Text Recognition:

- **image** (string, numpy array, byte) - Input image
- **horizontal\_list** (list, default=None) - see format from output of detect method
- **free\_list** (list, default=None) - see format from output of detect method
- **decoder** (string, default = 'greedy') - options are 'greedy', 'beamsearch' and 'wordbeamsearch'.
- **beamWidth** (int, default = 5) - How many beam to keep when decoder = 'beamsearch' or 'wordbeamsearch'
- **batch\_size** (int, default = 1) - batch\_size>1 will make EasyOCR faster but use more memory
- **workers** (int, default = 0) - Number thread used in of dataloader
- **allowlist** (string) - Force EasyOCR to recognize only subset of characters. Useful for specific problem
- **blocklist** (string) - Block subset of character. This argument will be ignored if allowlist is given.
- **detail** (int, default = 1) - Set this to 0 for simple output
- **paragraph** (bool, default = False) - Combine result into paragraph
- **contrast\_ths** (float, default = 0.1) - Text box with contrast lower than this value will be passed into model 2 times. First is with original image and second with contrast adjusted to 'adjust\_contrast' value. The one with more confident level will be returned as a result.

- **adjust\_contrast** (float, default = 0.5) - target contrast level for low contrast text box
- **Return** list of results [8]

The method we use:

```
reader.readtext(dst, add_margin=0.55, width_ths=0.7,
link_threshold=0.8, decoder='beamsearch',
blocklist='=-')
```

#### 4. Decoding (Beam Search Decoder)

Decoder helps us to improve text recognition results by avoiding spelling mistakes, allowing arbitrary numbers and punctuation marks and making use of a word-level language model. [9] Beam search decoding is a fast and well-performing algorithm with an integrated language model to decode the neural network output in the context of text recognition. Beam search decoding iteratively creates text candidates (beams) and scores them. CTC beam search decoding is a simple and fast algorithm and outperforms best path decoding. A character-level LM (Language Model) can easily be integrated.

The parameter we use:

```
decoder='beamsearch'
```

### 3.2 Text-to-Speech

The text-to-speech (TTS) is the process of converting words into a vocal audio form. The program, tool, or software takes an input text from the user, and using methods of natural language processing understands the linguistics of the language being used, and performs logical inference on the text. This processed text is passed into the next block where digital signal processing is performed on the processed text. Using many algorithms and transformations this processed text is finally converted into a speech format. This entire process involves the synthesizing of speech. [10] Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-

speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech. The reverse process is speech recognition. Below, is a simple block diagram to understand the same.

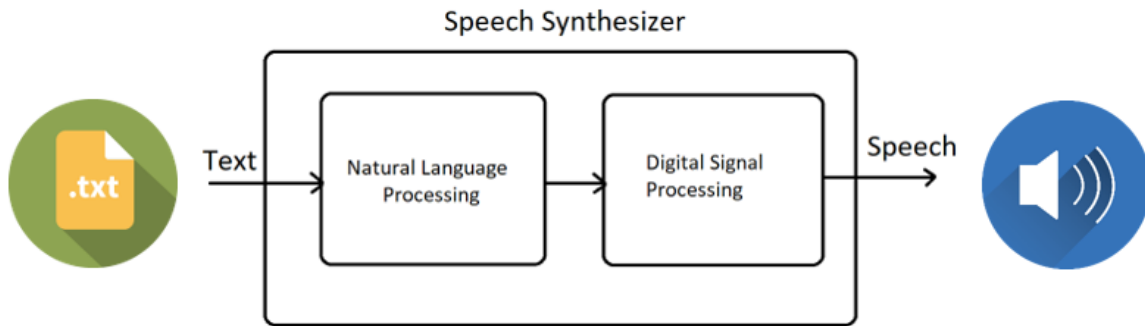


Fig. 6 Text-to-Speech Flow Diagram

From the block diagram, we can understand that the text being passed is firstly pre-processed with the help of natural language processing, and then using digital signal processing is converted to speech.

