Technical University "Gheorghe Asachi", Iași
Faculty of Automatic Control and Computer Engineering
Image Processing

Initial report - Text detection and text recognition in images for Text2Speech modules

TEAM: SA06_01

Eugen Leonid Cocalea Creoleanu (ELCC)
1408A

1 Project overview

The application we're going to develop is a Proof of Concept application that is attempting to identify text in images, recognize that text and output that in a format ready to be picked up by a 3rd party application.

Text detection and recognition applications can be used for a multitude of purposes like translating instructions (recipes, street directions, attraction descriptions, newspaper articles) written in a foreign language, helping visually impaired or illiterate people get around tasks that involve reading instructions, creating Augmented Reality applications that take information from the real world and present it to the user. The eventual end goal is to have the application run on portable / wearable computers that will accompany the user in the real world and continuously find information to work on in the surroundings.

The objective of the project is to achieve a reasonable rate of text detection and recognition in images that are already acquired and fed into the application while leaving its integration with 3rd party applications (like portable / wearable computers, translation engines, text2speech engines) for a later project.

Expected challenges are mostly related to the lack of field knowledge by the team and the scarcity of time to be allocated to the project.

Potential consumers: tourists, visually impaired people, illiterate people, kids learning to read, neglected kids that needs a story read to them one evening.

Competitors: Meta (1), Apple (3), Google (4)

2 State of the Art

Text recognition in images, also known as Optical Character Recognition (OCR), is an active area of research in the field of computer vision. The state-of-the-art approaches for text recognition in images typically use deep learning-based methods, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs). These methods have been shown to achieve high accuracy on various benchmarks and real-world datasets.

In recent years, researchers have been focusing on developing end-to-end OCR systems that can handle various types of text in images, such as curved or perspective text, as well as text in different languages and scripts. One of the most promising approach for handling such text is the use of attention-based models, such as the Attention-OCR and CRAFT.

Attention-based models are a type of deep learning model that have been used in recent years for various tasks, including text recognition in images. These models are designed to focus on specific parts of the input, rather than processing the entire input at once.

The Attention-based OCR models use a mechanism called attention, which allows the model to focus on specific parts of the image, such as individual characters or words, when making predictions. This mechanism is implemented by using a set of weights, called attention weights, that are used to weigh the importance of different parts of the input.

One of the most popular attention-based OCR models is the Attention-OCR, which uses a convolutional neural network (CNN) as the encoder and a recurrent neural network (RNN) as the decoder. The model first encodes the image features using the CNN, and then uses the RNN to generate a sequence of characters by focusing on specific parts of the encoded features.

Another attention-based model is CRAFT (Character-Region Awareness For Text detection) which is a two-stage framework. In the first stage, the model detects character regions, and in the second stage, it recognizes the text in the detected regions by using a sequence-to-sequence model with attention mechanism.

Another important area of research in text recognition is the development of models that can handle scene text, which refers to text that appears in natural images, such as street signs or shop signs. Scene text recognition is a challenging task due to the large variability in the appearance and layout of text in natural images.

In addition, researchers have been working on developing real-time text recognition systems that can run on resource-constrained devices, such as smartphones or cameras. This has been achieved by using lightweight models and efficient inference algorithms.

3 Tesseract

Tesseract is an optical character recognition (OCR) engine developed by Google. The Tesseract library for Python, pytesseract, is a wrapper for the Tesseract OCR engine. It allows developers to integrate Tesseract into their Python applications and perform OCR on images.

The steps that the library takes to identify text in an image are:

Pre-processing: The image is first pre-processed to enhance the quality of the image. This includes steps such as converting the image to gray scale, adjusting the contrast and brightness, and removing noise.

- Image binarization: The first step in pre-processing is to convert the input image into a binary image, where each pixel is either black or white. Tesseract uses a method called Adaptive thresholding, which calculates a threshold value for each pixel based on the local neighborhood and converts the pixel to white or black accordingly.
- Image deskewing: This step is used to correct any skew or rotation in the input image. Tesseract uses the Hough Transform algorithm for this purpose.
- Image enhancement: This step is used to improve the quality of the input image. Tesseract uses a method called Unsharp masking, which is a technique to sharpen the edges of an image.
- Image de-noising: This step is used to remove any noise from the input image. Tesseract uses a method called Median Filtering, which replaces each pixel's value with the median value of the pixels in its neighborhood.
- Image layout analysis: This step is used to identify the layout of the text in the input image. Tesseract uses a method called the Connected Components Analysis, which is used to find connected regions of the same color in an image.

