Digital/Electronic Signature for Lawyers/Judges using Serverless Applications on AWS

by

Seyedmorteza MALEKABADI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN SOFTWARE ENGINEERING
M.A.SC.

MONTRÉAL, 7 SEPTEMBRE 2023

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain April, Thesis Supervisor
Department of Software Engineering and Information Technology, École de technologie supérieure

Mr. Michel Kadoch, Chair, Board of Examiners
Department of Electrical Engineering at École de technologie supérieure

Mr. Abdelaoued Gherbi, External Examiner
Department of Software Engineering, and Information Technology, École de technologie supérieure

Mr. Alain Abran, External Examiner
Department of Software Engineering and Information Technology, École de technologie supérieure

THIS THESIS WAS PRENSENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

MONTREAL, JULY 28, 2023

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Signature numérique/électronique pour les avocats/juges utilisant des applications sans serveur sur AWS

Seyedmortseza MALEKABADI

RÉSUMÉ

Comme l'internet est de plus en plus utilisé, l'application des technologies de l'information s'est accélérée dans toutes les industries, y compris celle du droit. Le processus actuel, au Québec et au Canada, pour planifier un procès (c.-à-d. en utilisant un formulaire de protocole d'instance) nécessite des signatures manuscrites, ce qui entraîne beaucoup d'inefficacités. Le secteur juridique pourrait utiliser des technologiques en automatisant ce type de processus à l'avenir. De plus, les architectures d'applications infonuagiques émergentes qui utilisent des technologies sans serveur présentent de nombreux avantages, par exemple aucune gestion de serveur n'est nécessaire et l'application s'adapte automatiquement. Cette recherche vise à expérimenter une solution sans serveur pour automatiser la signature de protocoles d'instances au Québec.

Une première étape de cette recherche consiste à comprendre d'abord l'état de l'art du processus de signature électronique. Le processus de signature d'un protocole d'instance exige que chaque avocat signe individuellement le formulaire, ce qui démontre son consentement. Dans le cas d'une signature électronique, sans l'utilisation de logiciel commercial, chacune de ces signatures pourrait être soulevée du formulaire et apposée au côté des autres sur le document final. Cette opération nécessite l'utilisation de technologies de traitement d'images pour localiser et transférer les signatures individuelles.

Une fois que les défis du processus de signature est étudié et que les technologies de traitement d'image est investigué, les activités de recherche visaient à concevoir et à expérimenter un pipeline de techniques de traitement d'images pour proposer une solution à l'aide d'une étude de cas. Cette expérimentation est ensuite implantée dans un prototype de traitement de protocoles d'instance pour en valider la robustesse.

Mots-clés : Protocole d'instance, droit, signature électronique, applications sans serveurs, traitement d'images

Digital/Electronic Signature for lawyers/judges using serverless applications on AWS

Seyedmorteza MALEKABADI

ABSTRACT

As the Internet is used more and more widely, the application of information technology has deepened in all industries including law. Today, the process for planning a trial in Quebec and Canada (i.e., using a case protocol form) requires original handwritten signatures, which is inefficient and would benefit from the use of e-signatures. As well, emerging cloud application architectures that use serverless technologies carry many advantages, for example, no server management is necessary as applications scale automatically. The law industry could make use of these technological advantages as it automates its processes in the future.

A first step in this research is to understand the state of the art of the electronic signature process. The process of signing a case protocol requires each lawyer involved to individually sign the form, demonstrating their consent. In the case of an electronic signature, without the use of commercial software, each of these signatures could be lifted from the form and applied next to the others on the final document. This operation requires the use of image processing technologies to locate and transfer individual signatures.

Once the challenges of the signature process are studied and image processing technologies are investigated, the research activities aim to design and experiment with an image processing technique pipeline to propose a solution through a case study. This experiment is then implemented in a trial protocol processing prototype to validate its robustness.

Keywords: Case protocol, law, electronic signature, serverless applications, image processing

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABREVIATIONS

API	Application Programming Interface
AWS	Amazon Web Services
LoG	Laplacian of Gaussian
PDF	Portable Document Format
PNG	Portable Network Graphics
PIPEDA	The Personal Information Protection and Electronic Documents
GAN	Generative Adversarial Network
GC	Government of Canada
SES	Amazon Simple Email Service
S3	Amazon Simple Storage Service

INTRODUCTION

The first contemporary use of the term “digitalization” in conjunction with computerization appeared in a 1971 essay published in the North American Review (Brennen & Kreiss, 2016). In it, Robert Wachal discusses the social implications of the “digitalization of society” in the context of considering objections to, and potentials for, computer-assisted humanities research. In this digital world, electronic signature aims to become equivalent to handwritten signatures. An electronic signature (e-signature) is obtained by applying a series of cryptographic operations on the document to sign. Usually, these operations are based on the use of asymmetric cryptography and hash functions. The purpose of an electronic signature is to guarantee authentication and integrity of the information signed.

An e-signature is becoming legally equivalent to a handwritten signature and therefore, an e-signed document could be used in legal proceedings. In Europe, the conditions which an e-signature should fulfil to be legally equivalent to a handwritten signature are established in the directive 1999/93/EC of the European Parliament and the Council (Lentner & Parycek, 2016). Several Canadian jurisdictions, including the federal government, provinces and territories, have developed laws, policies and standards for electronic documents and electronic signatures since the mid-1990s (Government of Canada, 2017).

The process of planning a trial, in Quebec and Canada, is currently done using government forms (e.g., Case Protocols) that require handwritten signatures by all parties involved. This trial planning process is initiated by a leading lawyer that proposes a trial schedule for all the activities and then must exchange this planning with many parties involved until an agreement is reached and all parties sign the case protocol. Next, he contacts the court to file the case protocol. In a more modern setting where the case protocol is an electronic document, the agreement and verification process could be done using an e-signature (see Figure 0.1).

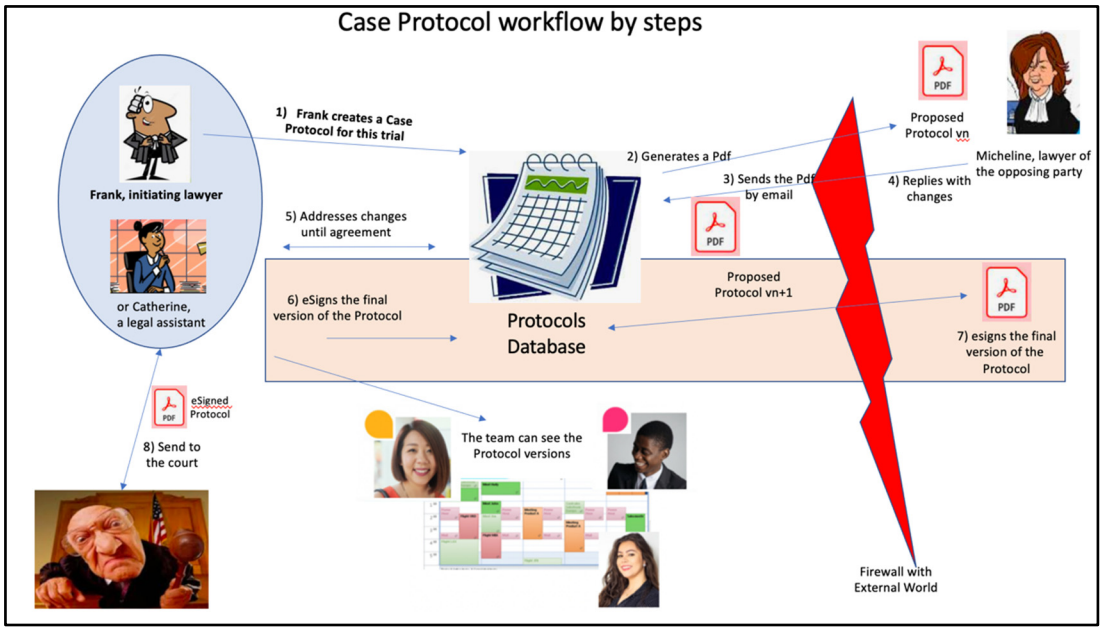

Figure 0.1 Typical workflow for an electronic document exchange for Protocols

In Figure 0.1, a case protocol is generated as a PDF document and sent as an attachment to an email seeking the approval of proposed dates. At step 7, the opposing party lawyer (which could include many different lawyers from different firms) needs to sign the approved protocol before it is sent, at step 8, to the judge. Today, the signatures are placed on the protocol in many ways. Some sign and scan the last page of the protocol, while others will e-sign the document using a signature technology. Some will even use their smartphone camera to take a picture of the last page of the protocol once signed. Since this process allows many different means of signing the protocol, this thesis considers the final signature process and researches an approach for an efficient e-signature process with associated technologies to address these many use cases.

The next chapter presents an overview of the e-signature domain and the research plan of this thesis using Basili's framework, which is helpful for planning and defining applied software engineering research projects. A validation method for the case study experimentation is presented followed by a description of the organization of this thesis.

CHAPTER 1

ELECTRONIC SIGNATURE OVERVIEW AND RESEARCH PLAN

1.1 Introduction

This introduction presents a short overview of e-signature challenges followed by the research plan which is presented using Basili's framework (V. R. Basili et al., 1986). The last section describes the structure of this thesis.

Using the internet helps to simplify interactions (L. Zhu & Zhu, 2012). Consequently, as the use of the internet spreads and digitalization grows, the use of traditional paper documents and handwritten signatures is changing. In the law profession and many other professions all over the world, open-source and commercial solutions are offered that provide many services regarding the management (workflow) of an electronic document that requires signatures (see Figure 1.1). Many steps are involved in the processes of preparing, signing, acting on and managing a document electronically.

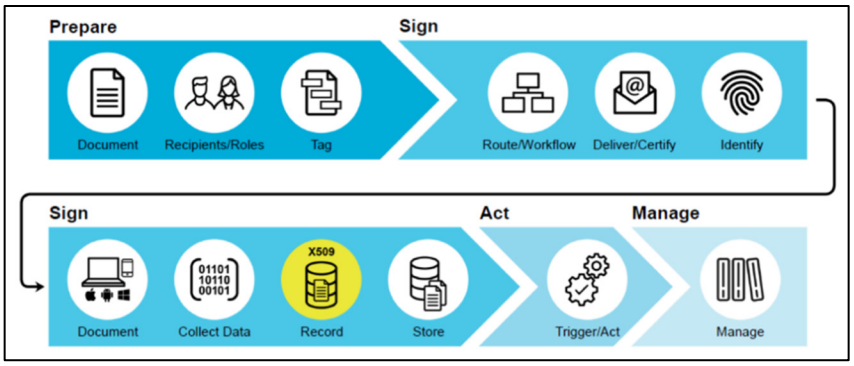

Figure 1.1 Example of DocuSign e-signatures document workflow
Taken from Wadvani, Pappas, & Rippel (2019)

Since e-signatures on a document are considered a guarantee of an agreement, and to ensure it is legally valid, an e-signature must satisfy three conditions: first, it must be bound to a

document or message, and it is not valid for any other message or document; secondly, it is associated to the signee's identity and only that signee can generate it; and thirdly, it has to be publicly verifiable, and therefore be possible to detect any later change in the signed data (Government of Canada, 2017).

Regarding implementation, by looking at recent technology trends and considering today's way of implementing software, public cloud-computing services are growing in popularity because of their many advantages. For instance, according to AWS, they provide their cloud computing services to more than 850 large companies such as Netflix, Twitch, Twitter, and Facebook. AWS provides dozens of services and capabilities at low cost that can also help small businesses, customers, and developers. This thesis deploys the whole system on AWS services to benefit from its advantages.

The next section presents the research-planning summary using Victor Basili's approach for software engineering research projects. The purpose of the section is to describe the objective, scope, and steps of this research.

1.2 Overview of Basili's framework for the thesis solution proposal

The following sections present the research plan based on Basili's framework. First the definition phase of this research is presented, followed by the planning phase, operations phase and finally, the interpretation phase.

1.2.1 Definition Phase

The definition phase, depicted in Table 1.1, is composed of the following activities:

- Describe what has motivated this research;
- Identify a high-level objective of the research;
- Describe specific objective of the research proposed; and
- Identify the potential beneficiaries of the results of the research.

This research is motivated by the fact that the current the case protocol process used by lawyers is manual and inefficient. Improving the process would benefit the industry. As a potential solution, this thesis has chosen the case protocol workflow where documents are exchanged between parties until an agreement is reached and signed by all the parties.

Table 1.1 Definition Phase of the research

I. Definition			
Motivation	Objective	Proposal	Audience
• How to improve the process of Case Protocol approvals/signatures in Canada and in Quebec	• Investigate how an e-signature process could improve this process	• Design and implement a signature transposal process that implicates many lawyers	• Students; • Researchers; • Lawyers.

The specific research proposal aims at designing and experimenting a signature transposition pipeline that consists mainly of sending an approved case protocol to be signed by many lawyers, having them e-sign the case protocol individually and assembling the many signatures on a resulting protocol (i.e., transposition of signatures) then sending it back to the originating lawyer. We have already stated in the introduction that lawyers currently use many different approaches to sign the final version of a protocol. This is caused by the fact that not all lawyers use the same e-signature technology. One of the challenges of this research is that every individual case protocol e-signed using different types of signatures needs to be transposed and each e-signature and placed on the final version of the case protocol for the judge. This requires imaging techniques be used to transpose individual e-signatures onto a final document (i.e., the final approved case protocol). A second challenge is to ensure that this process does not invalidate the signature as it alters the original documents.

1.2.2 Planning Phase

The planning phase of this research covers several state-of-the-art reviews to understand the literature and identify the existing knowledge that can be used to solve this problem. Victor Basili proposes that software engineers plan several steps before identifying a potential solution:

- Clearly identify the domains that need to be researched. In this research this includes e-signature commercial and open-source software solutions, existing law software that implement e-signatures and image transposition techniques;
- For each of these 3 domains, identify the sources (e.g., journals, conference papers, software companies and products, web sites, etc.) that will be surveyed;
- Conduct the literature review for each of the 3 domains;
- Identify the state-of-the-art techniques used today: 1) by existing law software workflow and e-signature software; and 2) image processing techniques to transpose e-signatures from one document to another;
- Determine proposed processes, workflow and technologies;
- List the deliverables planned during this first phase of the research.

The planning phase for this research is summarized in Table 1.2.

Table 1.2 Planning Phase of the research

II. Planning		
Project steps	Inputs	Deliverables
Conduct a state-of-the-art review of e-signature techniques and available software solutions	Journals, conference papers, websites, and other sources	Identify e-signature technology that could be used by this research (Chapter 2 of the thesis)
Conduct a state-of-the-art review of e-signature applications and imaging techniques that could be used to transpose e-	Discussions with lawyers, review of journals, conferences, image AI algorithms and frameworks	• Describe examples of types of signatures that will be encountered; • Identify promising imaging techniques that could be

signatures from one document to another		used in this research (Chapter 2 of this thesis).
Present a short introduction of the serverless software architecture offered by AWS and its advantages	Journals, conference papers, websites	Document the advantages of using serverless technologies on public clouds (Chapter 2 of this thesis)

1.2.3 Operations Phase

The next phase proposed by Basili for planning software engineering research (see Table 1.3) represents the core contribution of the research and consists of proposing a solution to the problem and conceiving a software prototype to prepare the experimentation of the proposed solution. Each step is described below:

1) Using the conclusions and recommendations of the previous phase (discussed in Chapter 2), propose a potential solution by first confirming the proposed workflow that the solution will implement; second, use different imaging techniques to transpose the e-signatures onto the approved case protocol; and third, design and implement an e-signature solution that will be tested on an AWS based serverless experimental prototype considering the imaging technique and proposed workflow. Finally, prepare the case study by identifying the goal and how to measure its success;
2) The second step is to execute the case study; and
3) Document the results of the case study.

Table 1.3 Operations Phase

III. Operations		
Development of a potential solution and case study preparation	Execution of the case study	Results of the operations steps
• Confirm the workflow to be implemented in the prototype; • Design a software prototype; • Develop the software prototype (*using the already identified technologies*) while experimenting imaging techniques; • Validate the software prototype; • Identify the experimental measures to be taken during the case study; • Prepare the case study to be executed.	• Execute the case study; • Produce and collect data during the case study experiment; • Validate the results; • Document the case study results.	• Propose an e-signature pipeline of image processing techniques; • Create a design diagram of the proposed serverless prototype; • Document the software prototype tests results; • Case study measures, observations, and results; • Represent the case study results graphically; • These elements are all part of Chapter 3 of this thesis.

1.2.4 Interpretation Phase

Finally, the last phase of this research is the interpretation of the results of the experimentation. This is done by conducting an analysis of all the findings and feedback (see Table 1.4).

Table 1.4 Interpretation Phase

IV. Interpretation		
Interpretation of results	Extrapolation of results	Future Work
• Discuss the results obtained in the limited context of the experiment (Chapter 4 of this thesis)	• Discuss the results obtained in the context of the potential of this proposal to be used on a larger scale (Chapter 4 of this thesis)	• Identify future work

First, the case study results are interpreted in the context of this specific experimentation. Then these results are extrapolated to understand if they could be applied generally for all case protocols in Canada and Quebec. Finally, future work is presented. The next section presents the validation method proposed for the experiment used in this research.

1.3 Experimental validation method proposal

Each research project must identify how it will validate its findings and results. This research will use a case study to validate its results and conduct a two-step validation approach: internal and external validation.

4) Internal validation: several internal validations will be done during the design and construction of the software prototype as they consist of typical software development tests (e.g., design review, unit, system, and alpha tests). This will be done to conduct a validation of the proposed workflow, e-signature status tracking and the test of the imaging technique that the software prototype will use. Iterative improvements, resulting from these tests will be made as the solution evolves towards a functioning solution.
5) External validation of the proposal (i.e., involving the opinion/feedback of a lawyer) will be done in two steps:
    a) Present the proposed case protocol workflow for e-signature and obtain a lawyer's opinion on its validity for this use case. Results of this interview will be reported in Chapter 2;
    b) Automate the proposed workflow in an AWS serverless software prototype and ask the same lawyer to experiment the e-signature process using the software prototype.

The next section presents how this thesis is organized.

1.4 Organization of the thesis

The introduction of this thesis provided a general overview of the research domain and the problem it intends to study. It was followed by a research plan based on Victor Basili's software engineering research methodology. Chapter 2 will present a literature review on: 1) the state of information technology in the law domain as well a description of the current case protocol process challenges and the commercial products typically used today; 2) a short review of opensource and commercial e-signature solutions that could inspire the research; 3) the e-signature process including the image processing technology involved; and finally, 4) the future of cloud based serverless applications. Chapter 3 proposes: 1) a process for e-signature adapted to case protocols; 2) an e-signature pipeline of image processing techniques; 3) the architecture of a software prototype using serverless technologies on AWS to experiment a potential solution; and 4) introduces a case study to experiment the solution. In Chapter 4, the case study results and limitations are presented, as well as future research directions that also involve experimenting with image processing technology to transfer e-signatures and a final summary of the results of this research are described. In the next chapter, we present the literature review.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides an overview of the current state of electronic signature (e-signature) applied to our specific use case. It aims to also present an overview of image processing techniques that could be used in our case study. The chapter starts by introducing the Government of Canada regulations that govern the use of electronic documents and signatures, and that applies in Quebec. This section provides an understanding of the legal framework within which e-signature software must operate in this country. The second section (sections 2.3 to 2.8) reviews a selection of both commercial and open-source e-signature solutions to assess their potential use in solving our use case. This review provides insights into the various approaches that these existing solutions use to address the e-signature process. This short overview is structured in a way that highlights the key features and benefits of each solution allowing us to compare and evaluate them.

The third section (section 2.9) presents the case study, explaining its objective and some of the edge cases that must be considered in the proposed solution. This section provides the context for the discussion of the potential image processing techniques that could be used to resolve our problem, which is presented in the fourth section (sections 2.10 to 2.14). This section provides an overview of potential techniques that process images where lawyers' signatures are present and explains why these techniques have the potential to solve the issues identified in our case study. Finally, the last section (section 2.15) provides an overview of the AWS cloud technologies that will be experimented, in a proof of concept, as a potential solution in this research.

2.2 Introduction to e-signature regulations in the Law domain in Canada

In 2017, the Government of Canada (GC) provided guidance on e-signatures. This document still applies today and should be considered when implementing an e-signature process, solution, or service in an organization in order for these signatures to be regards as legal. The regulations indicate that if a signature is paper-based or electronic, the fundamental purpose of the signature should be the same. A signature links a person to a document (or transaction) and typically provides evidence of that person's intent to approve or to be legally bound by its contents. The primary function of a signature is to provide evidence of the signatory's identity, intending to sign a binding agreement (Government of Canada, 2017).

There are several ways to implement and design an e-signature. Part 2 of The Personal Information Protection and Electronic Documents (PIPEDA) (Government of Canada, 2000) defines an e-signature as "a signature that consists of one or more letters, characters, numbers or other symbols in digital form incorporated in, attached to or associated with an electronic document." Essentially, an e-signature can be virtually any form of electronic representation that can be linked or attached to an electronic document or transaction, including:

1) user authentication to an internal application to approve something, such as when a supervisor logs into an application to approve a leave request;
2) using a stylus on a tablet touchscreen to write a signature by hand and capture it in electronic form;
3) a typed name or signature block in an email;
4) user authentication to access a website, coupled with a mouse click on some form of acknowledgment button to capture intent;
5) a scanned handwritten signature on an electronic document; and
6) a sound such as a recorded voice command (for example, a verbal confirmation in response to a question).

