Birzeit University

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

Perception-To-Action: Visually Guiding a Robot Gripper to Grab Objects Using OpenCV Library under ROS Operating System

Prepared by:

Sondos Araj	1172290
Wafaa Foqahaa	1172108
Yara Khourieh	1171564

Supervised by:

Dr. Hanna Bullata

A graduation project submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer System Engineering

BIRZEIT

June – 2022

Abstract

In this report, we will describe the manipulator part of the Librarian Robot which is a mobile robot built for Birzeit University library. Its main purpose is to help the librarians and students in Birzeit University for searching about the desired book by capturing the bookshelves which contain the book and extract the book information such as title, author name, year of publication, etc. from the image that the robot camera capture. Then, if the desired book is found, the robot gripper will grab the book and put it in a box.

The robot uses ROS operating system along with graphical software applications like GAZEBO and RVIZ to implement library environment and take advantage of simulated sensors like laser sensor, pressure sensor.

The main emphasis in this report is on the development of an algorithm to detect the boundaries of books and extract text from them.

نبذة مختصرة

سنقوم في هذا التقرير بوصف جزء المناور من الروبوت المكتبي وهو عبارة عن روبوت تم إنشاؤه لمكتبة جامعة بيرزيت؛ الهدف الأساسي منه هو مساعدة أمناء المكتبة والطلاب في جامعة بيرزيت للبحث عن الكتب التي يحتاجونها بشكل أسرع وأسهل. يقوم هذا الروبوت بالتقاط صورة لرف الكتب المتوقع بأنه يحتوي الكتاب المطلوب باستخدام الكاميرا الخاصة به، ومن ثمّ يعمل على معالجة هذه الصورة للقيام بعزل كل كتاب على حدا، ثم استخراج معلومات الكتاب مثل العنوان واسم المؤلف وسنة النشر وما إلى ذلك من صور الكتب المعزولة. بعد ذلك، إذا تم العثور على الكتاب المطلوب، سيقوم الروبوت بسحب الكتاب باستخدام مقبضه ويضعه في صندوق الكتب.

يستخدم الروبوت نظام التشغيل ROS جنبًا إلى جنب مع تطبيقات البرامج الرسومية مثل GAZEBO و RVIZ لمحاكاة بيئة المكتبة للاستفادة من أجهزة الاستشعار المحاكاة مثل مستشعر الليزر ومستشعر الضغط.

ينصب التركيز الرئيسي في هذا التقرير على تطوير خوارزمية لتعيين حدود الكتب لفصل كل كتاب على حدا من صور أرفف الكتب وبالتالي استخراج النص منها.

Table of Content

Acronyms and Abbreviations

ROS	Robot Operating System
RVIZ	ROS Visualization
OCR	Optical Character Recognition
UJI	University Jaume-I
DOF	Degree Of Freedom
MEGA-D	Megapixel Digital Stereo Head
HCI	Human-Computer Interface
RGB	Red Green Blue
FSA	Finite State Automation
PR	Projection of Rectangle
MWIS	Maximal Weighted Independent Set
CNN	Convolutional Neural Networks
R-CNN	Region-Based Convolutional Neural Networks
ROI	Region of Interest
RPN	Region Proposal Network

List of Figures

Chapter 1

Introduction

Robotics is a field that combines science, engineering, and technology to make machines called robots that do (or mimic) human tasks [1]. Robots are changing the world in mostly positive ways. They may be taking over some human jobs, but they also create better efficiency. This in turn boosts economic activity, which generates more opportunities for humans to generate income. Also augment physical abilities and serve in areas where interaction with people is needed [2].

1.1 What is a robot

One can define a robot as it is "*a product of the robotics domain, which involves the creation of programmable robots that can aid humans or duplicate their operations"* [1]. Other would say a robot is *"a physically embodied artificially intelligent agent capable of acting in ways that affect the physical world"* [3]. So, there is no specific definition of the term "robot", but simply it can be defined as it is a self-contained machine capable of sensing its surroundings, performing calculations to make judgments, and acting in the actual world [4].

1.2 Robot Components

Robots are designed to meet a wide range of demands and serve a number of functions, and as a result, they require a wide range of specialized components to execute these duties. However, three main components are essential to the creation of every robot which are mechanical construction, electrical components and some level of computer programming. In general, robotics components can be divided into five categories [1]:

- **Control system:**

The level of computer programming that each robot need is known as robot's central processing unit or robot's control system. Control systems are programmed to teach a robot how to use its specialized components in order to execute a job, similar to how the human brain transmits signals throughout the body [1].

- **Sensors:**

Sensors are the main components of robots. They offer inputs to a robot in the form of electrical data, which the controller processes and allows the robot to interact with the outside environment [1].

- **Actuators:**

Only a device with a moveable frame or body may be considered a robot. The components that cause this movement are known as actuators. These components are made up of motors that receive signals from the control system and work together to carry out the movements required to perform the task. Actuators are constructed of a number of materials, including metal and elastic, and are often powered by compressed air or oil, although they come in a variety of configurations to best perform their particular duties [1].

- **Power Supply:**

Power is necessary for robots, just as it is necessary for the human body to function. Power supply sources differs according to robot types, work, characteristics and other reasons, but mainly two types of power supply sources are used. Stationary robots, such as those seen in factories, may be powered by an AC outlet, while other robots are more typically powered by an internal battery [1].

- **End Effectors:**

End effectors are the physical components that allow robots to complete their jobs. They are usually external [1].

1.3 Types of Robots

Mechanical robots differ from each other in design, functionality and degree of autonomy. Generally, there are five types of robots:

- **Pre-Programmed Robots**

  Pre-programmed bots are notified of their functions in advance because they operate in a simple and controlled environment as well as perform a simple and specific task so that they do not need artificial intelligence to function well. The most famous example is the mechanical arm on an automobile assembly line. The arm performs a specific function such as welding car doors or inserting a specific part into the engine [1].

- **Humanoid Robots**

  As the name suggests, they are robots that resemble and/or mimic human nature, so they are expected to perform human-like behaviours such as running, jumping or even carrying objects, etc., as they work with humans as well as help people with physical disabilities. Two examples of this are Atlas Sophia and Boston Dynamics for Hanson Robotics [1].

- **Autonomous Robots**

  Autonomous robots operate independently of human supervision because they are designed to operate in open environments. These robots are intelligent machines that can detect the world around them through sensors. Depending on the data obtained from the sensors and their mission, these robots make their decision to move to the next optimal step by employing decision-making structures (such as a computer). An example of this type of robot is the Roomba vacuum cleaner, which uses sensors to roam around the house freely [1].

- **Tele-operated Robots**

  These robots are controlled remotely by a human. There are many types of robots of this family, such as the robotic arm found in the space shuttle. The underwater robots that helped solve the oil spill in the Gulf of Mexico are also remotely operated robots. Perhaps the most famous type of these robots will be medical robots in the near future. These robots are controlled by a surgeon or a doctor, so we are happy to say that your doctor can be present remotely in your town, even if you are in China [5].

- **Augmenting Robots**

  Augmented robots contribute to enhancing human capabilities, and moreover, they can replace capabilities that humans may lack. These robots can turn science fiction into reality through the field of human augmentation, as these robots can redefine humanity to make humans faster and stronger. An example of these robots currently in use is robotic prosthetic limbs used to lift heavy weights [1].

1.4 Librarian Robot

Librarian robot is a movable and helpful robot that works in the library to assist the librarian and the customer is bringing from or returning books to shelves. It accepts the customer’s/librarian's request for a book, checks if it exists in the database, and if it does, searches the library for a quick path to that book type's location. After that, it begins reading book titles from the beginning of the shelf until it finds the required one, at which point it takes up the book with a gripper and places it in a basket to be returned to the customer.

1.5 Project Goal

Our project main goal is to build a gripper (software part) for a librarian robot manipulator that will grab a book and put it in a box. A simulated environment under Robot Operating System ROS will be built to do such work. Mainly, it will read the book spine using camera after identifying the book boundaries on the shelf, furthermore it will extract the book title using Optical Character Recognition software available such Tesseract OCR Engine. After that, if the right book is found, the coordination perception-action must take place until the book is grabbed and put in a box with the assistance of some sensors such as a distance sensor, and a pressure sensor.

1.6 Report Organization

The report is organized in six chapters and the rest of this report is organized as follows:

- Chapter 2 presents a literature reviews about librarian robot system description, gripping process, what is a gripper, how it works, exposes some librarian robots' projects already developed and created and also exposes some book boundary detection algorithms.
- Chapter 3 introduces the concept of text extraction with Tesseract OCR Engine with OpenCV library and explains how it works. It also presents image pre-processing techniques and displays what changes they do on an image, and presents some boundary detection algorithms.
- Chapter 4 shows a new technique used these days in book boundary detection using AI and Deep learning.
- Chapter 5 shows some simulated results obtained from using deep learning model and text extraction python code.

The last chapter contains a conclusion of the work presented in this project and draws up some possible future works.

Chapter 2

Literature Review

These days, there is a lot of research on autonomous and semi-autonomous robots. Some authors classified this type of robots as service robots. The International Federation of Robots indicates that service robots are for the welfare of humans, with the exception of industrial robots. Service robots are divided into two types: professional service robots and personal service robots. These robots perform many functions such as assisting with housework, searching and monitoring, assisting people with special needs, and most importantly, these robots are used for dangerous or impossible manual tasks [6].

Research is still active in the field of robotics, and many applications have been presented to date. But robots designed to perform library tasks are still relatively rare. In this context, an autonomous solution for librarian robots is presented; this solution aims to extract a book requested by a user from the bookshelf if this book is located [7].

2.1 Librarian Robot Overall System Description

In accordance with the aforementioned objective, here is a summary of the functions of the system of any librarian robot in general with an emphasis on the manipulation strategy [7].

- To allow the user to interact with the system, a "User Input" service is provided, whereby the system can receive voice commands from the user and then translate them to the corresponding label using the system's databases.
- In order to search for the desired book on the bookshelves, the system provides a "scanning mechanism". This mechanism uses Computer Vision, specifically Optical Character Recognition (OCR) technology, to process the label of any book on the shelf (expected to contain the requested book).

- “Success?” Each time a book label is identified, a comparison is made between it and the label requested by the user. This comparison continues until the process is successful, at which point the system executes the "Grasping Module".
- When the desired book is found, the system provides a "Grasping Module" responsible for autonomous manipulation of the book. In order to achieve this goal, it needs processing force and visual sensory information.

The following flow chart shows the above system functionality:

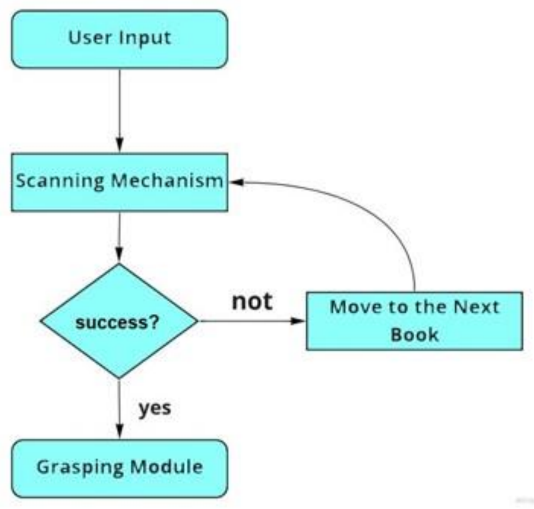


Figure 2-0-1: System functionality flow chart

2.2 Gripping Operation Analysis

Gripper design depends on several factors, the most important of which are the maximum book size and weight and minimum thickness, in addition to the maximum shelf depth. The book grasping process involves two main steps: picking up the book and repositioning it [6].

To pick up the book, the hand approaches the book and tilts it around its base using one of the fingers (usually the forefinger). Once this happens, the other fingers grab the book and then pull it out. However, the process of repositioning the book is done by approaching the gripped book from the shelf, in this case the edges of the lower corner are the first to be placed and then the body of the book is pushed into the shelf and readjusted in a vertical position [6].

We use the UJI Robot as a use case of a librarian robot that has grippers and we will discuss it in details.

2.3 UJI Robot

The UJI (i.e. the acronym of the University Jaume-I, Castellón, Spain) librarian robot is a robot designed to help with daily tasks and is considered a service robot that contains a prototype mobile manipulator. At first, it employed as an assistant in public libraries looking for books on the shelves, collecting them and transporting them to a certain place [6].

The demo setup, shown below, requires a camera in hand (MEGA-D Megapixel Digital Stereo Head) configuration, a robot arm (Mitsubishi PA-10, 7 DOF parallel jaw gripper), and a mobile platform (Nomadic). In addition, it is necessary to make a special design for the fingers in the parallel-jaw gripper [6].

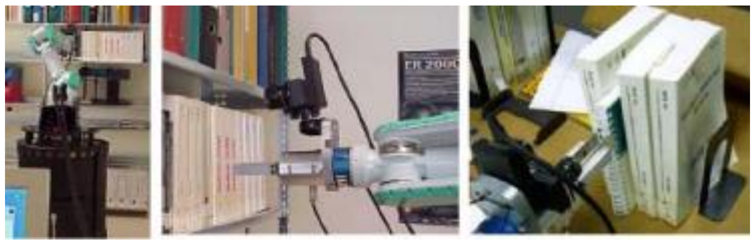


Figure 2-0-2: The system in action. (M. Prats et al., 2004)

All tests of this system were carried out with a 266 MHz Pentium II Processor. Initial tests showed good performance, for example, it takes less than a second to localize a label on an image. Furthermore, it always needs 5-10 seconds to the next image processing, including OCR (Optical Character Recognition), to identify the label. In addition to all of this, we need a ratio of 5 seconds per centimeter of the book for gripping it, that is, a book of 4 cm thickness takes 20s [6].

Recently, recent experiments have shown that the system is powerful enough to process non-conventional books, as presented in Figure 2-2. These experiments used another mobile platform (ActivMedia PowerBot) and a 1.2 GHz Pentium IV Processor. These modifications showed a significant improvement in terms of book locating, identifying and extracting. Now, we need 1.5 seconds for every 1 cm of the book for gripping it, that is, a book of 4 cm thickness takes 6s [6].