### 1. Natural Language Processing

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

### 2. Digital Signal Processing

The Digital Signal Processing (DSP) component handles the actual machine 'pronunciation' of words, phrases and sentences, analogous to human speech articulation (based on input to the Digital Signal Processing component). This component can be implemented in two ways, namely rule-based synthesis and concatenative synthesis.

## CHAPTER 4

### RESULTS

#### Step 1: Installing and Importing Libraries

```
[21] #install and import opencv library to do denoising for the image
!pip install opencv-python-headless==4.1.2.30
import cv2

Requirement already satisfied: opencv-python-headless==4.1.2.30 in /usr/local/lib/python3.7/dist-packages (4.1.2.30)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python-headless==4.1.2.30) (1.19.5)

[22] #install and import easyocr library to read text from image
!pip install easyocr
import easyocr

Requirement already satisfied: easyocr in /usr/local/lib/python3.7/dist-packages (1.4.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from easyocr) (1.4.1)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.7/dist-packages (from easyocr) (4.1.2.30)
Requirement already satisfied: Pillow<8.3.0 in /usr/local/lib/python3.7/dist-packages (from easyocr) (7.1.2)
Requirement already satisfied: torch in /usr/local/lib/python3.7/dist-packages (from easyocr) (1.10.0+cu111)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages (from easyocr) (0.18.3)
Requirement already satisfied: python-bidi in /usr/local/lib/python3.7/dist-packages (from easyocr) (0.4.2)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (from easyocr) (3.13)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from easyocr) (1.19.5)
Requirement already satisfied: torchvision>=0.5 in /usr/local/lib/python3.7/dist-packages (from easyocr) (0.11.1+cu111)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch->easyocr) (3.10.0.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from python-bidi->easyocr) (1.15.0)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-packages (from scikit-image->easyocr) (2021.11.2)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from scikit-image->easyocr) (1.2.0)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->easyocr) (2.6.3)
Requirement already satisfied: matplotlib>=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->easyocr) (3.2.2)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packages (from scikit-image->easyocr) (2.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!>=2.1.2,!>=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0,>=2.0.0->scikit-image->easyocr) (0.11.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0,>=2.0.0->scikit-image->easyocr) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0,>=2.0.0->scikit-image->easyocr) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0.0,>=2.0.0->scikit-image->easyocr) (1.3.2)

[23] #install and import googletrans library to translate the text from image
!pip install googletrans==3.1.0a0
from googletrans import Translator

Requirement already satisfied: install in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: googletrans==3.1.0a0 in /usr/local/lib/python3.7/dist-packages (3.1.0a0)
Requirement already satisfied: httpx==0.13.3 in /usr/local/lib/python3.7/dist-packages (from googletrans==3.1.0a0) (0.13.3)
Requirement already satisfied: chardet==3.* in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (3.0.4)
Requirement already satisfied: idna==2.* in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (2.10)
Requirement already satisfied: httpcore==0.9.* in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (0.9.1)
Requirement already satisfied: sniffio in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (1.2.0)
Requirement already satisfied: rfc3986<2,>=1.3 in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (1.5.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (2021.10.8)
Requirement already satisfied: hstspreload in /usr/local/lib/python3.7/dist-packages (from httpx==0.13.3->googletrans==3.1.0a0) (2021.12.1)
Requirement already satisfied: h2==3.* in /usr/local/lib/python3.7/dist-packages (from httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0) (3.2.0)
Requirement already satisfied: h1<0.10,>=0.8 in /usr/local/lib/python3.7/dist-packages (from httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0) (0.9.0)
Requirement already satisfied: hyperframe<6,>=5.2.0 in /usr/local/lib/python3.7/dist-packages (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0) (5.2.0)
Requirement already satisfied: hpack<4,>=3.0 in /usr/local/lib/python3.7/dist-packages (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==3.1.0a0) (3.0.0)

[24] #install and import gTTS library to convert text-to-speech
!pip install gTTS
from gtts import gTTS

#import IPython.display.audio to directly play the audio in the python notebook
from IPython.display import Audio

Requirement already satisfied: gTTS in /usr/local/lib/python3.7/dist-packages (2.2.3)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from gTTS) (1.15.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from gTTS) (2.23.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from requests) (7.1.2)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->gTTS) (2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->gTTS) (3.0.4)
```