As stated above, any kind of signature is acceptable as long as it links a person to a document. In our case study email is used for authorization by each signee, any kind of signature, including e-signature, that is used by the signee is legal.

Section 37 of PIPEDA explicitly mentions that “The electronic document is retained for the specified period in the format in which it was made, sent or received, or in a format that does not change the information contained in the electronic document that was originally made, sent or received”. Also, section 42 part b of PIPEDA (Government of Canada, 2000), explicitly states that: “The electronic document contains a secure electronic signature that was added when the electronic document was **first generated in its final form and that can be used to verify that the electronic document has not been changed since that time**”.

Regarding the above rules, the thesis case study requirement is to identify any type of signature (handwritten, e-signature, etc.) apposed on the case protocol by an individual signee and then transpose it onto the final case protocol document. Therefore, the main obligation of the send, receive, and retain of a legal document is to make sure that no changes are made during the signature process.

To conclude, to ensure the proposed solution does not invalidate the signature process, two factors should be considered. Firstly, link each person to a signature. Secondly, make sure that document contents will not change during sending, receiving, and retaining.

2.3 Review e-signature technologies that could be used in this research

E-signature technologies are available either from commercial vendors or as open-source software. The next sections present an overview of solutions available in these two markets.

2.4 Commercial e-signature solutions

Commercial e-signature solutions are provided by third parties that offer electronic e-signature as a service as well as commercial services. These commercial e-signature solutions are not open-source, and they are not free. Open-source e-signature solutions are also available. Investigating commercial solutions gives us insight into the functionalities they perform which could help with the design of a solution for this research project. In the following sections, several popular commercial products and open-source software are discussed.

2.5 Review of commercial e-signature functionalities

As stated in section 2.2, the proposed case study will send and collect signatures from signees. This function is also known as the Transaction Management Process. Closing Folder software, which is part of IManage, uses this term to describe this function. To comply with Government of Canada (GC) rules concerning electronic documents, each user of this type of commercial solution must be authenticated and the document must maintain its integrity. The types of software therefore provide four functions:

1) Authentication;
2) Integrity;
3) Transaction Management; and
4) API's to interface with the solution.

After reviewing and investigating commercial e-signature solutions, six applications were reviewed and functionalities that could be used for this research project were identified.

Table 2.1 Commercial functionalities

Name	Authentication	Integrity	Transaction Management	API
DocuSign	✓	✓	×	✓
IManage	✓	✓	✓	✓
HelloSign	✓	✓	×	✓
Adobe	✓	✓	×	✓
ESignature	✓	✓	×	×
SignNow	✓	✓	×	✓

Table 2.1 shows the features required for the case study. An × means does not support that functionality. A ✓ indicates support for those functionalities. As is shown in Table 2.1, IManage provides all four of the requirements that the case study needs. However, there is a problem with using this tool. If lawyers want to manually sign the document, IManage does not support that method.

2.6 Open-Source e-signature solutions

The purpose of this section is to review open-source solutions that could be utilized during implementation of the case study based on several criteria: 1) the availability of e-signature functionality; 2) alignment with Canadian legislation requirements; 3) how active the community of developers is; 4) programming language; and 5) the licensing model offered to enable us to utilize the software. Open-source e-signature software is freely available. The most popular candidates were considered and scrutinized. The following sections describe the most popular candidates.

2.7 Comparison of open-source e-signature solutions

The research aim is to develop an e-signature proof of concept prototype that operates on AWS serverless technology and meets the rules and regulations imposed by the Government of Canada so that documents are considered legal. To assess which open-source solution could be used in this research and to get some insight regarding the implementation and functional features they offer, five criteria have been chosen to evaluate these e-signature open-source solutions:

1) Functionalities:
    a) Transaction Management;
    b) API.
2) Alignment with the Canadian regulations:
    a) Authentication;
    b) Integrity.
3) Active open-source community:
    a) Number of forks;
    b) Number of stars;
    c) Number of commits;
    d) Number of releases;
    e) Latest release;
    f) Open Issues.
4) Programming language;
5) Licensing model.

As Table 2.2 shows, open-source projects do not provide all of the functionalities required such as transaction management (the ✓ indicates the criteria is met and the × indicates it is not met). Moreover, they are not compatible with cloud architecture. In addition, all these tools provide e-signatures, so in the case that someone wants to print the document, sign it and scan it again, these tools are not appropriate for the case study.

Table 2.2 Open-source functionalities overview

Name	Transaction Management	Authentication	Integrity	API	Programming Language	Licensing Model	Forks	Stars	Commits	Releases	Final Release	Open Issues
Odoo	×	✓	✓	✓	Java Script Python	GPL v3	15.7k	24.1k	15k	16	10/21	2104
SignServer	×	✓	✓	×	Java	GPL v2	2	4	8k	64	04/21	×
OpenKM	×	✓	✓	×	Java	GPL v2	221	462	238	9	06/21	15
FOXopen	×	✓	✓	×	Java	GPL v3	27	47	764	4	07/16	0

2.8 Open-source and commercial e-signature solutions conclusion

Sections 2.5 and 2.7 mentioned functionalities which are necessary for the case study. The solution needs to be able to handle handwritten signatures and images that are taken with a phone as well as those included in PDF files. After reviewing all the available commercial solutions, none of them met these particular requirements. Therefore, although reviewing these products gave us insight into the big picture of what would be appropriate for the case study, it is not possible to use these solutions directly. In addition to commercial solutions, open-source solutions have some shortcomings which make them ineffective for thesis case studies. Most of them are deprecated and no longer supported, making them obsolete. Moreover, none of them use serverless architecture and are cloud compatible. Lastly, they provide no transaction management service. The next section presents the case study goals and requirements.

2.9 Case study objective

For this specific research case study, a PDF document containing the protocol to be signed is sent by email for signature to every lawyer involved. After opening the PDF file and signing the protocol, the lawyer replies to the email and sends back the signed document. A first challenge here is to receive each signed PDF and transpose the signatures onto a final copy. Figure 2.1 describes this process.

On the left side of the figure, each lawyer has signed his or her copy of the protocol. On the right side of Figure 2.1, the system must transpose each signature onto a final signature page where all the signatures appear. To do this transposition of individual signatures, an image classification or categorization task which determines whether handwriting is preent or not is required. Note that the documents received may include several types of signatures, i.e., these signatures can be made using Adobe Acrobat, DocuSign, SignNow or by other means. Furthermore, some of the signatures can be handwritten and then photographed or scanned.

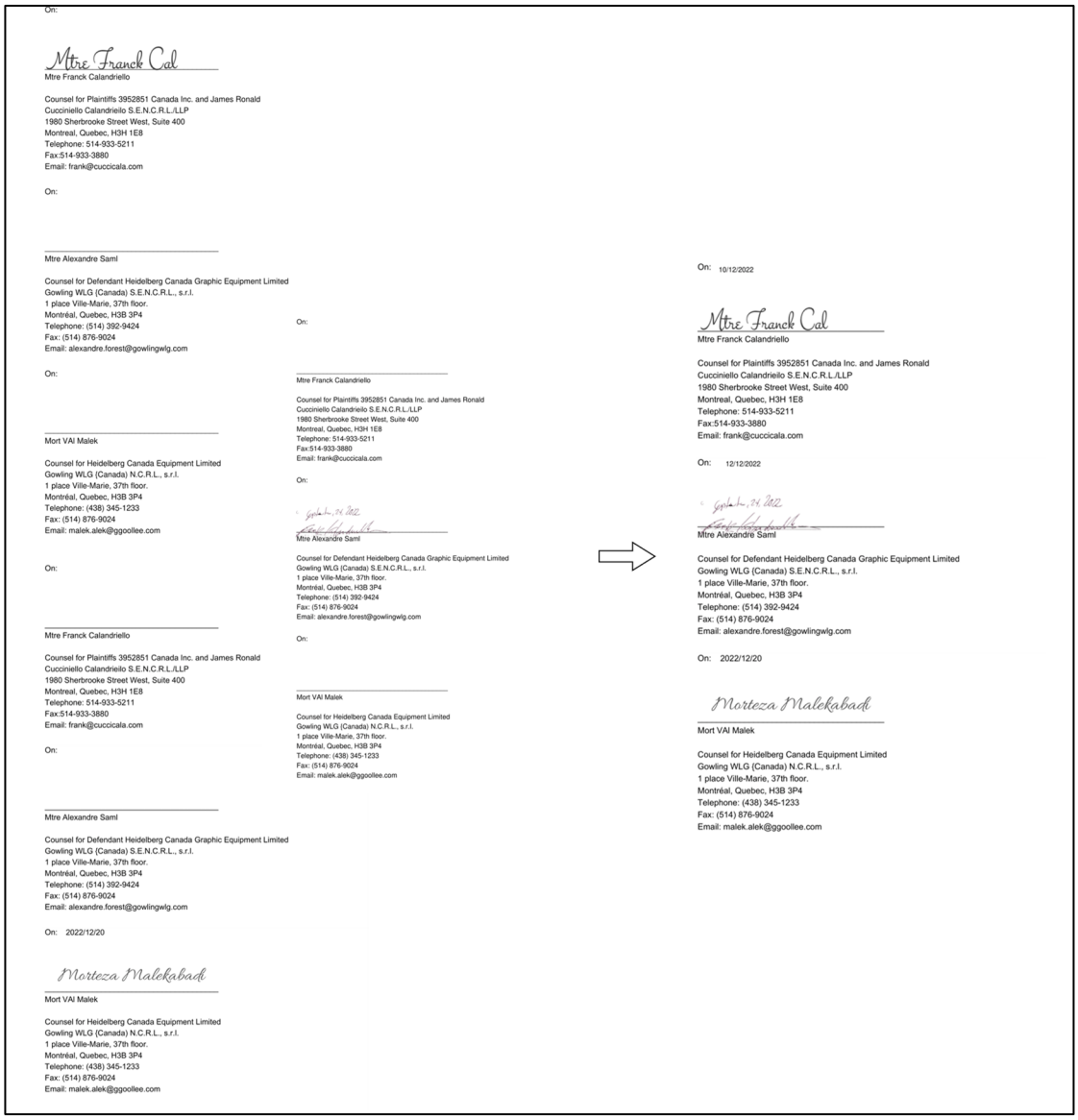

Figure 2.1 Project overview, inputs, and output

The following figures illustrate three examples of particular signature situations.

1) Signature of the protocol using electronic signature tools (Figure 2.2)


On:

Mtre Franck Cal

Mtre Franck Calandriello

Counsel for Plaintiffs 3952851 Canada Inc. and James Ronald
Cucciniello Calandrieilo S.E.N.C.R.L./LLP
1980 Sherbrooke Street West, Suite 400
Montreal, Quebec, H3H 1E8
Telephone: 514-933-5211
Fax:514-933-3880
Email: frank@cuccicala.com

On:

Mtre Alexandre Saml

Counsel for Defendant Heidelberg Canada Graphic Equipment Limited
Gowling WLG (Canada) S.E.N.C.R.L., s.r.l.
1 place Ville-Marie, 37th floor.
Montréal, Quebec, H3B 3P4
Telephone: (514) 392-9424
Fax: (514) 876-9024
Email: alexandre.forest@gowlingwlg.com alexandre.saml@gowllngwlg.com

Figure 2.2 E-signed document

2) Signature of the protocol using handwritten signature and scanned document (Figure 2.3)

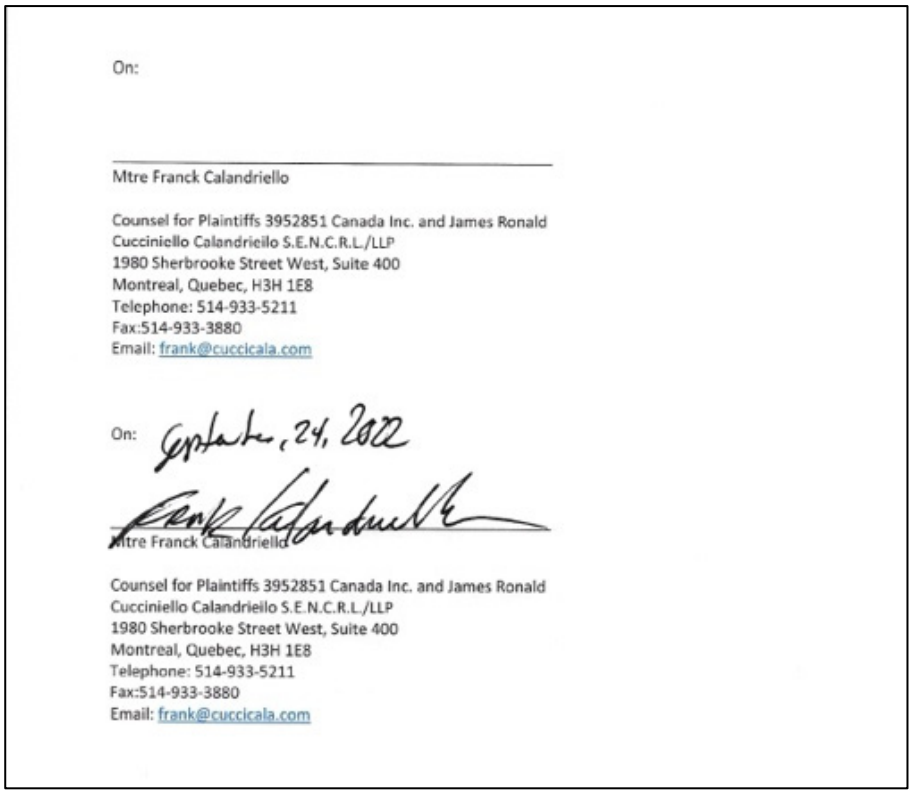

Figure 2.3 Handwritten-signed scanned document

3) Signature of the protocol using handwritten signature and a photo of the page (Figure 2.4).

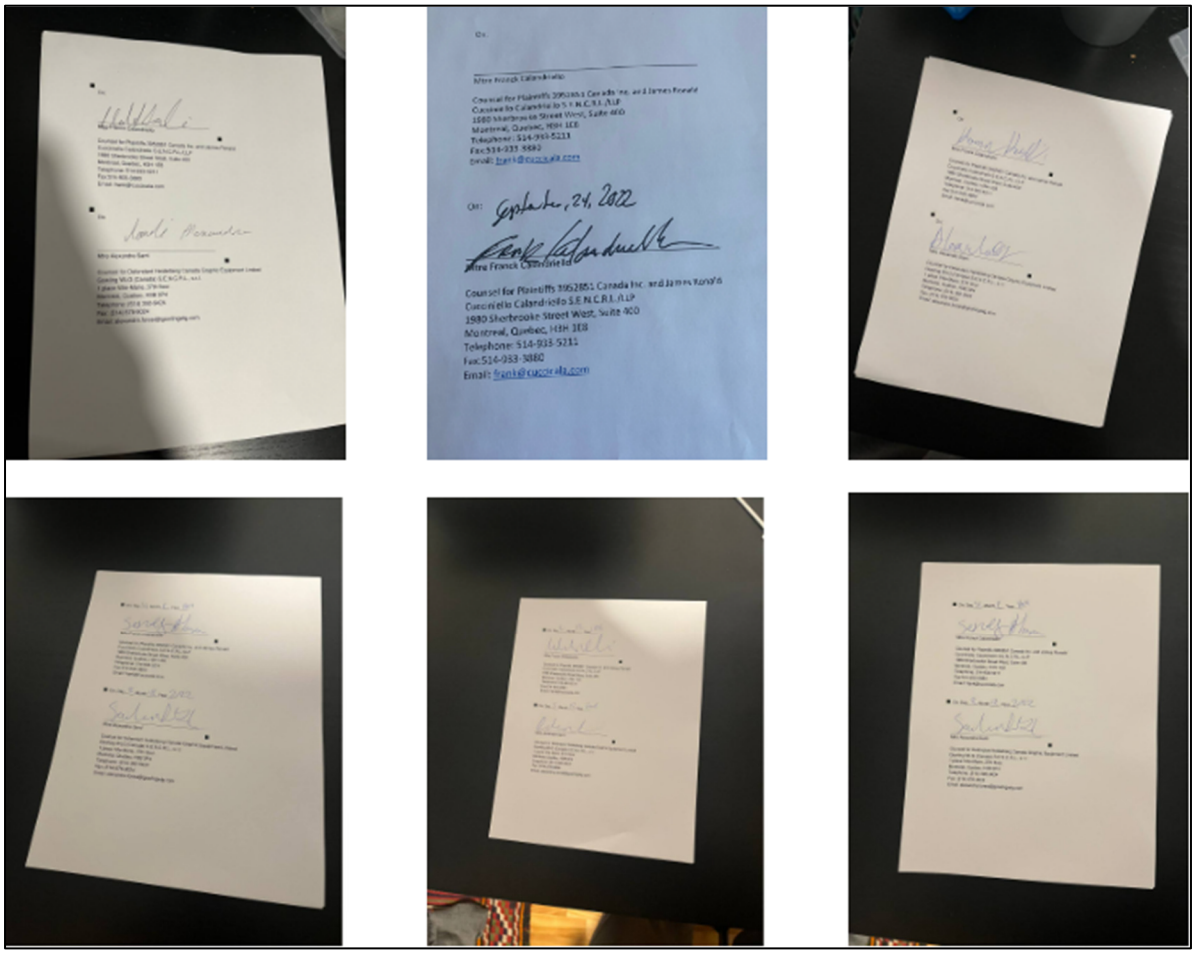

Figure 2.4 Examples of handwritten signed and photo taken document

As the different situations show, the automated signature transposition can become challenging. The main challenge of this research project is centered around extracting the signature from these many types of situations. Proposed approaches to solve this problem are discussed in the next chapter along with how these approaches perform, their drawbacks and advantages.

2.10 Potential solutions

In this section, an overview of the potential solutions that were identified and taken from the state of the art are presented. In the next chapter, a more in-depth examination and discussion of the capabilities and limitations of each of these potential solutions will be provided. The goal of this literature review is to identify potential solutions that best meet the requirements of the case study and can satisfy our use cases. A closer look at each potential solution will allow for the assessment of their strengths and weaknesses. Ultimately, the information gathered in this literature review will be used to make an informed decision on which solution to experiment for this research project. Below is an initial list of potential solutions:

1) Use of existing image processing libraries;
2) Use of an image coordinates technique; and
3) Use of an image contouring technique.

2.10.1 Use of existing image processing libraries

After reviewing the literature and searching for existing image processing solutions, several existing image processing libraries were identified as potential solutions. These are: Apache PDFBox (https://pdfbox.apache.org/), IceBlue (https://www.e-iceblue.com/), Aspose (https://www.aspose.com/) and PDFtron (https://www.pdftron.com/). A first issue is that these libraries are all commercial libraries and consequently they are not free for use. In addition, after experimentation, they did not completely meet the case study requirements. For example, if the document received is not in a PDF format or if the signature is handwritten and then scanned, these libraries will not work. When experimenting using these libraries, the edge cases presented could not be satisfied.

2.10.2 Use of image coordinates technique

After considering the use of existing image processing libraries, other avenues had to be investigated. The literature proposes that image coordinate techniques have a potential to solve

the issues identified (Figure 2.5). After converting PDF files to images and considering the images as a cartesian plane, it could be possible to store the coordinates of all signatures in a database. This is possible because the PDF sent to each lawyer is created by our automated system and is consistent every time. Using this technique, each pixel would be considered as a unit of a cartesian plane. In this potential solution, after receiving the signed document, the coordinates could be retrieved from this database and the image could be cropped based on the coordinates. However, we think that this technique will not be applicable in all edge cases identified, for example, if the received file is an image, or if the picture is angled, or if the scanned document size is smaller or larger than the original document. Hence, this method could only be beneficial if the format and the size of the file does not change from the original file sent for signature.

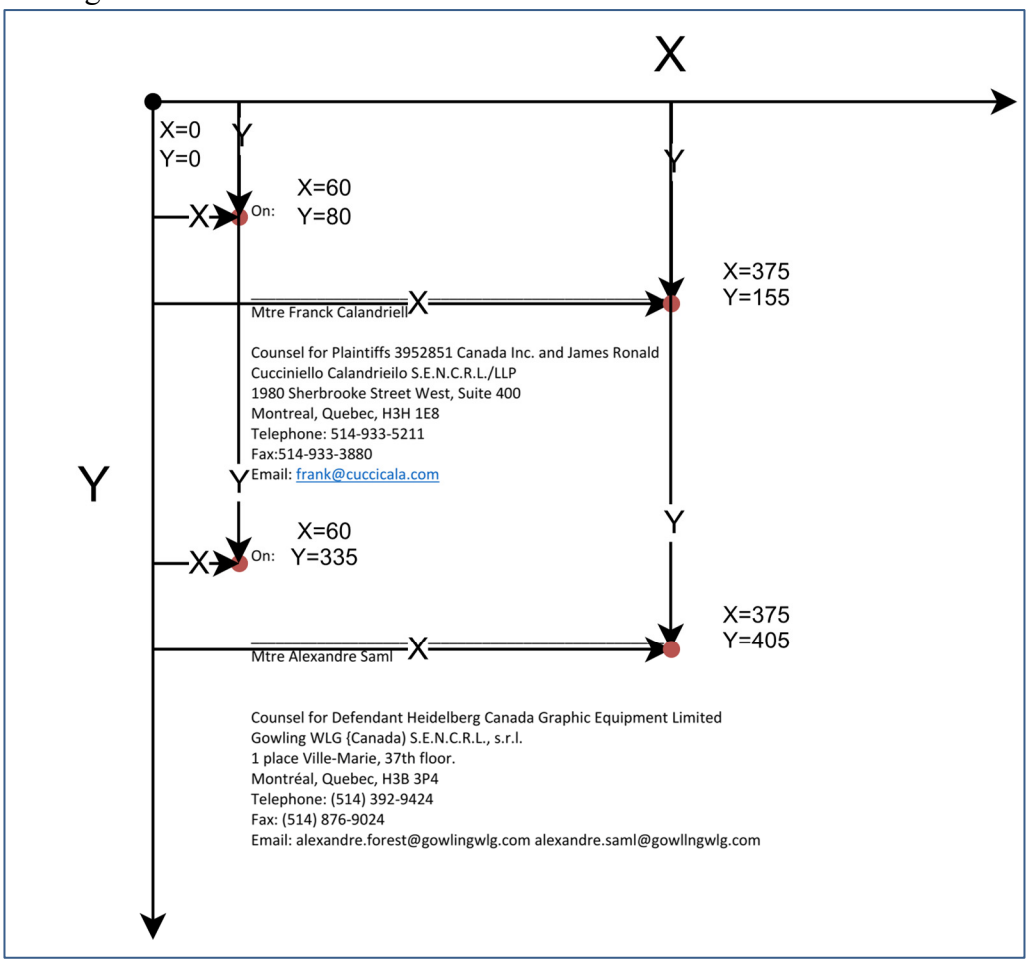

Figure 2.5 Example of Coordinate Plane on Document

2.10.3 Use of image contouring technique

An alternative to the coordinates technique, and to enable the future system to be more robust, contours of the signature can be considered. Image contouring is the process of identifying structural outlines of objects in an image. In this proposed approach, each signature location is identified using two black squares inserted in the background of the signature (see Figure 2.6).