The researchers placed tactile sensors on the robot's fingertips and placed it into a real application: a robot pulling a book from a bookshelf. They monitored whether the robot could assess whether the initial contact was good in order to test force/pressure control strategies [6].

They found that the robot could autonomously guarantee a good initial contact at the top of the book even from a very initial approximation [6].

Figure 2-3 presents three behaviors applied to the robot, the first of which controls the force that the manipulator applies to the book through the contact at the fingertip, and the second tries to maximize the contact area in order to ensure that the reference force is reached. The third and final behavior is extracting the book [6].

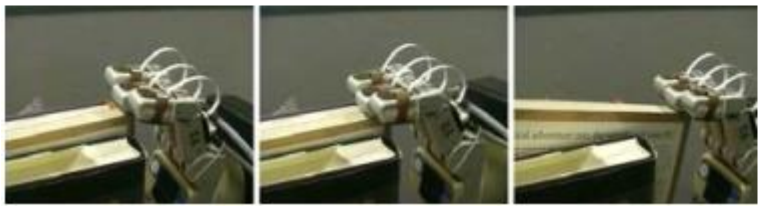


Figure 2-0-3: The robot grasping the book. Left: initial contact. Middle: Maximizing contact surface. Right: doing the task. (A. Morales et al., 2007)

The researchers confirmed that there will be more actions that will be applied to the gripper so that the UJI robot can handle objects of different shapes [6].

2.4 Model-based Book Boundary Detection

To detect the boundaries of each book in the pictures captured by the robot camera, there is a model-based book boundary detection technique. For improved detection accuracy, this technique employs a bookshelf model. The model is intended to describe the structural aspects of bookshelves and is represented as a finite state automation (FSA), with each state representing a component of bookshelves, such as the boundary and title [8] [9].

Along with this model, the issue of detecting book boundaries is described as a model fitting problem to identify the state at each horizontal location. Finally, the horizontal positions identified as the boundary state are recognized as the locations of book boundaries. Furthermore, the local slant angle at each horizontal location is best predicted for dealing with non-uniformly slanted books [8] [9].

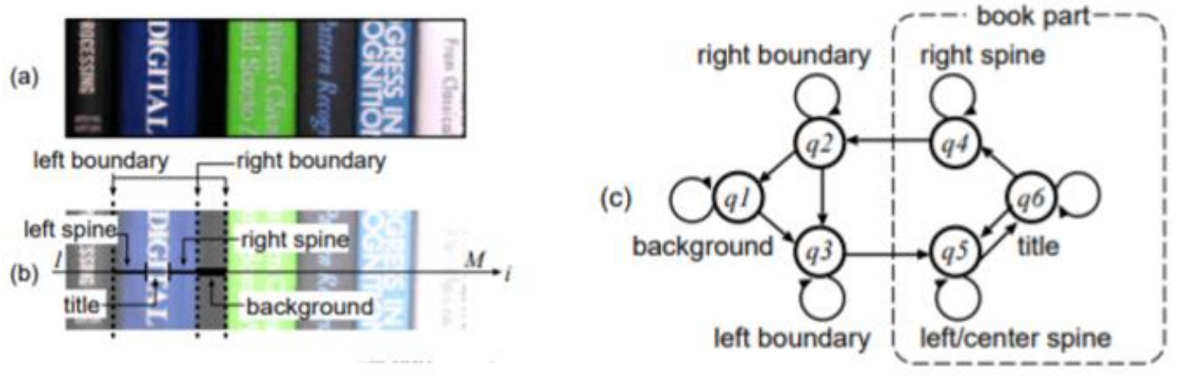


Figure 2-0-4: (a) A bookshelf image. (b) The components of (a). (c) An FSA representation of (b).

Thus, the problem of detecting book boundaries is described as a model-based optimization based on both the state and the angle at each horizontal location [8] [9].

- **Algorithm:**

As we can see in Figure 2-4 (a), it shows a bookshelf image taking from the middle part($M(width) \times N(height)$), doing this to ignore the difference in the heights of the books and the bookshelf boards [8] [9].

The concept of this algorithm is to optimally estimate the component at each horizontal position of a bookshelf image such as Figure 2-4 (a). As shown in Figure 2-4 (b), bookshelf images have several components, such as: boundary, spine, title, and bookshelf background and the order of those components is naturally governed by some rules [9].

The rules can be represented by an FSA model composed of 6 states ($Q = \{q_1, q_2, q_3, q_4, q_5, q_6\}$) as shown in Figure 2-4 (c). With this model, the estimation problem is expressed as a model-based optimization problem of the sequence $s_1, \ldots, s_i, \ldots, s_M$ where $s_i \in Q$ denotes the state of the component at horizontal position i [9].

According to Figure 2-5 (a), the books are non-uniformly slanted. For coping with non-uniformly slanted books, the slant angle at each horizontal position should be estimated optimally. This estimation issue can be expressed as an optimal estimation problem of the sequence $p_1, \ldots, p_i, \ldots, p_M$ [9].

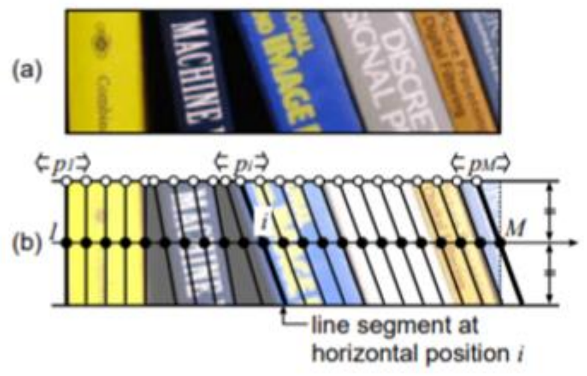


Figure 2-0-5: (a) Non-uniformly slanted books. (b) A sequence of line segments representing the slants of books.

As shown in Figure 2-5 (b), $pi \in [i - W, i + W]$ is the horizontal position of the top end of the line segment which comes through the center of the i_{th} column, where W is a positive integer to specify compensable slant angles. For practical simplicity, we use pi instead of some real-valued angle [9].

The following maximization formula represents these problems:

$$MAX\ (\sum_{i=1}^{M} f_i(s_i, p_i))\ ,\ w.r.t.\ s_i, p_i \qquad (1)$$

Where $f_i(s_i, p_i)$ is a criterion function to evaluate $p_i$and s_i at horizontal position$i$. The maximization formula undergo to the transition rule of the FSA model. The following equation employed to limit the interval between p_i and p_{i-1}:

$$p_i = \begin{cases} p_{i-1} + 1 & if\ s_i \in \{q_4, q_5, q_6\} \\ & or\ s_{i-1} \in \{q_4, q_5, q_6\}, \\ p_{i-1} + \{0,1,2\} & otherwise \end{cases} \qquad (2)$$

By (2) constraint, any fluctuation at the angle is not allowed if $s_i \in \{q_4, q_5, q_6\}$ or $s_{i-1} \in \{q_4, q_5, q_6\}$, which means that each book spine region has its fixed slant. Otherwise, angle fluctuation is allowed with a limited degree at boundaries and bookshelf background parts [8] [9].

The criterion function $f_i(s_i, p_i)$ is designed based on the following observations:

- Long contiguous edges with near-vertical directional feature are often detected around book boundaries.
- Most edges with near-horizontal directional feature are detected around the title characters and the illustrations on book parts.
- Edges are rarely detected in bookshelf background parts and book spine parts.

2.5 Viewpoint-Independent Book Spine Segmentation

Lior Talker proposed this method for precisely segmenting books on bookshelves in images taken from general viewpoints. The proposed segmentation algorithm overcomes challenges caused by book spine text and texture, different book orientations under perspective projection, and book proximity. As a first step, a shape dependent active contour is used to generate a set of book spine candidates [22].

Books can be oriented differently, and the image can have perspective distortion. The visible part of a book on a bookshelf is the spine, and the rest of the book is usually occluded. As a result, this method significantly decreases the task of book segmentation to the segmentation of their spines (see Figure 2-6) [22].

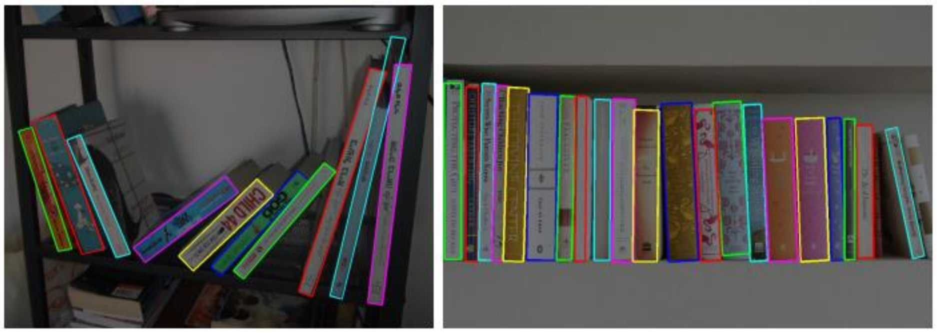


Figure 2-0-6: Book spine segmentation results

There are two phases in this method. The first allows us to generate a set of spine candidates with a high number of false positives but a low number of false negatives. The set of spine candidates is filtered in the second phase using spatial constraints [22].

The first phase involves modifying the active contours paradigm to include a predefined contour shape, a rectangle perspective projection - PR. All book spines are assumed to be coplanar or lie on parallel planes (identical 3D normals), but they may have different orientations within these planes. As a result, while each spine's edges are consistent with a pair of orthogonal vanishing points, different spines may have different vanishing points. All spines, however, have the same vanishing point. A PR's shape is constrained by these observations [22].

In the second phase, the set of detected spine candidates is filtered based on size statistics and relative spatial location. The selection problem is specifically formulated as the maximal weighted independent set (MWIS) of a graph that corresponds to the spatial relations between the spine candidates [22].

Book Spine Segmentation

This approach takes as input an image of books on bookshelves. All book spines are assumed to run parallel to a plane π. The result is a group of spines, each of which is a perspective projection of a rectangle (PR). The procedure of this method is divided into two stages. The initial step is to identify a set of PR candidates that match the image gradients and expected perspective. The second step is to filter this set based on the location, relative size, and spatial relations of the candidates [22]:

1 Identifying PR Candidates

The authors employed a method to identify a set of PRs that are candidates for book spine segmentation. The PR detection problem is expressed as a five-parameter minimization problem [22].The identification of the PR candidates goes through several stages:

- PR Parameterization
- PR Expansion
- Orthogonal Pairs Of Vanishing Points

2 PR Selection

The computed set of PRs includes the desired book spines as well as a large number of additional PRs which should be discarded. PRs that segment numerous books together, sub-regions of a spine, other objects in the scene, or simply noise are examples. Furthermore, many PRs may segment nearly the same region of the image. Using the assembly of books, the proposed algorithm was utilized to choose the set of PRs that most likely represented book spines [22].

Chapter 3

Text Extraction from Images Using OpenCV and Tesseract Engine

This chapter talks about the concept of extraction text from images using OpenCV library with Tesseract Optical Character Recognition Engine, how it works and what prerequisites needed like image pre-processing and image boundaries detection.

3.1 What is OpenCV

As well as humans can see with eyes, detect objects and track motion, computers nowadays have the ability to detect objects, take pictures, understand and manipulate them which is known under the term of Image Processing using the process of Computer Vision. OpenCV which stands for Open Source Computer Vision is a huge open source library that contains more than 2500 algorithms used in Computer Vision Applications in areas powered by Artificial Intelligence or Machine Learning algorithms, and for completing tasks that need image processing. OpenCV has a lot of application that are for both academic and commercial such as object detection, ego motion estimation, gesture recognition, human-computer interface (HCI), and motion understanding and tracking. Using OpenCV, we can process images and videos to identify objects, faces, or even the handwriting of a human [10].

3.2 Tesseract OCR Engine

OCR or Optical Character Recognition is the process of picking up printed or handwritten text from a two-dimensional image and convert it into machine-readable text. To execute as accurately as feasible, OCR usually comprises of the following sub-processes [11]:

- Preprocessing of the Image
- Text Localization
- Character Segmentation
- Character Recognition
- Post Processing

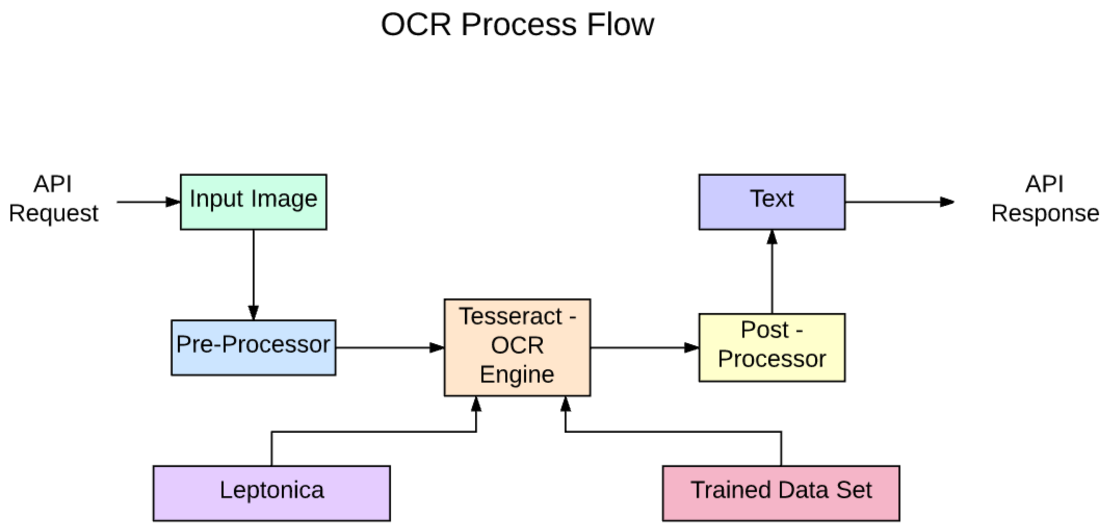


Figure 3-1: OCR Process Flow

There are many OCR engines that have been developed and used over the years such as OCRopus, Ocular and SwiftOCR, but the most popular and used one is the Tesseract OCR Engine. It is an open-source OCR engine available under the Apache 2.0 license that began as a Ph.D. research project in HP Labs, Bristol [11].

