UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament de Teoria del Senyal
i Comunicacions

telecos
BCN

FINAL REPORT

Mapping physical stores using Google Street View

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Adrià González Mestre

In partial fulfilment

of the requirements for the degree in

BACHELOR'S DEGREE IN TELECOMMUNICATIONS TECHNOLOGIES AND SERVICES ENGINEERING

Barcelona, June 2020

Abstract

Understanding textual information in natural scenes images has a crucial role in fields like text detection and text recognition. This information could be widely used to automatize what are now manual jobs as annotating and classifying nouns and find a matching a pairing in a database. The development of these detection and recognition techniques is what will make these methods more reliable.

The robustness of these two tasks will be separately implemented and analyzed. For that purpose, it will be necessary a labeled database made by ourselves where experiments and metrics will be carried out, both for text detection as a first step, where the system obtains coordinates where the text is detected in the image, and for text recognition as a second step, where the system extracts word strings for each coordinate previously obtained.

Finally, as this project aims to automatize the mentioned task, annotating and pairing, the extracted strings will be used to geolocate the position associated with the textual information obtained, and download through Google Street View a natural view of the geolocalization to compare the correspondence with the database.

Resumen

Comprender la información textual en imágenes de escenas naturales tiene un papel crucial en campos como la detección de texto y el reconocimiento de texto. Esta información podría usarse ampliamente para automatizar lo que ahora son trabajos manuales como anotar y clasificar sustantivos y encontrar un emparejamiento que coincida en una base de datos. El desarrollo de estas técnicas de detección y reconocimiento es lo que hará que estos métodos sean cada vez más confiables.

La solidez de estas dos tareas se implementará y analizará por separado. Para ese propósito, será necesario una base de datos etiquetada hecha por nosotros donde se llevarán a cabo experimentos y métricas, tanto para la detección de texto como un primer paso, donde el sistema obtiene coordenadas donde se detecta el texto en la imagen, y para el reconocimiento de texto como segundo paso, donde el sistema extrae cadenas de palabras para cada coordenada obtenida previamente.

Finalmente, como este proyecto tiene como objetivo automatizar la tarea mencionada, la anotación y el emparejamiento, las cadenas extraídas se utilizarán para geolocalizar la posición asociada con la información textual obtenida, y descargar a través de Google Street View una vista natural de la geolocalización para comparar la correspondencia con la base de datos.

Resum

Comprendre la informació textual en imatges d'escenes naturals té un paper crucial en camps com la detecció de text i el reconeixement de text. Aquesta informació es pot utilitzar àmpliament per automatitzar el que ara són treballs manuals com numerar i classificar substantius i trobar una parella que concordi en una base de dades. El desenvolupament d'aquestes tècniques de detecció i reconeixement és el que farà que aquests mètodes siguin cada vegada més fiables.

La solidesa d'aquestes dues tasques s'implementarà i analitzarà per separat. Per a aquest propòsit, caldrà una base de dades etiquetada feta per nosaltres on es duran a terme experiments i mètriques, tant per a la detecció de text com un primer pas, on el sistema obté coordenades on es detecta el text a la imatge, i per el reconeixement de text com a segon pas, on el sistema extreu cadenes de paraules per a cada coordenada obtinguda prèviament.

Finalment, com aquest projecte té com a objectiu automatitzar la tasca esmentada, l'anotació i l'aparellament, les cadenes extretes s'utilitzaran per localitzar la posició associada amb la informació textual obtinguda, i descarregar a través de Google Street View una visió natural de la geolocalització per comparar la correspondència amb la base de dades.

Acknowledgements

First, I want to thank my tutor Josep Ramon Morros and Elisa Sayrol for proposing me to be part of this project and letting me work in the field where I am interested in. Also, I want to express my gratitude for support when assistance was needed and helping in the creation of the database.

And finally, I really want to thank my parents for all the dedication and effort they have had with me to this point of life, supporting when I had problems or in difficult times.

Revision history and approval record

Revision	Date	Purpose
0	04/06/2020	Document creation
1	27/06/2020	Document revision
2	28/06/2020	Document revision

Document distribution list

Name	e-mail
Adrià González Mestre	adria.gonzalez.mestre@gmail.com
Josep Ramon Morros	ramon.morros@upc.edu
Elisa Sayrol	elisa.sayrol@upc.edu

Written by:		Reviewed and approved by:	
Date	04/06/2020	Date	28/06/2020
Name	Adrià González Mestre	Name	Josep Ramon Morros
Position	Project Author	Position	Project Supervisor

Table of contents

1. Introduction

This project was conceived with the intention of finding a solution to an issue presented by the DIBA (Diputació de Barcelona). The goal of the project is to detect unregistered economic activities in our municipalities, a task that is at the present time done manually going through each economic activity in our streets. In other words, this project aim is to detect and recognize physical stores from captured images and, after that, check if they are present in a municipal list. The approach that has been followed is to detect text in the image, as the name written in a sign of a store, that can be used to identify afterwards the location.

Although image detection is an object of research that has been improved exponentially thanks to the evolution and developing of systems based on Machine-learning and Deep-learning techniques, most of the challenges are faced to handwritten text but our scenario involves scene text detection (or as it is known text in the wild). However, understanding textual information in natural scenes has become recently important so extracting scene text is now state-of-the-art.

1.1. Statement of purpose

The main objective of the project is the implementation of a system capable of detecting and recognizing text from pictures that could be obtained from high-resolution cameras to smartphone cameras or Google Street View images. Moreover, Google Street View is a tool that can be used to obtain images from practically anywhere, this feature will be used to download data starting from a coordinate number or a string that can be related to either the name of a shop or a street. Once a database composed of shops and other establishments is created, the text will be detected using different architectures which will provide bounding boxes over each detection and, finally, recognition will be applied over these obtained boxes in order to extract characters and words to compare the result with a labeled list. In addition, we are working with different languages like English, Spanish, and Catalan. For this reason, the project also aims to correct word understanding mistakes.

1.2. Requirements and specifications

Firstly, no official agreement has yet been reached with the DIBA, it implies that a database of store images has been created from scratch, where given the current circumstances of social isolation has complicated the initial task. The captured images are manually annotated, both bounding boxes to perform detection and sign texts to observe recognition. A minimum resolution is needed to be able to extract relevant information, parallel pictures help object detection and some recognition mistakes due to language or calligraphy can take place. On the other hand, the system is executed using Python 3.8 interface, using PyTorch 1.2 libraries to develop the Deep-learning methods. A GPU is recommended to speed up the process.