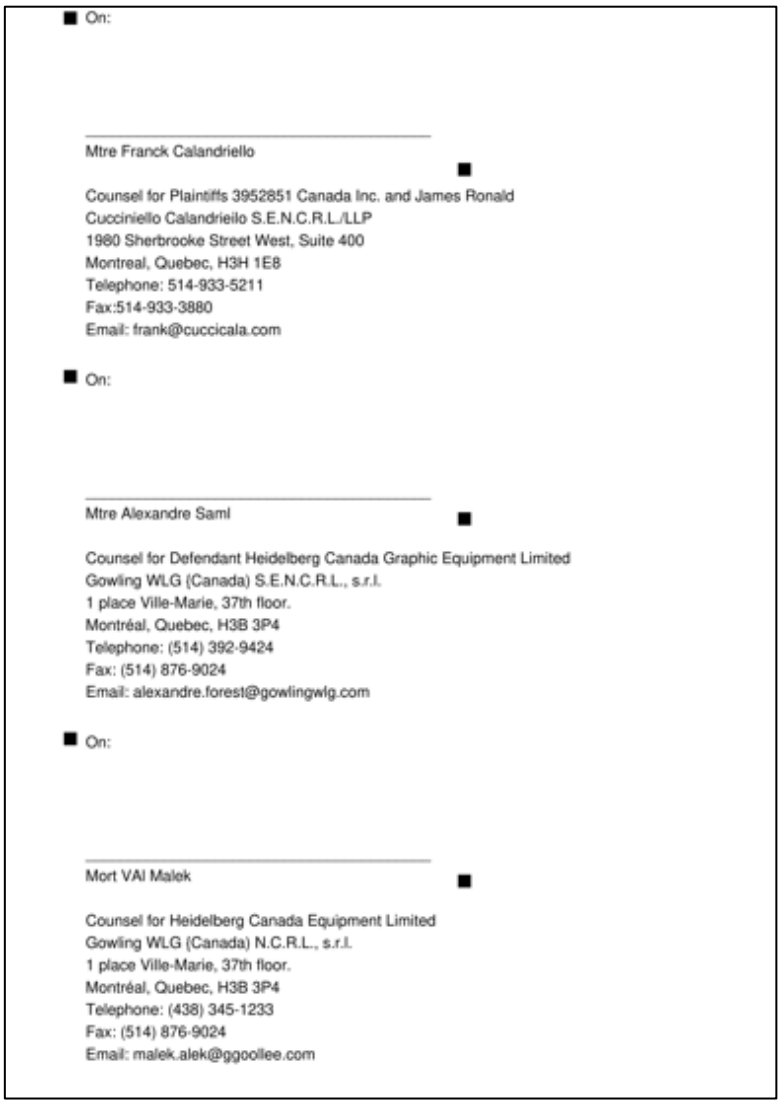

Figure 2.6 Sample document with contours

These black squares can be considered as contours. Contours can be a useful tool for shape analysis and object detection and recognition. As shown in Figure 2.6, it is possible to add small black squares to the document that is sent. If the lawyer uses that same document and signs it, it is possible to locate these squares and crop the part of the document containing the signatures. Hence, the purpose of adding these contours is to find the coordinates of the signature without storing them in a database. Considering the many situations presented earlier, contour finding could become a partial solution since the received documents could be either PDFs or images. Moreover, these different inputs could cause the same issues as the previous approaches. To address these issues, it will be necessary to apply image processing techniques to straighten the signed document and adjust its dimensions to a suitable size.

2.11 Image processing

The purpose of this section is to identify techniques that can be used to perform preprocessing and postprocessing to resize, fix the perspective problem, ensure the quality of the image of the signature received and clean the signature images of noise and extraneous characters. This process typically involves multiple stages (i.e., a sequence of tasks also named a pipeline), and its objective is to produce a perfectly angled, noise-free and unified image size from a signed document received from the lawyers.

As mentioned in sections 2.10.2 and 2.10.3 there are some edge cases where the signed documents from the lawyers were not successfully processed using the techniques presented. Hence, to cover those edge cases, preprocessing and postprocessing could be used to try to clean up the image so the technique is more successful.

For example, Figure 2.4 showed images of case protocol signatures that have an angle or have other objects in them. In these cases, locating the signature using the previously described techniques could cause problems.

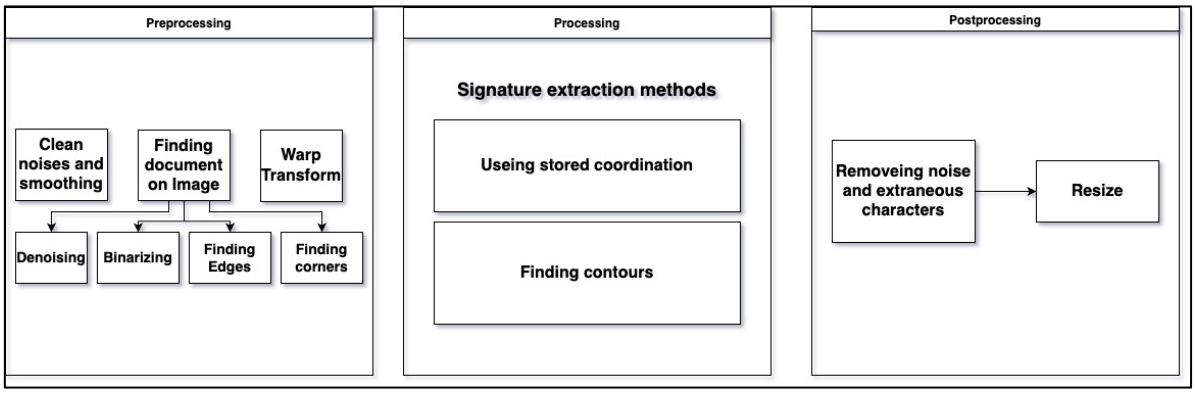

Figure 2.7 Processing pipeline proposed

A potential solution (see Figure 2.7) could be designed to try to resolve these issues by first removing other objects from the image, cleaning and extracting the edges to only keep the document, and then cropping the location of the signature. Finally, deleting all other text and contours from the cropped part of the document that contains the signature. The following subsections will cover all the image processing techniques discussed in relation to this proposed solution.

2.11.1 Denoising

Denoising refers to the process of removing noise or unwanted disturbances from an image or video. Noise in an image can be caused by various factors such as poor lighting conditions, low-quality cameras, or transmission errors during image acquisition. The goal of denoising is to enhance the quality of the image by removing these unwanted disturbances while preserving the important information in the image (Ahmad et al., 2019; Elad & Aharon, 2006).

Efficient image denoising remains a significant challenge. Despite the recent development of advanced techniques, most algorithms are not yet suitable for practical use. Although they exhibit exceptional performance when the image model aligns with the algorithm's assumptions, they tend to generally fail or eliminate fine image details (Buades et al., 2005).

There are various methods available to tackle this issue. Some popular methods include linear filters like the Gaussian filter and median filter, which smooth out noise by averaging the pixel values in a small neighborhood. Another potential solution consists of using a wavelet transform which is a mathematical technique that is effective in removing high-frequency noise by thresholding coefficients in each frequency band. Other non-linear methods like total variation denoising can also be used to preserve the fidelity of the original image while minimizing its total variation. Finally, non-local means filter is also widely used in image denoising, where it preserves edges and textures while removing noise, and preserves edges by considering both spatial proximity and intensity similarity (Fan et al., 2019).

2.11.2 Binarizing

In digital image processing, a binary image is a type of digital image that has only two possible values for each pixel: 0 or 1. In other words, a binary image consists of black and white pixels only, where black represents 0 and white represents 1 (Szeliski, 2010). Binary images are useful in many image processing tasks such as object detection, edge detection, and pattern recognition. The main advantage of using binary images is that they simplify the image data, making it easier to process and analyze. This is because the information in a binary image is reduced to a binary code that can be easily processed by a computer. In addition, binary images can be used to extract features such as the shape, size, and orientation of objects in the image, which are important for many computer vision applications (Ballard & Brown, 1982).

Binarization is an image transformation technique where the objective is to convert a grayscale or color image into a binary image. Various techniques are available for binarizing an image, and the choice depends on the type of image, lighting conditions, and the desired level of accuracy (Sezgin & Sankur, 2004). Global thresholding is a common binarizing technique that uses a single threshold value to assign pixels to either the foreground or background. Another technique called adaptive thresholding is useful for images with uneven illumination or varying contrast, as it calculates the threshold value for each pixel based on its local neighborhood.

Another technique named the Otsu's technique is a thresholding method that uses the between-class variance of pixel intensities to determine the threshold value. Edge-based and clustering-based methods are also widely used for binarization, where the former identifies the edges of objects and applies a threshold to separate foreground and background pixels, and the latter groups the pixels into clusters based on their intensities and applies a threshold to separate the clusters with the highest and lowest intensities. Finally, local thresholding divides the image into non-overlapping blocks and applies a threshold to each block. Each of these techniques has its strengths and weaknesses, and the selection of a specific technique will depend on the image characteristics and the application requirements (Pal & Pal, 1993; Sezgin & Sankur, 2004).

2.11.3 Perspective Transformation

As understood from the name of this image processing technique, a perspective transformation is associated with the change in an image viewpoint. This type of transformation does not necessarily preserve parallelism, length, and angle, but it preserves the collinearity and incidence of an image. This is called an affine transformation which is a linear mapping technique that preserves points, straight lines, and planes. This means that the straight lines will remain straight even after a transformation. As shown in Figure 2.8, for an affine transformation, the projection vector is equal to 0. Thus, an affine transformation can be considered as a particular case of a perspective transformation. Since the transformation matrix (M) is defined by eight constants (i.e., degrees of freedom), to find this matrix we first select four points in the input image and map these four points to the desired locations in the unknown output image according to the use-case; in this way we will have eight equations and eight unknowns and that can easily be solved. Once this transformation matrix is calculated, we could apply a perspective transformation to the entire input image to get the final transformed image (Croft et al., 1991; Weisstein, n.d.).

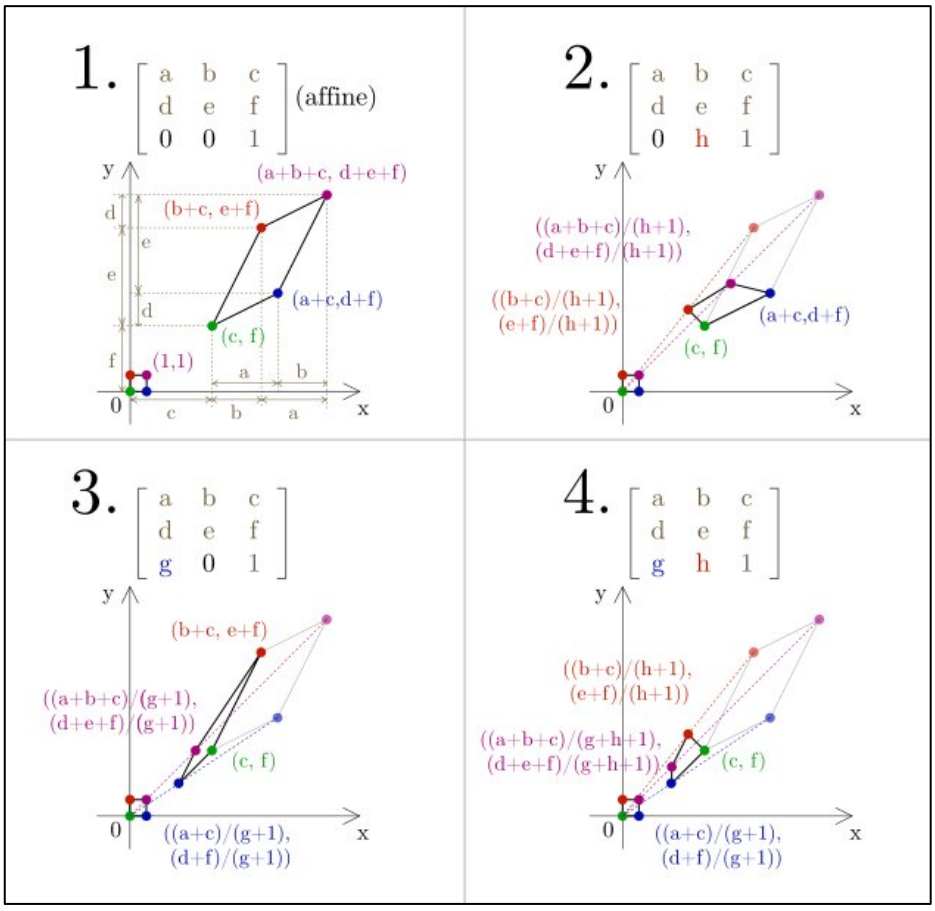

Figure 2.8 Perspective Transformations Overview
Taken form D.Stebani (2016)

To extract the signatures from documents like the one shown in the left part of Figure 2.9, it is necessary to change the perspective and the size of the image. In this image processing process, the corners must be found and the perspective changed. In the next section, the topic of finding the corner of document in the image will be discussed.

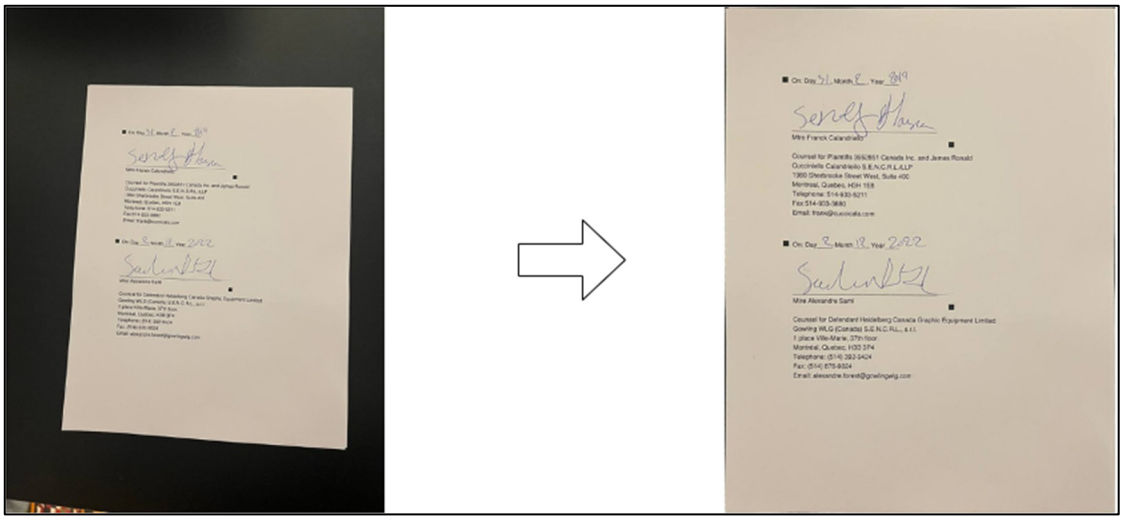

Figure 2.9 Perspective Transformation Example

As seen in Figure 2.9 the perspective transformation goal is to make all the signed documents have the same shape and size and more importantly, have the same perspective to ensure that the contour technique works all the time.

2.11.4 Edge detection

Edge detection is an image processing technique that is used to identify the boundaries between objects or regions in an image. Several techniques are available for edge detection, each with its unique strengths and weaknesses. First, the Sobel operator and Prewitt operator are two common techniques that use convolution kernels to detect horizontal and vertical edges separately. Another technique, the Roberts Cross operator, detects diagonal edges by using two 2x2 convolution kernels. Another approach named the Laplacian of Gaussian (LoG) operator applies the Laplacian operator to the Gaussian-smoothed image to detect edges of varying scales. Zero-crossing edge detection identifies the edges by detecting the zero-crossings in the second derivative of the image intensity. The Canny edge detector is a popular method that involves multiple stages, including smoothing with a Gaussian filter, gradient magnitude and direction computation using Sobel operators, non-maximum suppression, and hysteresis

thresholding to select the strongest edges (Marr et al., 1997). The choice of method depends on the nature of the image, the desired level of accuracy, and the application requirements. Edge detection is an essential preprocessing step for many image processing tasks, including object recognition, segmentation, and tracking (Raman & Aggarwal, 2009).

In the upcoming section, a technique will be discussed which aims to eliminate noise and extraneous characters from a signature image, retaining only the signature itself, as demonstrated in Figure 2.12. This method facilitates the process of transferring the signature onto a base PDF file.

2.12 Generative Adversarial Network (GAN) technique

The image-to-image translation problem is a class of vision and graphics problems in which the goal is to learn the mapping between input and output images.

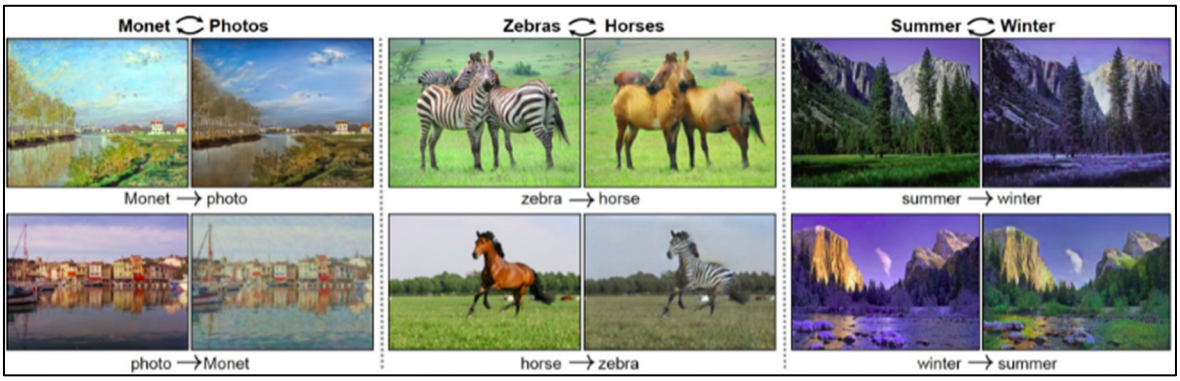

Figure 2.10 GAN Overview Example
Taken from J.-Y. Zhu et al. (2017)

With unpaired images, Cycle GAN can perform image-to-image translation. Therefore, we can translate between images in domains A and B (horse and zebra) without training the model with matching images (Figure 2.10). A large dataset of paired examples is typically required to train a model for image-to-image translation. Preparing these datasets can be time-consuming and expensive. For example, in Figure 2.11 we can see that a noisy signature was

translated into a clean signature using CycleGAN. This technique tries to improve image quality from the cropped image.

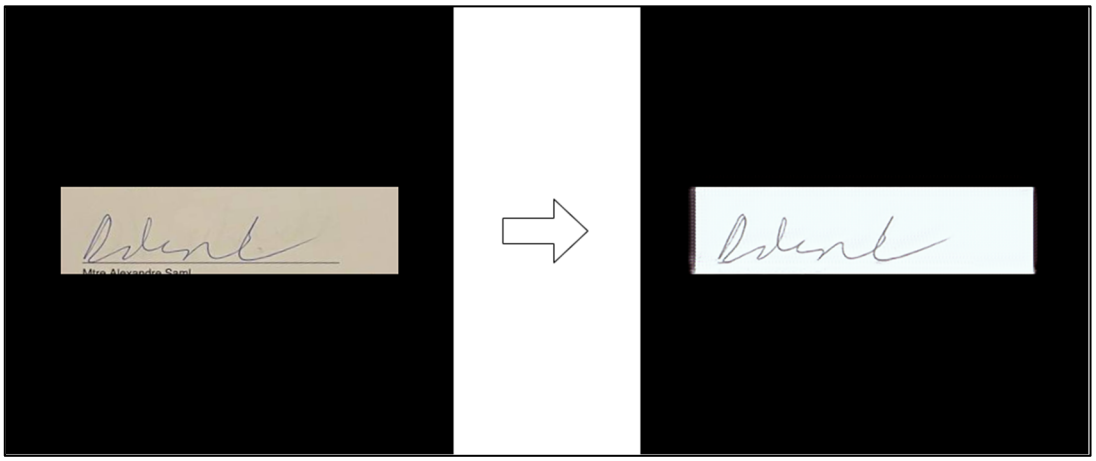

Figure 2.11 Removing noises with GAN example

Figure 2.11 shows the exact purpose of using GAN, as we can see a signature that has partially printed name and a horizontal line in the left image, which are not part of the signature. Hence, GAN removes this noise and a clean cropped image results.

