VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
INFORMATION TECHNOLOGIES STUDY PROGRAM

Problem-Based Project

Digitalization of a business card

Done by:
Narbutas Renaldas signature

Supervisor:
Dr. Bukauskas Linas

Vilnius
2024

Contents

Terminology

1. **OCR** - optical character recognition.
2. **CardDAV** - address book client/server protocol.
3. **NER** - named entity recognition.
4. **Spacy** - open source library that has NER capabilities.
5. **Tesseract** - open source OCR library.

Abstract

The goal of Pullout is to create a UNIX CLI program that can take Lithuanian business card images and convert them into a standardized digital version. It allows you to get rid of large bundles of physical business cards and instead save them on your digital machine. The user client processes the image and stores the contacts locally using SQLite. Image scanning is done by first cropping out the business card and then blurring to reduce noise in image. This filtered image is then sent to the Tesseract-OCR pretrained model to scan text. Afterwards, using the Spacy NER custom-trained statistical language model, it gets entities such as name, email, telephone, etc. After storing the virtual contact locally, it is possible to synchronize it using the CardDav protocol. The purpose of a server is for multiple clients to synchronize the same contacts. Since the client has local data, it is easy to query, modify, delete cards, and create new address book of cards without the need for any internet connection. In the future, the project could be tested with more CardDav servers, enhanced with a larger variety of languages, and the image processing could be detached from the client. It could be put on a separate server in order to make the Pullout client a more lightweight and scalable application.

Keywords: Image processing, CardDav, Tesseract-OCR, Spacy, Image to text

Santrauka

Vizitiniu korteliu skaitmenizavimas

Pullout tikslas yra sukurti UNIX komandines eilutes programą, kuri gali priimti lietuviškas vizitnių kortelių nuotraukas ir paversti į standartines skaitmenines korteles. Programa panaikina reikiamumą nešiotis daug fizinių vizitinių kortelių, pakeisdama į skaitmenines versijas. Nuotraukų skaitymas prasideda nuo korteles radimo, tada iškirpimo iš nuotraukos ir tuomet pridedamas nedidelis teksto suliejimas. Ši išfiltruota nuotrauka yra siunčiama Tesseract-OCR ištreniruotam modeliui skanuoti tekstą. Toliau yra tęsiama naudojant Spacy NER specialiai ištreniruotas kalbos modelis. Jis randa informacija toks kaip vardas, elektroninis laiškas, telefonas, etc. Kai kontaktas įrašytas vietiniam kompiuteryje, yra galimybė sinchronizuoti su serveriu naudojant CardDav protokolą. Serverio tikslas yra surišti daugybe klientų, kad galėtu visuose būti vienodi pakeitimai. Kontaktus yra lengva pridėti, ištrinti, pakeisti, išfiltruoti pagal užklausa, nes viskas vyksta tame pačiame kompiuteryje, be interneto. Ateityje, šis projektas galėtu būti testuojamas su daugiau CardDav serverių, patobulintas su daugiau įvairoves kalbų, ir nuotraukų procesavimas gali būti atskirtas nuo kliento. Galima perkelti į serverį, kuris tai atlieka, tokiu budu padarydamas klientą lengvesnį, ir labiau praplečiama programą.

Raktiniai žodžiai: Nuotraukų procesavimas, CardDav, Tesseract-OCR, Spacy, Image to text

1 Introduction

Pullout is a CLI program that will help you convert physical business cards to digital ones easily and effortlessly. By providing images of contact information it has possibilities to extract the following data: full name, phone number, company name, address, email, job title, URL. Extracted data will be saved to local storage, that is why it works offline, but if storing locally is not satisfactory you can synchronize addressbook data with CardDav servers. Local data can also be exported to PDF format.

2 Related work

There has been numerous research done for image to text scanning

Most projects that have been looked over use Tesseract for scanning text. The difference is image preprocessing, entity gathering and, lastly, data storage.

2.1 Research

Automatic recognition and license plate detection model based on OpenCV and Machine Learning [10], this is a project using matching libraries and tools, like OpenCV and Tesseract.

A more similar project to this one [4] made an IOS app for business card scanning.