## Step 2: Selecting the language to extract or to read the text

```
✓ [25] #select the language to extract the text
      #for the below example we want to extract it from the image that contains the Japanese language
      reader = easyocr.Reader(['ja'])

      #Translator class used to translate text from one language into another language
      translator = Translator()

      CUDA not available - defaulting to CPU. Note: This module is much faster with a GPU.
```

## Step 3: Inputting the image to convert

```
✓ [26] #import PIL (Python Imaging Library) which provides the python interpreter with image editing capabilities
      import PIL

      #import ImageDraw module from PIL to draw the bounding boxes in the given image
      from PIL import ImageDraw

      #open and read the image that wants to extract the text from
      im = PIL.Image.open('story(jpn).png')
      im
```

ある日、キツネは食べ物を探しに行くととてもお腹がすいた。彼は高低を検索しましたが、彼が食べることができるものを見つけることができませんでした。

## Step 4: Denoising the inputted image

```
✓ [27] #import matplotlib library to show the denoising images result
      from matplotlib import pyplot as plt

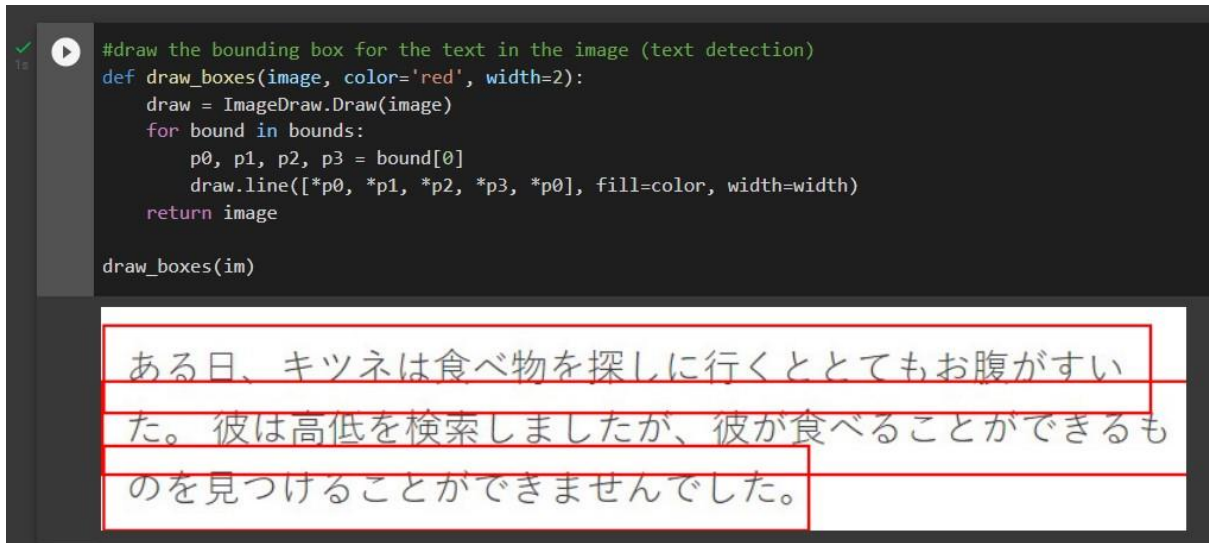
      #do denoising images to improve the OCR accuracy
      img = cv2.imread('story(jpn).png')
      dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
      plt.figure(figsize=(50,50))
      plt.subplot(121), plt.imshow(img)
      plt.subplot(122), plt.imshow(dst)
      plt.show()
```