Tesseract engine does not have a built-in GUI, it can be used directly through command line or using an API. It is compatible with many programming languages and frameworks like C++, Python, Windows, Linux, and Mac [11].

For Python language, Tesseract OCR Engine has a special wrapper that is called Python-tesseract or Pytesseract that recognizes and “reads” the text embedded in different types of images such as jpeg, png, gif, bmp, tiff, and others [11].

3.3 Image Pre Processing

Tesseract Engine and Pytesseract have some limitations where it can drop the output of the OCR operations if the input image is not perfectly taken or has noise on the background or has bad size or contrast or lightning [11]. So to avoid this from happening, image pre-processing can help to make changes on the original image which will help in obtaining better OCR output. There exist various image pre-processing techniques that are used for different aims and in different situations such as:

- **Rescaling**

  Rescaling or called Gray-scaling is converting a full color image that has different red, green and blue components values in RGB color space to a gray-scale image containing only shades of gray and no color. In gray color images, the red, green and blue components all have equal intensity in RGB color space. So, in contrast to a full color image, where each pixel requires three intensities, each pixel in a gray-scale image just requires a single intensity value that is stored as an 8-bit integer giving 256 possible different shades of gray from black to white [12].

- **Binarization (Thresholding)**

  Binarization or called Thresholding technique is to turn a gray-scale (multi-tone) image to a black-and white (two-tone) image by first determining the gray-scale threshold value then deciding whether or not a pixel has a certain gray value. This means that if the gray value of the pixels exceeds the threshold, they are transformed to white; else they are transformed to black, and this is the simplest thresholding method [13]. Thresholding can be used to create binary images from gray-scale images.

Moreover, there exist a lot of thresholding methods that are categorized into six groups where each of them selects the threshold value differently, and each of them has multiple types of thresholding such as histogram, clustering, entropy and others [14]:

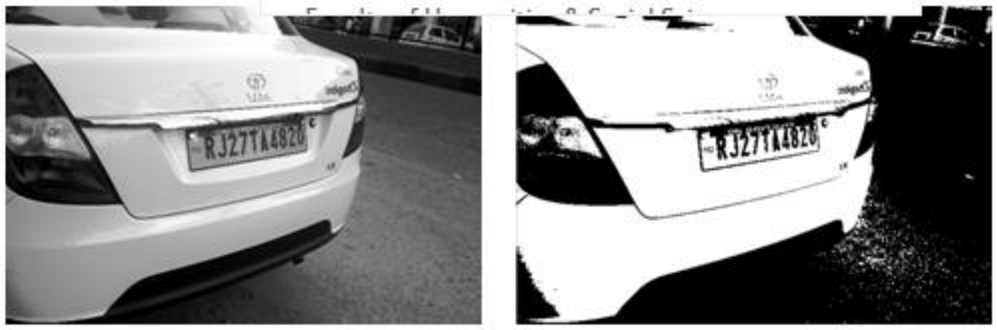


Figure 3-2: Binarization example

- **Noise Removal**

During image acquisition, coding, transmission or processing steps, images can be influenced by noise that could hit any pixel of the image and change its value. To remove this noise and deal with better images, noise removal a.k.a. image filtering method is applied, where filters are used to remove noise from images by maintaining the same details. There are different techniques of filtering and various types of filters that are chosen according to the filter behavior and type of data. The simplest way of removing noise is by replacing the value of each pixel with average of pixel around it. Gaussian Filter, Mean Filter, Median Filter and Bilateral Filter are some of many types of filters [15].

- **Morphological Operations**
    - **Dilation**

In the field of Morphological image pre-processing operations, Dilation is one of the two basic ones, the other being Erosion. Dilation is an operation typically applied to binary images gained from Thresholding that gradually expands the boundaries of foreground pixel areas while holes within those regions become smaller [16]. In other words, dilation make objects in binary images thicker based on a matrix of 1's and 0's called structuring element. This method can help in narrowing or connecting gaps in images, which is sometimes useful in retrieving important details in images for example retrieving correct letters.

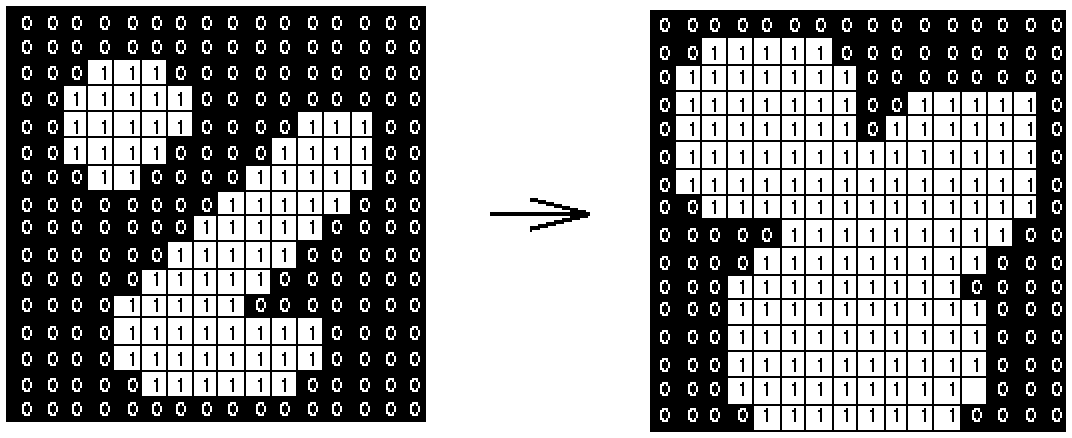


Figure 3-3: Dilation Example

- **Erosion**

Erosion is the reverse operation of dilation in morphological processing. It erodes away the boundaries of areas of foreground pixels, and holes within those areas become larger [17] and one of the benefits of this method is removing undesired details in images.

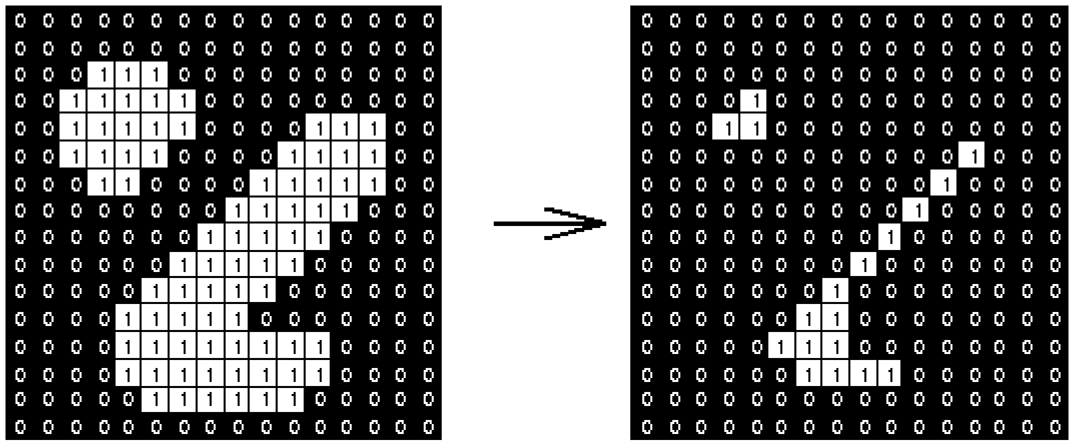


Figure 3-4: Erosion Example

- **Opening and Closing**

By mixing the two morphological operations together; erosion and dilation, new image processing techniques gained which are opening and closing. In opening, the erosion operation is performed first and then the dilation takes place. On the contrary, closing is applied by first perform the dilation operation followed by the erosion operation.

- ❖ Opening removes the thin protrusions of the obtained image.
- ❖ Opening is used for removing internal noise of the obtained image.
- ❖ Closing erases the small holes from the obtained image.
- ❖ Closing is used for smoothening of contour and fusing of narrow breaks.

- **Contour Filtering**

Contours can be defined as a curve that connects all continuous points (along the border) that have the same color or intensity. Contours are an excellent tool for shape analysis as well as object detection and recognition [17].

For better accuracy, the contours filtering process should be after the binarization of the images. So, apply threshold or canny edge detection before searching for contours [17].

Finding contours in OpenCV is like to finding a white object against a black background. So keep in mind that the object to be found should be white and the background should be black [17]. So, if the object in the image after binarization was black, the image should be inverted to make the object white and the background black.

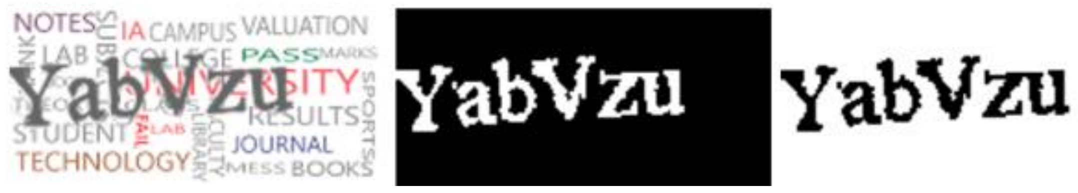


Figure 3-5: Contour Filtering Example

Chapter 4

Book Boundary Detection

Several methods based on bookshelf image analysis have been investigated for the purpose of automating bookshelf inspection in libraries and bookstores. A book boundaries detecting method must be used for the images of the bookshelves in the library to guarantee that the required book is easily found. After the camera has captured a photo of a bookshelf, the initial step is to recognize its boundaries. Detecting the boundaries of each book in the image simplifies the text extraction process [17].

In traditional systems, book boundaries are initially detected in order to extract individual books using a common line detection approach, such as the Hough transform. Their detection accuracies, however, are not always enough [17].

The latest technology in image segmentation is Mask R-CNN. Mask R-CNN is a convolutional neural network that detects the objects in an image and gives a high-quality segmentation of each of them [23].

In this context, the stages of development of Mask R-CNN will be presented, starting from Convolutional Neural Networks (CNN) [23].

4.1 What is a Convolutional Neural Network (CNN)?

A convolutional neural network is the building block in the task of computer vision for image recognition, processing, and segmentation. This network consists of three basic layers:

1 Convolutional Layer: This layer uses filters and the kernel to abstract the input image into a feature map.

2 Pooling Layer: This layer summarizes the presence of features in the feature map resulting from the previous layer to help to down sample the feature maps.

3 Fully Connected Layer: This layer connects every neuron in one layer to every neuron in another layer.

The combination of the previous layers contributes to recognizing and identifying the desired object in the image. However, to detect single object in the image, simple convolutional networks are designed, but these networks are not suitable in complex cases, such as the presence of several objects in the image, and in this case, Mask R-CNN is used, which is a developed environment based on R-CNN [23].

The following Figure shows how a convolutional neural network works.

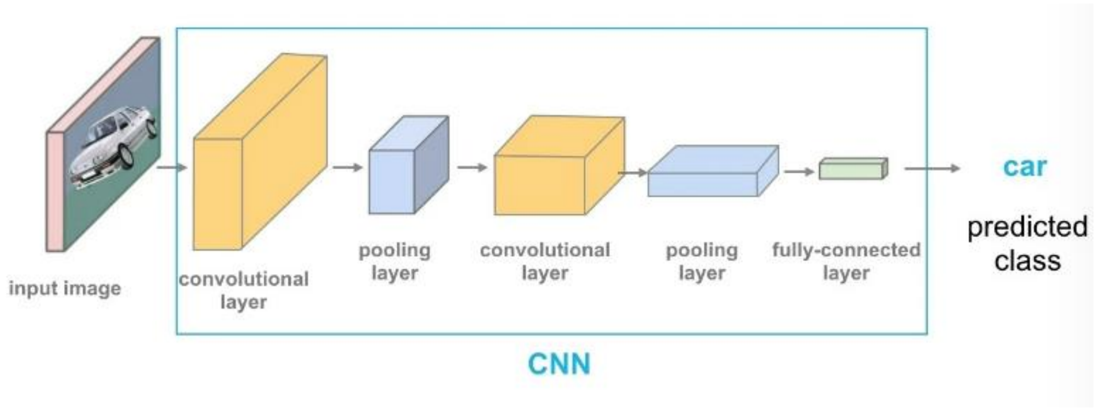


Figure 4-1: How a convolutional neural network works

Later, some improvements were made to CNN, so an upgraded version appeared, called RCNN (Region-Based Convolutional Networks).

4.2 What is R-CNN?

R-CNN is one of the models used in machine learning in computer vision tasks, especially the detection and identification of objects in an image. R-CNN is an acronym for Region-Based Convolutional Networks [23].

- **How does R-CNN work?**

This model places bounding boxes on the regions of the object, which in turn evaluates the convolutional networks independently in all regions of interest (ROI) and then classifies them in the proposed category as shown below [23].

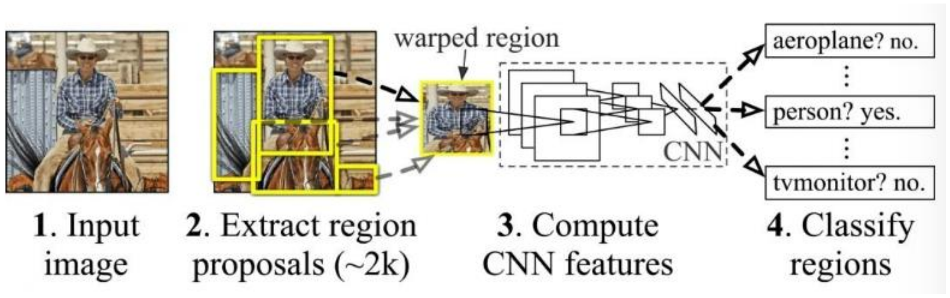


Figure 4-2: Concept of R-CNN – Region-based Convolutional Networks

Later, R-CNN was developed into Faster R-CNN, and then it was used as an infrastructure for building Mask R-CNN [23].

4.3 What is Faster R-CNN?

As mentioned earlier, Faster R-CNN is an upgraded version of R-CNN which has two phases:

Region Proposal Network (RPN): Is a neural network that suggests multiple objects within an image [23].