1.3. Methods and procedures

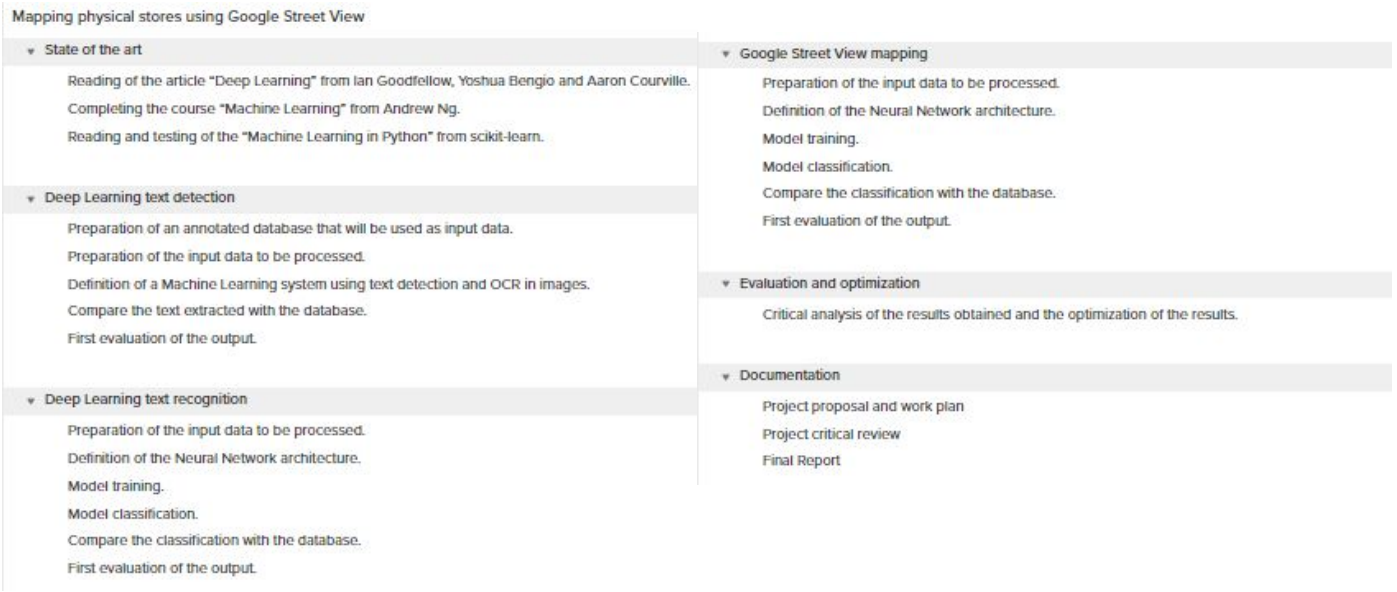
The idea of the project is proposed by Josep Ramon Morros together with Elisa Sayrol Clols, both researchers of the Image Processing Group at the Universitat Politècnica de Catalunya.
The project is a new task that started from scratch, but it still feeds from recent studies and software implementations by other authors state-of-the-art that will be cited in each step through the memory. Moreover, as this project starts in a main point that is shared between three projects, where the other two simultaneous projects are made by Álex Ramírez and Sergi Gispert, it has to be cleared that there are some tasks that are theoretically in common in the path of the development but they are implemented individually and are independent between the different projects. In each project, there are delimited goals and the unique task that has been done in common is the creation of the database as it has to be done from zero for the three projects and the annotation of ground truths in the database needed for text detection and text recognition that was equally done by Álex and me.

1.4. Work Plan

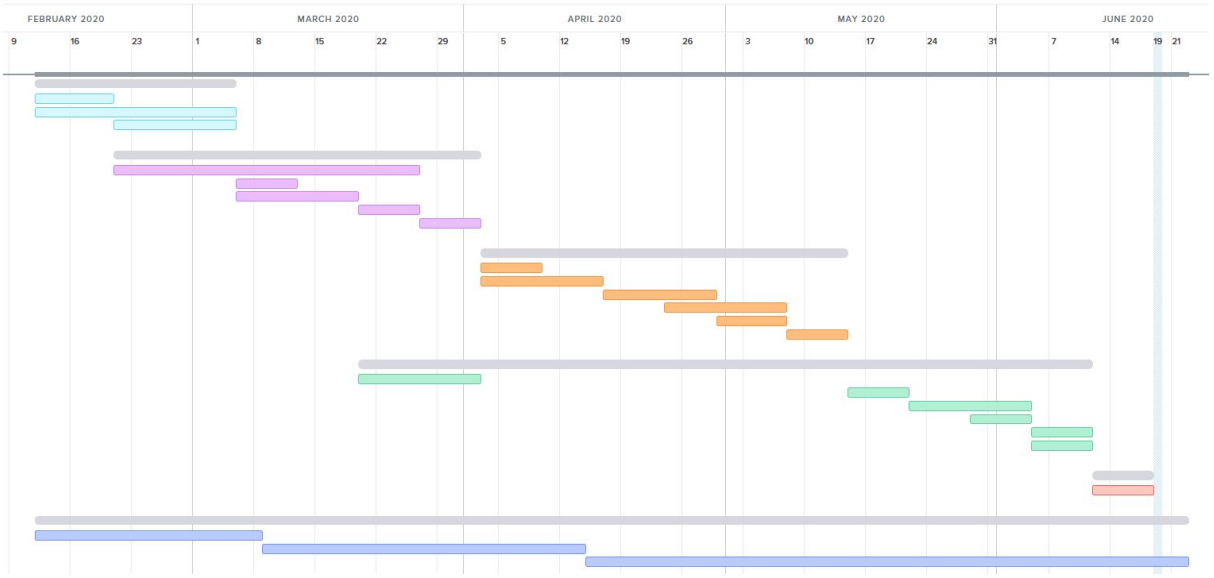
This project was structured using the following work packages and the Gantt diagram.

1.4.1. Work Packages and tasks

- Work Package 1. State of the art
- Work Package 2. Deep Learning text detection
- Work Package 3. Deep Learning text recognition
- Work Package 4. Google Street View mapping
- Work Package 5. Evaluation and optimization
- Work Package 6. Documentation



1.4.2. Gantt diagram



1.5. Incidences

Initially, it was planned to take pictures manually from a large variety of stores, the objective was to create a database that could be used to train Machine-learning and Deep-learning systems. In addition, two pictures of each store in a lapse of three months would be used to test the hardiness of the system when capturing images from the same store with different climatologies or light effects but because work had to be interrupted by the mandatory confinement, the database could not be finally prepared in this path of work due to the lack of images.

Moreover, the idea of using text detection and Optical Character Recognition (OCR) in Google Street View images should be taken into consideration as a future step as it needs a Google Cloud Platform Premium Plans that is no longer available for sign up or new customers to be able to have access to high-resolution images where OCR could be applied correctly.