2.2 Github projects

There are two similar github projects[7, 14]. They both use Tesseract and Python programming language to achieve digitization of business cards. The first project [14] does not use any image filtering and the NER is based on a dictionary of common words. This is a naive approach as not all images will be good quality and common words have limitations in their scope. The second project[7] has tools similar to our. It has a good NER system, with image filters and cropping.

3 Architecture

The program seen in Figure 1 is interacted with the command line interface. It allows you to process images, query local data, export to PDF. Data is stored in a local SQLite database. Addressbooks can optionally be synchronized with a CardDav server, this allows multiple clients on different machines to have the same contact information.

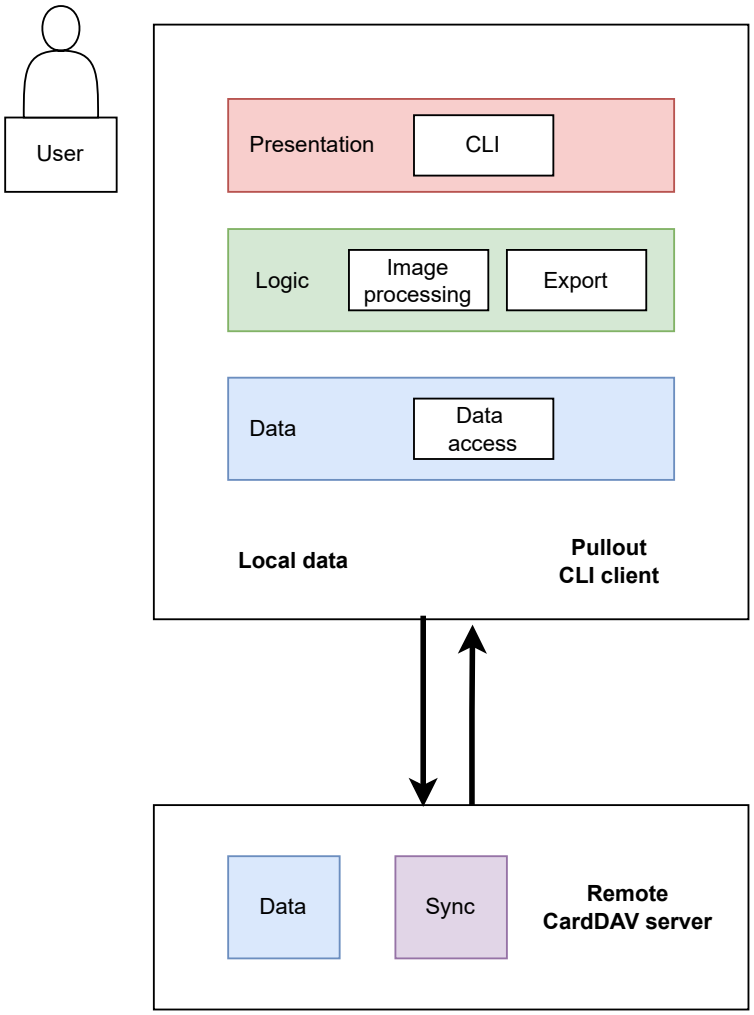


Figure 1. Architecture diagram

3.1 CardDav client-server system

Pullout is a CardDav client and it can synchronize with CardDav servers. The project focuses on having only one CardDav server to one CardDav client and does not handle cases with many to many relationships. The principle can be seen in Figure 2. We can see that there are two Pullout clients on different machines and one CardDav server. If one machine adds, modifies, deletes a contact and the other synchronizes, it will change on the other machine as well. More on CardDav can be seen in section 5

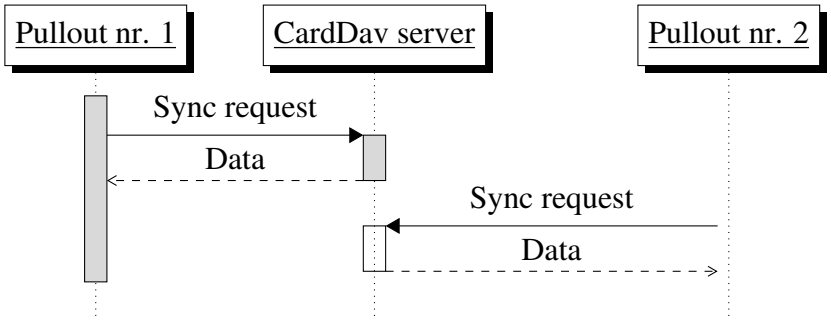


Figure 2. Sequence diagram of server-client system

3.2 E-R diagram

The local database E-R diagram can be seen in Figure 3. This database is not dependant on any users and can live without it, saving it only on local machine. The client does not store data in vCard (vtf) format because it is difficult to query information in that state. It would be difficult because each vCard would need to be parsed before being able to search through it. Instead each contact is added to vcard table where each of the contact components is a separate attribute. More about ctag, etag, href can be seen in section 5.