Fast R-CNN: It extracts the small features from each candidate box using region of interest pooling (RolPool) and then performs the classification and bounding-box regression [23].

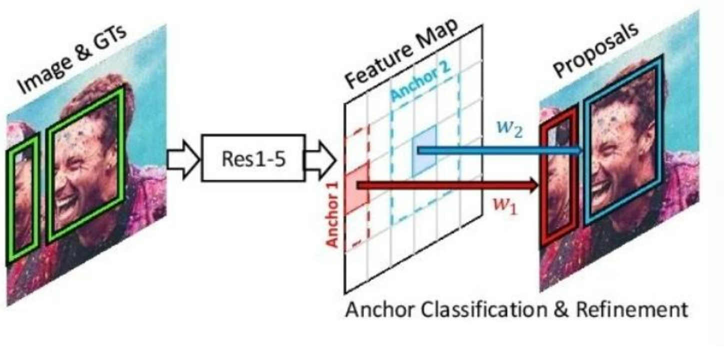


Figure 4-3: Concept of Region proposal Networks (RPN)

Faster R-CNN is faster and optimized version from R-CNN because you do not have to feed 2000 region proposals to the convolutional network every time as in RCNN, but it is enough to make convolution once for each image and a feature map is generated from it. Moreover, the main difference between them is that Faster RCNN uses a Region Proposal Network (RPN) while RCNN uses selective search to generate regions of interest [23].

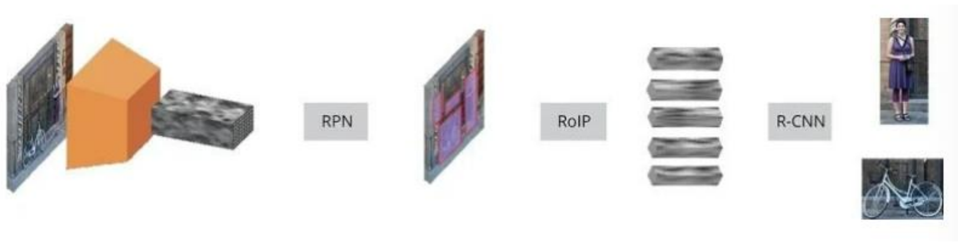


Figure 4-4: Faster R-CNN Stages

4.4 What is Mask R-CNN?

Mask R-CNN is a convolutional neural network that is an upgraded version of Faster R-CNN and is the state-of-the-art in terms of image segmentation and instance segmentation [23].

To understand the Mask R-CNN, we must first clarify what is meant by the image segmentation.

- **Image Segmentation**

Image segmentation is a computer vision task in which a digital image is divided into multiple segments (sets of pixels, also known as image objects) that are used to locate objects and boundaries (lines, curves, etc.) [23].However, there are two types of image segmentation that Mask R-CNN uses:

- **Semantic Segmentation**

In this case each pixel is categorized into a fixed set of classes without distinguishing the instances of the object. That is, it classifies identical objects as a single pixel-level class [23].

- **Instance Segmentation or Instance Recognition**

In contrast to semantic segmentation, instance segmentation deals with the correct detection of all objects in an image while each instance is precisely segmented [23].

The Figure below shows the difference between semantic segmentation and instance segmentation. In the first case (semantic segmentation), all objects are categorized as a single entity (person) while in the second case (instance segmentation), the process separates each person as a single entity [23].

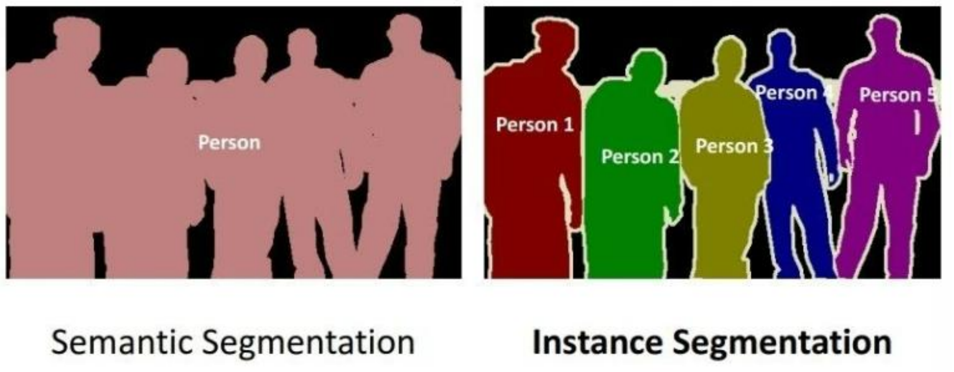


Figure 4-5: Differences of Semantic Segmentation versus Instance Segmentation

- **How does Mask R-CNN work?**

As mentioned earlier, the Mask R-CNN is an upgraded version of Faster R-CNN. In addition to the two outputs for each candidate object in Faster R-CNN (a class label and a bounding-box offset), Mask R-CNN adds a third branch that produces the object mask. The output of the additional mask differs from the output of the class and box, which It requires a more accurate spatial layout extraction of the object. In other words, Mask R-CNN adds a branch for predicting region of interest (object mask) in parallel to the existing branch for bounding box recognition [23].

Mask R-CNN is distinguished by its simplicity, high efficiency and performance, in addition to flexibility.

4.5 Library Book Detection

As the saying goes don't reinvent the wheel, just realign it. In our project, an already built Mask R-CNN model for Book Spine Detection that has been done by Mabouseif was used as a starting point of the whole system. This work was taken from the Github page of Mabouseif and was named Library Book Detection [25].

This model helped us in book boundary detection; it was tested on a group of bookshelf images taken from the main library of BZU.

Chapter 5

Simulated Results

This chapter presents the results we obtained from applying the Mask-RCNN model on bookshelf images in addition to the results of applying text extraction code on each horizontally aligned book spine image.

To be noticed, all images in this chapter are taken by us from Birzeit university main library. Some are for books that are uniformly vertical, some are horizontal and some are slant angles.

5.1 Mask-RCNN Model Result

After applying Mask-RCNN model on our test images, these are some results on each book status and the success rate of each one.

- **Uniformly Vertical Books**

  Figure 5-1 shows an image of 6 books presented vertically on the bookshelf. The result of this image after running the R-CNN algorithm on it is shown in Figure 5-2, in which the algorithm detected 6 books out of 6 which indicates that the success rate of this image is 99%. After getting the book spines, each spine image was fed into a background and noise removal code that display the text in black and the background in white in order to get better extraction results. After this step, each of the processed images was fed to the text extraction code to get the title, but because the obtained book spines were uniformly vertical, a 90° rotation piece of code was applied before Tesseract work. Figure 5-3 shows the result of the 6 books and the success rate is 67%.

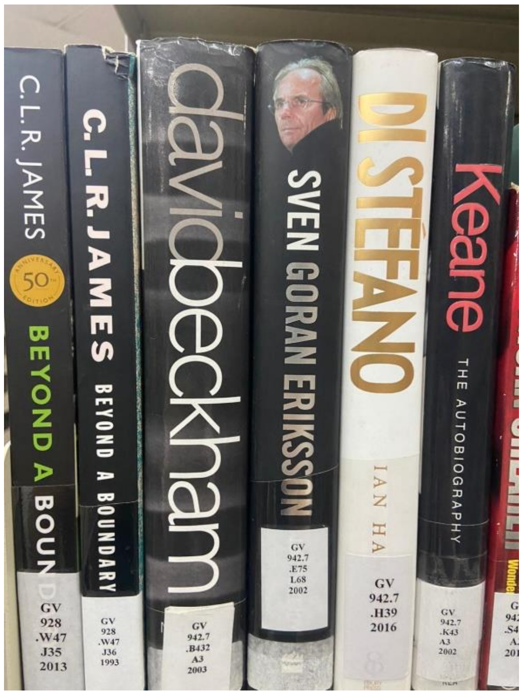


Figure 5-1: Bookshelf image of 6 vertical books

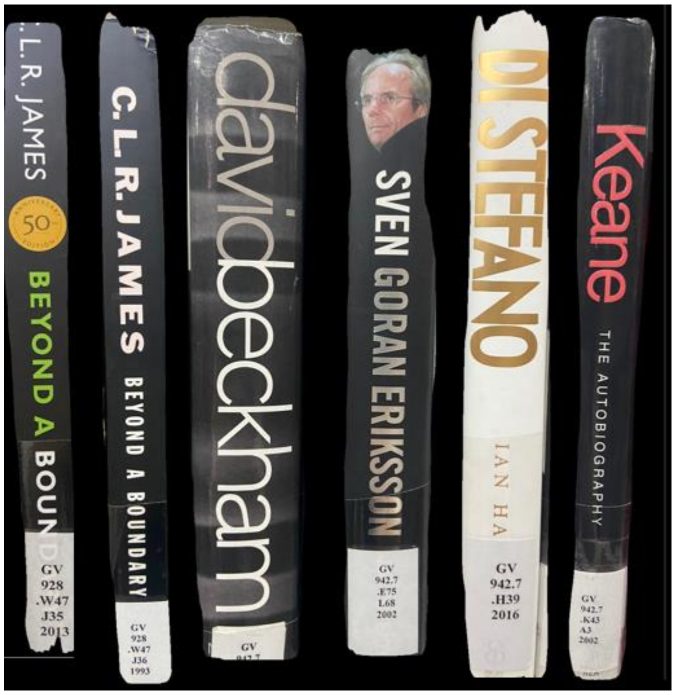
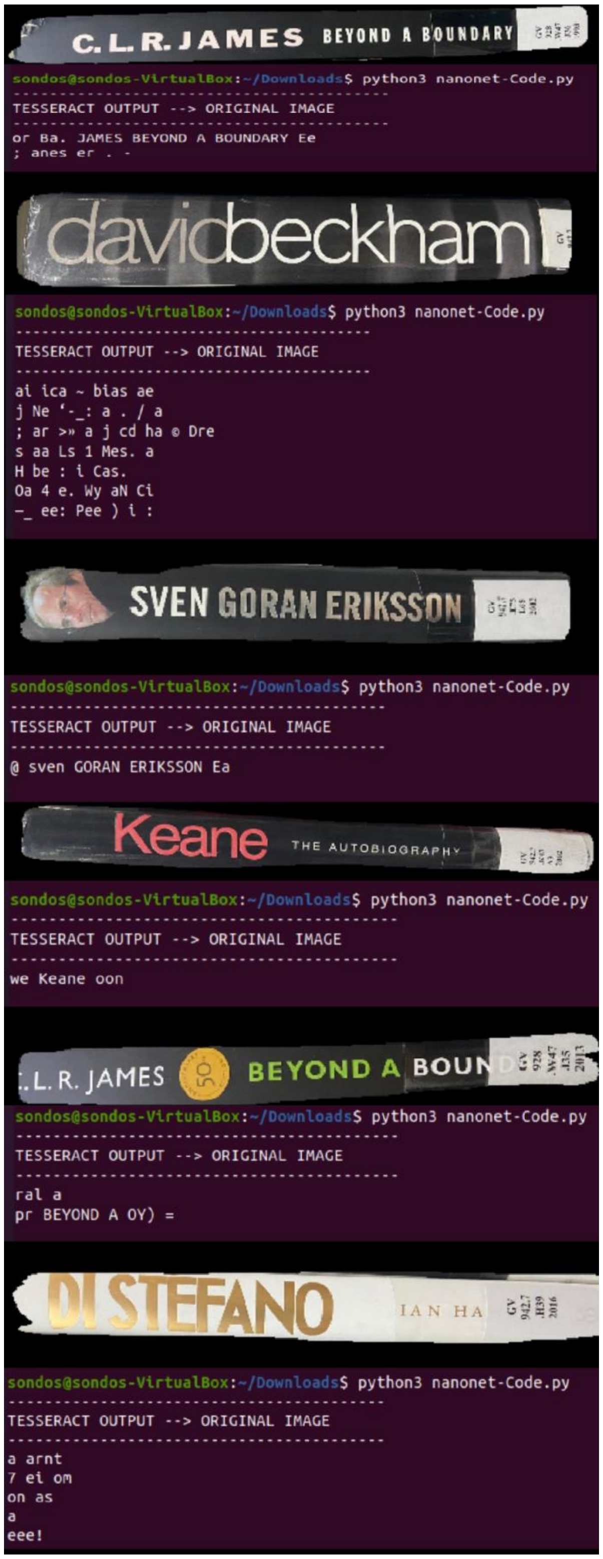


Figure 5-2: Mask R-CNN result of bookshelf image of 6 vertical books



```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
or Ba. JAMES BEYOND A BOUNDARY Ee
; anes er . -
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
ai ica ~ bias ae
j Ne ‘-_: a . / a
; ar >» a j cd ha © Dre
s aa Ls 1 Mes. a
H be : i Cas.
Oa 4 e. Wy aN Ci
—_ ee: Pee ) i :
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
@ sven GORAN ERIKSSON Ea
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
we Keane oon
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
ral a
pr BEYOND A OY) =
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
a arnt
7 ei om
on as
a
eee!
```

Figure 5-3: Tesseract result of bookshelf image of 6 vertical books

- **Horizontal Books**

Now, the horizontally-aligned books on bookshelf were tested. Figure 5-4 shows an image of 4 books presented horizontally on the bookshelf. After running the R-CNN algorithm, the result of this image is shown in Figure 5-5, where 4 books out of 4 were captured almost perfectly. The success rate is 99%.

Later, the background and noise removal code was used then Tesseract did the job for extracting the titles and rotation was not needed in this case, the result of these books is shown in Figure 5-6 and the success rate is 75%.