2. State of the art: Image detection and recognition

2.1. Introduction: Computer Vision

One of the most important developing fields where Artificial Intelligence (AI) is spreadly used is Computer Vision. Various aspects are included in this field that are practical for new applications as image recognition, object detection, image generation, and more.

In the first place, object detection is defined as the capability of software systems to locate objects in an image and identify each object or different ones. These systems have been highly developed as it has been introduced in face and vehicle detection, security or driverless cars, and more. Software developers have exploited most of the well-know old Machine-learning methods as OpenCV [1], a public library originally from Intel.

Early classical implementations involving OpenCV had limited results, but thanks to the improvements that led Deep-learning (which is also based on Machine-learning) more complex algorithms appeared such as R-CNN [2], Faster-RCNN [3], RetinaNet [4] or YOLO [5] that achieved better performances under different conditions.

2.2. Evolution: Deep-learning

While the employment of GPUs has been improved in Deep-learning as technology is progressing efficiency and computational speed, former algorithm methods supported on manual Machine-learning are no obsolete in front of the big data.

In this project, we are going to focus on PyTorch [6], an open-source library utilized in Python, a programming language that is nowadays drawing attention as is commonly used for applications like computer vision and natural language processing. Additionally, it is because one of the benefits is the ability to run on the GPU or graphics card, where Deep-learning bases its power to run faster scripts, allowing to accelerate processes like the training of models or testing databases that are frequently very slow.

Our algorithm will use what we call an Artificial Neural Network (ANN), a system of nodes interconnected in an orderly manner, distributed in layers, through which an input signal propagates to produce an output. They are known in this way because they aim to easily simulate the functioning of the biological neural networks found in the animal brain.

They consist of an input layer, one or more hidden layers, and an output layer and can be trained to learn, to recognize certain patterns. This feature is what includes them within the

ecosystem of technologies known as Artificial Intelligence (AI). ANNs have several decades of history, but have recently become very relevant due to the availability of the large amounts of data and the necessary computing power to use them to solve complex problems.

2.3. State-of-the-art

There are several algorithms that are normally used in text detection. However, the detection of scene text involves a different set of characteristics and challenges that require unique methodologies and solutions unlike in handwritten text. For this reason, it might be more suitable to classify these algorithms analyzing the characteristic advantages that differentiate them instead of the detection steps that they implement.

2.3.1. Pipeline Simplification

We will follow the categorization in [7]. Most methods before the time of Deep-learning, and a few early methods that use Deep-learning, have multi-step pipelines. Newer methods have largely simplified and much shorter pipelines, which is a key to reduce error propagation and simplify the training process. Recently, it has been more common to find methods that use one-staged trained methods instead of two-stages methods as it was done formerly. A base standard for detection of this algorithm bounding box prediction method is the widely known EAST [8].

Inside this category, we can differentiate mainly between two methods. Firstly, multi-step methods are early Deep-learning based methods that take the task of text detection into a multistep process. In multi-step a convolutional neural network is employed to predict whether each pixel within the input image belongs to a character or not, basically, it checks if this character is inside the text region, and another feature is that it identifies the text orientation around the pixel. Indeed, when characters present proximity between them and a common orientation, they will be categorized as a detection of a text region and, for this reason, grouped as a bounding box.

On the other hand, we found the most recent approach which is a simplified pipeline. Now modern methods follow a two-step pipeline, the two steps are first an end-to-end trained neural network and second a post-processing step. The idea behind the simplification is that the post-processing has become smoother. These modern methods that use this simplified pipeline have taken inspiration from general techniques in the field of object detection and have modified the detection regions that were obtained to produce bounding boxes with the intention to directly localize text instances.

2.3.2. Decomposing into Sub-Text

An end-to-end fully convolutional neural network learns to generate a dense prediction map where each pixel within the original image belongs to text instances or not. Post-processing methods then groups pixels together depending on which pixels belong to the common text instance.

Pixels are clustered accordingly to their color consistency and edge information. The fused image segments are called superpixel. To sum up, these obtained superpixels are then that used to extract chains of characters and detect the text instance. An example algorithm that aims to improve text detection that relies in the use of this dense predictions is CRAFT [9].

2.3.3. Specific Targets

The first target to take into account is Long Text, unlike general object detection, the text usually comes in varying aspect ratios. They have a much larger width-height ratio, and thus general object detection framework would fail.

Another distinction from general text detection is Multi-Oriented Text, it means that text detection is rotation sensitive and rotated text is common in the real world, while using traditional axis-aligned prediction boxes would incorporate noisy background that would affect the performance of the following text recognition algorithm.

In this section, another distinction that can be taken into account is that text can present irregular shapes, in natural scene text we will not encounter text that can be included in a rectangular box, it will be found often as curved text. Moreover, this curved text presents a different challenge to the other standards since the rectangular bounding box now needs to include a proportion of background that did not appear in our boxes before. This captured background could include irrelevant text instances that could probably increase the difficulty in the following step of text recognition. The proposed model in [10] called TextSnake which advantage is the adaptability to detect this irregular non-rectangular text aims to solve these rotations.

3. Methodology: Theoretical contents

As previously has been mentioned, the project has been divided in three parts, being the first one corresponding to architectures based on text detection, the second one implements the use of the obtained boxes to text recognition and, the last part, the code necessary to use the prepared data to map a specific store.

As previously has been mentioned, the project has been divided into three parts, being the first one corresponding to the code necessary to use the prepared data to map a specific store, the second one implements architectures based on text detection, the last part, is based on the use of the obtained figures called bounding boxes to carry out text recognition.

3.1. Architectures for Geolocation

3.1.1. Google Street View API

The Street View Static API is used to obtain raw image data as it gives you permission to download a static Street View panorama or picture using your own code with the condition of using a limited personal key and signature, the code also has to be programmed using Python language. The input transmitted to the API is defined with geolocation, a number of coordinates, or string parameters, that could be a word from a sign or a line from the name of a street, it is sent through a Hypertext Transfer Protocol (HTTP) request, and after that process is returned a static image if the emitted key and signature is confirmed.

A complete argument list with multiple parameters is called when requesting this static image, each parameter requested is location, size, heading, field of view (fov), and pitch. One of the main disadvantages is that the maximum size is limited to users at 640x640 pixels and quality can not be improved or upgraded. For this reason, it can be used to locate coordinates or search for a location of a particular list of strings with the objective of getting image data but carrying out Optical Character Recognition (OCR), text detection and recognition, in the vast majority of the pictures is completely out of the question as the text is too blurred.

On the other hand, if a developer Google Maps Platform Premium Plan which is nowadays no longer available could be used, this disadvantage would cease and the limitation would decrease to 2048x2048 pixels scaled images in high resolution where OCR would be then totally feasible.