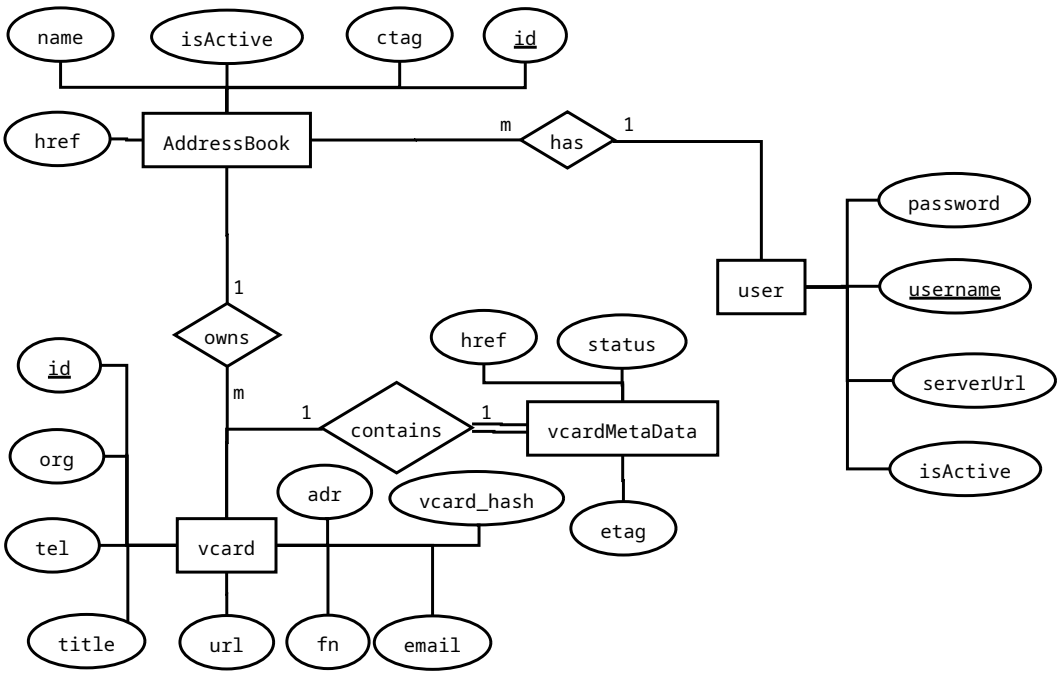


Figure 3. E-R diagram

To avoid contact duplication, each vCard is hashed with the md5 hashing algorithm and checked with the hash in vcard table. The reason hash is not the primary key of vcard table is because it is very long and the only way to delete a vcard in Pullout client is by specifying the id. The id attribute, which is auto incremented, is not too large.

3.3 Requirements

Every good software needs to adhere to some requirements.

3.3.1 Functional

1. The application should be able to accept an image of a business card as input.
2. Support for common image formats: JPEG and PNG.

3. Scan image and recognize data fields: full name, phone number, company name, address, email, job title, URL.
4. Program should be able to easily assemble and disassemble vcard information.
5. Users able to view, edit, create groups and delete digital contacts.
6. Display the scanned cards in an easy to understand format.

3.3.2 Non-functional

1. OCR accuracy at least 90% on Lithuanian business cards with good quality picture.
2. Actions taken by the user, such as picture submission, should provide textual feedback to indicate success or any issues.
3. Clear documentation, detailing program functions and user capabilities, should be provided to help end-users utilize the system effectively.

4 Image processing

Image processing [1] has three major steps: filtering image, scanning text (OCR), recognizing entities in scanned text (NER). Diagram can be seen in Figure 4. If everything is good, then we store it in the database.