ある日、キツネは食べ物を探しに行くととてもお腹がすいた。彼は高低を検索しましたが、彼が食べることができるものを見つけることができませんでした。

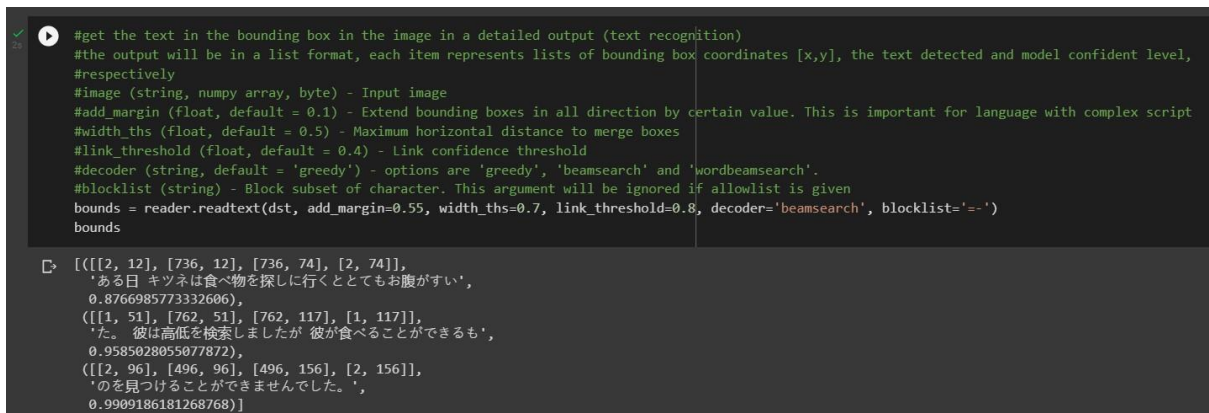
ある日、キツネは食べ物を探しに行くととてもお腹がすいた。彼は高低を検索しましたが、彼が食べることができるものを見つけることができませんでした。



## Step 5: Drawing bounding box to detect text in the inputted image (text detection)



## Step 6: Getting the text from the bounding box in a detailed output (text recognition)



## Step 7: Getting the text from the bounding box in a simple output (text recognition)

```
✓ [30] #get the text in the bounding box in the image in a simple output by using detail=0 parameter (text recognition)
#the output will be in a list format but in a simple way, just the text detected, without a bounding box coordinates [x,y] and model confident level
#image (string, numpy array, byte) - Input image
#add_margin (float, default = 0.1) - Extend bounding boxes in all direction by certain value. This is important for language with complex script
#width_ths (float, default = 0.5) - Maximum horizontal distance to merge boxes
#link_threshold (float, default = 0.4) - Link confidence threshold
#decoder (string, default = 'greedy') - options are 'greedy', 'beamsearch' and 'wordbeamsearch'.
#blocklist (string) - Block subset of character. This argument will be ignored if allowlist is given
#detail (int, default = 1) - Set this to 0 for simple output
text_list = reader.readtext(dst, add_margin=0.55, width_ths=0.7, link_threshold=0.8, decoder='beamsearch', blocklist='-', detail=0)
text_list

❏ ['ある日 キツネは食べ物を探しに行くととてもお腹がすい',
    'た。彼は高低を検索しましたが 彼が食べることができるも',
    'のを見つけることができませんでした。']
```

## Step 8: Combining the detected text in the image into the single-line text

```
✓ [31] #combine/join the detected text in the image from multiple lines into the single-line text
text_comb=' '.join(text_list)
text_comb

'ある日 キツネは食べ物を探しに行くととてもお腹がすい た。彼は高低を検索しましたが 彼が食べることができる ものを見つけることができませんでした。'
```