3.1.2. Detecting street directions in 360° images

One of the other tasks that have to be solved is the detection of a parallel image to the store. It is a problem that comes from the fact that the Google Street View API does not provide a

direction when requesting data images, it is needed to give the location as it was expected and as one request with individual values of heading a field of view would give a totally random direction (it does not follow the path of the Google Street View car in charge of taking the photos), it is necessary to request a chain of multiple headings and field of views to extract a combination of images that put together complete a 360° angle.

For this reason, a Canny edge detector [11] algorithm, a widely used method used to detect all the edges in an image-based in the use of the first gradient in pixel regions where a change of gray level is present, a fact that means we have detected and located and edge. The first gradient is used because regions where the intensity is constant the gradient is zero and where the intensity changes, as was mentioned with the gray levels, it takes a value.

Finally, some information is obtained as the orientation of the gradient in each pixel, we will use this orientation to only detect edges that are horizontal in the image, discarding vertical or crossed edges. This will solve the task that previously proposed because if we apply the Canny edge detector in each image that is downloaded from the 360° total composition and a postprocessing method as Hough Line Transform [12] from OpenCV [1] is used to detect straight lines, we will easily find the parallel pictures as for nature two of the images that are relevant for us will present more edges than the discarded ones, be it for the very shape of the street or the line edges formed by the facades.

3.2. Architectures for Text Detection

3.2.1. EAST (Efficient and Accurate Scent Text)

This robust Deep-learning method is one of the most known and used algorithms for text detection based on this paper [8]. The main advantages of this method are that it can search in an image horizontal and rotated bounding boxes, we understand bounding boxes as the figure coordinates where the text is detected a priori. This feature means that words and lines can be predicted even when text adopts arbitrary orientations. It can be used a posteriori in combination with any text recognition method to extract the text in the bounding box. In this architecture, the text detection pipeline that has been explained previously has excluded intermediate and redundant steps that have been drastically reduced to two stages.

EAST is the idea behind many other Deep-learning architectures that will also be explained, they have in common that uses a fully convolutional network to create word text level or line text level predictions. The produced predictions which are rectangles or rotated rectangles as has been mentioned have multiple figures for the same text but the different bounding boxes are processed using an algorithm called Non-maximum Suppression (NMS) in [13] step to reduce the multiple outputs to a final single one.

Moreover, since EAST is a Deep-learning model, it is feasible to reduce some computationally expensive sub-algorithms that for other text detectors are normally needed. One of these eluded sub-algorithms for example is the step of word partitioning because EAST directly obtains the word.

Finally, it has to be mentioned that EAST is composed by three sections, a Feature extractor steam that consists of four convolution stages (PVANet), a Feature-merging branch and an Output layer that outputs a score map of the reliability and a text quadrangle coordinates, the already mentioned bounding boxes.

Apart from that, in the final implemented algorithm there are three small changes from the original EAST paper implementation [8]: In the project the implementation a ResNet-50 convolution model is used instead of the mentioned PVANet, the optimization of the parameters is done using Intersection over Union rather than the Balanced Cross Entropy, and finally, instead of Staged learning rate decay it has been changed to a Linear learning rate decay.

3.2.2. CRAFT (Character Region Awareness for Text Detection)

CRAFT is a text detector [9]recently considered state-of-the-art that has the objective to detect words using character level detection in scene text, one of the main tasks that are needed to solve during this project. In this algorithm the text is is detected by exploring each individual character region, and also the affinity that is produced between each character with the others. In this case, the bounding box that is obtained from the text detection is produced by consecutively finding minimum bounding rectangles. This mentioned rectangles com from a binary map after thresholding what will be marked as character regions and calculated affinity scores for each character.

In addition, the region score and affinity scores are produced thanks to a convolutional neural network that consists in only two steps [9]: a first convolutional section of 6 VGG16 convolutional stages where the image is introduced as an input, a second decoding step that receives the data and processes the output at word-level to obtain and detect the region and affinity score. In the first place, the region score gives us the information relevant to locate and detect the individual characters and the affinity score is the value that provides the probability chance of a character to come from a chain of characters that will compose a single word. One of the advantages that present CRAFT is that it performs state-of-the-art and is Decomposing into Sub-text and also can be used if it is needed, in Multi-Oriented Text. One of the main disadvantages that could be considered when working with character regions is that the system is vulnerable to perspective deformations so taking parallel pictures in relation to the text will always increase the robustness of the algorithm.

Finally, it has to be mentioned that if the confidence score is low the character will not be estimated and it also can be applied when post-processing because it relies on the scoring, meaning that algorithm as NMS will not be used in this case.

3.2.3. TextSnake

This last detector, TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes [10], has the main objective of solving the Specific Targets that are related to irregular text figures. The advantages of this method are the generalization of the forms that can adopt the different curved texts. As curved text commonly appears on street signs but there are not many databases available, the generalization ability is one of the strong features of this system.
Briefly, the predictions for text detection uses local attributes that follow a line connecting each character in order to form the point list and orientation, this line is called Text Center Line (TCL) and is at the final prediction used to extract the local geometry to obtain the irregular pixel text region.

3.3. Architectures for Text Recognition

3.3.1. Pytesseract

Tesseract [14] was one of the first implementations from HP Laboratories within the nineties. At first, initial implementations of Tesseract algorithm were mainly limited to structured text because it would perform poorly when noise was present. Under those circumstances, tasks that involved unstructured text failed to meet the expectations. Eventually, as Deep-learning based methods began to perform better for unstructured text data, more advanced development in Tesseract included a Deep-learning architecture together with a Long short-term memory (LSTM) network, a type of Recurrent Neural Network that is supported on Optical Character Recognition (OCR), text recognition, which was focused on the line text level recognition.

When using Tesseract OCR with Python, a standard used library is Python-tesseract, this library is packaging from Google's Tesseract OCR Engine in [14]. Furthermore, this recent architecture can use all image types supported by a well-known library in image processing called Pillow.

3.3.2. Pyspellchecker

Pyspellchecker is a dictionary that uses the Levenshtein Distance [15] algorithm to seek out permutations within an edit distance of two from the first word. After that, compares all multiple possible permutations, which include insertions if there is a lack of letters, deletions if it presents letters that are not needed, replacements when a change of a letter reduces the distance, and finally, transpositions which exchange two elements and keep all others fixed. Indeed, all of those word permutation changes are referenced and compared along with a

limited word frequency list that needs to be updated. In this frequency list, words have a score that represents when a word is found more often when a result word is more proper to appear in this e frequency list.