2.13 Deployment of the potential approaches

In sections 2.9 and 2.10, it was mentioned that different techniques were identified and could be used to successfully extract signatures from the protocol. In this section, first, logically combining these techniques in an image processing pipeline is discussed and second, the AWS cloud technology intended to be used to experiment these techniques is presented.

2.14 Proposing an image processing pipeline

In this section, the proposed signature extraction process steps will be explained. Figure 2.12 shows how all these image processing techniques can be orchestrated.

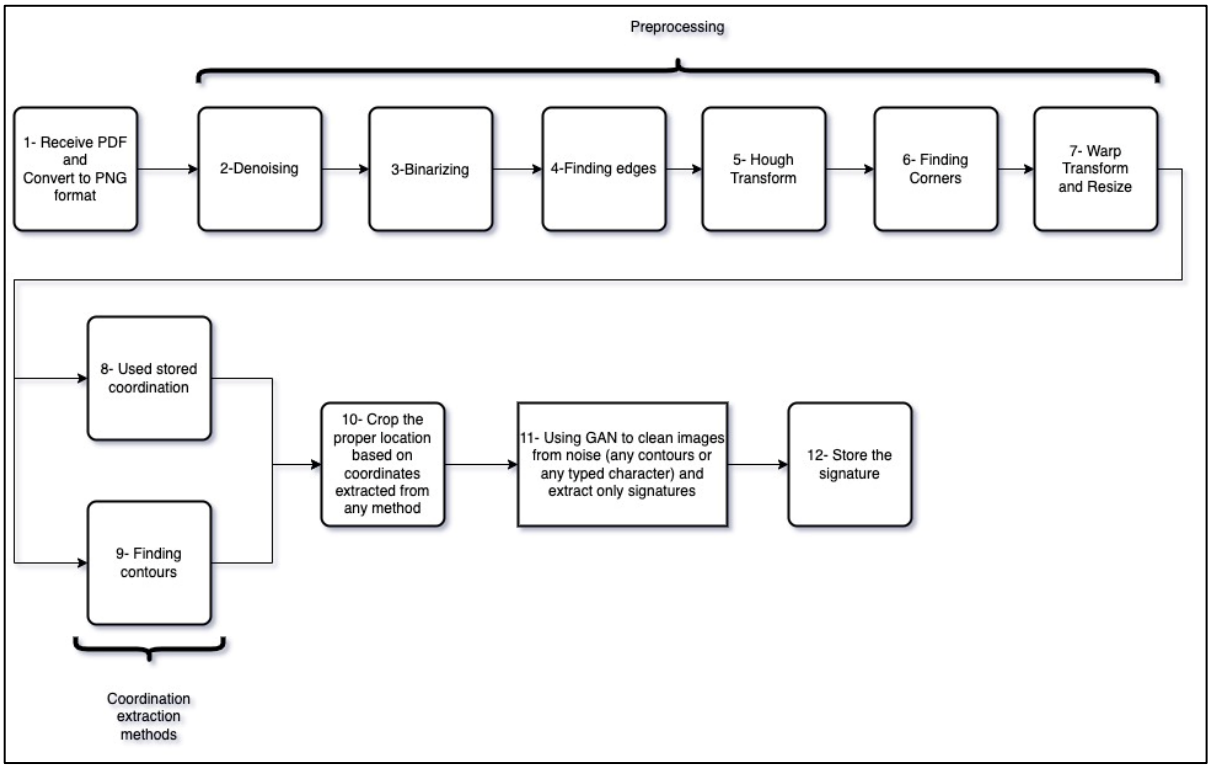

Figure 2.12 Proposed signature extraction process overview

The proposed process of signature extraction from a document includes six steps:

1) Receive PDF and Convert to PNG format;
2) Preprocessing:
    a) Denoising;
    b) Binarizing;
    c) Finding edges;
    d) Hough Transform
    e) Finding corners
    f) Warp Transform and Resize image.
3) Coordination extraction: three possible approaches experimented:
    a) Add contours to the document and find them;
    b) Store coordinates;

4) Crop the image;
5) Use GAN to clean image from any noise and characters and only keep signature;
6) Store signature image in a file (e.g., an AWS S3 bucket).

The next section will present the AWS technologies that have been chosen to experiment this signature extraction process.

2.15 Experimental cloud technology

In this research, an existing case protocol prototype software has already been developed where the proposed signature transposition solution will be experimented. This case protocol prototype has been developed as a serverless application on the AWS cloud. This section will present how the proposed signature extraction process will integrate into the existing serverless architecture of this existing software prototype.

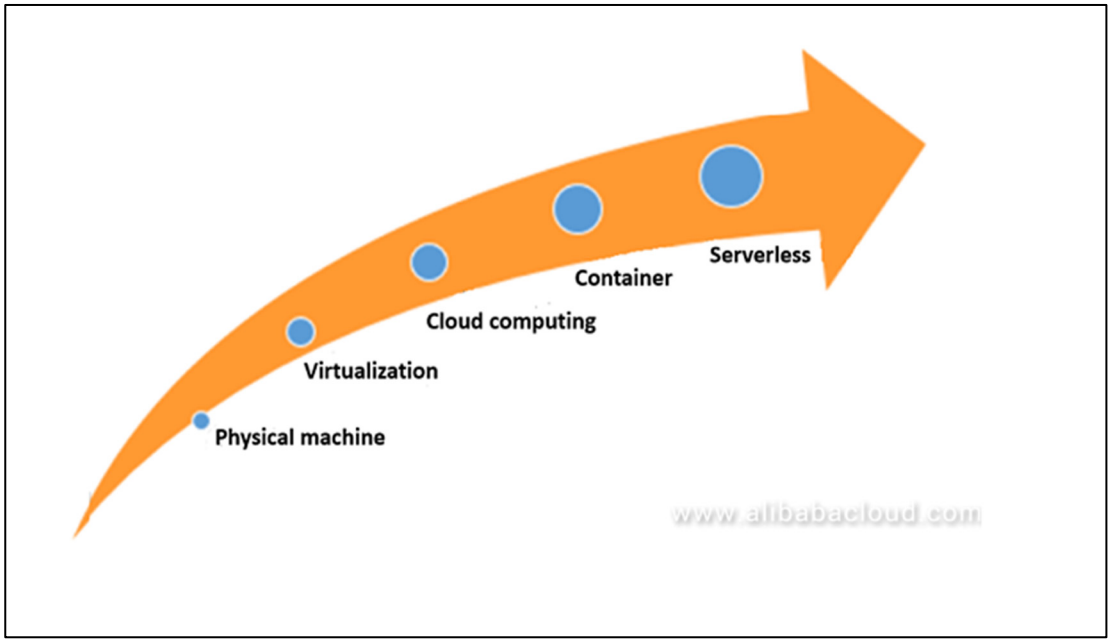

Figure 2.13 Evaluation of software architecture
Taken from DZone (2018)

2.15.1 Cloud based serverless applications

The rapid development of system infrastructure technologies, especially virtualization and the cloud, presents new opportunities and challenges in software design. Software architects, who are evolving from bare metal hardware deployments to traditional virtualization, cloud IaaS, container technology, and now cloud events, deploy software to design and implement optimized configurations (price, performance) (Baldini et al., 2017).
Cloud computing is the on-demand consumption of compute power, storage, database, applications, and all IT resources through the Internet following a pay-as-you-go pricing model (McGrath & Brenner, 2017).

Serverless computing is an execution model in which cloud service providers dynamically manage the allocation of server computing resources. Consumers are charged for the actual number of resources they consume, rather than paying for units of pre-purchased computing capacity. This model was developed to improve the scalability of applications in the cloud with optimal cost and minimal configuration work (McGrath et al., 2016; Rajan, 2018). In serverless computing, developers define a function to handle an event, and the serverless framework horizontally scales the application as needed (Al-Ali et al., 2018). Recently added to this continuum is a set of services that are classified as cloud events, specifically, new commercial services such as Amazon Web Service's Lambda. Cloud events are in many ways the next reductive step in IaaS abstraction; replacing coder's concerns about hardware and software dependencies with conceptually simpler function calls to act upon various other cloud services or cloud resident data sets (Baldini et al., 2017).

This thesis proposes a system on AWS that is compatible with the case protocol and is developed using AWS software.

2.16 Conclusion

In conclusion, this chapter provides a comprehensive overview of the current state of e-signature software. The regulations governing the use of electronic documents and signatures were introduced and explained in the first section. The second section presents a review of various commercial and open-source e-signature solutions, their key features, and benefits.
The case study has also been presented, which helps to illustrate the various issues and edge cases that must be considered when proposing a solution. This case study provides the context for the discussion of potential methods and technologies that could be used to resolve the identified issues. Potential solutions and technologies are explored, presenting a pipeline to fulfill the requirements of the case study. This exploration includes a detailed overview of image processing techniques that are relevant to solving the issues identified in the case study. In addition, an overview of AWS cloud technologies that can be used to experiment with is provided. In the next chapter, the proposed solution using AWS cloud technologies and how it can be made compatible with the case protocol is highlighted. Moreover, the experimentation process will be delved into, detailing each proposed technique, and explaining the entire application process, from file creation and reception to transposition to the main document.

CHAPTER 3

PROPOSAL OF AN ESIGNATURE SOLUTION FOR CASE PROTOCOL SIGNATURE TRANSPOSITION USING A CLOUD SERVERLESS APPROACH

3.1 Introduction

This thesis focuses on the specific activities depicted in the red square of Figure 3.1. These are part of a larger project that aims to design and develop software that generates/manages a case protocol PDF file, sends it to lawyers (i.e., the recipients), retrieves a signed protocol PDF file, extracts their signatures, and integrates all the resulting signatures into a final case protocol that can be sent to the court. All these actions are currently executed on AWS utilizing various services such as Lambda Functions for implementing the business logic, SES as the mail server, CloudWatch for monitoring and scheduling tasks, S3 for object storage, and DynamoDB as the database.

This chapter begins with a description of the design, implementation and experimentation of the image processing techniques that have the potential to solve the issues raised in previous chapters. Then we will explore how the image processing techniques will be experimented to extract signatures from individual protocols received from lawyers. The proposed processing pipeline experimentation includes three phases: preprocessing (image preparation), processing (signature extraction), and postprocessing (signature cleaning).

3.2 Existing case protocol prototype

The implemented case protocol software architecture is depicted in Figure 3.1, with a red rectangle indicating the segment that is the focus of this thesis. This segment must be integrated into the existing case protocol software which was built using AWS services, including Lambda function. To facilitate integration, the red segment must also be built using the same service.

In the proposed architecture, each Lambda function is responsible for one task to make use of microservice benefits such as scalability. In addition, each Lambda function is triggered by another Lambda function or by another AWS service. To ensure our proposed solution integrates seamlessly with the current software architecture, all the functions will be written in Java except for the extract signature function, which will be written in Python.

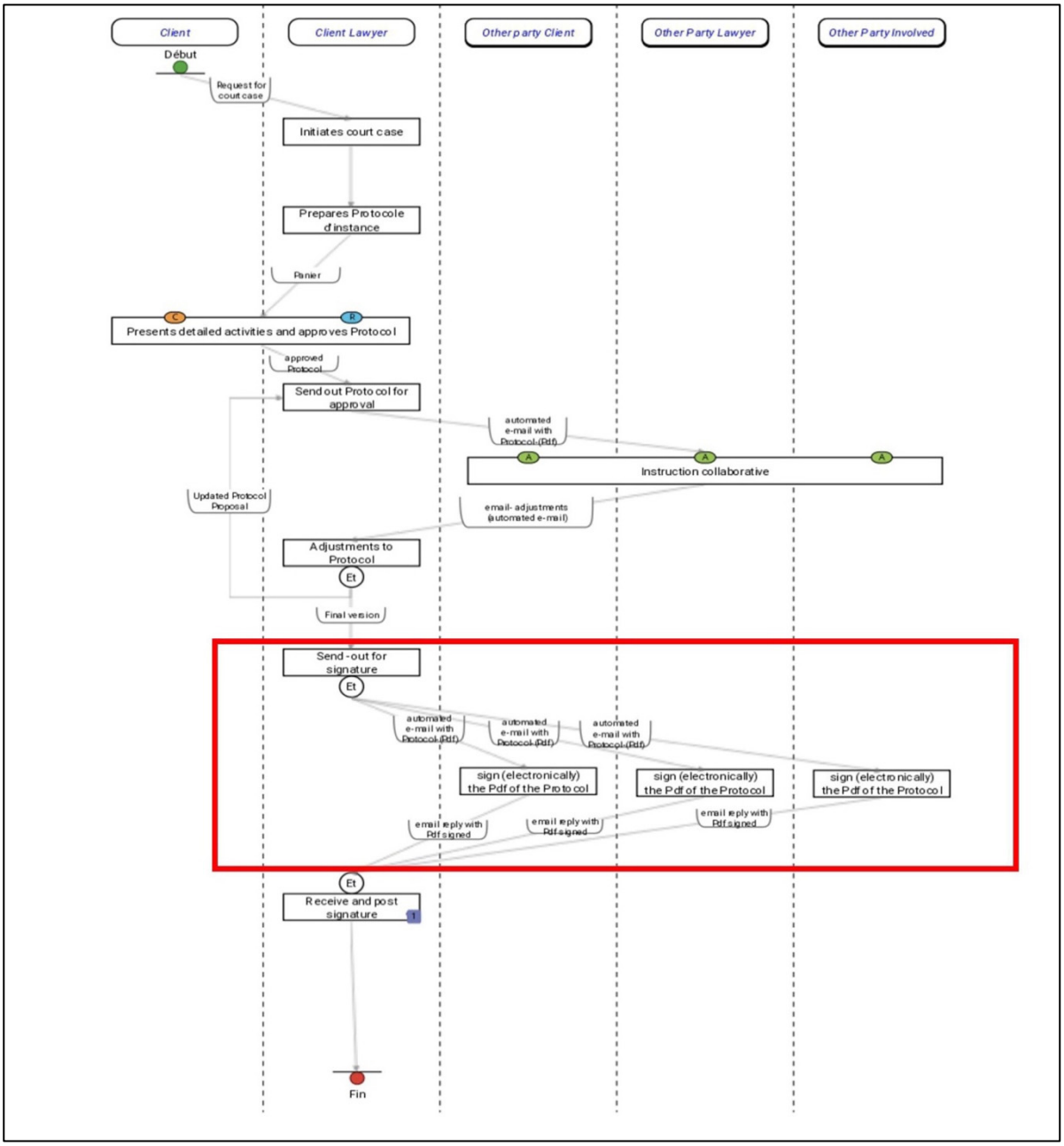

Figure 3.1 Software prototype overview

These choices are based on programming language capabilities, available libraries, and convenience. They will be discussed in the section corresponding to each Lambda function. In designing the project prototype, other AWS services will also be used. The details of all these AWS services and why they have been chosen will be explained in subsequent sections. The proposed software architecture for the thesis proof of concept is depicted in Figure 3.2, which also highlights the connections between each proposed AWS service. Finally, this chapter will provide a comprehensive overview of the entire system and offer detailed explanations for each of its components.

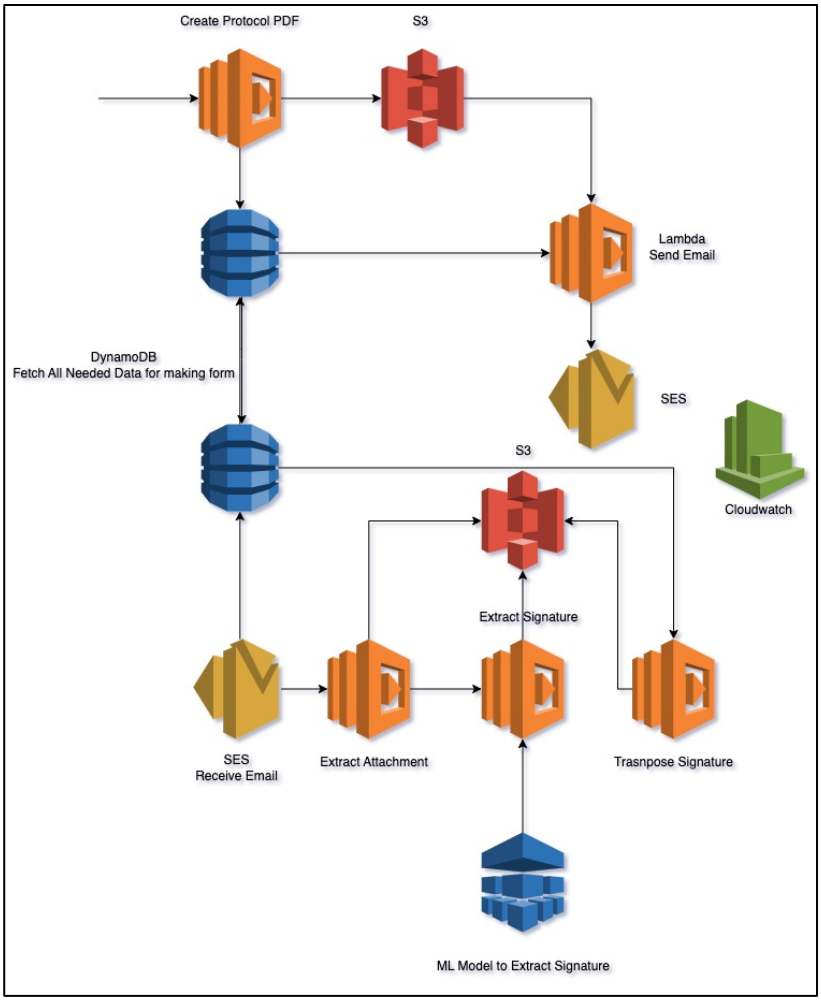

Figure 3.2 Proposed AWS software prototype architecture overview

3.3 Architecture of the proposed solution

The proposed solution software uses a serverless architecture based on the Lambda services of AWS. AWS Lambda services will also be used to experiment the signature extraction process to ensure that it integrates seamlessly. A proposed system design to be experimented is shown in Figure 3.3. The business logic of the proposed solution includes six Lambda functions.

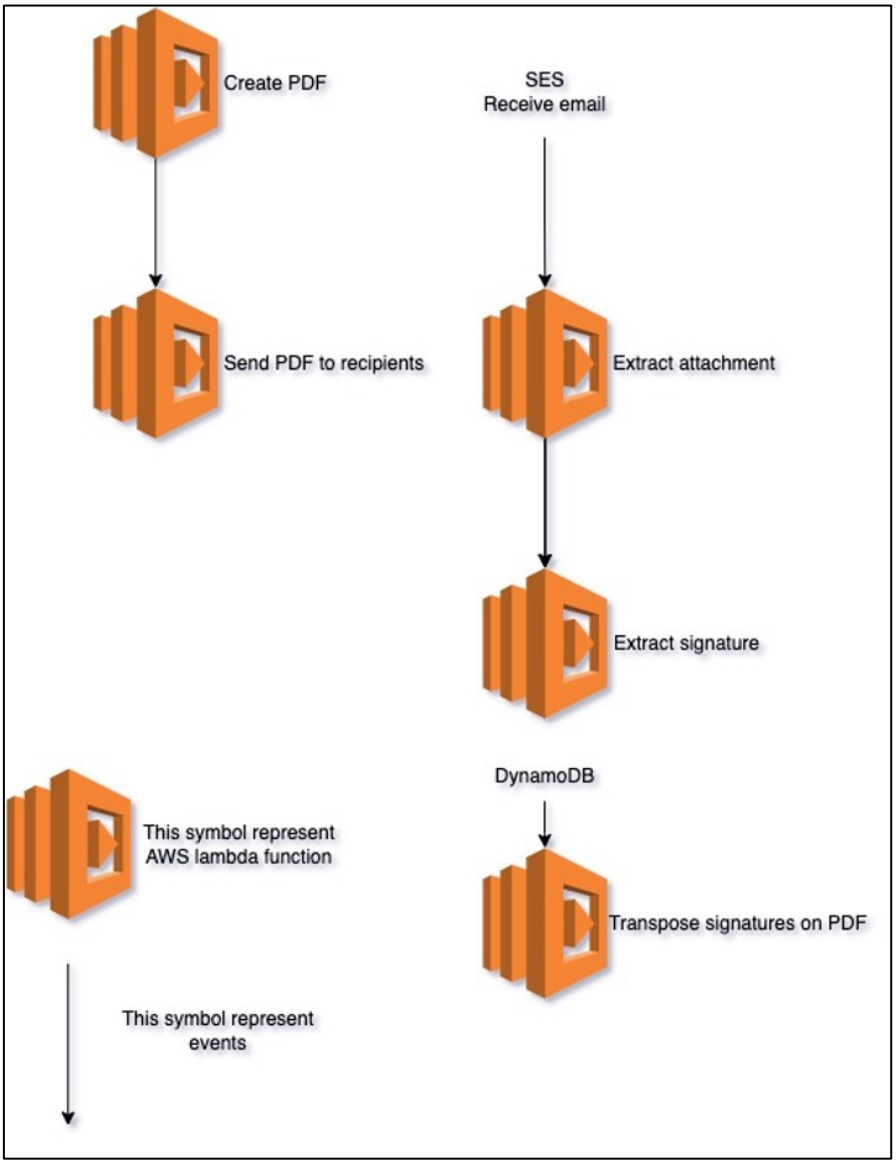

Figure 3.3 Proposed serverless system design overview

3.4 Proposed pipeline

As presented in Figure 3.1, the case study of the thesis aims to be able to automatically transpose the signatures of many lawyers onto a final case protocol to be submitted to the court. This process starts when a finalized and case protocol is sent, by email, as an attachment requiring consent and approval by signature. This email is sent to lawyers who must sign it to confirm their consent. If they consent, then they must reply to this email with their signed protocol attached. Since the process of forwarding the case protocol to lawyers is simultaneous, there is only one signature needed on each case protocol forwarded. It means that, for example, if the court protocol needs two consents, two different emails containing an unsigned case protocol will be emailed to those two lawyers at the same time. Afterward, each lawyer will reply to this email independently, with the signed case protocol as an attachment. Then, each of the signatures must be lifted from their individually signed case protocol and transposed onto a final version to be submitted to court. To achieve this, a pipeline is designed, and, in this chapter, the experimentation will be evaluated.