## Step 9: Detecting the text language

```
✓ [32] #detect the language of the text in the image using translator
print(translator.detect(text_comb))

Detected(lang=ja, confidence=1)
```

## Step 10: Translating the detected text into another language

```
✓ [33] #translate the detected language of the text into a default language (English)
text_en = translator.translate(text_comb, src='ja')
print(text_en.text)

One day the fox was very hungry when he went looking for food. He searched high and low but couldn't find him even though he could eat.

✓ [38] #translate the detected language of the text into the Indonesian language
text_id = translator.translate(text_comb, src='ja', dest='id')
print(text_id.text)

Suatu hari rubah sangat lapar ketika dia pergi mencari makanan. Dia mencari tinggi dan rendah tetapi tidak dapat menemukan apa pun yang bisa dia makan.
```

### Step 11: Reading and saving the text audio (text-to-speech)

```
✓ [34] #read the text audio in a default accent (English)
0s ta_tts = gTTS(text_en.text)

#save the text audio file
ta_tts.save('trans.mp3')

✓ [39] #read the text audio in an Indonesian accent
0s ta_tts_id = gTTS(text_id.text, lang='id')

#save the text audio file
ta_tts_id.save('trans_id.mp3')
```

### Step 12: Playing the text audio file

```
✓ [35] #play the text audio file in an English accent
0s Audio('trans.mp3', autoplay=True)
```



An audio player interface for the file 'trans.mp3'. It shows a play button, a progress bar at 0:09 / 0:09, a volume icon, and a settings menu icon.

```
✓ [40] #play the text audio file in an Indonesian accent
0s Audio('trans_id.mp3', autoplay=True)
```



An audio player interface for the file 'trans\_id.mp3'. It shows a play button, a progress bar at 0:13 / 0:13, a volume icon, and a settings menu icon.

## **CHAPTER 5**

### **CONCLUSION**

In this project or program, we have implemented various techniques for efficient image processing, especially when converting the image to text. We have also used different techniques or methods for processing the input and output in our program. We hope that our program can be useful in accordance with our program's goal, which is to be able to help people who cannot read or even blind people to still be able to obtain information from a text image by hearing the text audio and also help other people to translate an image containing text into various languages and can also listen to the text audio.

## REFERENCES

- [1] O'Reilly. (n.d). Information Literacy in the Digital Age. O'Reilly Online Learning. Retrieved June 14, 2022, from <https://www.oreilly.com/library/view/information-literacy-in/9781843345169/xhtml/B9781843345152500019.htm>
- [2] Montoya, S. (2018). *Silvia Montoya - UNESCO*. UNESCO. Retrieved June 14, 2022, from [https://gaml.uis.unesco.org/wp-content/uploads/sites/2/2018/12/4.6.1\\_07\\_4.6-defining-literacy.pdf](https://gaml.uis.unesco.org/wp-content/uploads/sites/2/2018/12/4.6.1_07_4.6-defining-literacy.pdf)
- [3] Kementerian Koordinator Bidang Pembangunan Manusia dan Kebudayaan, & Novrizaldi. (2021, November 19). *Tingkat Literasi Indonesia Memprihatinkan, Kemenko PMK Siapkan Peta Jalan Pembudayaan Literasi Nasional | Kementerian Koordinator Bidang Pembangunan Manusia dan Kebudayaan*. Kementerian Koordinator Bidang Pembangunan Manusia Dan Kebudayaan. Retrieved June 14, 2022, from <https://www.kemendikopmk.go.id/tingkat-literasi-indonesia-memprihatinkan-kemenko-pmk-siapkan-peta-jalan-pembudayaan-literasi>
- [4] Dataman, C. K. (2021, March 20). *What Is Image Recognition? - Dataman in AI*. Medium. Retrieved June 14, 2022, from <https://medium.com/dataman-in-ai/module-6-image-recognition-for-insurance-claim-handling-part-i-a338d16c9de0>
- [5] Contributor, T. (2021, October 5). *image recognition*. SearchEnterpriseAI. Retrieved June 14, 2022, from <https://www.techtarget.com/searchenterpriseai/definition/image-recognition>
- [6] Itransition. (n.d.). *OCR algorithms: a complete guide – Itransition*. Retrieved June 14, 2022, from <https://www.itransition.com/blog/ocr-algorithm>
- [7] Karandish, F. (2019, November 25). *The Comprehensive Guide to Optical Character Recognition (OCR)*. Moov AI. Retrieved June 14, 2022, from <https://moov.ai/en/blog/optical-character-recognition-ocr/>
- [8] *Jaided AI: EasyOCR documentation*. (n.d.). Jaided AI. Retrieved June 14, 2022, from <https://www.jaided.ai/easyocr/documentation/>