Adjacent to it, one of main the advantages that gives Pyspellchecker is that this library has support for different languages including English and Spanish, where other dictionaries only include English, and other languages are also available German, French, and Portuguese. Dictionaries used in this library were generated using the Frequency Word List Generator project, original idea from [16] that is supported on translations from an updated web page called OpenSubtitles. Additionally, this data is known to be reused by various open-source projects as Wikipedia among others.

3.3.3. Custom dictionary

Continuing the previous point, given that the project is also working with a language that not included in the previous algorithm, Catalan, we must carry out certain prior corrections with capitals, accents or signs, in addition to adding a custom dictionary working as an annex in order not to correct those words corresponding to proper names or the most common and frequent Catalan words that would be wrongly corrected by the Pyspellchecker module. In other words, a white list is going to be implemented to not erroneously autocorrect results.

4. Experimentation

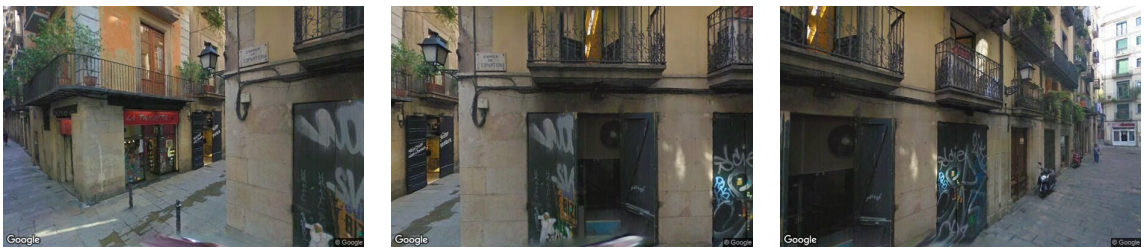
4.1. Results for Geolocation

Google Street View results analysis should be divided into two main sections, the first one is using the system as the opening step of the project, using the coordinates or the name of a street as a technique to obtaining pictures from a specific store with the objective to consecutively apply Optical Character Recognition to the data extracted.

The system has been fully implemented to perform this step, even included the recognition, but this last step has been tested with our own database following the same procedure as if it was obtained from Google Street View API due to the lack of quality. In the same way, we can send one request of a location or multiple locations simultaneously using different types of strings.

For example, we could use in the system different approaches: “Abaixadors10” a name of a shop, “Carrer dels Abaixadors, 10, 08003 Barcelona” the name of the same street and building number or “41.40252328588935, 2.169704480334489” coordinates, and all requests would be accepted equally and extracted as the exact same image data. In addition, even if the coordinates are not totally accurate the project transforms the coordinates and has set a threshold to find the closest street location where a Google Street View image is available, solving some mistakes that could be produced by phones GPS.

As explained in the methodology, when requesting this image we will not be able to obtain the relevant image that interests us directly as Google Street View API does not have a direction parameter that can be tracked because the camera position in their cars is random, heading 0° will not show the front image and heading 90° or 180° will not obtain the parallel images to the stores, for that reason we can only download the full view of 360° in pieces obtaining for example:



“Fig. 4.1.1. Images facing headings at 0°, 45° and 90° respectively”

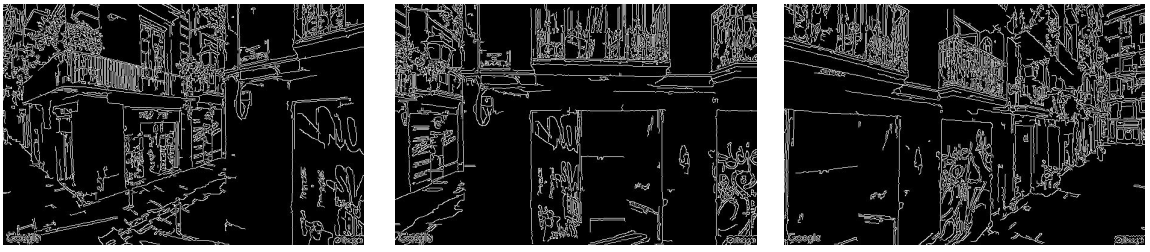
"Fig. 4.1.2. Images facing headings at 135°, 180° and 225° respectively"

As we can observe in the downloaded images, if the combination of headings from 0° to 360° taking samples every 45° and fovs are properly established at 90°, we can obtain the parallel pictures that are directly headed to the stores and if the fov is adequate there will be practically no distortion of the panoramic camera. In addition, by using this technique it is now possible to save the values obtained from the correct heading to know the direction where is actually facing the camera in the street and use this information to continue downloading data in the following (or previous) positions where data is available.

The first step to find where the direction is facing is applying a transformation to the image from RGB to gray, then the Canny edge detector [11] is used with a first threshold for the hysteresis procedure set to 50, this hysteresis procedure is used for edge points grouping, a second hysteresis threshold is also applied to search for initial segments of strong edges that is set to 150 and, finally, after applying an aperture of size 3 we can obtain the edges in the binary images that can be observed in figures 4.1.3.

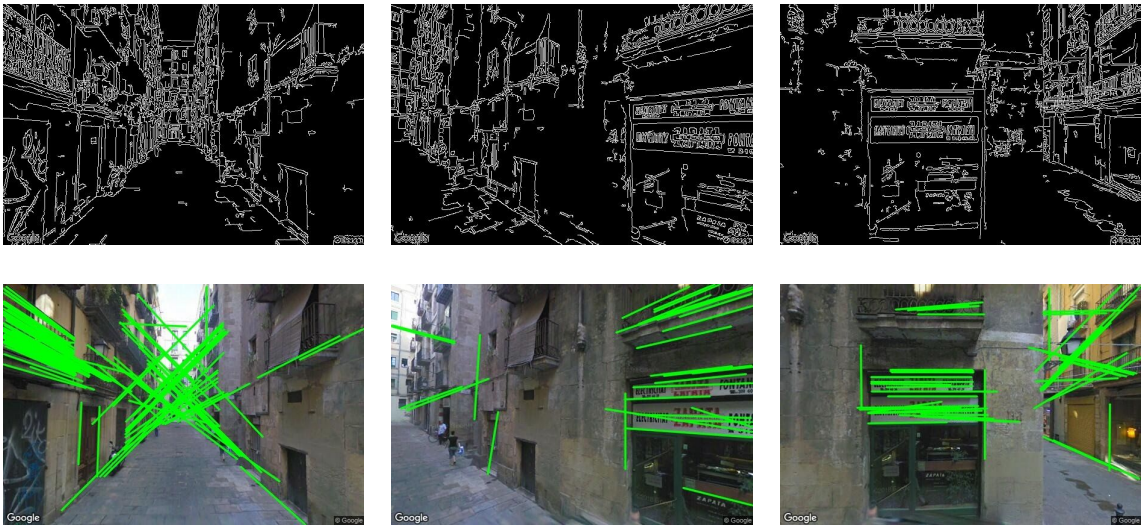
Once these edges are obtained, we can use the Hough Line Transform [12] to detect straight lines in a binary image, this method keeps track of the intersections between curves in the whole image, if some intersections are accumulated above a threshold fixed at 100, it identifies what corresponds to a line. Other values also have to be taken into account in this method as the minimum line length or the maximum line gap that the algorithm can consider as a line, which values are 100 and 10 pixels respectively.