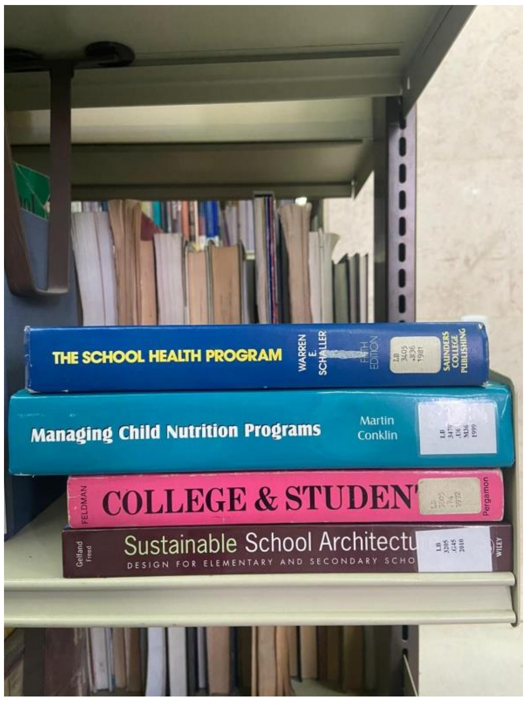

THE SCHOOL HEALTH PROGRAM
WARREN E. SCHALLER
FIFTH EDITION
SAUNDERS COLLEGE PUBLISHING
Managing Child Nutrition Programs
Martin Conklin
COLLEGE & STUDEN
FELDMAN
Pergamon
Sustainable School Architectu
DESIGN FOR ELEMENTARY AND SECONDARY SCHO
Gelfand Freed
WILEY

Figure 5-4: Bookshelf image of 4 horizontal books

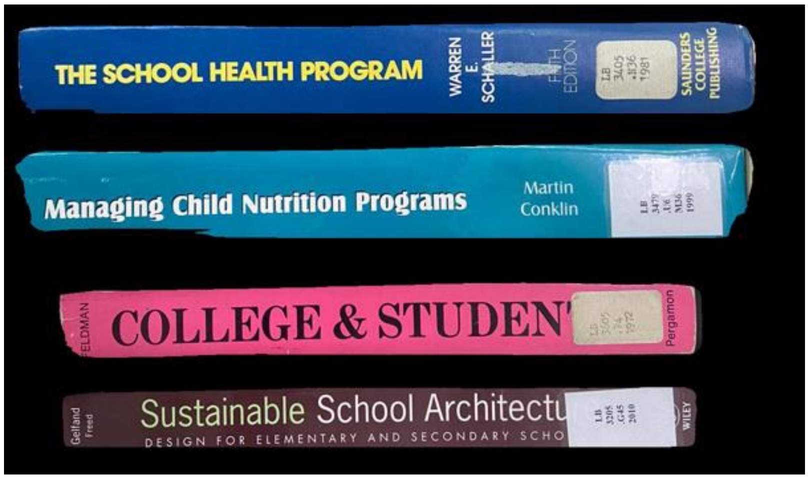
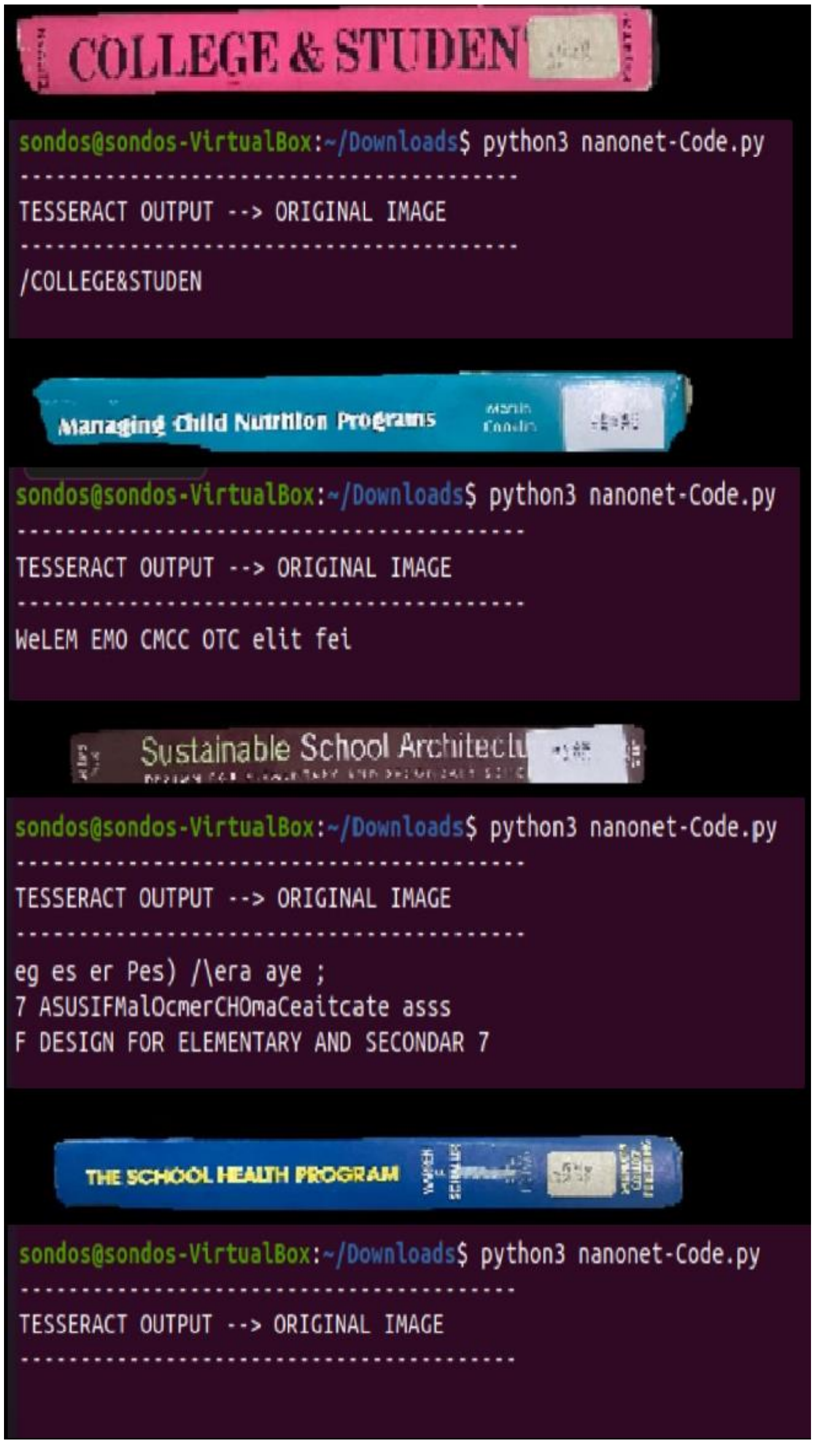

THE SCHOOL HEALTH PROGRAM
WARREN E. SCHALLER
FIFTH EDITION
LB 3405 .S36 1981
SAUNDERS COLLEGE PUBLISHING
Managing Child Nutrition Programs
Martin Conklin
LB 3479 .U6 M36 1999
FELDMAN
COLLEGE & STUDENT
Pergamon
Sustainable School Architectu
DESIGN FOR ELEMENTARY AND SECONDARY SCHO
Gelfand Freed
LB 3205 .G45 2010
WILEY

Figure 5-5: Mask R-CNN result of bookshelf image of 4 horizontal books


COLLEGE & STUDEN
Managing Child Nutrition Programs
Sustainable School Architectu
THE SCHOOL HEALTH PROGRAM

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
/COLLEGE&STUDEN
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
WeLEM EMO CMCC OTC elit fei
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
eg es er Pes) /\era aye ;
7 ASUSIFMalOcmerCHOmaCeaitcate asss
F DESIGN FOR ELEMENTARY AND SECONDAR 7
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
```

Figure 5-6: Tesseract result of bookshelf image of 4 horizontal books

- **Slanted Books**

The most challenging case of book status is the slanted books on bookshelf. Different slant angle books were tested and different results obtained according to the slant angle. In Figure 5-7, 4 slanted books by 10°-15° were tested and the result of the R-CNN algorithm is shown in Figure 5-8 where 3 books out of 4 were detected and the success rate is 75%.

In Figure 5-9, the result of text extraction after applying background and noise removal is shown. The success rate of this case is 33%.

Another slanted books image was tested and gave a bad result which is shown in Figure 9-10. The image consists of 11 thin books and when running the R-CNN algorithm it did not detect any book perfectly.

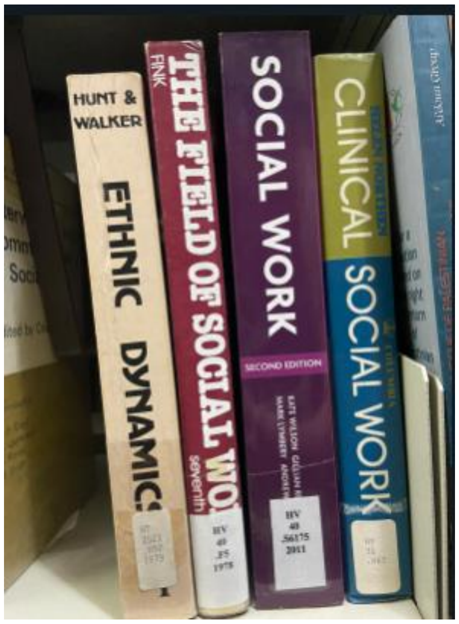


Figure 5-7: Bookshelf image of 4 slanted books

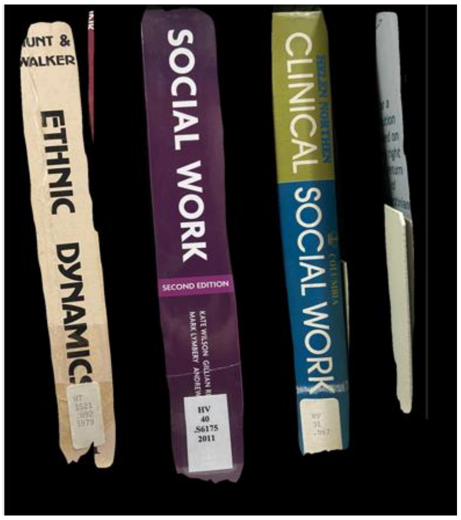
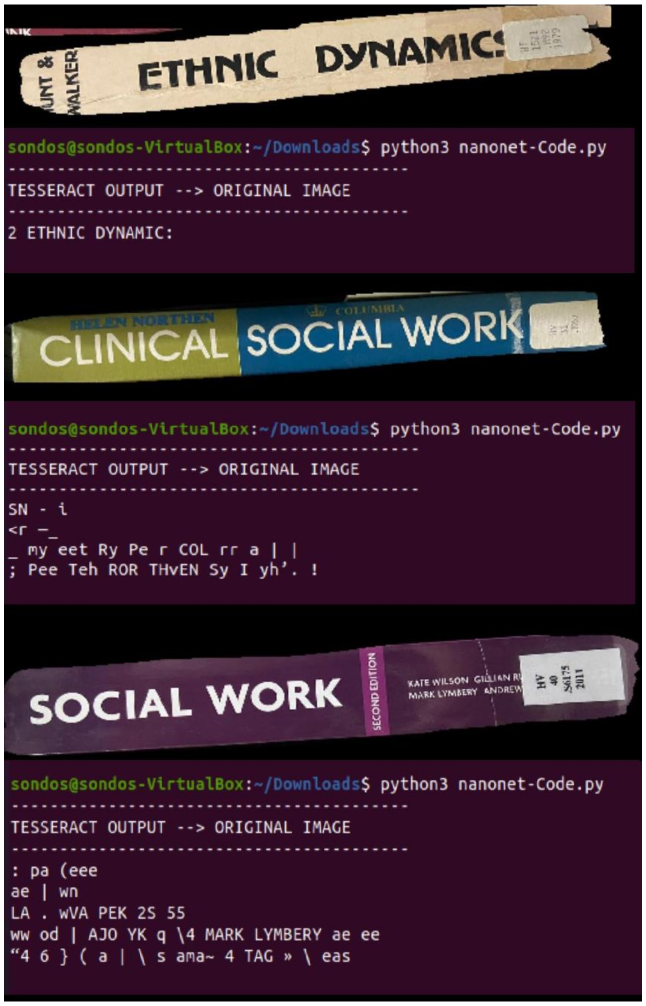


Figure 5-8: Mask R-CNN result of bookshelf image of 4 slanted books



```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
2 ETHNIC DYNAMIC:
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
SN - i
<r —_
_ my eet Ry Pe r COL rr a | |
; Pee Teh ROR THvEN Sy I yh’. !
```