In Figure 2.12, a pipeline was proposed. In this section, it will be scrutinized and examined in detail, with each component being comprehensively discussed.

1) Receiving a PDF and converting it to PNG format

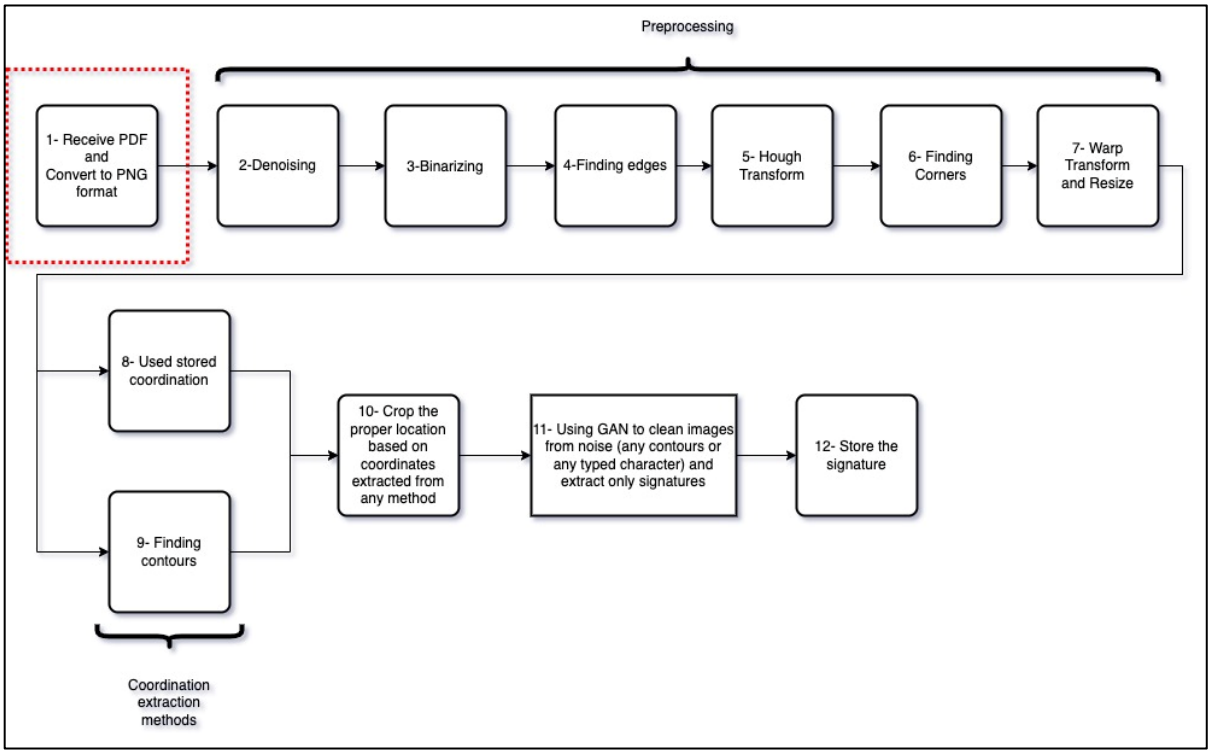

Figure 3.4 Receiving PDF and Converting it to image step

The process of receiving a PDF file involves utilizing AWS SES to extract the email attachment and subsequently storing it in S3. To convert the PDF to PNG format, the pdf2image package is employed. The convert_from_path function is utilized to read the PDF file, and the resulting image is saved as a PNG file using the page.save (f ‘`file name`’, `file format`) (Belval, 2023) line of code (see Figure 3.4 activity 1).

In order to ready the image for subsequent procedures such as detecting contours and refining it with CycleGAN, a sequence of image processing techniques in six steps are employed to

achieve this goal (see Figure 3.5). All six steps will be thoroughly discussed and examined using this image.

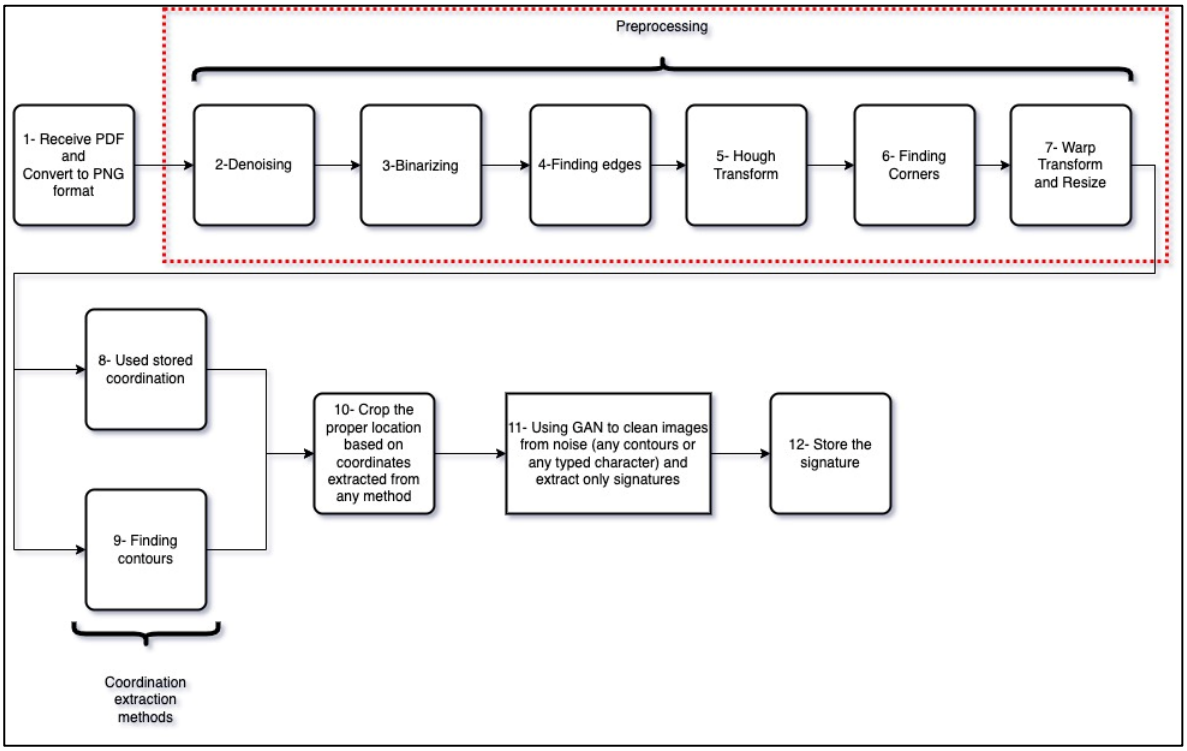

Figure 3.5 Image preprocessing steps of the image processing pipeline proposed

2) Denoising

In order to denoise the image, (see Figure 3.5 activity 2) from any gaps and holes whilst keeping the edges, the following technique is experimented with the result shown in Figure

3.6. The cv2.fastNlMeansDenoising (opencv, 2023)method can effectively remove noise from the image while preserving the edges and details. The strength of the denoising can be controlled by adjusting the strength parameter.

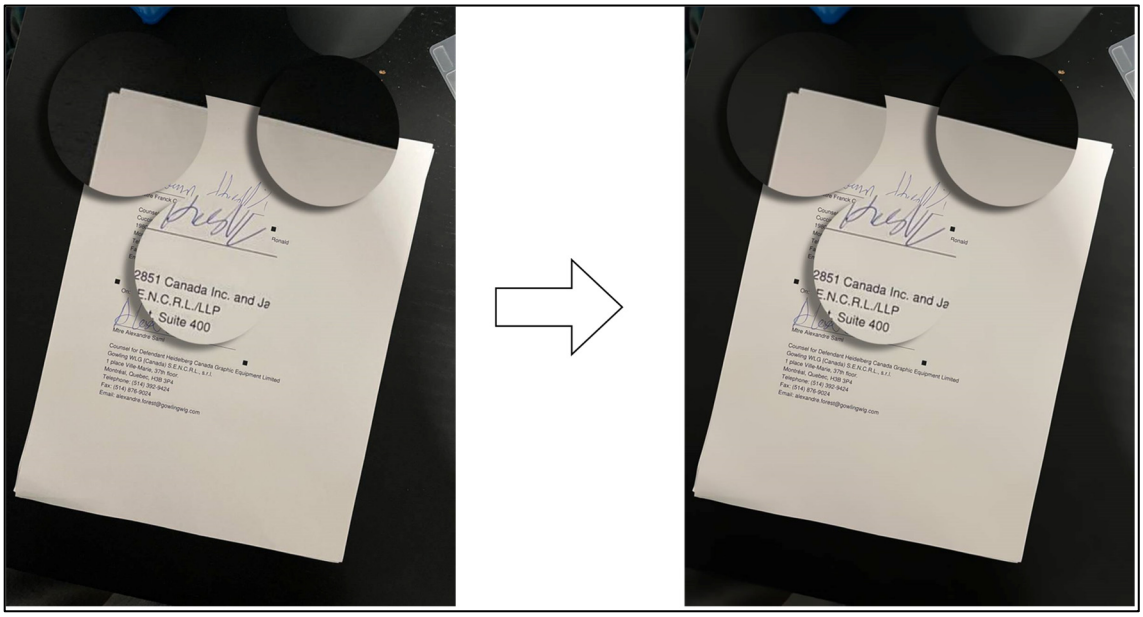

Figure 3.6 Denoised image

This method has been chosen due to its non-linear denoising technique, which enables the preservation of image details while eliminating noise. Moreover, it features an adaptive denoising capability that allows for the adjustment of the strength of denoising based on local image characteristics, making it particularly useful when dealing with images that have varying levels of noise. The cv2.fastNlMeansDenoising method considers two parameters. The first parameter, image, specifies the input image to be denoised. The second parameter, h, is a value that controls the strength of the denoising filter. Higher values of h will result in stronger denoising.

3) Binarizing

In order to find edges in the image, binarizing (see Figure 3.5 activity 3) is used. The Otsu method was experimented, whereby the image was first converted to grayscale and then the Otsu thresholding algorithm was applied using the cv2.threshold (opencv, 2023) function.

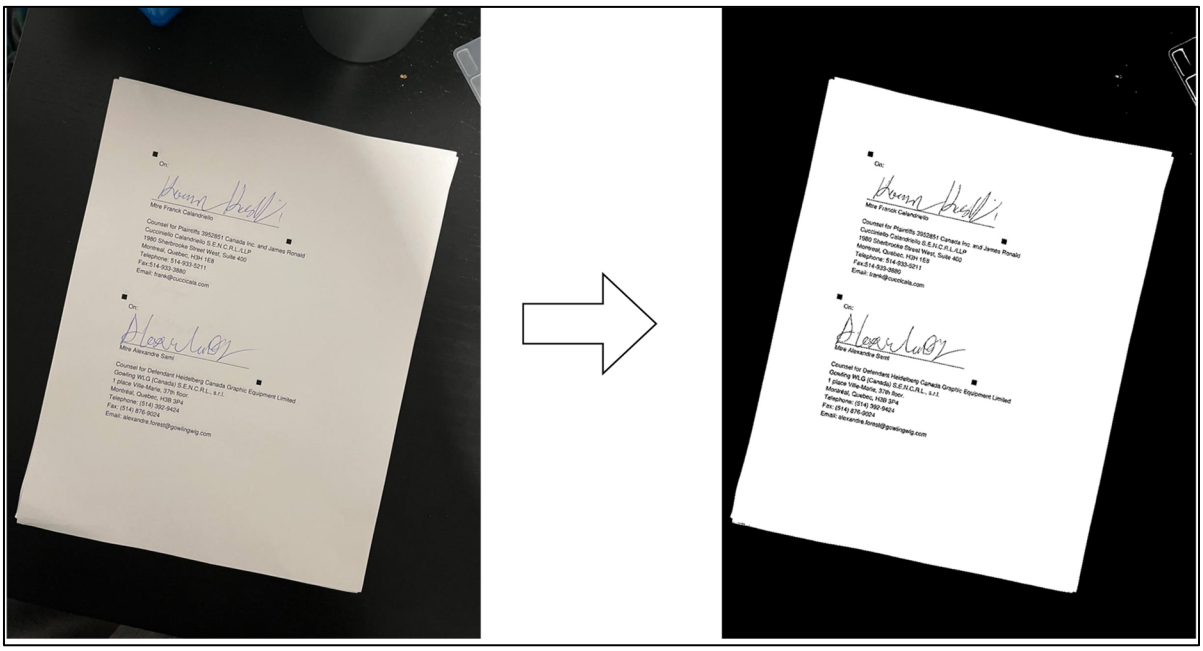

Figure 3.7 Binarized image

Moreover, this method is used since it automatically calculates the optimal threshold value, which minimizes the intra-class variance between the two groups of pixel values. This can lead to a more accurate and consistent thresholding result, as compared to manual thresholding methods that rely on trial and error. To remove imperfections on the image, morphological operations can be used to both fill gaps and remove small holes, as well as remove small objects and smooth the image. The cv2.morphologyEx (opencv, 2023) function can be used to apply these operations. To fill gaps and remove small holes, the morphological closing operation is used, which works by dilating the image followed by erosion using a structuring element of a specified size. On the other hand, to remove small objects and smooth the image, the

morphological opening operation is used, which works by erosion followed by dilation using a structuring element of a specified size. The result is shown in Figure 3.7b

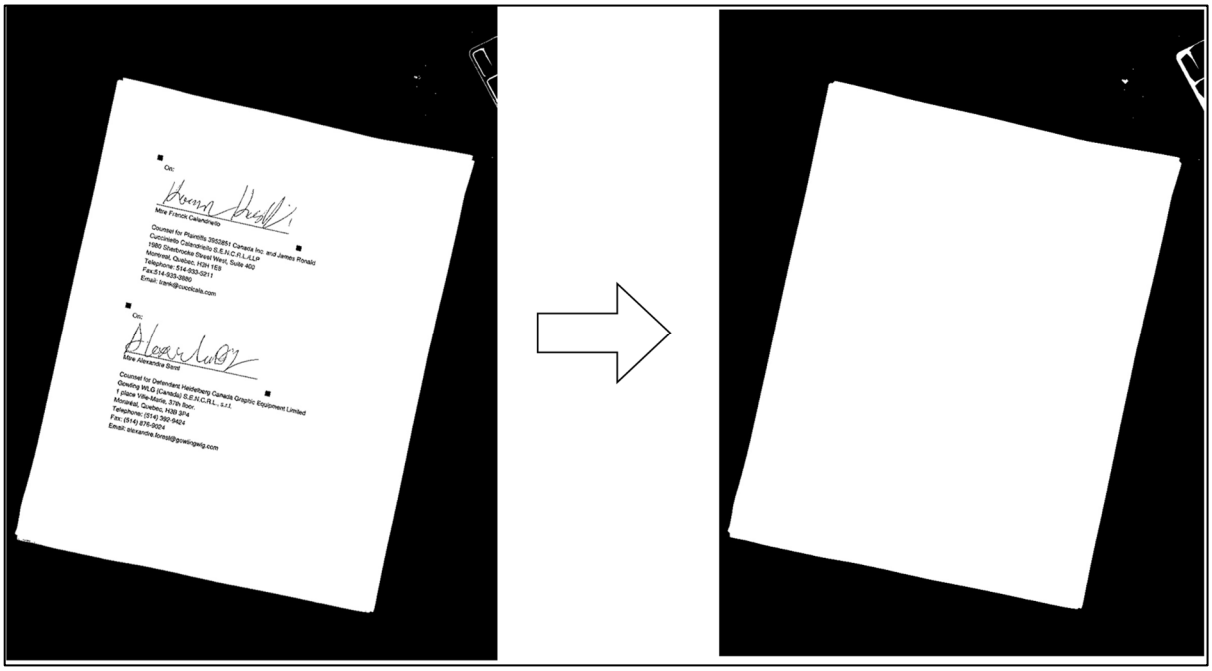

Figure 3.7b Morphological Transform

4) Finding edges

The Canny edge detection algorithm is commonly used to detect edges in an input image (see Figure 3.5 activity 4). This algorithm requires three parameters to operate: thresh1, thresh2, and apertureSize. The thresh1 and thresh2 parameters are used to set the lower and upper thresholds for the hysteresis procedure in the Canny algorithm. Meanwhile, the apertureSize parameter is used to adjust the size of the Sobel kernel used for edge detection. By using the cv2.Canny(opencv, 2023e) function in Python, this method returns a binary image with the detected edges. This is a straightforward approach to detect edges in an input image. Figure 3.8 shows the result of this activity.

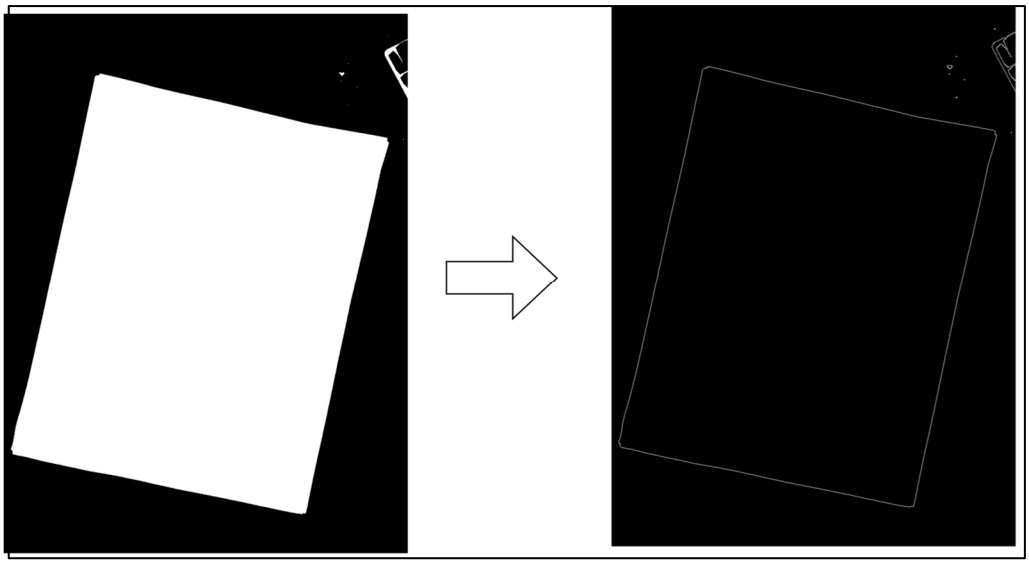

Figure 3.8 Edge detection of the protocol

5) Hough Transform

The Hough transform is employed to detect lines on the edges of the image which will be used to identify the corners of a document (see Figure 3.5 activity 5). To accomplish this, the cv2.HoughLines(opencv, 2023) function utilizes three parameters: rho_acc, theta_acc, and thresh. rho_acc and theta_acc represent the distance and angle precision of the Hough transform, respectively. thresh is the Hough transform threshold, which establishes the minimum number of votes required to validate a detected line (Figure 3.9).

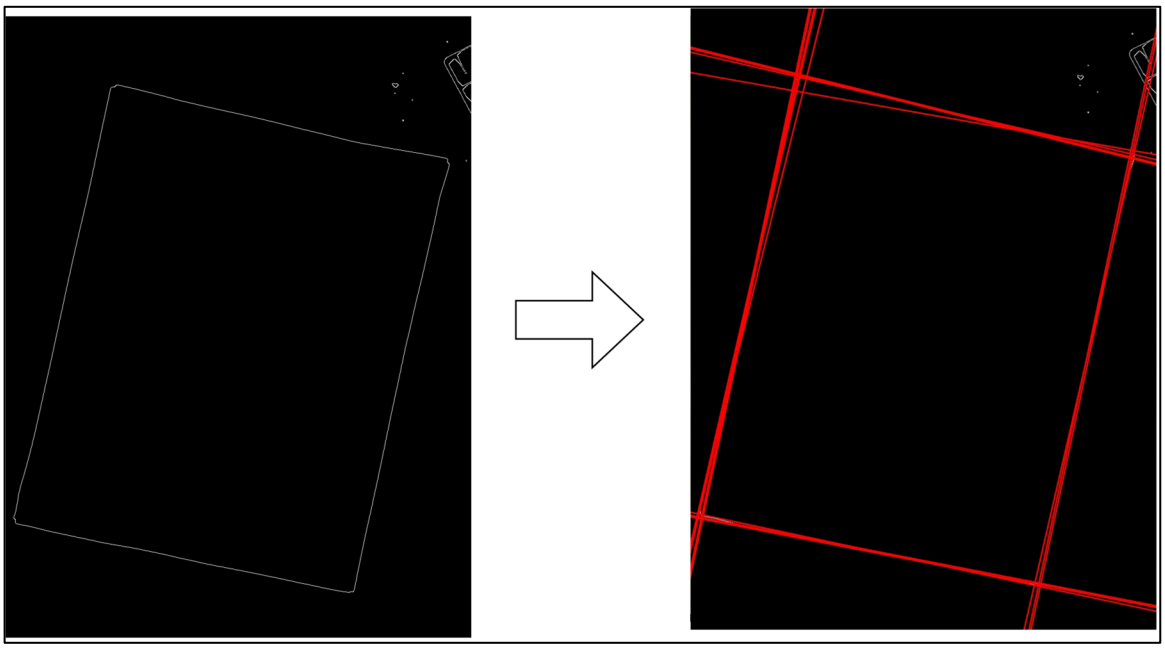

Figure 3.9 Hough Lines of the edge detected

6) Finding Corners

To locate the quadrilaterals (document on the image) (see Figure 3.5 activity 6) and perform a warp transform on the document, it is necessary to first find the intersection between Hough lines. To find intersection lines, the list of all the lines detected by the Hough transform is first retrieved, each pair of lines is iterated over, and the angle between them is calculated. If the angle is between 80 and 100 degrees, another method is used to calculate the intersection point between the two lines. If the intersection point falls within the range of the image dimensions (0 <= x <= width, 0 <= y <= height), the point is appended to the intersections list.