Once this procedure is done, we will continue to discard the images that are facing diagonal or vertical lines because we are only searching horizontal edges, and we will approve the two images that contain a major number of thick edges, which will discard the image where is facing the Google Street View car.



"Fig. 4.1.3. Images obtained from using Canny edge detector facing headings at 0°, 45° and 90°"

"Fig. 4.1.4. Images obtained after applying Hough Line Transforms facing headings at 0°, 45° and 90°"



"Fig. 4.1.5. Images obtained from using Canny edge detector above and after applying Hough Line Transforms below, facing headings at 135°, 180° and 225° respectively"

On the other hand, as the project has finally tested the functionality with images from our own database and annotating manually a list of coordinates or street names could be labored, a feature that has been added is the automatic extraction of geodata from a full directory of images, so if the picture was taken with GPS location as the database was intentionally made, the whole directory can be analysed and obtained the correspondent Google Street View image.

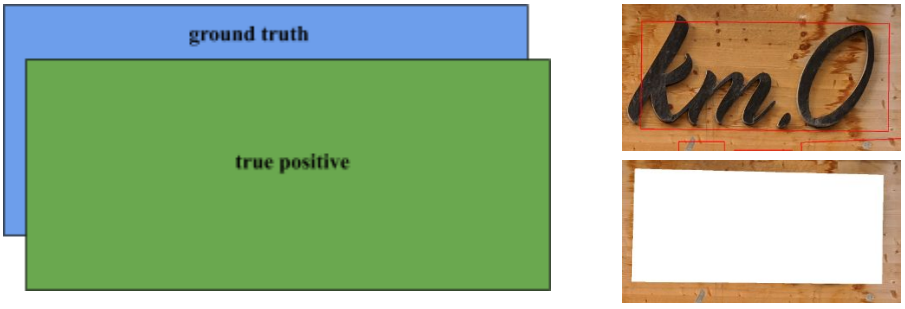
4.2. Metrics for Text Detection: Intersection over Union

Intersection over Union (IoU) [17] is an evaluation metric that is used when evaluating any algorithm that provides bounding boxes predicted as an output of an algorithm, for example, a text detection architecture, that can be calculated using this metric if we can compare the are within the box and a reference ground truth box. More precisely, if we want to apply IoU metric to evaluate an object detector we need basically two objects: The mentioned ground truth bounding boxes that are hand-labeled and the resulting predicted bounding boxes from our model, that will be obtained from either EAST or CRAFT.

IoU metric calculates a proportion of areas that involve: the area of overlap, which is the intersection rectangle between the ground truth and the predicted bounding box, and is after computed by dividing the overlap area by the sum of the prediction and the ground truth areas minus the intersection area, known as the area of union. To sum up, IoU is basically calculated by taking the area of overlap and dividing it by the area of the union as we can see in [17].

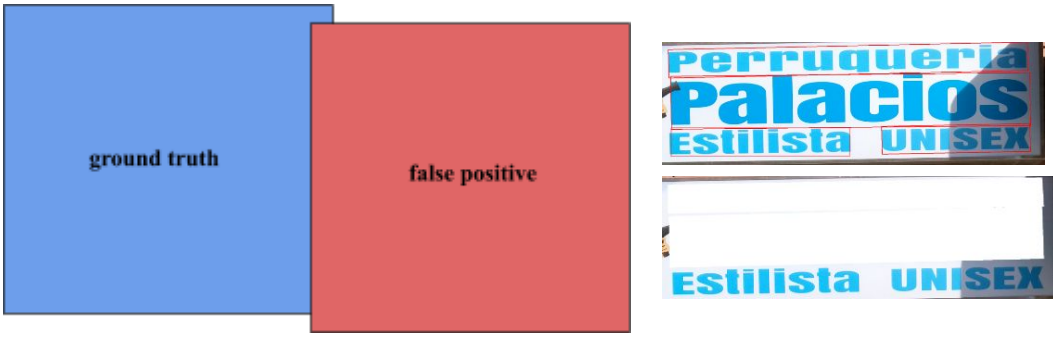
As we are taking and comparing each ground truth box with all the detected boxes, in the ground truth we could face a false positive from the box above. But on the other hand, if both ground truth are overlapped as we are manually drawing the ground truth boxes, it could not be discarded. so this is why the overlap value could lead to a miscalculation when counting true positives. In other words, we could consider a not relevant small box as a true positive.

As we can see, in the text recognition we face some issues where boxes are overlapping a common ground truth box. We can differentiate between four differentiated cases:



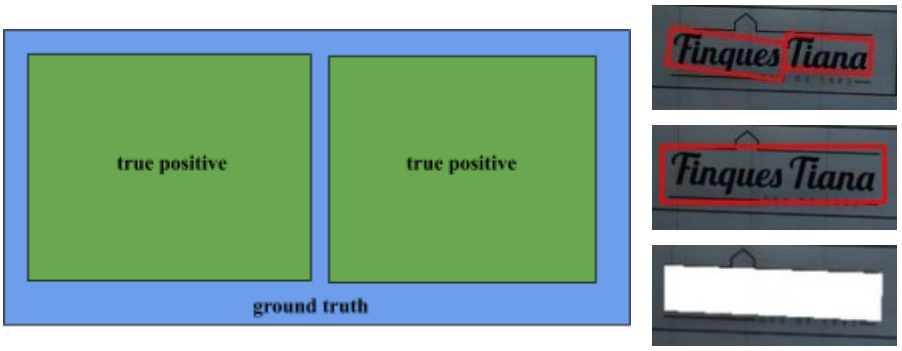
"Fig. 4.2.1. Graphics, bounding box and ground truth"

A correct detection as we can see in figure 4.2.1. is considered correct when our box includes at least a 50% threshold of the total area in our ground truth, in this case our metric will be designated as a high overlapping.