```
sondos@sondos-VirtualBox:~/Downloads$ python3 nanonet-Code.py
----------------------------------------
TESSERACT OUTPUT --> ORIGINAL IMAGE
----------------------------------------
: pa (eee
ae | wn
LA . wVA PEK 2S 55
ww od | AJO YK q \4 MARK LYMBERY ae ee
“4 6 } ( a | \ s ama~ 4 TAG » \ eas
```

Figure 5-9: Mask R-CNN result of bookshelf image of 4 slanted books

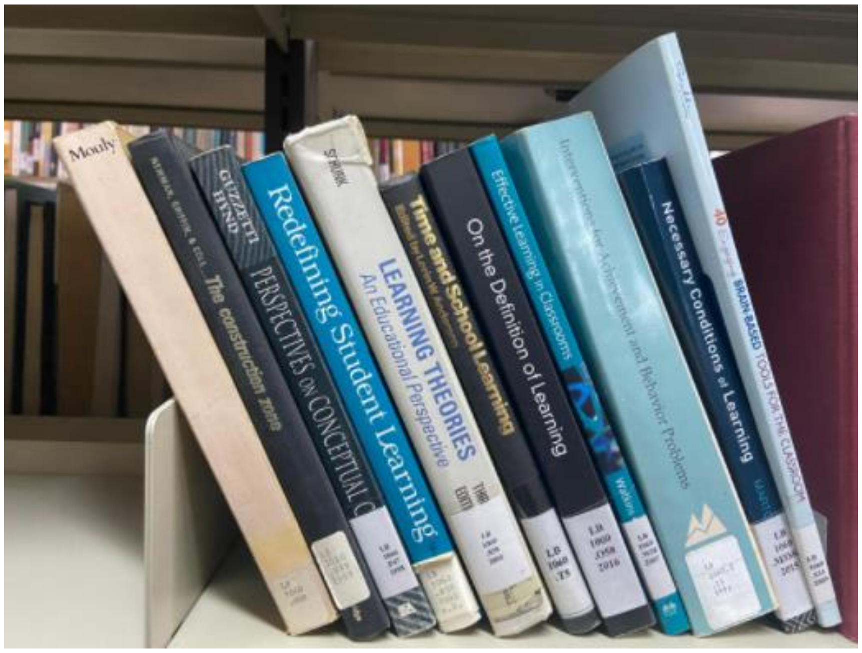


Figure 5-10: Bookshelf image of 11 slanted books

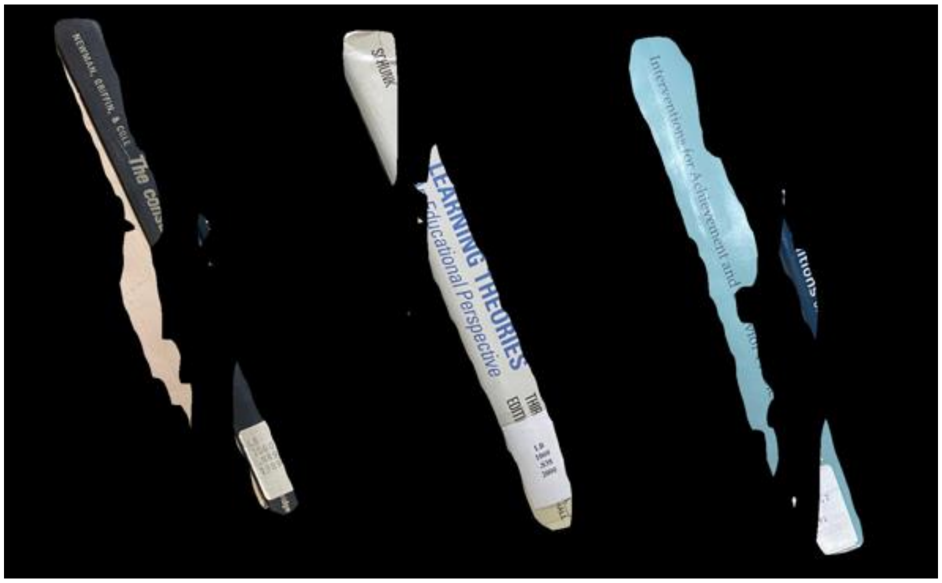


Figure 5-11: Mask R-CNN result of bookshelf image of 11 slanted books

After many experiments on taken images, we came out with these results for each case (vertical, horizontal and slant):

→ **For Vertical Books Case**:

From 39 images of vertical books which contains 440 books the accuracy of the boundary detection was 97%.

→ **For Horizontal Books Case**:

From 8 images of vertical books which contains 36 books the accuracy of the boundary detection was 95%.

→ **For Slant Angled Books Case**:

From 12 images of vertical books which contains 131 books the accuracy of the boundary detection was 47%.

5.2 Text Extraction Result

The following figures show some example of image processing and background and noise removal that has been applied on taken images. Figure 5-12 and 5-14 show two different images before being processed. While Figure 5-13 and 5-15 show these images after being processed. One of them had good processing and the text was displayed perfectly while the other suppressed the text and the image turned into a white block.

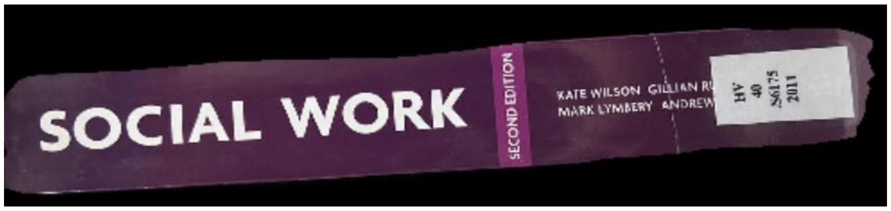


Figure 5-12: Book spine before processing

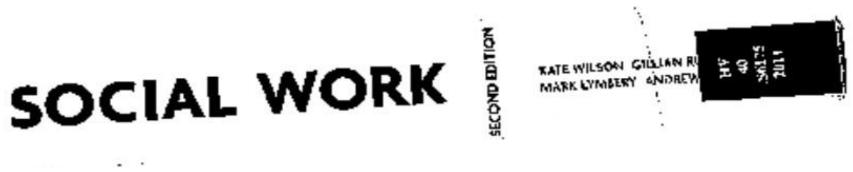


Figure 5-13: Book spine after processing with good result

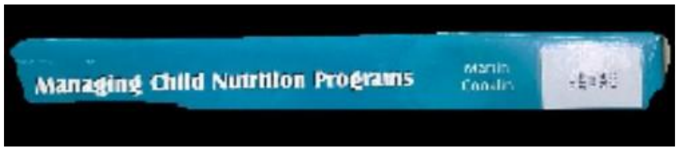


Figure 5-14: Book spine before processing

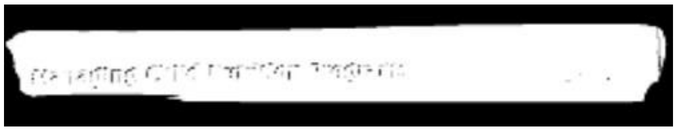
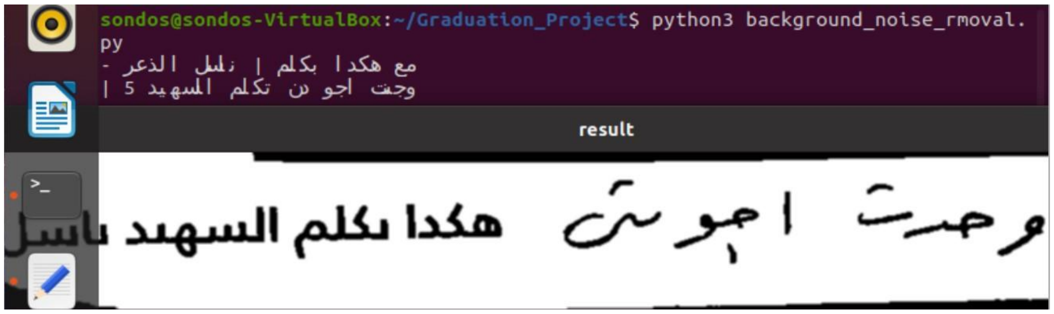


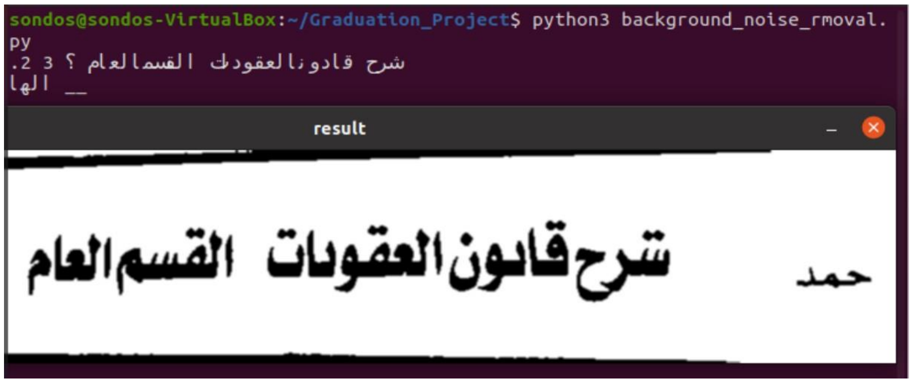
Figure 5-15: Book spine after processing with bad result

Furthermore, we have tried to detect text from Arabic books. The following figures show the results. Where the accuracy of the Arabic language was not as good as English and this is due to many reasons one of them is the diversity of fonts in Arabic which makes the extraction harder.



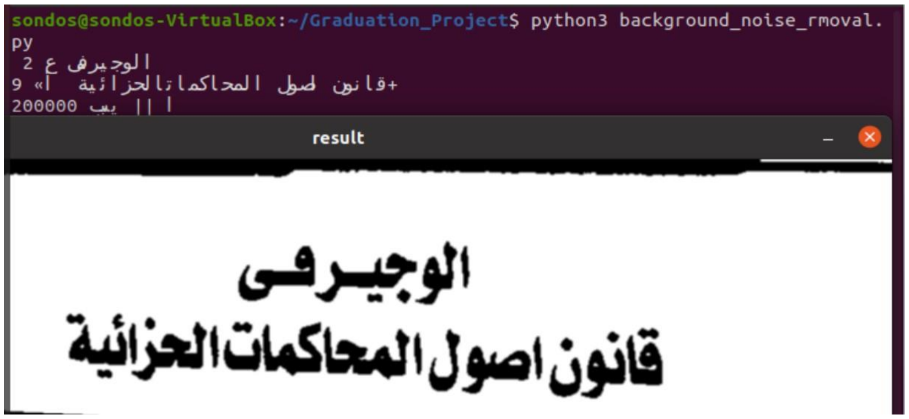
```
sondos@sondos-VirtualBox:~/Graduation_Project$ python3 background_noise_rmoval.
py
- مع هكدا بكلم | نليل الذعر
| 5 وجت اجو ن تكلم السهيد
```

Figure 5-16: Arabic Text Extraction Result



```
sondos@sondos-VirtualBox:~/Graduation_Project$ python3 background_noise_rmoval.
py
.2 3 ؟ شرح قادونالعقودك القسمالعام ا
الها __
```

Figure 5-17: Arabic Text Extraction Result (2)