[9] Scheidl, H. (2022, January 8). *Word Beam Search: A CTC Decoding Algorithm - Towards Data Science*. Medium. Retrieved June 14, 2022, from <https://towardsdatascience.com/word-beam-search-a-ctc-decoding-algorithm-b051d28f3d2e>

[10] K, B. (2021, December 15). *How to get started with Google Text-to-Speech using Python*. Medium. Retrieved June 14, 2022, from <https://towardsdatascience.com/how-to-get-started-with-google-text-to-speech-using-python-485e43d1d544>

## ATTACHMENTS

```
[70] #draw the bounding box for the text in the image (text detection)
def draw_boxes(image, color='red', width=2):
    draw = ImageDraw.Draw(image)
    for bound in bounds:
        p0, p1, p2, p3 = bound[0]
        draw.line([*p0, *p1, *p2, *p3], fill=color, width=width)
    return image

draw_boxes(im)
```

```
[71] #get the text in the bounding box in the image in a simple output by using detail=0 parameter (text recognition)
#the output will be in a list format but in a simple way, just the text detected, without a bounding box coordinates [x,y] and model confident level
#image (string, numpy array, byte) - Input image
#add_margin (float, default = 0.1) - Extend bounding boxes in all direction by certain value. This is important for language with complex script
#width_ths (float, default = 0.5) - Maximum horizontal distance to merge boxes
#link_threshold (float, default = 0.4) - Link confidence threshold
#decoder (string, default = 'greedy') - options are 'greedy', 'beamsearch' and 'wordbeamsearch'.
#blocklist (string) - Block subset of character. This argument will be ignored if allowlist is given
#detail (int, default = 1) - Set this to 0 for simple output
text_list = reader.readtext(dst, add_margin=0.55, width_ths=0.7, link_threshold=0.8, decoder='beamsearch', blocklist='-', detail=0)
text_list
```

['ある日 キツネは食べ物を探しに行くととてもお腹がすい',  
'た。 彼は高低を検索しましたが 彼が食べることができる',  
'のを見つけることができませんでした。']

## Image-to-Text

```
[72] #combine/join the detected text in the image from multiple lines into the single-line text
text_comb=' '.join(text_list)
text_comb

'ある日 キツネは食べ物を探しに行くととてもお腹がすいた。 彼は高低を検索しましたが 彼が食べることができるものを見つけることができませんでした。 '
```

```
[73] #detect the language of the text in the image using translator
print(translator.detect(text_comb))

Detected(lang=ja, confidence=1)
```

```
[74] #translate the detected language of the text into a default language (English)
text_en = translator.translate(text_comb, src='ja')
print(text_en.text)


One day the fox was very hungry when he went looking for food. He searched high and low but couldn't find him even though he could eat.
```

## Language Translation for Detected Text

```
✓ [75] #read the text audio in a default accent (English)
0s ta_tts = gTTS(text_en.text)

#save the text audio file
ta_tts.save('trans.mp3')

✓ [76] #play the text audio file in an English accent
0s Audio('trans.mp3', autoplay=True)
```



Text-to-Speech