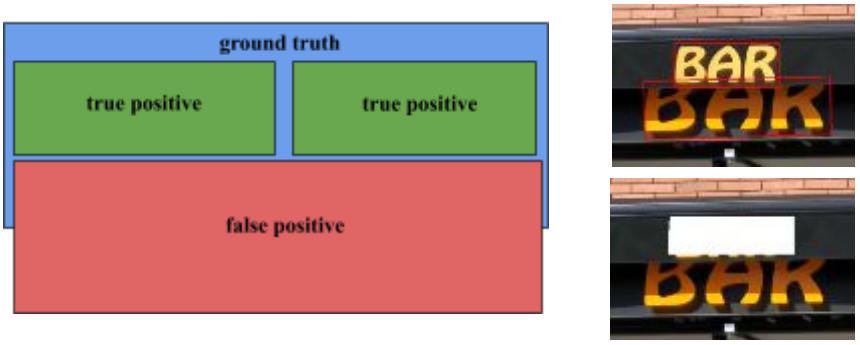
"Fig. 4.2.2. Graphics, bounding boxes and ground truth"

It is considered a correct detection when we are discarding boxes that are overlapped a 20% threshold or lower in relation to our total ground truth area. In figure 4.2.2. we see in red the false positive, it is a false positive because there is an overlap but the relation between both rectangles is much smaller than the total area of both of them separately.



"Fig. 4.2.3. Graphics, bounding boxes, merged bounding box and ground truth"

In the case shown in figure 4.2.3. we should include both detections as true positives but individually they do not surpass the IoU that is considered as a threshold 0.50. To solve this case it has to be added in the system a relation between both true positives where if both boxes are merged in one unique group they should surpass the true positive threshold. We can introduce this feature in the system by merging boxes that are placed at closed distances, taking both bounding boxes, and creating a new one than englobes both old boxes, using this method we will ensure that the new IoU will be surpassing the true positive threshold.



"Fig. 4.2.4. Graphics, bounding boxes and ground truth"

In this final case seen in figure 4.2.4. we can see that the red box associated with a false positive is in our ground truth and it could surpass our true positive threshold so it will not be discarded directly by the 20% are threshold. For this reason, we have to discard it using a new overlap parameter that involves the relationship between the area overlapped of the false-positive itself with the total area that it has, the systems considers that if less than the 50% of the box is inside our ground truth it will be discarded.

Once the results have been processed, we can assure that all our handmade ground truths are recognized by the system and taken into account some confidence thresholds. A high threshold or overlap value is set to 0.90 and a lower one to 0.20.

As a result, the system has 1864 true positives and zero false negatives from 218 images, those images have a total of 540 bounding boxes, where the obtained Recall is 1.00, this value means

all bounding boxes that we want are detected. In the other hand, as we are also detecting other text boxes that are not relevant for us, we have 2689 false positives, this means our system is detecting a bigger number of boxes that will be not used afterward, leading to a Precision 0.41 and F-score 0.58, this last value could be a worrying indicator in other types of Machine-learning detections but, in this case, it will not bring disadvantages to the final system.

EAST results from the whole annotated database:

· True positive = 240 · False negative = 0 · False positive = 802
· Precision = 0.23 · Recall = 1.00 · F1-score = 0.37

CRAFT results from the whole annotated database:

· True positive = 1864 · False negative = 0 · False positive = 2688
· Precision = 0.41 · Recall = 1.00 · F1-score = 0.58

The optimized parameters needed to obtain this CRAFT metrics are set to a text confidence threshold of 0.7 and a text low-bound threshold of 0.4. It also has to be mentioned the standards corresponding to the link confidence threshold that is set to 0.4.

4.3. Metrics for Text Recognition: Levenshtein distance

Levenshtein distance is a string metric for measuring the difference between two sequences, it is used in information theory, linguistics, and computer science. In summary, the Levenshtein distance between two words is calculated as the minimum number of single-character edits corresponding to insertions, deletions, or substitutions required to change one word into the other.

Following this procedure, we can analyze our handmade ground truth list of strings corresponding to each store with the list of strings obtained using Tesseract recognition in the output obtained from each detection, using EAST or CRAFT. Some quality standards that are in relation to the extension of each word. In addition, three different cases can be set:

Firstly, words with four letters or less can not present errors, it means the maximum Levenshtein distance is zero. Secondly, words between five and nine letters could have one error, meaning the maximum distance is one or less. Thirdly, large words that have ten or more letters can only present two errors. Finally, we assume other combinations can not be accepted as a good OCR so will be counted as misses. When taking into account these parameters, working in a word level with a total of 1032 annotated words.

EAST results from the whole annotated database:

· Hits = 302 · Misses = 711 · Accuracy = 0.30

CRAFT results from the whole annotated database:

· Hits = 435 · Misses = 597 · Accuracy = 0.42

Next, as we are working with different languages as Catalan that is not incorporated in the Tesseract, a solution could be to transform capitals to lowercase and remove accents. Additionally, a custom dictionary was added that incorporates a whitelist with 20656 Catalan words. Disappointingly, the module used to create a dictionary Pyspellchecker which returns the most probable result for the misspelled word requires an excessive increase in time and the statistics in the general output get worse.

EAST results from the whole annotated database:

· Hits = 274 · Misses = 758 · Accuracy = 0.27

CRAFT results from the whole annotated database:

· Hits = 339 · Misses = 693 · Accuracy = 0.33

In the other hand, as we are also interested to find the whole name of a shop, we can calculate metrics using the same standards in a sentence level. When working with sentences each manually annotated string will correspond to one image so the total database presents 626 sentences. Following the criteria explained previously, we can see now some differences.

EAST results from the whole annotated database:

· Hits = 164 · Misses = 462 · Accuracy = 0.26

CRAFT results from the whole annotated database:

· Hits = 200 · Misses = 426 · Accuracy = 0.32

As we can see, CRAFT tends to work with more bounding boxes and the text extracted from its features are better as the boxes are more accurate, the system also uses all permutations in the output to calculate the Levenshtein distance, meaning that probabilistically the results would be better.

Finally, if we try combining both bounding boxes that are obtained from EAST and CRAFT we can try solving the differences between methods and find a robust system. To sum up, as it was predicted in the following results, the combination of both methods let to a more compatible system that complements the results obtained from each architecture.

EAST & CRAFT results from the whole annotated database at word level:

· Hits = 475 · Misses = 557 · Accuracy = 0.46

EAST & CRAFT results from the whole annotated database at sentence level:

· Hits = 222 · Misses = 404 · Accuracy = 0.35

5. Budget

The software that has been used in the project includes open-source licences as the programming language Python and the libraries OpenCV, Pytorch, Pytesseract and Pyspellchecker. On the other hand, the library from Google Street View API requires a billing that works under a monthly pay-as-you-go model: Street View Static API costs $0.007 per image up to 1,000,000 images and $0.0056 per image till reaching 5,000,000 images, and Geocoding API costs $0.005 per request up to 1,000,000 requests and $0.004 per request till reaching 5,000,000 requests. Besides that, every month the platform provides a $300 usage free of charge that has not been surpassed during the project.