To calculate the intersection point, a method was developed which has two parameters, line1 and line2, each represented as a tuple (rho, theta) in Hesse normal form, which are used to calculate the intersection point of these lines. A 2x2 matrix A is created, where the first row contains the cosine and sine of theta1, and the second row contains the cosine and sine of theta2. Then, a 2x1 matrix b containing rho1 and rho2 is created. Using the numpy linalg

(numpy, 2023) solve method, the values of x0 and y0 that satisfy the matrix equation Ax = b are obtained. Finally, the values of x0 and y0 are rounded to the nearest integer, and the intersection point is returned as a list containing [x0, y0].

Afterwards, for each corner, multiple points are identified, as shown in Figure 3.10. To consolidate nearby points and obtain only four points, the (scikit-learn, 2023) algorithm was utilized to cluster intersection points into four groups that represent the centers of four quadrilaterals. Figure 3.11 shows the result.

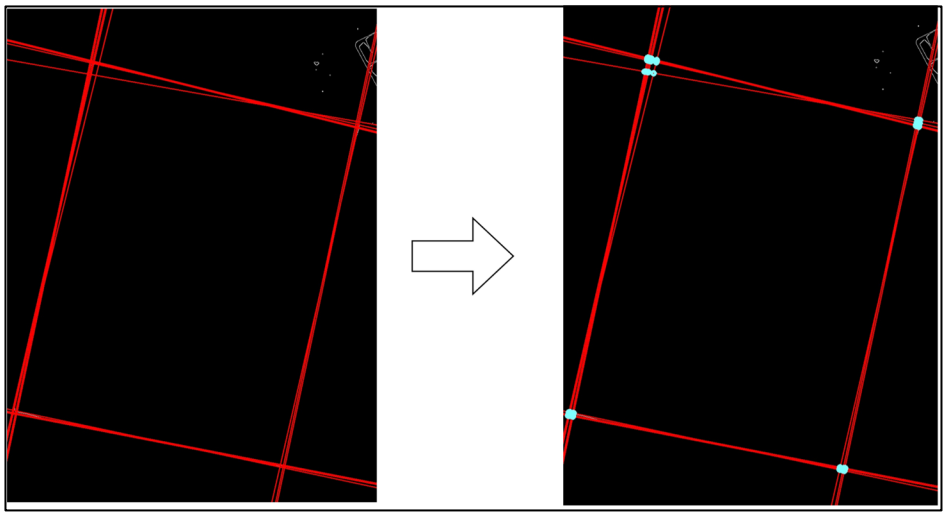

Figure 3.10 Intersection points

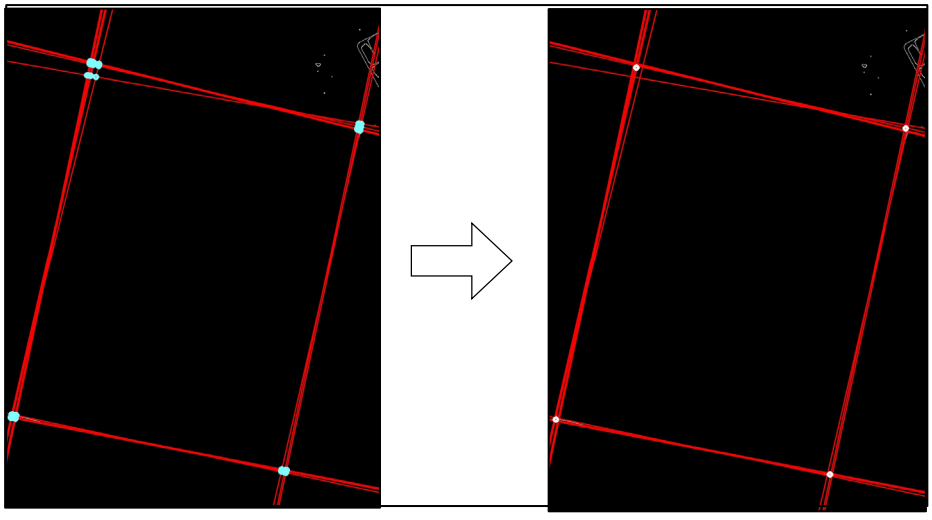

Figure 3.11 Grouped intersection points

7) Warp Transform and Resize

Once the four corners of the quadrilaterals have been found, the nodes should be ordered before applying the warp transform on them (see Figure 3.5 activity 7). In order to determine the correct order of the four corners of a quadrilateral, the developed method calculates the sum of the x and y coordinates for each point. It then selects the point with the smallest sum as the top-left corner and the point with the largest sum as the bottom-right corner.

Next, it calculates the difference between the x and y coordinates for each point. The point with the smallest difference is assigned as the top-right corner, while the point with the largest difference is assigned as the bottom-left corner.

Based on the distance between the ordered points, the method computes the width and height of the new image and creates a new array of destination points specifying the top-left, top-right, bottom-right, and bottom-left corners of the new image. Afterwards the perspective transform matrix is computed using the ordered points and the destination points, and the image is transformed using the cv2.warpPerspective (opencv, 2023) function.

To resize an image, the height and width ratios are calculated, and then the cv2.resize (opencv, 2023) function is used to adjust the image dimensions accordingly. The height of the image is set to a desired value, and then the ratio of the new height to the original height is calculated. This ratio is then used to calculate the new width of the image. The dimensions of the new image are then specified as a tuple, and the cv2.resize function is used to resize the image accordingly using interpolation with the cv2.INTER_AREA method.

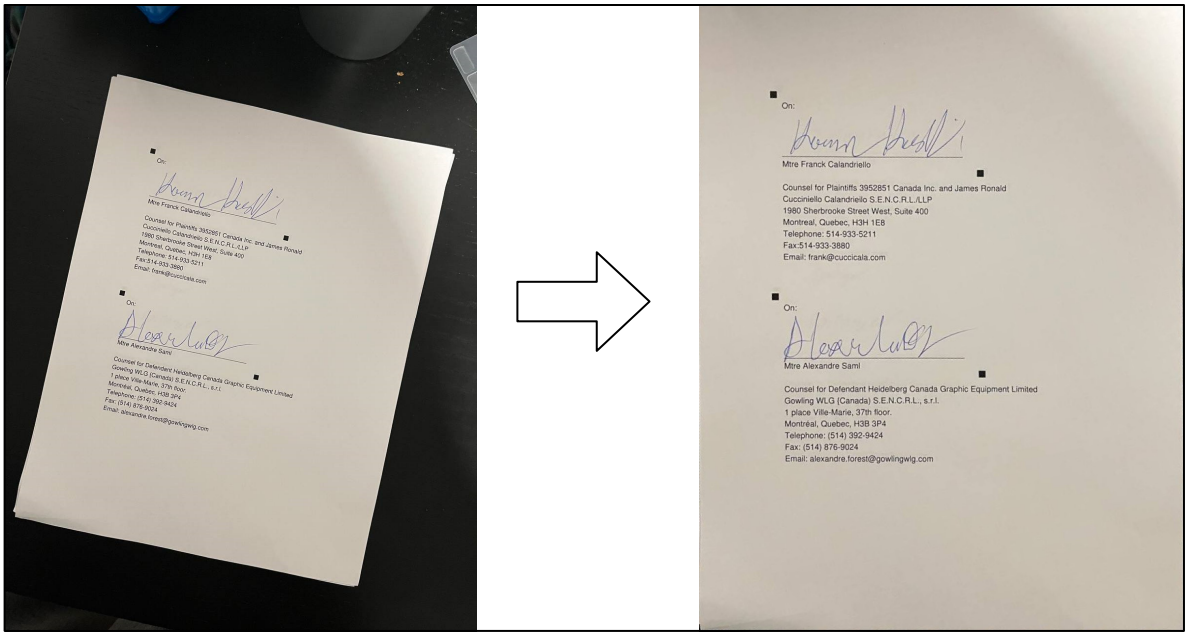

Figure 3.12 Extracted image

8) Use of stored coordination

After converting a PDF into an image, the signature is then placed on the image as if it were a Cartesian plane, allowing for precise location and placement (see Figure 3.13 activity 8). After the signature has been added, the document can be cropped based on the stored location of the signature upon receipt. As shown in Figure 3.15, this approach is highly effective for e-signed documents since their size and perspective match those of the original PDF file. However, if the document's size and perspective differ significantly from the original file, as illustrated in Figure 3.12, the resulting outcome may deviate significantly from the intended objective.

9) Use of image contouring technique

In order to ensure the previous proposed technique is applicable regardless of the size and perspective of each signature (see Figure 3.13 activity 9), two black square contours are added to original PDF (as previously shown in Figure 2.6). Once the document is received, this

method utilizes cv2.findContours to search for contours, extract their coordinates, and crop them in a similar manner to the use of stored coordinates.

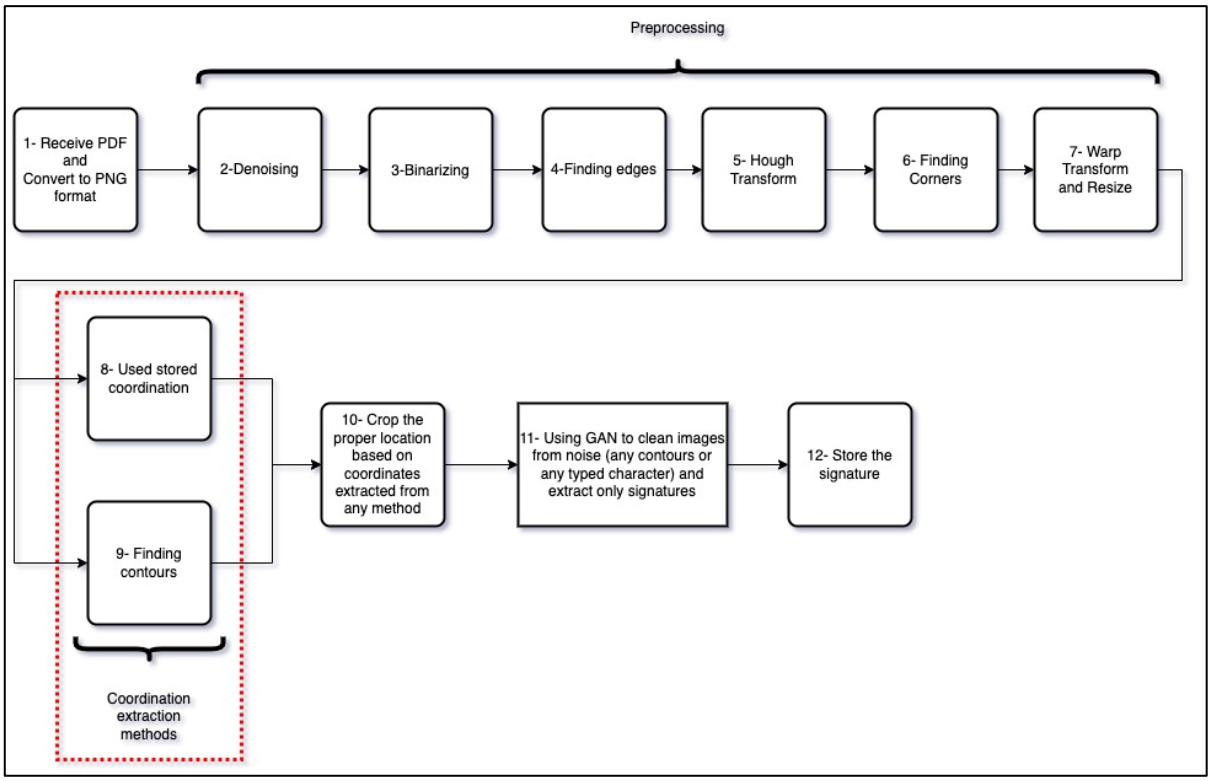

Figure 3.13 Coordination extraction step

10) Crop using Apache PDFBox

The image cropping process was performed using pdfbox after obtaining the coordinates of the contours in the previous step (see Figure 3.14 activity 10).

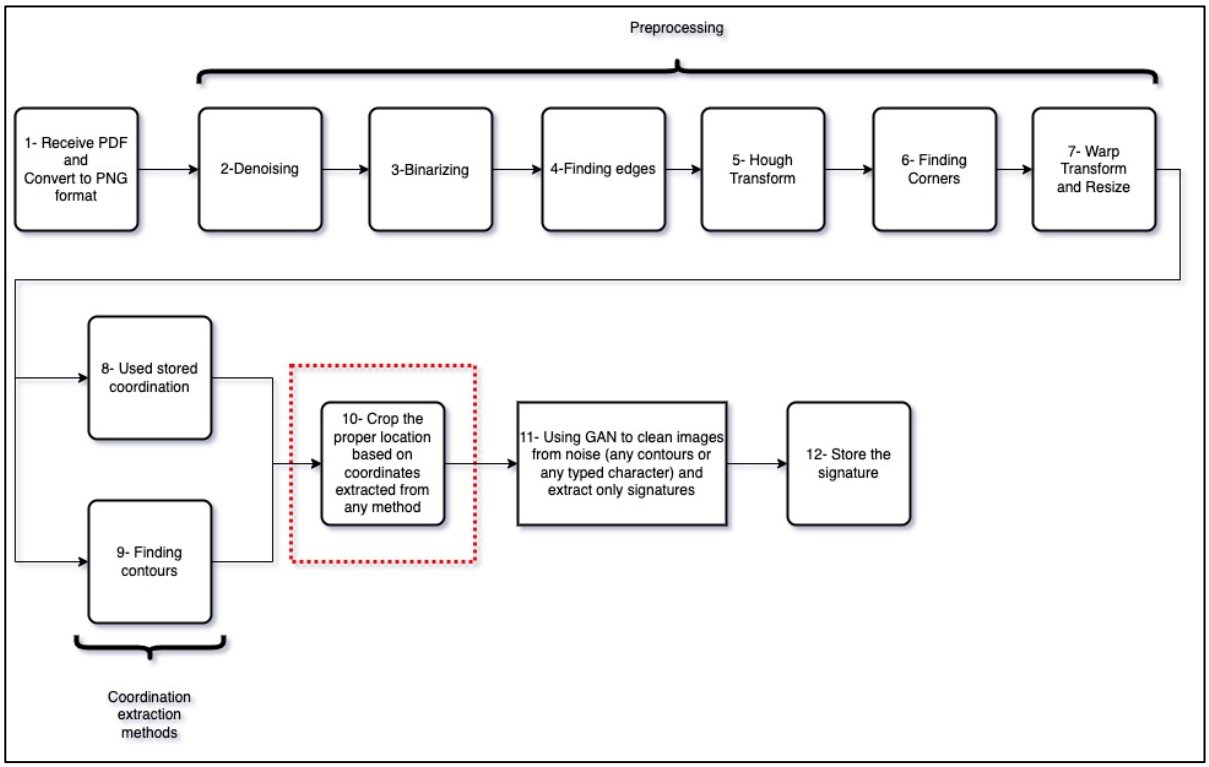

Figure 3.14 Cropping the image step

This allowed for the precise extraction of the relevant region of interest from the original image. A black margin is added to the extracted image since our generative model, the GAN, was trained with square images (Figure 3.15).

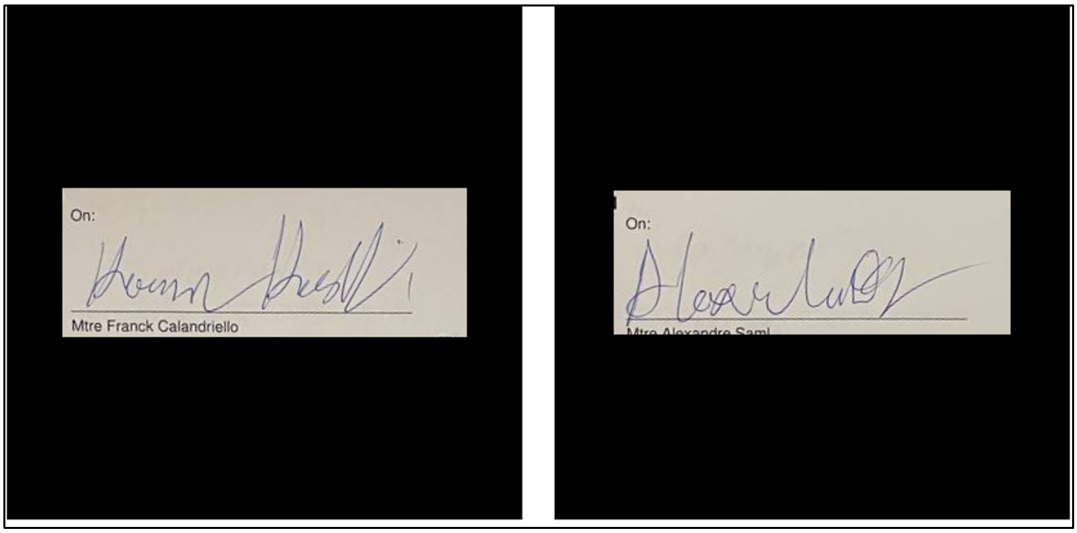

Figure 3.15 Cropped signatures

11) Cleaning the noise, Using CycleGAN to clean the image

A signature image with noticeable noise artifacts is depicted in the Figure 3.17, which was mitigated using a machine learning approach (see Figure 3.15 activity 11). Specifically, a generative model was employed to eliminate noise and extraneous characters (Figure 3.16).

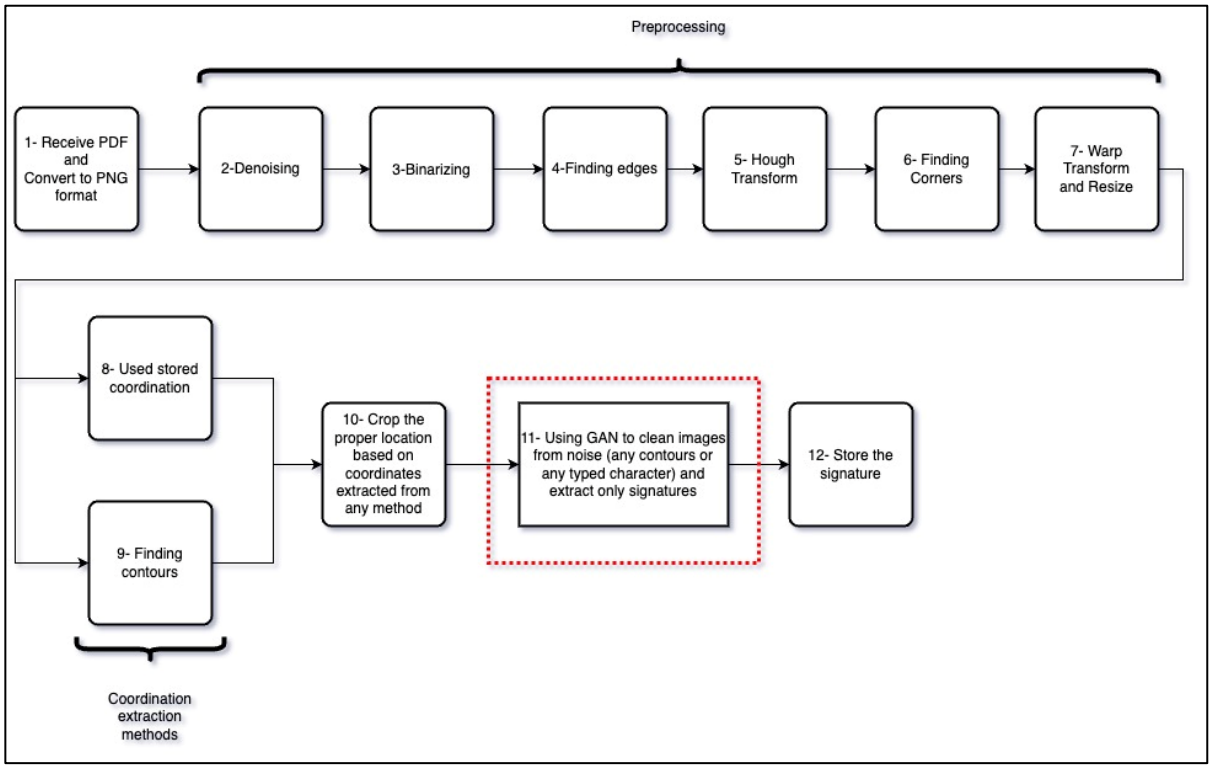

Figure 3.16 Cleaning the noise step

CycleGAN is a generative model that can perform image-to-image translation with unpaired images, allowing for translation between images in domain A (noisy signature) to an image in domain B (clean signature) without having to train the model with matching pairs of images. Typically, training a model for signature verification requires a large dataset of paired examples, which can be difficult, expensive, or impossible to prepare. CycleGANs use two generators and discriminators to translate images from the domain of noisy signatures to the domain of clean signatures. The generator for the noisy signature domain tries to translate the image into the clean signature domain, and the discriminator for the clean signature domain

predicts whether the image is a real clean signature or a generated clean signature. From the generated clean signature image, the generator for the clean signature domain must reconstruct the original noisy signature. This cycle-consistency behavior allows the model to translate a noisy signature image to a clean signature image without the need for matching data pairs. The utilization of the CycleGAN technique involved the training of the model on both clean and noisy images, which were readily accessible on Kaggle (RobinReni, 2019). Also, this GitHub repository contains the source code to prepare the image (Joseph, 2022). For each training session, the CycleGAN was trained for one day on a Google Cloud Deep Learning Machine equipped with a V100 graphics card, 16 GB of memory, and NVLink Ring networking operating at 300 GB/s.