Segmentation: The image is then segmented into regions of interest, where each region corresponds to a character or a group of characters. The segmentation is performed to separate the text from the background and to identify individual characters.

- Text Line Segmentation: This step is used to segment the text in an image into individual lines. Tesseract uses a method called the Stroke Width Transform (SWT) algorithm, which

is based on the idea that the width of the strokes in text is relatively constant, while the width of the strokes in non-text regions varies greatly.

- Character Segmentation: This step is used to segment the text in an image into individual characters. Tesseract uses a method called the Connected Components Analysis (CCA), which is used to find connected regions of the same color in an image.
- Word Segmentation: This step is used to segment the text in an image into individual words. Tesseract uses a method called the Dynamic Programming algorithm, which is used to find the optimal segmentation of characters into words.

Feature extraction: Features are extracted from the segments identified in step 2. These features are used to represent the character in a way that the OCR engine can understand.

- Normalization: This step is used to normalize the size and orientation of the characters, so that they can be easily compared. Tesseract uses a method called the Affine Normalization, which is based on the idea of aligning all the characters to a common baseline, and scales them to a common height.
- Feature extraction: This step is used to extract features from the normalized characters that can be used for recognition. Tesseract uses a method called the Quadratic classifier, which is based on a set of features extracted from the normalized character images, such as the intensity of the pixels in certain regions of the image, the width of the strokes, and the ratio of the width to the height.
- Feature encoding: This step is used to encode the features extracted from the characters, so that they can be easily compared. Tesseract uses a method called the Neural Network based classifier, which is a type of deep learning algorithm that is trained to recognize patterns in the feature vectors.

Recognition: The features extracted in step 3 are passed through the OCR engine, which recognizes the characters based on the features.

- Neural network-based classifier: Tesseract uses a neural network-based classifier, which is trained to recognize patterns in the feature vectors extracted from the characters. The neural network architecture used in Tesseract is based on the Long Short-Term Memory (LSTM) algorithm, which is a type of Recurrent Neural Network (RNN) that is particularly well-suited for sequential data such as text.
- Language model: Tesseract uses a language model to improve the recognition accuracy by taking into account the context of the text being recognized. The language model used in Tesseract is based on a statistical approach, which estimates the probability of a sequence of words or characters based on the frequency of occurrence of these words or characters in a large training dataset.
- Dictionary based recognition: Tesseract also has the ability to perform recognition based on a pre-defined dictionary of words, which can be useful in cases where the text to be recognized is limited to a specific set of words, such as license plates, or when there is a need to recognize specific terms and names.

Post-processing: The recognized text is post-processed to correct any errors that might have been introduced during the recognition process. This step includes correction of spelling mistakes, grammar errors, and formatting errors.

- Spelling correction: Tesseract uses a spelling correction algorithm to correct any errors in the recognized text. It uses a combination of statistical algorithms and dictionary-based methods to check the recognized text against a list of known words and suggest corrections for any misspelled words.
- Text-to-Speech (TTS) conversion: Tesseract uses TTS conversion to convert the recognized text into spoken words. It uses a combination of phonetic and statistical algorithms to convert the text into the corresponding speech sounds.
- Text-to-Braille conversion: Tesseract uses Text-to-Braille conversion to convert the recognized text into braille. it uses a combination of algorithms to convert the text into the corresponding braille characters.

- Text-to-XML conversion: Tesseract uses Text-to-XML conversion to convert the recognized text into an XML file. it uses a combination of algorithms to format the recognized text into an XML file with proper tags, such as headings, paragraphs, and lists.
- Text-to-PDF conversion: Tesseract can also convert the recognized text into a PDF file, preserving the original format and layout of the text.

4 Development approach

Task allocation

Task ID	Task description	Team member	Estimated time
Research	Research on existing solutions and technologies	ELCC	8h
Design	Application design Choosing relevant technologies	ELCC	8h
Implementation	Application development Application testing	ELCC	48h
Documentation	Writing application documentation	ELCC	8h

Git repository: https://github.com/Pucster/sa06t2s

References

[1] Meta AR/VR

[2] Google OCR

[3] Apple Glasses

[4] Google Glasses