A hardware resource that is used from the Image Processing Group servers are the GPUs. It has not implied any cost because it was a research project, but we can compare it with a paid platform as Google Cloud to estimate the price. It is public that a NVIDIA® Tesla® T4 model that has 16GB, a similar model than the one that is used in our servers, is paid in the Londres server at $0.41 (0.36€) per hour. If the GPU has been used approximately 6 hours each week during 5 months, 43.2€ are spent in a server.

This project has been done during 20 weeks, it has included the learning hours, development hours and documentation. It is estimated that at least 20 hours of development and 1 hour of meeting have been employed each week, and a standard salary for a Junior Engineer in the Universitat Politècnica de Catalunya is 8€/hour. A total of 420 hours have been used during the project that correspond to a sum of 3360€ in salary and 3403.2€ in the whole project.

	Cost / hour	**Number of hours**	**Cost**
Server with GPU	0.36€/h	6h/week × 20 weeks = 120h	43.20€
Junior Engineer	8.00€/h	21h/week × 20 weeks = 420h	3360.00€
		Total	**3403.20€**

"Table 5. Calculation of costs with the final value of the total budget"

6. Conclusions and future development

6.1. Conclusions

As we have seen during the project, after creating an end-to-end system, the Google Street View section could be practically used as an easy, fast and automatic method to search the geolocation of a store that is in a public database with the objective to compare the results that are obtained from and OCR. This means that the idea behind this project is really possible to be applied in official institutions to check whether or not a store is in their records.

The project independently of the circumstances has a section where text detection meet all expectations and can obtain the text bounding boxes with right the regions of interest, the Recall of 1.00 lets us know that we will for sure get the information that we want to use afterward in next steps. In addition, the F-score has achieved 0.58, and in the first approaches was 0.10, which means that we are also taking more information that will be irrelevant to us but it is much less than what was expected. On the other hand, the next section where text recognition uses this bounding boxes data has not met the project confidence expectations in the same level that were achieved in detection, we want a final robust project to also perform geolocation and using our defined quality standard in the metrics it reaches a 46% of the words correctly recognized.

After obtaining these results, if we wanted to feedback the system using the application in another project, for example using the Google Street View after the text recognition, it could be affected as it would be needed an optimal string.

6.2. Improvements

This project could solve in a future implementation the unsolved section using high-quality images from Google Street View if the obtained data were to be in high resolution. First, the software implementation is already done to work with unscaled raw images in order to work with different types of resolutions, and secondly, OCR is already tested in our database using our geodata as if it was obtained from Google Street View.

Finally, the second development that could be done would be a dictionary that instead of using a whitelist for the Catalan language used a dictionary that already incorporated this language, it would improve the results as now the system is able to correct English and Spanish words but if it presents a Catalan one it can only compare if exists within a limited list. I believe this would be one of the main improvements as our manually made database is mostly in Spanish and Catalan.

Bibliography

[1] Gary Bradski. "The openCV library". *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". *UC Berkeley*, 2013. https://arxiv.org/pdf/1311.2524.pdf. [Accessed: 10 June 2020].

[3] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". [Online] Available: https://arxiv.org/pdf/1506.01497.pdf. [Accessed: 10 June 2020].

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollár. "*Focal Loss for Dense Object Detection". Facebook AI Research (FAIR)*, 2018. [Online] Available: https://arxiv.org/pdf/1708.02002.pdf. [Accessed: 10 June 2020].

[5] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". *University of Washington, Allen Institute for AI, Facebook AI Research*, 2016. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91. [Online] Available: https://arxiv.org/pdf/1506.02640.pdf. [Accessed: 10 June 2020].

[6] Paszke, A. et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". pp. 8024–8035. *Curran Associates, Inc.,* 2019. [Online] Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. [Accessed: 10 June 2020].

[7] Shangbang Long, Xin He, Cong Yao. "Scene Text Detection and Recognition: The Deep Learning Era". [Online] Available: https://arxiv.org/pdf/1811.04256.pdf. [Accessed: 11 June 2020].

[8] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, Jiajun Liang. "EAST: An Efficient and Accurate Scene Text Detector". *Megvii Technology Inc.*, 2017. [Online] Available: https://arxiv.org/pdf/1704.03155v2.pdf. [Accessed: 11 June 2020].

[9] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, Hwalsuk Lee. "Character Region Awareness for Text Detection". *Clova AI Research, NAVER Corp.*, 2019. [Online] Available: https://arxiv.org/pdf/1904.01941.pdf. [Accessed: 11 June 2020].

[10] Shangbang Long, Jiaqiang Ruan, Wenjie Zhang, Xin He, Wenhao Wu, Cong Yao. "TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes". *Peking University, Megvii (Face++) Technology Inc.*, 2018. [Online] Available: https://arxiv.org/pdf/1807.01544.pdf. [Accessed: 12 June 2020].

[11] Jorge Valverde-Rebaza. "Detección de bordes mediante el algoritmo de Canny". *Escuela Académico Profesional de Informática Universidad Nacional de Trujillo,* 2007. [Online] Available: https://www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny. [Accessed: 27 June 2020].

[12] Richard O. Duda and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures". *Commun. ACM 15, 1,* 1972. [Online] Available: https://doi.org/10.1145/361237.361242. [Accessed: 28 June 2020].

[13] Rasmus Rothe, Matthieu Guillaumin and Luc Van Gool. "Non-Maximum Suppression for Object Detection by Passing Messages between Windows". *Computer Vision Laboratory, ETH Zurich, Switzerland*, 2015. LNCS. 9003. 10.1007/978-3-319-16865-4_19.

[14] R. Smith, "An Overview of the Tesseract OCR Engine". Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Parana, 2007, pp. 629-633, doi: 10.1109/ICDAR.2007.4376991. [Online] Available: https://ieeexplore.ieee.org/document/4376991. [Accessed: 12 June 2020].

[15] Haldar, Rishin and Debajyoti Mukhopadhyay. "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach." Web Intelligence & Distributed Computing Research Lab Green Tower, 2011. [Online] Available: https://arxiv.org/ftp/arxiv/papers/1101/1101.1232.pdf. [Accessed: 19 June 2020].

[16] Jorg Tiedemann. "Finding Alternative Translations in a Large Corpus of Movie Subtitles". *Department of Modern Languages, University of Helsinki*, 2016. [Online] Available: http://www.lrec-conf.org/proceedings/lrec2016/pdf/62_Paper.pdf. [Accessed: 15 June 2020].

[17] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid and Silvio Savarese. "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression". *Computer Science Department, Stanford University, United states, School of Computer Science, The University of Adelaide, Australia, Aibee Inc, USA*, 2019. [Online] Available: https://arxiv.org/pdf/1902.09630.pdf. [Accessed: 10 June 2020].