To train the model, the following command is used:

```
python train.py --dataroot ./datasets/dataset_name --name model_name --model cycle_gan
```

To test the model, the following command is used:

```
python test.py --dataroot datasets/dataset_name/testA --name model_name --model test --no_dropout
```

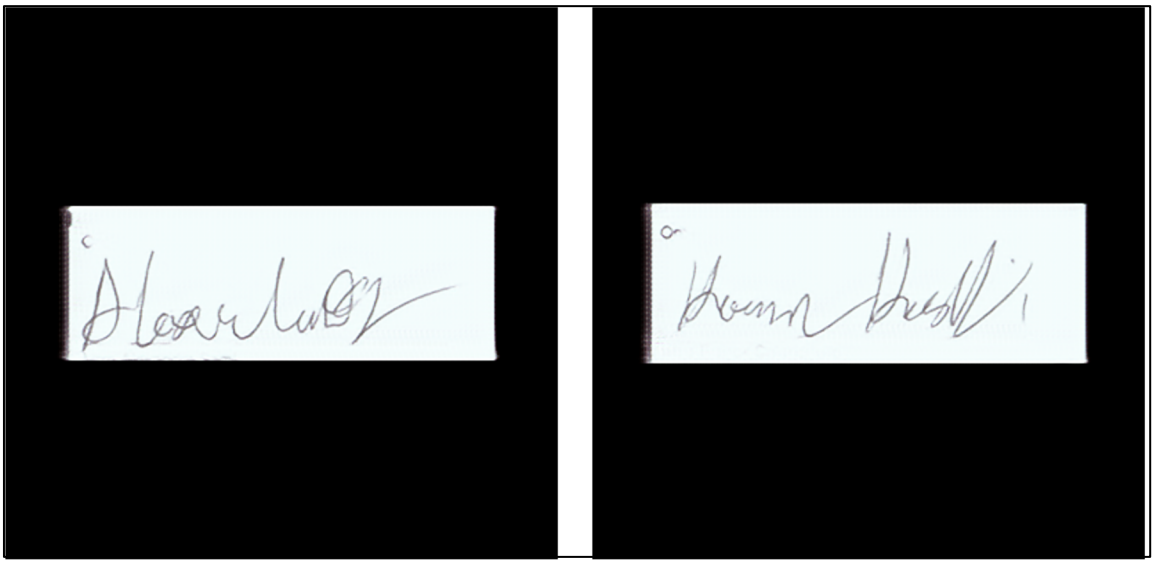

Figure 3.17 Remove noises from signatures

12) Storing the Signature

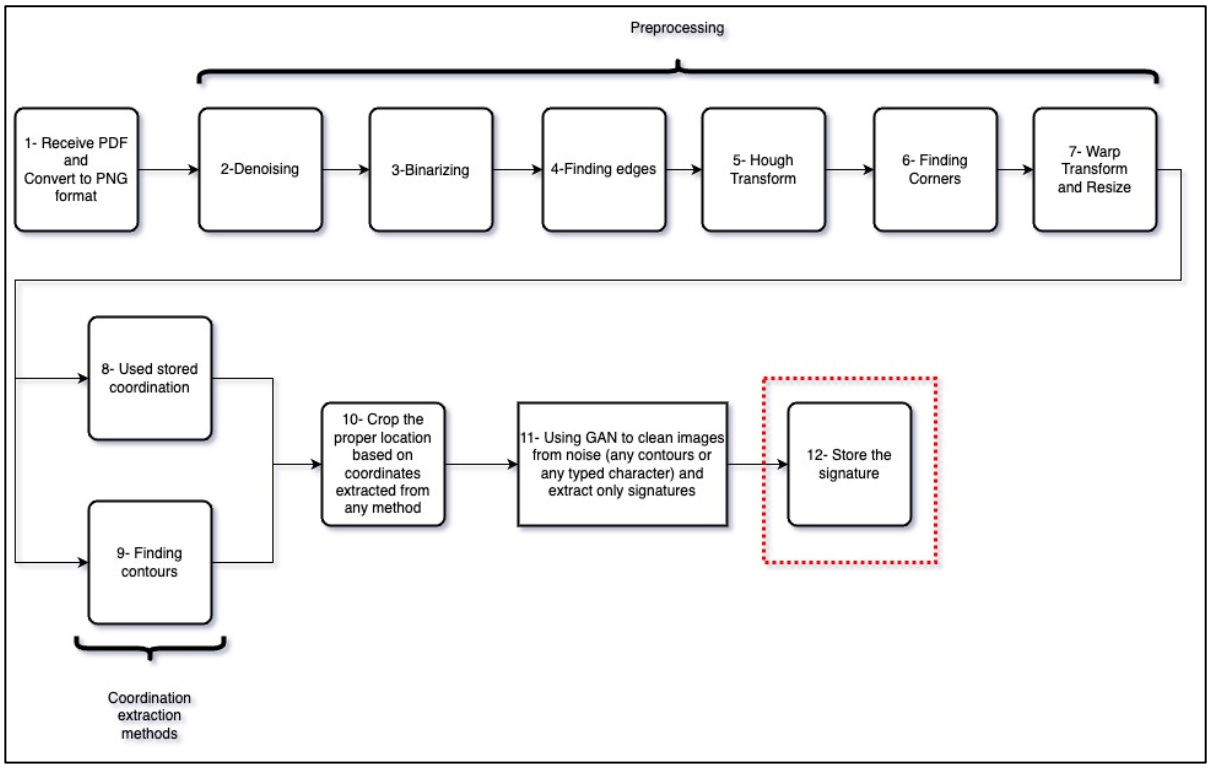

Figure 3.18 Storing the resulting signature

After the signatures are processed (i.e., after CycleGAN denoising) they will be stored in S3 (see Figure 3.18 activity 12) This step is done using AWS Lambda function and it is written in Java.

3.5 Conclusion

This chapter has presented a comprehensive description of the project's signature image transformation prototype and its objectives. It explained the existing case protocol prototype and how the proposed solution could be used. Additionally, the chapter highlights the image processing methods and techniques chosen and experimented to extract signatures from documents, including the three phases of preprocessing, processing, and postprocessing.

Overall, this chapter establishes the foundation for the rest of the project, providing a clear understanding of the necessary steps required to achieve the project's goals.

CHAPTER 4

RESULTS OF THE CASE STUDY AND FUTURE WORK

4.1 Introduction

This chapter begins with a summary of the initial research objectives and key findings (section 4.2). Following that, a detailed overview of the proposed solution is provided (section 4.3), along with a discussion and interpretation of the case study results (section 4.4). Finally, recommendations for future research are presented (section 4.5) to conclude the chapter.

4.2 Summary of the research

This thesis has provided a comprehensive overview of the current state of e-signature software and proposes a solution for a specific case study. The second chapter begins by introducing the regulations governing the use of electronic documents and signatures in Canada. Various commercial and open-source e-signature solutions, highlighting their key features and benefits are reviewed. During the review, it became apparent that none of the available commercial solutions met the project's needs to handle handwritten signatures and images taken with a smartphone. Open-source solutions, on the other hand, were deemed inadequate due to their lack of support, outdated technology, and absence of transaction management services. Thus, a custom solution is proposed to meet the specific requirements of the project, which was explained in Chapter 3.

Chapter 3 explained the case study objective which involves simulating sending a PDF document for signature to multiple lawyers, receiving each individual signature, and transposing them onto a final signature page. Potential solutions for transposing the signatures

are presented, including experimenting with existing image processing libraries, image coordinate techniques, and image contouring techniques.

Then, the major contribution of this thesis is discussed. This involves the design, implementation and testing of a signature pipeline for the case study where a case protocol is sent to multiple lawyers for individual e-signatures, which are then transposed onto a final version of the protocol to be submitted to the court. The challenges include dealing with different e-signature technologies and types of signatures used by the lawyers, transposing the signatures onto the final document without invalidating them, and tracking the status of individual signatures to ensure timely completion of the process.

The proposed image transposition pipeline involves a signature extraction process that includes three steps:

1) Denoising, binarizing, edge detection, corner detection and perspective transformation as a preprocessing;
2) Coordination extraction and cropping the image as processing;
3) Using GAN to clean the image as a post processing.

This chapter presents the experimental results of each image processing technique, such as denoising, binarizing, perspective transformation, edge detection, and the use of Generative Adversarial Networks (GANs).

4.3 Interpretation of the implemented solution

To evaluate the effectiveness of the designed pipeline, a comprehensive testing process was conducted using a wide range of documents. These documents were carefully categorized into five distinct types, which share specific characteristics such as the type of signature, the document's background, and the angle at which the document was captured. These factors were chosen considering the performance and effectiveness of the designed pipeline for each type.

The documents are categorized into five categories as follows:

1) E-signed as in Figure 4.1

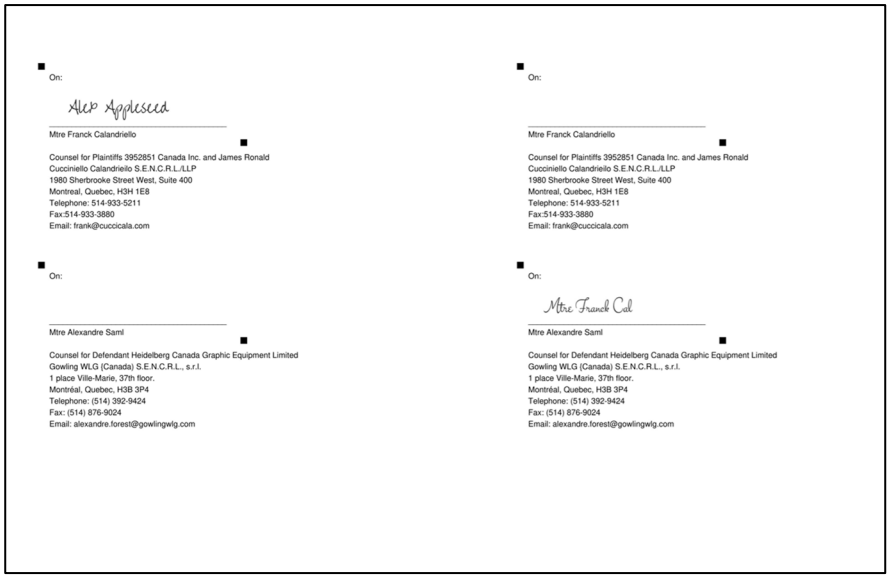

Figure 4.1 E-signed samples

2) Angular and crooked as in Figure 4.2

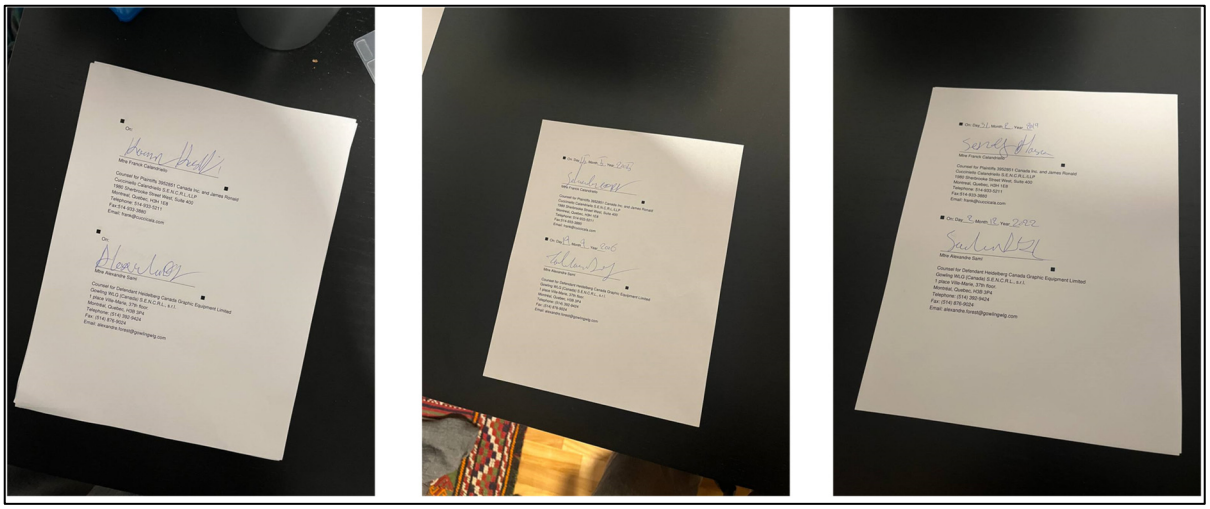

Figure 4.2 Angular and crooked samples

3) Dark Background as in Figure 4.3

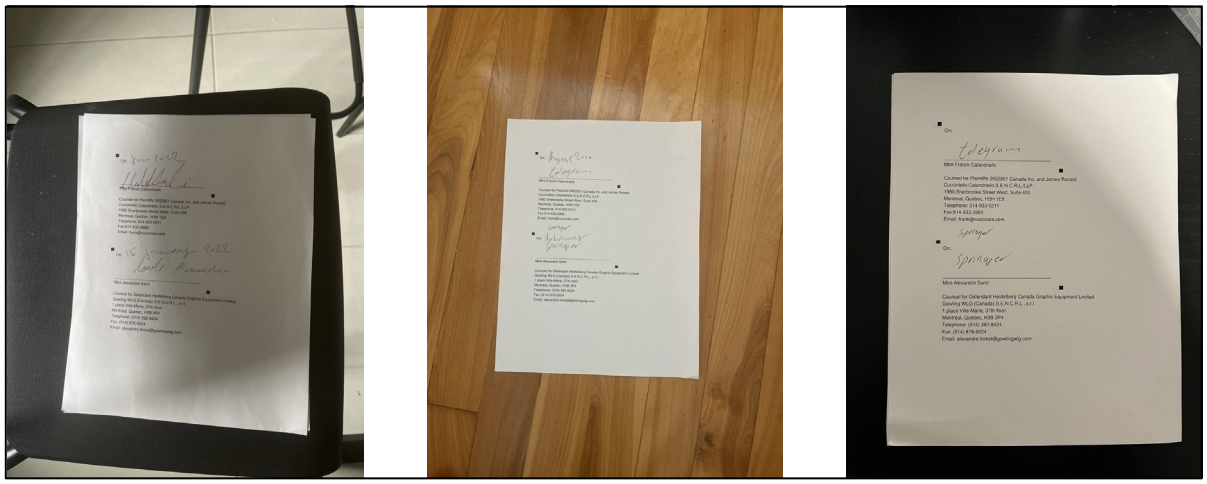

Figure 4.3 Dark background samples

4) Light Background as in Figure 4.4

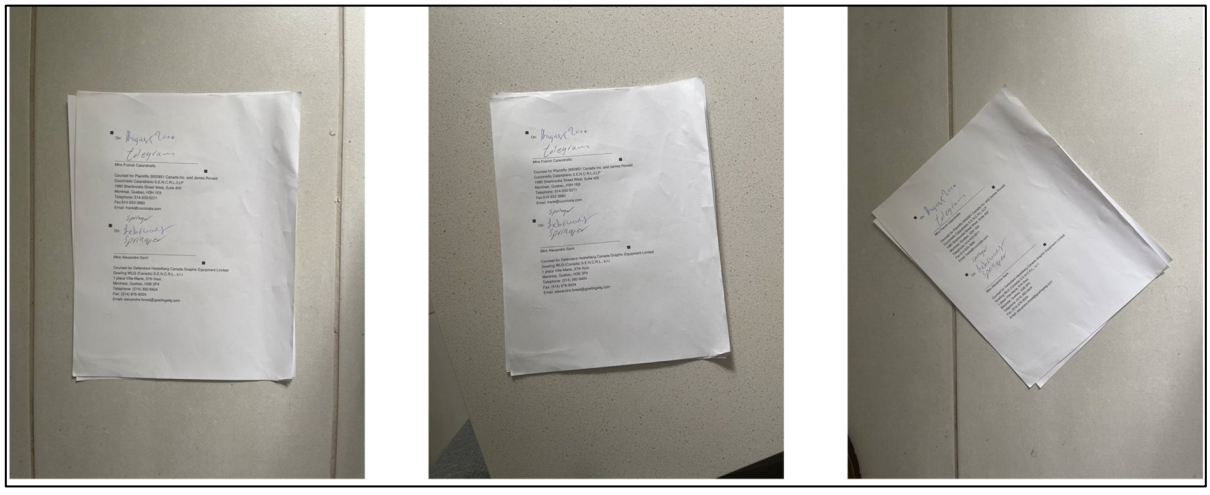

Figure 4.4 Light Background samples

5) Other and edge cases as in Figure 4.5

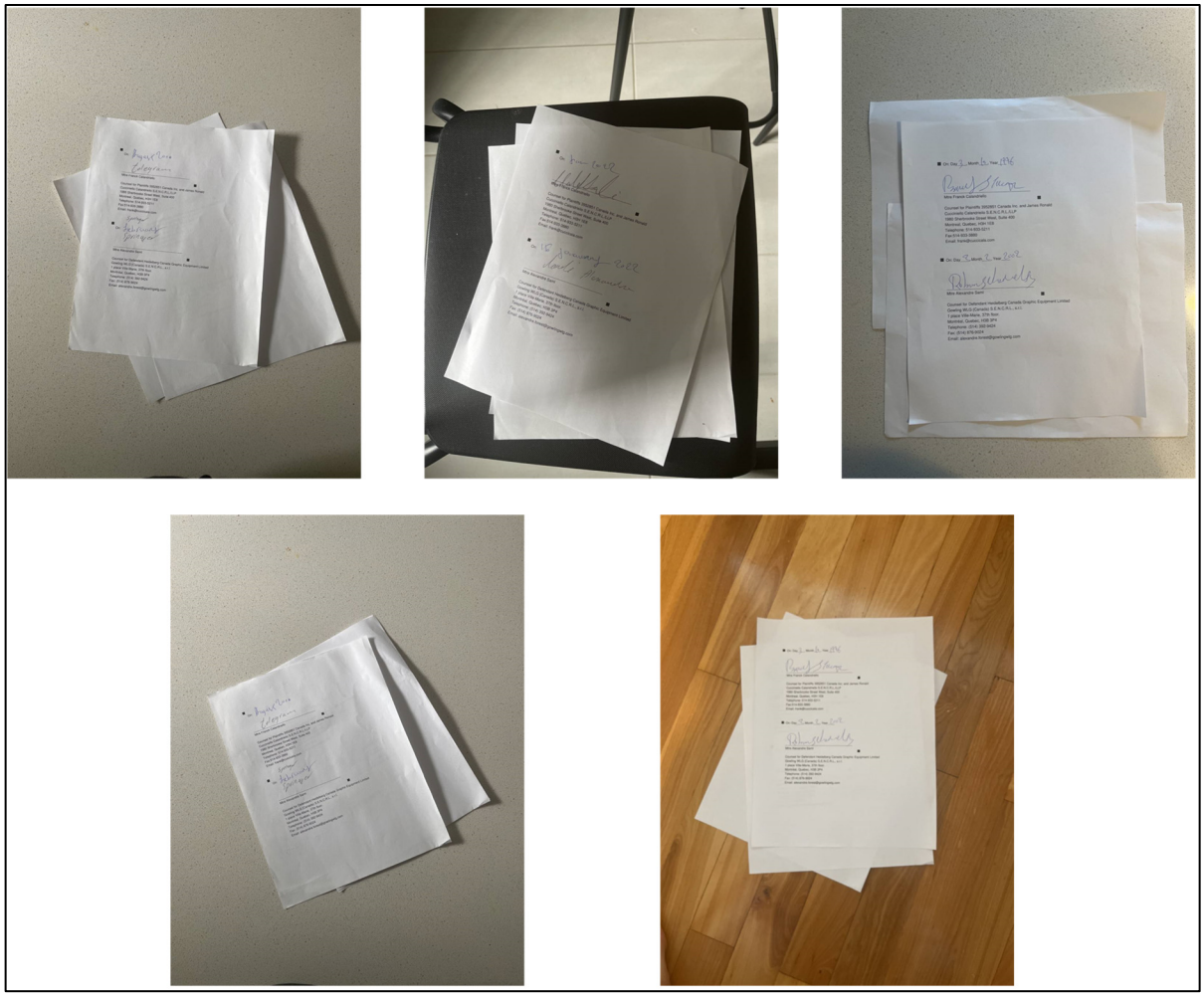

Figure 4.5 Other and edge cases samples

The pipeline functions flawlessly for the first three categories, accurately detecting and extracting both the document and signature. However, when dealing with documents featuring a light background, the preprocessing stage struggles to produce satisfactory results, making it difficult to identify contours. Consequently, this negatively impacts the overall functionality of the process. Nevertheless, if the photo quality is high and closely resembles the original

document, the pipeline can still perform adequately. In such cases, the preprocessing stage is bypassed, and contours can be successfully identified.