```
sondos@sondos-VirtualBox:~/Graduation_Project$ python3 background_noise_rmoval.
py
 2 الوجيرف ع
9 «ا  +قانون لصول المحاكماتالحزائية
200000 يبب || ا
```

Figure 5-18: Arabic Text Extraction Result (3)

As we can see from the previous samples, the extraction of texts in English is more accurate than Arabic ones.

→ Accuracy of English books text extraction was 66.5%.

→ Accuracy of Arabic books text extraction was 53.2%.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

As a conclusion on this report, some new concepts have been achieved and discussed such as the Text Extraction and Optical Character Recognition concepts which are very important and distributed ones used in many fields in the real life such as building license plate readers, digitizing invoices and digitizing ID cards. Furthermore, Image Pre-Processing techniques were studied and shown as there exist many of them which are used frequently and are required in different applications.

Some python codes using OCR engines like Tesseract and different libraries such as OpenCV, NumPy and others were needed and discussed in order to be the first step in building the gripper functionality to get sufficient performance. However, the efficiency of these codes was different and dependent on the image pre-processing techniques and python libraries used. Overall, one of these codes was the best for us that have the ability of detect and extract text from images in a good manner.

We found the Mask-RCNN model for book boundary detection is the best to split each book individually from bookshelf images.

6.2 Limitations

Several limitations and restriction have been faced in performing the process of text extraction such:

- **Limitation of Tesseract**

  Tesseract OCR Engine is quite powerful but does have several limitations in dealing with images:

  - Not very good quality of the image (size, contrast and lightening).
  - If the book title on the spine is written in a vertical format.
  - If an image contains languages outside of those given in the -l LANG arguments, results may be poor.
  - For Arabic language, the enhancement on the book spine image according to coloring caused some letters losing some of its parts like points.
  - Tesseract does not support some special fonts.

- **Boundary Detection Limitation**

  Book boundary detection is important in text extraction because it makes it easy to extract text from each book. But, we faced a problem in splitting slant angle books and bad quality book shelf image.

- **Limitation of book Organization in the Library Itself**

  Some limitations that we faced were from the organization of the books in Birzeit university library, such as:

  - Books placed so randomly that it is difficult to find the spines of the books.

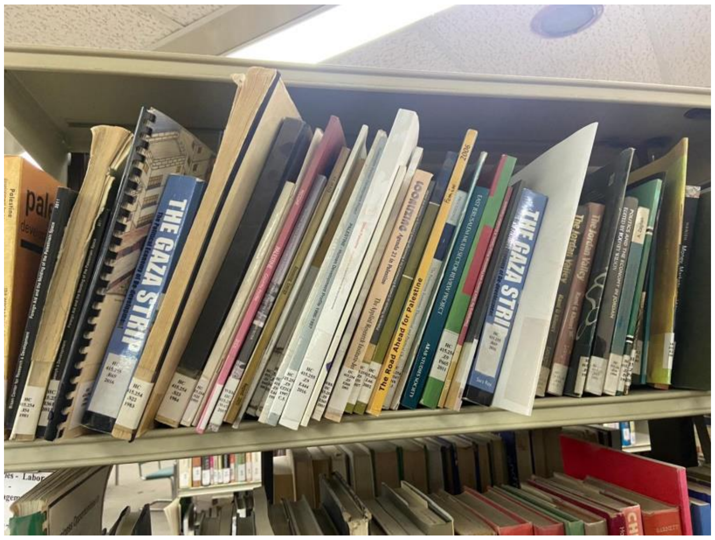


Figure 6-1: Limitation on bookshelf case 1

- Books are placed upside down so that the robot's camera cannot see the main part of the book, which is the title part.

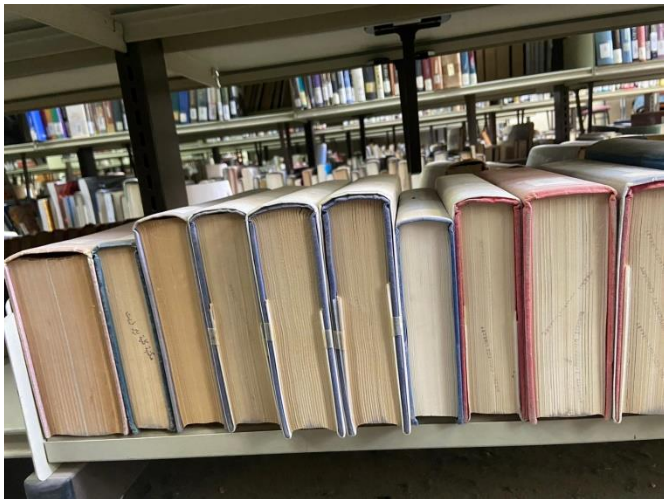


Figure 6-2: Limitation on bookshelf case 2

- The presence of dust on books, which reduces the quality of the images and makes it harder to extract the text from them.
- Some books are covered with a transparent glossy layer that makes them glow when captured, making it difficult to extract the text from these books.
- Some old books, usually the font is not clear due to its age, and some of the font is etched and takes the same color as the background of the book, making it difficult to extract the text from these books.

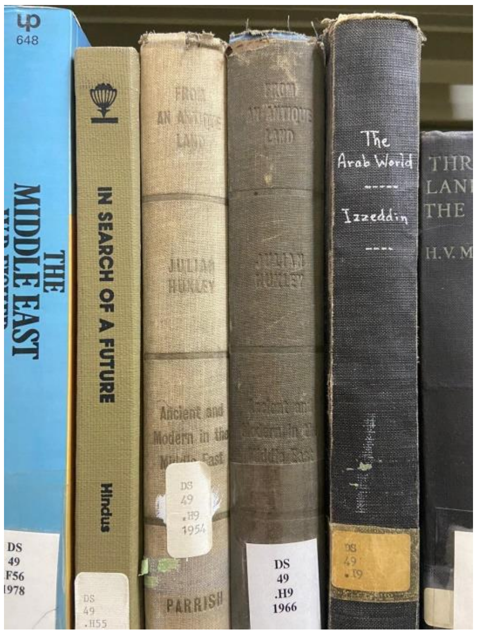


Figure 6-3: Limitation on bookshelf case 3

6.3 Future Work

Whatever we accomplished in this project until now, there is a lot of work and effort required to complete and achieve the main aim of our project which is build an efficient manipulator for the librarian robot. For short-term future work, the boundary detection of non-uniformed and slanted books should be resolved. In addition, more language should be added to the algorithm in order to extract any text from any language. For the long-term, the robot should be assembled and programs added to it in order to start working in the library. Furthermore, the organization of the books in the library bookshelves should be more uniformed.

References

[1] : Built in. Robotics Technology. https://builtin.com/robotics. Retrieved Dec 2021

[2] : Bernadine Racoma. Day Translations. How will the robot change the world? https://www.daytranslations.com/blog/will-robots-change-world/. Retrieved Dec 2021

[3] : Matt Simon. Wired. What is a Robot? https://www.wired.com/story/what-is-a-robot/. Retrieved Dec 2021

[4] : Erico Guizzo. Robots. What is a Robot? https://robots.ieee.org/learn/what-is-a-robot/. Retrieved Dec 2021

[5] : Jeremy Gottlieb and David Leech Anderson. The Mind Project. Kinds of Robots. https://mind.ilstu.edu/curriculum/medical_robotics/kinds_of_robots.html. Retrieved Dec 2021

[6] : Corina Monica Pop, Gheorghe Leonte Mogan, "ROBOTIC GRIPPERS FOR HANDLING BOOKS IN LIBRARIES", Faculty of Mechanical Engineering, Transilvania University of Brasov, Romania, 28-30 May 2015.

[7] : Ramos-Garijo R, Prats M, Sanz PJ, Del Pobil AP, "An Autonomous Assistant Robot for Book Manipulation in a Library", Computer Science Department, Jaume-I University, Castellón, Spain, November 2003.

[8] : Eiji TAIRA, Seiichi UCHIDA, Hiroaki SAKOE, "A MODEL-BASED BOOK BOUNDARY DETECTION TECHNIQUE FOR BOOKSHELF IMAGE ANALYSIS", Faculty of Information Science and Electrical Engineering, Kyushu University 6–10–1 Hakozaki, Higashi–ku, Fukuoka–shi, 812-8581 Japan.

[9] : Eiji TAIRA, Seiichi UCHIDA, Hiroaki SAKOE, "Book Boundary Detection and Title Extraction for Automatic Bookshelf Inspection", Faculty of Information Science and Electrical Engineering, Kyushu University 6–10–1 Hakozaki, Higashi–ku, Fukuoka–shi, 812-8581 Japan.

[10] : Young Wonks. What is OpenCV? https://www.youngwonks.com/blog/What-is-OpenCV. Retrieved Jan 2022

[11] : Filip Zelic and Anuj Sable. Nanonets. A comprehensive guide to OCR with Tesseract, OpenCV and Python. https://nanonets.com/blog/ocr-with-tesseract/ Retrieved Jan 2022

[12] : HIPR. Grayscale Images. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm Retrieved Jan 2022.

[13] : Madhav University. Binarization Process. https://madhavuniversity.edu.in/binarization-process.html. Retrieved Jan 2022

[14] : Parthima Guruprasad, Kushal S Mahalingpur and Manjesh T.N, "OVERVIEW OF DIFFERENT THRESHOLDING METHODS IN IMAGE PROCESSING", Vivekananda Institute of Technology, Bangalore-74.

[15] : Anisha Swain. Image Vision. Noise Filtering in Image Processing. https://medium.com/image-vision/noise-filtering-in-digital-image-processing-d12b5266847c Retrieved Jan 2022

[16] : HIPR. Dilation. https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm. Retrieved Jan 2022

[17] : HIPR. Erosion. https://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm. Retrieved Jan 2022.

[18] : ShivamKumar1. GeeksforGeeks. Difference between Opening and Closing in Digital Image Processing. https://www.geeksforgeeks.org/difference-between-opening-and-closing-in-digital-image-processing/#:~:text=Opening%20is%20a%20process%20in,then%20erosion%20operation%20is%20performed. Retrieved Jun 2022

[19] : Adrian Rosebrock. PyImageResearch. OpenCV Text Detection. https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/?fbclid=IwAR26xpwdQCAIFa9s_1bORRyszzQku80NXIsAyO-QlMzJ96N6hSmjHgw5SLU . Retrieved Jan 2022

[20] : Filip Zelic and Anuj Sable. Nanonets. A comprehensive guide to OCR with Tesseract, OpenCV and Python. https://nanonets.com/blog/ocr-with-tesseract/ Retrieved Jan 2022

[21] : Project Gurukul. Python Project – Text Detection and Extraction with OpenCV and OCR. https://projectgurukul.org/python-text-detection-extraction-opencv-ocr/?fbclid=IwAR27oZWD8d1aK5J8f1pJkiNEJTDdV5sZVPM3Bt7AHO1D-EsIx7tpQU4oy-I. Retrieved Jan 2022

[22] : Lior Talker, Yael Moses, “Viewpoit-Independent Book Spine Segmentation”, The Interdisciplinary Center Herzliya 46150, Occupied Palestine, March 2014.

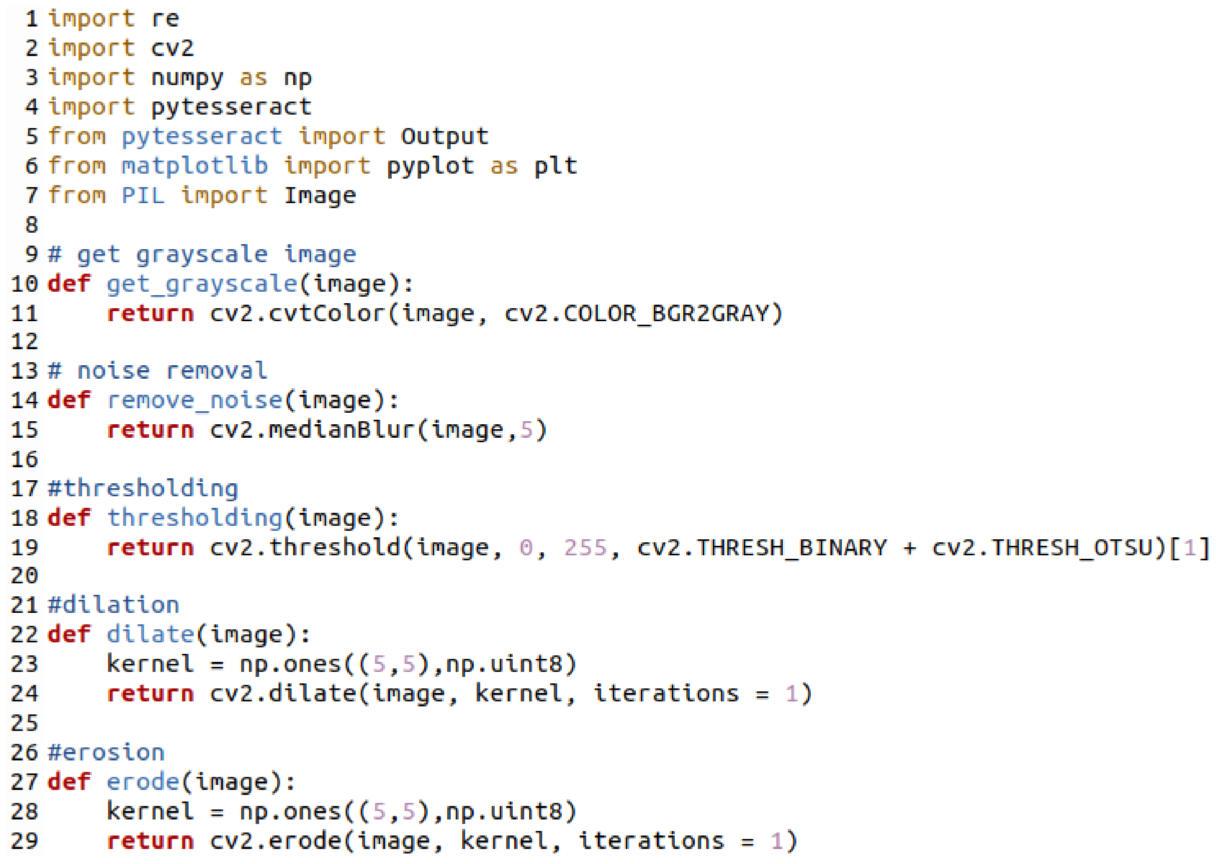
[23] : Mask R-CNN: A Beginner’s Guide, https://viso.ai/deep-learning/mask-r-cnn/?fbclid=IwAR0tRGh7N0mFy3bMi8SzsXvgT6k4uJkARfsnJycPWmeaP3PqbAB4u6jZRGQ#:~:text=Mask%20R%2DCNN%20is%20a,segmentation%20mask%20for%20each%20instance . Retrieved Jun 2022

[24] : Contours: Getting Started, https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html . Retrieved Jun 2022

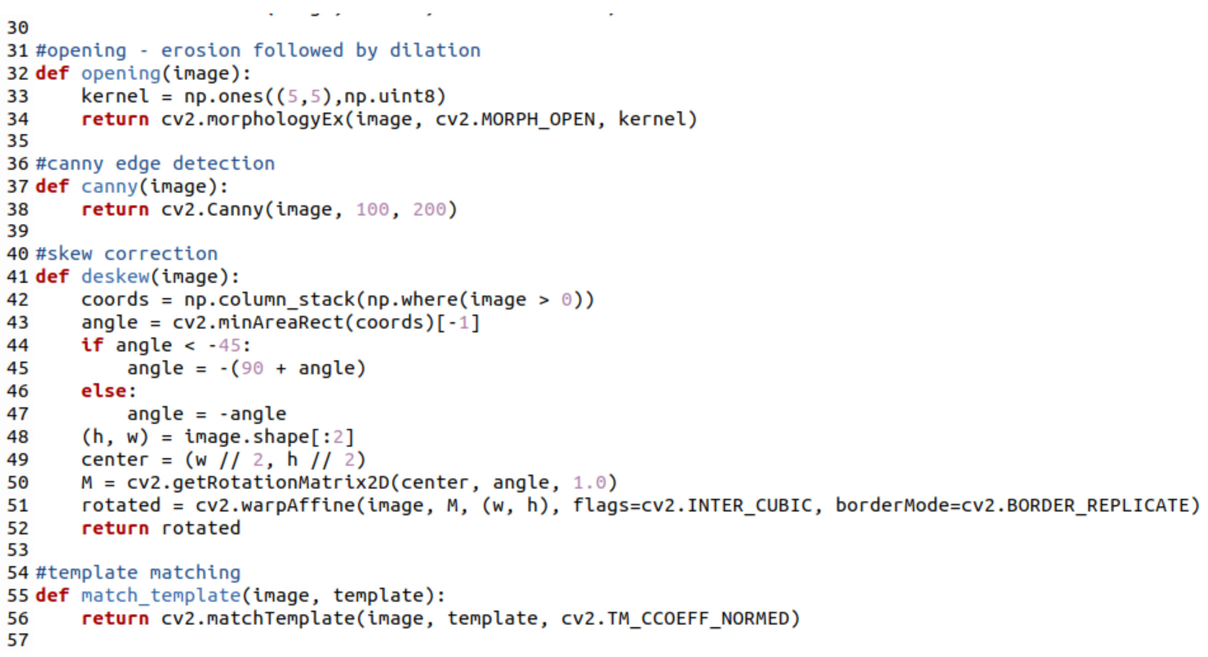
[25] : Library Book Detection https://github.com/mabouseif/Library_Book_Detection?fbclid=IwAR29eMziR9Rj5cIzawXyq4OOrsjRplW05Vez4ErA-m-Sb_-ZXvP2gqyO9lw

Appendices

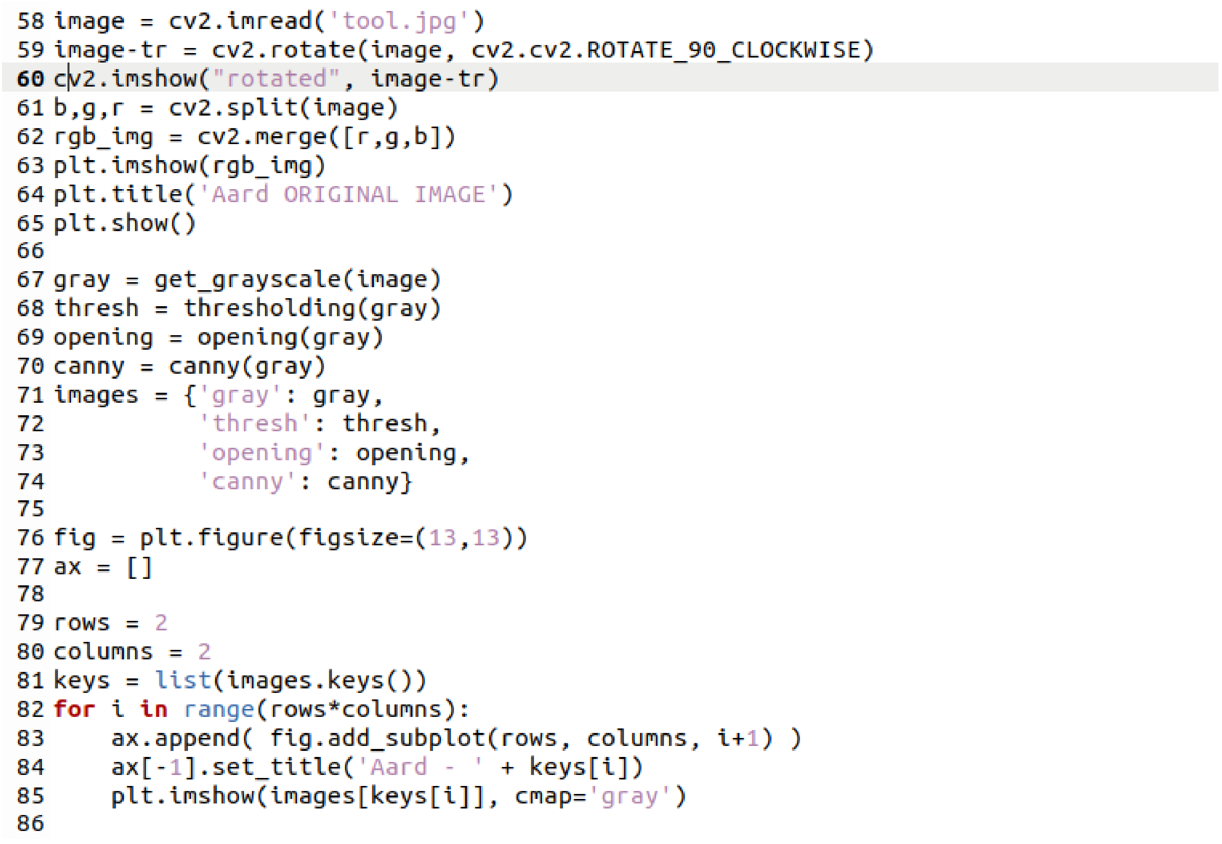
Appendix A Text Extraction Code



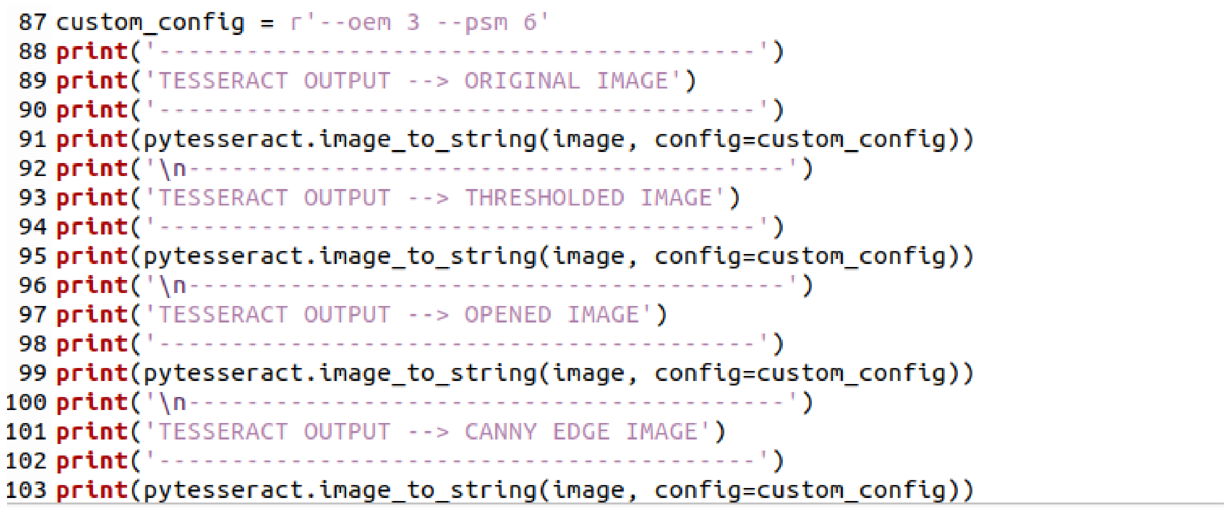
```python
import re
import cv2
import numpy as np
import pytesseract
from pytesseract import Output
from matplotlib import pyplot as plt
from PIL import Image