The results for “Other and edge cases” are unsatisfactory for several reasons. Firstly, the background is indistinguishable, making it difficult for the preprocessing part to locate the document. Secondly, the documents are often crooked and/or angled with a light background, further complicating the preprocessing step. Additionally, there are numerous noises in the picture and the camera is not positioned close enough to the document, resulting in difficulties in finding contours. As a result, while the pipeline performs well for the first three categories, it falls short for the last two categories due to these photographic challenges. In section 4.4, two machine learning methods are recommended as potential approaches to achieve acceptable results for images in categories 4 and 5.

Regarding the resource usage of the pipeline Table 4.1 provides an overview of the resource requirements for each stage, including time, CPU, RAM, disk, and GPU usage.

Regarding postprocessing and during the IT training phase, which is either once only or whenever improvements are desired, typically takes around 22 to 25 hours on a Google Cloud VM dedicated to machine learning, equipped with a V100 graphics card and 16 GB of memory.

In conclusion, the designed pipeline demonstrates satisfactory performance with various types of documents. However, there is significant potential for further improvement, which will be discussed in the following two sections. Additionally, it is worth noting that the pipeline operates with minimal resource utilization and remarkable speed, making it ideal for the case study.

Table 4.1 Resource usage of each part

STEP	Time (Seconds)	CPU (Core)	RAM (GB)	Disk (GB)	GPU (Core)
Preprocessing (Cleaning document to Hough transform)	2.45 - 5.45	Less than 1	Less than 1	Less than 1	None
Processing (Finding contours)	1.15 - 2.20	Less than 1	Less than 1	Less than 1	None
Postprocessing (Cleaning signature via CycleGAN)	5.40 – 6.30	Less than 1	Less than 1	Less than 1	None

4.4 Strength and weakness of the techniques used

One of the key benefits of the designed pipeline is its flexibility. The pipeline design used for extracting the signature is modular, meaning that it can be easily adapted and used on any document, regardless of its format or content. This makes it a versatile tool for a wide range of applications, from analyzing scientific data to processing legal documents. The modular nature of the pipeline also makes it easy to integrate with other document processing tools, allowing users to create custom workflows that meet their specific needs. Overall, the pipeline is a powerful tool for enhancing the readability and accessibility of documents, and its modular design makes it a flexible solution for a variety of use cases.

The novelty of this thesis lies in the use of an approach to signature identification. Specifically, the technique involves utilizing two black squares, placed on the top left and bottom right corners of the signature area, that serve as reference points for identifying the signature location. This approach offers several benefits, including improved accuracy and efficiency in

signature identification. Overall, the use of two black squares as contours for signature identification represents a novel and innovative approach that offers numerous benefits and has significant potential for a wide range of applications in document processing and analysis.

For the signature identification pipeline to function effectively, it is important that the background of the document differs slightly from the signature area. This is because the algorithm relies on differences in color or texture to identify the signature area, and if the background is too similar, the algorithm may not be able to distinguish between the signature and the surrounding area.

Additionally, if the signature is larger than the intended signature area, or if it contains a shape as big as the black contours that are placed on the document, the algorithm may not be able to extract the signature accurately. In such cases, it may be necessary to adjust the size or position of the signature area, or to use alternative techniques to extract the signature, such as machine learning or artificial intelligence algorithms.

4.5 Future work

4.5.1 Retrieve document from any image

Currently, the system is limited in terms of its compatibility with image formats, as it is only capable of processing PNG and JPG images. However, there is potential for the system to be enhanced so that it can work with other types of image formats as well. This would allow for greater flexibility and enable the system to be used more widely across a range of applications and industries.

4.5.2 Make the system detect automatically using ML

To improve the robustness of the system and eliminate the need for black contours, it may be possible to utilize machine learning models such as YOLO and SSD. These models are capable of object detection and can identify the signature area without the need for reference points.

The YOLOv5 object detection algorithm can be utilized to detect signatures in images. By training the YOLO model on signature data, it can accurately detect the presence and location of signatures within an image. The algorithm works by dividing the image into NxN regions and predicting whether a signature is present in each region. If a signature is detected, the algorithm also predicts the coordinates of the bounding box surrounding the signature. Non-max suppression is used to ensure that duplicate signatures are not detected. This method of signature detection can be applied to various fields such as forensics, document processing, and verification. YOLOv5's high accuracy and speed make it a valuable tool for detecting signatures in images efficiently and effectively.

Single Shot Detection (SSD) is a versatile object detection algorithm that can be applied to detect signatures in images. By training the SSD model on signature data, it can accurately detect the location of signatures within an image. SSD works by predicting the location and class of objects at multiple scales within the network. This allows it to detect signatures of various sizes within a single pass through the network. The algorithm can also detect multiple signatures in a single image, making it useful for processing document images. SSD can be fine-tuned for specific signature detection tasks, and its high accuracy and efficiency make it a valuable tool for automating signature detection processes. The application of SSD to signature detection can aid in various fields such as document verification, forensic analysis, and identity verification. Overall, SSD's ability to detect signatures in images accurately and efficiently makes it a promising method for automating signature detection processes.

4.6 Conclusion

In conclusion, this chapter has presented a comprehensive summary of the research conducted, the proposed solution for the case study, the evaluation of the signature pipeline's performance,

and potential areas for future improvement. The designed pipeline demonstrates satisfactory performance, and its modular design and flexibility make it a valuable tool for enhancing the readability and accessibility of documents. With further advancements and refinements, the pipeline has the potential to enhance signature processing and analysis in various fields.

APPENDIX I

IMAGE PROCESSING PIPELINE

As explained in Chapter 3, some of the image processing techniques used are denoising, binarizing, perspective transformation, edge detection, and the use of Generative Adversarial Networks. In this section, the implementation of these techniques is demonstrated.

1) Importing necessary libraries

```
import cv2
import numpy as np
from skimage.filters import threshold_otsu
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
from itertools import combinations
from collections import defaultdict
import argparse
from math import atan
```

2) Implementation of denoising and binarizing

```
https://github.com/Shakleen/Python-Document-Detector
def denoise(image):
    temp = cv2.fastNlMeansDenoising(image, h = 7)
def closer(image):
    kernel = cv2.getStructuringElement(
        cv2.MORPH_ELLIPSE,
            (kernel_size, kernel_size)
        )
        closed = cv2.morphologyEx(
            image,
            cv2.MORPH_CLOSE,
            kernel,
            iterations = iterations
        )
    return closed

def threshold(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    T_, thresholded = cv2.threshold(image, thresh1, thresh2,
cv2.THRESH_BINARY)

    return thresholded


def opener(image):
   kernel = cv2.getStructuringElement(
            cv2.MORPH_ELLIPSE,
            (kernel_size, kernel_size)
        )
        opened = cv2.morphologyEx(
            image,
            cv2.MORPH_OPEN,
            kernel,
            iterations = iterations
        )
    return opened
```

3) Implementation on edge detecting

```
def detect_edge(image, thresh1 = 50, thresh2 = 150, apertureSize =
3):
    edges = cv2.Canny(image, thresh1, thresh2, apertureSize =
apertureSize)
    return edges
```

4) Implementation of Hough transform

```
def get_hough_lines(image, rho_acc = 2, theta_acc = 360, thresh =
100):
    lines = cv2.HoughLines(
            image,
            rho_acc,
            np.pi / theta_acc,
            thresh
```

```
        )
    draw_hough_lines(lines)

    return lines


def draw_hough_lines(lines, rho_acc = 2, theta_acc = 360, thresh =
100):
        for line in lines:
            rho, theta = line[0]
            a, b = np.cos(theta), np.sin(theta)
            x0, y0 = a * rho, b * rho
            n = 5000
            x1 = int(x0 + n * (-b))
            y1 = int(y0 + n * (a))
            x2 = int(x0 - n * (-b))
            y2 = int(y0 - n * (a))

            cv2.line(
                hough_line_output,
                (x1, y1),
                (x2, y2),
                (0, 0, 255),
                2
            )

        cv2.imwrite('output/hough_line.jpg', hough_line_output)
```

5) Finding corners

```
def get_intersections(image, lines):
        """Finds the intersections between groups of lines."""
    lines = lines
    intersections = []
    group_lines = combinations(range(len(lines)), 2)
    x_in_range = lambda x: 0 <= x <= image.shape[1]
    y_in_range = lambda y: 0 <= y <= image.shape[0]

    for i, j in group_lines:
```

```
        line_i, line_j = lines[i][0], lines[j][0]

        if 80.0 < get_angle_between_lines(line_i, line_j) < 100.0:
             int_point = self._intersection(line_i, line_j)

        if x_in_range(int_point[0][0])and y_in_range(int_point[0][1]):
             intersections.append(int_point)

         draw_intersections(intersections)

         return intersections


def find_quadrilaterals( intersections ):
    X = np.array([[point[0][0], point[0][1]] for point in
intersections])
    kmeans = KMeans(
        n_clusters = 4,
        init = 'k-means++',
        max_iter = 100,
        n_init = 10,
        random_state = 0
     ).fit(X)


    return [[center.tolist()] for center in kmeans.cluster_centers_]


def draw_quadrilaterals(lines, kmeans):
    grouped_output = get_color_image()

    for idx, line in enumerate(lines):
            rho, theta = line[0]
            a, b = np.cos(theta), np.sin(theta)
            x0, y0 = a * rho, b * rho
            n = 5000
            x1 = int(x0 + n * (-b))
            y1 = int(y0 + n * (a))
            x2 = int(x0 - n * (-b))
            y2 = int(y0 - n * (a))

            cv2.line(
```

```
                grouped_output,
                (x1, y1),
                (x2, y2),
                (0, 0, 255),
                2
            )

    for point in kmeans.cluster_centers_:
            x, y = point

            cv2.circle(
                grouped_output,
                (int(x), int(y)),
                5,
                (255, 255, 255),
                5
            )

     cv2.imwrite('output/grouped.jpg', grouped_output)
```

6) Warp perspective and resize

```
def resize(image):
    if image.shape[0] <= height: return image
    ratio = round(self._height / image.shape[0], 3)
    width = int(image.shape[1] * ratio)
    dim = (width, height)
    resized = cv2.resize(image, dim, interpolation = cv2.INTER_AREA)

    return resized

def extract_page():
    # obtain a consistent order of the points and unpack them
    # individually
    pts = np.array([
            (x, y)
            for intersection in self._intersections
            for x, y in intersection
        ])
    rect = order_points(pts)
    (tl, tr, br, bl) = rect
```

```
        # compute the width of the new image, which will be the
        # maximum distance between bottom-right and bottom-left
        # x-coordiates or the top-right and top-left x-coordinates
    widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))
    widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2))
        maxWidth = max(int(widthA), int(widthB))

        # compute the height of the new image, which will be the
        # maximum distance between the top-right and bottom-right
        # y-coordinates or the top-left and bottom-left y-coordinates
    heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2))
    heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2))
    maxHeight = max(int(heightA), int(heightB))

        # now that we have the dimensions of the new image, construct
        # the set of destination points to obtain a "birds eye view",
        # (i.e. top-down view) of the image, again specifying points
        # in the top-left, top-right, bottom-right, and bottom-left
        # order
    dst = np.array([
            [0, 0],                          # Top left point
            [maxWidth - 1, 0],               # Top right point
            [maxWidth - 1, maxHeight - 1],   # Bottom right point
            [0, maxHeight - 1]],             # Bottom left point
            dtype = "float32"                # Date type
    )

        # compute the perspective transform matrix and then apply it
    M = cv2.getPerspectiveTransform(rect, dst)
    warped = cv2.warpPerspective(self._image, M, (maxWidth,
maxHeight))



        # return the warped image
    return warped

 def order_points(pts):
        """
        Function for getting the bounding box points in the correct
        order
```

```
        Params
        pts        The points in the bounding box. Usually (x, y)
coordinates
        Returns
        rect    The ordered set of points
        """
        # initialzie a list of coordinates that will be ordered such
that
        # 1st point -> Top left
        # 2nd point -> Top right
        # 3rd point -> Bottom right
        # 4th point -> Bottom left
    rect = np.zeros((4, 2), dtype = "float32")

        # the top-left point will have the smallest sum, whereas
        # the bottom-right point will have the largest sum
    s = pts.sum(axis = 1)
    rect[0] = pts[np.argmin(s)]
    rect[2] = pts[np.argmax(s)]

        # now, compute the difference between the points, the
        # top-right point will have the smallest difference,
        # whereas the bottom-left will have the largest difference
    diff = np.diff(pts, axis = 1)
    rect[1] = pts[np.argmin(diff)]
    rect[3] = pts[np.argmax(diff)]

        # return the ordered coordinates
    return rect
# Shakleen #
```

7) Implementation of Finding contours and cropping image

```
def find_contours(img):
#convert from BGR to HSV color space
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#apply threshold
    thresh = cv2.threshold(gray, 100, 255, cv2.THRESH_BINARY)[1]
```

```
# find contours and get one with area about 180*35
# draw all contours in green and accepted ones in red
    contours = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = contours[0] if len(contours) == 2 else contours[1]
#area_thresh = 0
    min_area = 100.0
    max_area = 200.0
    result = img.copy()
    for c in contours:
        area = cv2.contourArea(c)
    # cv2.drawContours(result, [c], -1, (0, 255, 0), 1)
        if area > min_area and area < max_area:
            cv2.drawContours(result, [c], -1, (0, 0, 255), 1)
            n = c.ravel()
            # print(c)
            i = 0
            # print(result)
            # print(area)
            for j in n :
              if(i % 2 == 0):
                  x = n[i]
                  y = n[i + 1]

                  # String containing the co-ordinates.
                  string = str(x) + " " + str(y)

                  # print(area)
                  print(x,y)
              i = i + 1

# save result
    cv2.imwrite("box_found.png", result)

# show images
   show_image(thresh)

def crop_image(img, h, w):
```

```
    image = cv2.copyMakeBorder(crop_img, int((400-h)/2), int((400-
h)/2), int((400-w)/2), int((400-w)/2), cv2.BORDER_CONSTANT)
    show_image(image)
```

8) Using CycleGAN

To download and install the model using this command:

```
git clone https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
pip install -r requirements.txt
pip install visdom
pip install dominate
```

After the training is complete, the latest Generator(B)

```
/checkpoints/latest_net_G_B.pth
```

should be copied and saved as latest_net_G.pth in the

```
/checkpoints/latest_net_G.pth
```

directory using the command

```
cp ./checkpoints/latest_net_G_B.pth ./checkpoints/latest_net_G.pth
```

To train the algorithm, this dataset was used:

```
python pytorch-CycleGAN-and-pix2pix/test.py --dataroot ./testB --checkpoints_dir
./drive/MyDrive/signature  --name gan_signdata_kaggle --model test --no_dropout --
crop_size 400 --load_size 400 --display_winsize 400
```

```
https://github.com/amaljoseph/Signature-Verification_System_using_YOLOv5-and-
CycleGAN/blob/master/Training/CycleGAN/SignatureCleaningCycleGAN.ipynb
```

BIBLIOGRAPHY

Ahmad, K., Khan, J., & Iqbal, M. S. U. D. (2019). A comparative study of Different Denoising Techniques in Digital Image Processing. *2019 8th International Conference on Modeling Simulation and Applied Optimization ICMSAO*, (pp. 1–6). Manama, Bahrain

Al-Ali, Z., Goodarzy, S., Hunter, E., Ha, S., Han, R., Keller, E., & Rozner, E. (2018). Making Serverless Computing More Serverless. *2018 IEEE 11th International Conference on Cloud Computing CLOUD*, (pp. 456–459). San Francisco, CA, USA: IEEE

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., & Suter, P. (2017). Serverless Computing: Current Trends and Open Problems. In S. Chaudhary, G. Somani, & R. Buyya (Eds.), *Research Advances in Cloud Computing* (pp. 1–20). Springer Singapore. Retrieved from: https://doi.org/10.1007/978-981-10-5026-8_1

Ballard, D. H., & Brown, C. M. (1982). *Computer Vision*. Prentice Hall. Englewood cliffs, New-Jersey

Belval. (2023). *pdf2image*. Retrieved from: https://github.com/Belval/pdf2image

Brennen, J. S., & Kreiss, D. (2016). Digitalization. *The International Encyclopedia of Communication Theory and Philosophy*, 1–11.

Buades, A., Coll, B., & Morel, J. M. (2005). A Review of Image Denoising Algorithms, with a New One. *Multiscale Modeling & Simulation*, *4*(2), 490–530. Available online at: https://doi.org/10.1137/040616024

Croft, H. T., Falconer, K. J., & Guy, R. K. (1991). *Unsolved problems in geometry*. Springer-Verlag. Available online at: https://doi.org/10.1007/978-1-4612-0963-8

D.stebani. (2016). *Perspective_transformation_matrix_2D* Available online at: https://en.wikipedia.org/wiki/Transformation_matrix#/media/File:Perspective_transformation_matrix_2D.svg

DZone. (2018). *4 Use Cases of Serverless Architecture*. Retrieved from: https://dzone.com/articles/4-use-cases-of-serverless-architecture

Elad, M., & Aharon, M. (2006). Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing*, *15*(12), 3736–3745. Available online at: https://doi.org/10.1109/TIP.2006.881969

Fan, L., Zhang, F., Fan, H., & Zhang, C. (2019). Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, *2*(1), 7. Available online at: https://doi.org/10.1186/s42492-019-0016-7

Government of Canada. (2000). *Personal Information Protection and Electronic Documents Act*. Available online at: https://laws-lois.justice.gc.ca/eng/acts/p-8.6/FullText.html

Government of Canada. (2017). *Government of Canada Guidance on Using Electronic Signatures*. Available online at: https://www.canada.ca/en/government/system/digital-government/online-security-privacy/government-canada-guidance-using-electronic-signatures.html

Joseph, A. (2022). *Signature-Verification_System_using_YOLOv5-and-CycleGAN*. Available online at: https://github.com/amaljoseph/Signature-Verification_System_using_YOLOv5-and-CycleGAN

Lentner, G. M., & Parycek, P. (2016). Electronic identity (eID) and electronic signature (eSig) for eGovernment services – a comparative legal study. *Transforming Government: People, Process and Policy*, *10*(1), 8–25. Available online at: https://doi.org/10.1108/TG-11-2013-0047

Marr, D., Hildreth, E., & Brenner, S. (1997). Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *207*(1167), 187–217. Available online at: https://doi.org/10.1098/rspb.1980.0020

McGrath, G., & Brenner, P. R. (2017). Serverless Computing: Design, Implementation, and Performance. *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops ICDCSW*, (pp. 405–410). Available online at: https://doi.org/10.1109/ICDCSW.2017.36

McGrath, G., Short, J., Ennis, S., Judson, B., & Brenner, P. (2016). Cloud Event Programming Paradigms: Applications and Analysis. *2016 IEEE 9th International Conference on Cloud Computing CLOUD*, (pp. 400–406). Available online at: https://doi.org/10.1109/CLOUD.2016.0060

numpy. (2023). *routines.linalg.html*. Retrieved from: https://numpy.org/doc/stable/reference/routines.linalg.html

opencv. (2023a). *group__imgproc__transform*. Retrieved from: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html

opencv. (2023b). *group__photo__denoise*. Retrieved from: https://docs.opencv.org/4.x/d1/d79/group__photo__denoise.html

opencv. (2023c). *tutorial_hough_lines*. Retrieved from: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

opencv. (2023d). *tutorial_js_morphological_ops*. Retrieved from: https://docs.opencv.org/4.x/d4/d76/tutorial_js_morphological_ops.html

opencv. (2023e). *tutorial_py_canny*. Retrieved from: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

opencv. (2023f). *tutorial_py_thresholding*. Retrieved from: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, *26*(9), 1277–1294. Available online at: https://doi.org/https://doi.org/10.1016/0031-3203(93)90135-J

Rajan, R. A. P. (2018). Serverless Architecture - A Revolution in Cloud Computing. *2018 Tenth International Conference on Advanced Computing ICoAC*, (pp. 88–93). Available online at: https://doi.org/10.1109/ICoAC44903.2018.8939081

Raman, M., & Aggarwal, H. (2009). Study and Comparison of Various Image Edge Detection Techniques. *International Journal of Image Processing*, *3*. Available online at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.7883&rep=rep1&type=pdf

RobinReni. (2019). *Signature_Verification_Dataset*. Available online at: https://www.kaggle.com/datasets/robinreni/signature-verification-dataset

scikit-learn. (2023). *KMeans*. Available online at: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Sezgin, M., & Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, *13*, 146–168. Available online at: https://doi.org/10.1117/1.1631315

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications* (1st ed.). Springer-Verlag. Available online at: https://szeliski.org/Book/download.php

V. R. Basili, R. W. Selby and D. H. Hutchens, Experimentation in software engineering, in IEEE Transactions on Software Engineering, vol. SE-12, no. 7, pp. 733-743, July 1986, https://doi.org/10.1109/TSE.1986.6312975.

Weisstein, E. W. (n.d.). *Affine Transformation*. From MathWorld--A Wolfram Web Resource.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *2017 IEEE International Conference on Computer Vision ICCV*, (pp. 2242–2251). Available online at: https://doi.org/10.1109/ICCV.2017.244

Zhu, L., & Zhu, L. (2012). Electronic signature based on digital signature and digital watermarking. *2012 5th International Congress on Image and Signal Processing*, 1644–1647. Available online at: https://doi.org/10.1109/CISP.2012.6469828