# get grayscale image
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# noise removal
def remove_noise(image):
    return cv2.medianBlur(image,5)

#thresholding
def thresholding(image):
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

#dilation
def dilate(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.dilate(image, kernel, iterations = 1)

#erosion
def erode(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.erode(image, kernel, iterations = 1)
```



```
#opening - erosion followed by dilation
def opening(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)

#canny edge detection
def canny(image):
    return cv2.Canny(image, 100, 200)

#skew correction
def deskew(image):
    coords = np.column_stack(np.where(image > 0))
    angle = cv2.minAreaRect(coords)[-1]
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(image, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
    return rotated

#template matching
def match_template(image, template):
    return cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)

```

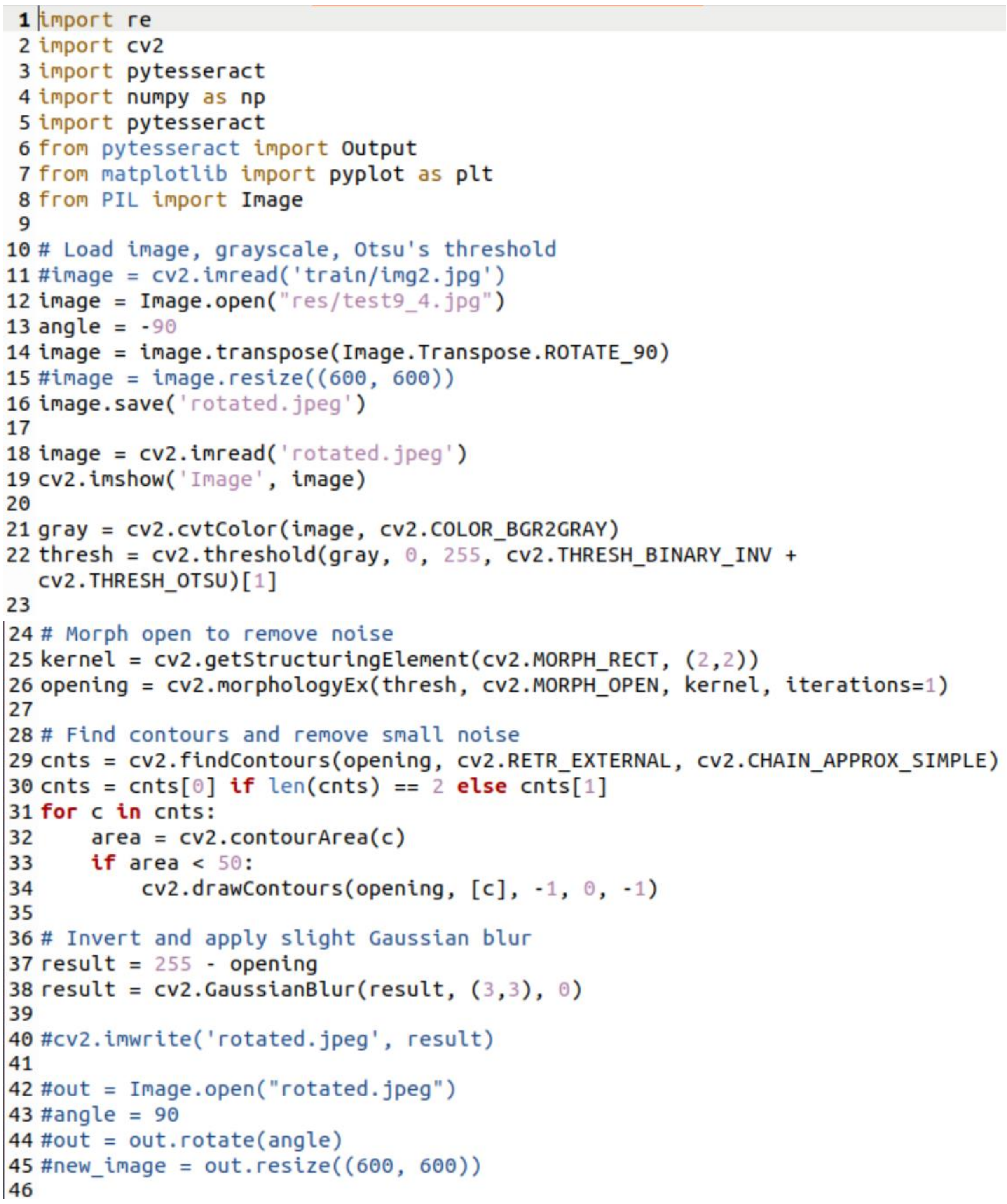


```
image = cv2.imread('tool.jpg')
image-tr = cv2.rotate(image, cv2.cv2.ROTATE_90_CLOCKWISE)
cv2.imshow("rotated", image-tr)
b,g,r = cv2.split(image)
rgb_img = cv2.merge([r,g,b])
plt.imshow(rgb_img)
plt.title('Aard ORIGINAL IMAGE')
plt.show()

gray = get_grayscale(image)
thresh = thresholding(gray)
opening = opening(gray)
canny = canny(gray)
images = {'gray': gray,
          'thresh': thresh,
          'opening': opening,
          'canny': canny}

fig = plt.figure(figsize=(13,13))
ax = []

rows = 2
columns = 2
keys = list(images.keys())
for i in range(rows*columns):
    ax.append( fig.add_subplot(rows, columns, i+1) )
    ax[-1].set_title('Aard - ' + keys[i])
    plt.imshow(images[keys[i]], cmap='gray')

```

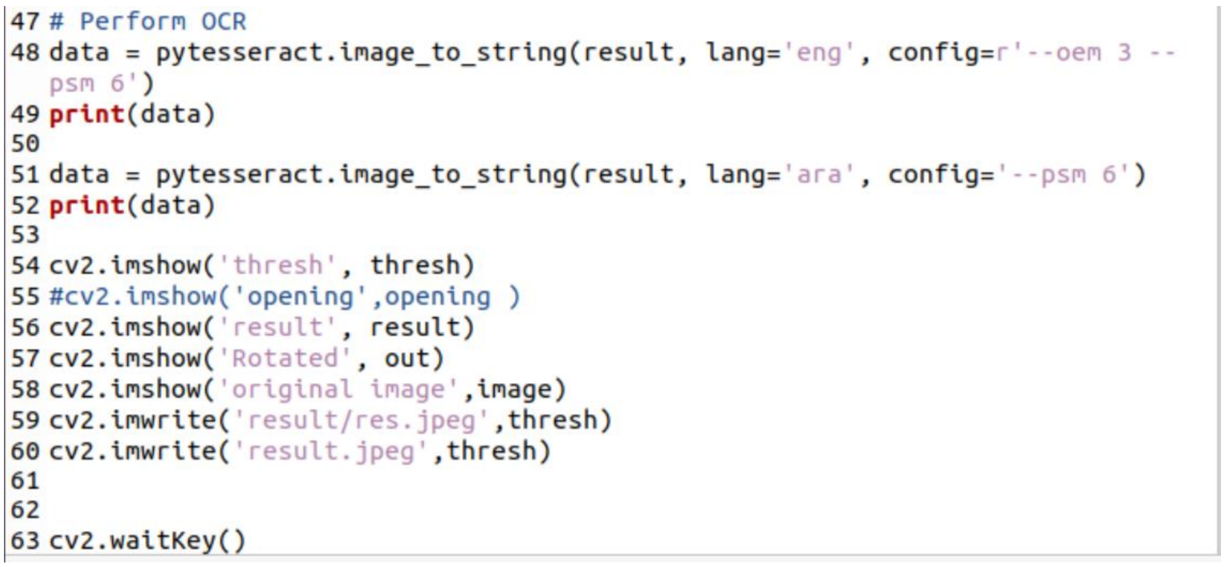


```python
custom_config = r'--oem 3 --psm 6'
print('-----------------------------------------')
print('TESSERACT OUTPUT --> ORIGINAL IMAGE')
print('-----------------------------------------')
print(pytesseract.image_to_string(image, config=custom_config))
print('\n-----------------------------------------')
print('TESSERACT OUTPUT --> THRESHOLDED IMAGE')
print('-----------------------------------------')
print(pytesseract.image_to_string(image, config=custom_config))
print('\n-----------------------------------------')
print('TESSERACT OUTPUT --> OPENED IMAGE')
print('-----------------------------------------')
print(pytesseract.image_to_string(image, config=custom_config))
print('\n-----------------------------------------')
print('TESSERACT OUTPUT --> CANNY EDGE IMAGE')
print('-----------------------------------------')
print(pytesseract.image_to_string(image, config=custom_config))
```

Appendix A Background and Noise Removal Code



```python
import re
import cv2
import pytesseract
import numpy as np
import pytesseract
from pytesseract import Output
from matplotlib import pyplot as plt
from PIL import Image

# Load image, grayscale, Otsu's threshold
#image = cv2.imread('train/img2.jpg')
image = Image.open("res/test9_4.jpg")
angle = -90
image = image.transpose(Image.Transpose.ROTATE_90)
#image = image.resize((600, 600))
image.save('rotated.jpeg')

image = cv2.imread('rotated.jpeg')
cv2.imshow('Image', image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

# Morph open to remove noise
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (2,2))
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=1)

# Find contours and remove small noise
cnts = cv2.findContours(opening, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]
for c in cnts:
    area = cv2.contourArea(c)
    if area < 50:
        cv2.drawContours(opening, [c], -1, 0, -1)

# Invert and apply slight Gaussian blur
result = 255 - opening
result = cv2.GaussianBlur(result, (3,3), 0)

#cv2.imwrite('rotated.jpeg', result)

#out = Image.open("rotated.jpeg")
#angle = 90
#out = out.rotate(angle)
#new_image = out.resize((600, 600))

```



```
# Perform OCR
data = pytesseract.image_to_string(result, lang='eng', config=r'--oem 3 --psm 6')
print(data)

data = pytesseract.image_to_string(result, lang='ara', config='--psm 6')
print(data)

cv2.imshow('thresh', thresh)
#cv2.imshow('opening',opening )
cv2.imshow('result', result)
cv2.imshow('Rotated', out)
cv2.imshow('original image',image)
cv2.imwrite('result/res.jpeg',thresh)
cv2.imwrite('result.jpeg',thresh)


cv2.waitKey()
```