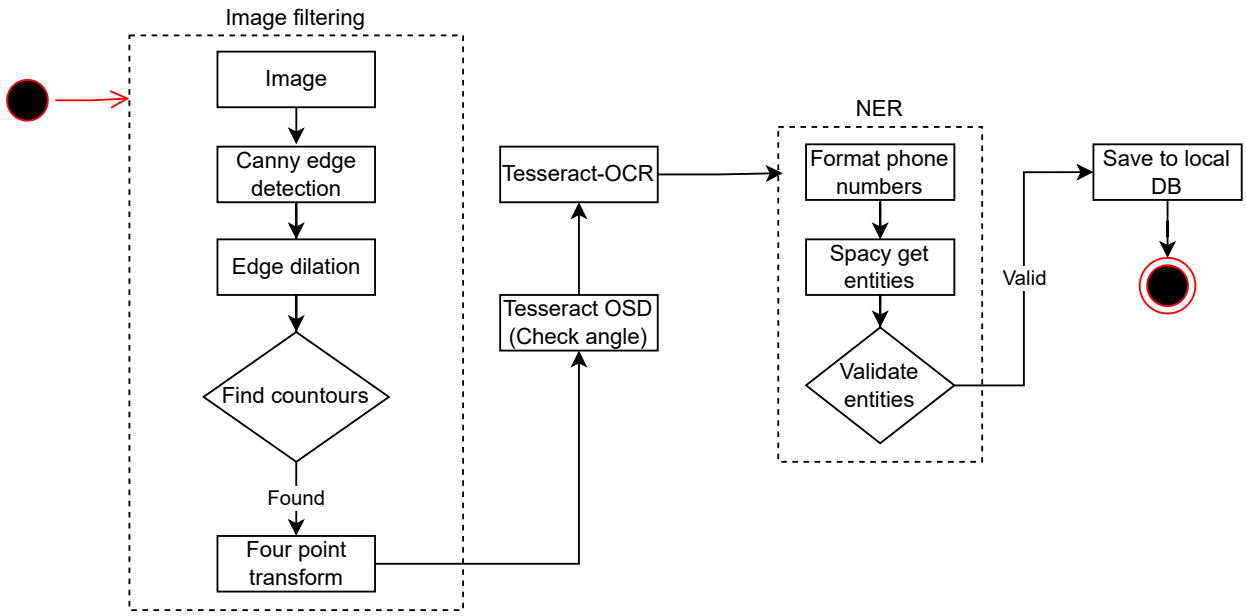


Figure 4. Image processing flowchart

4.1 Image Filters

Filters are necessary because it improves Tesseract accuracy, removes unnecessary noise in background and makes it faster to scan. Ideas taken from [11][10][5].

Image filtering steps:

1. Edge detection using canny edge algorithm
2. Dilate edges
3. Find 4 corners of the card
4. Transform and crop original image using the 4 points

The edges detected from canny edge algorithm are thin and may have some disconnecting points, that is why the edges are dilated to make it into one solid card outline. Afterwards, the following algorithm is used to find the four coordinate points of the card.

INPUT	OUTPUT
Input image $image$	Four coordinates of the image
Explanation	Pseudo-code expression
Detect areas of the image that have the same intensity, which correspond to the boundaries of objects. Return only the start and end points of lines.	$contours \leftarrow$ findContours$(image)$
sort them from largest area to lowest to find the matching rectangle quicker. It will be quicker because the card will be a large contour	$sortedContours \leftarrow$ sortByArea$(contours)$

Find perimeter from all contours that have a closed shape, reduce the number of polygon points from contour by approximating. The difference of perimeters from approximated shape to original shape can be at most 2%. $approx$ contains the points of the polygon. If there are only 4 points, width > 100, height > 100 of bounding rectangle and if each corner of the shape is between 80 < angle < 100 degrees, the business card has been found.	**for** $contour \in sortedContours$ **do**
	  $peri \leftarrow$ findPerimeter$(contour)$
	  $approx \leftarrow$ approxPolygon$(contour, 0.02 * peri)$
	  $w, h \leftarrow$ boundingRect$(approx)$
	  if len$(approx) \rightarrow 4 \ \& \ w > 100 \ \& \ h > 100$ **then**
	  if checkAngles$(approx)$ **then**
	  $fourPoints \leftarrow approx$

crop the input image to the four point coordinates using a perspective transformation	$croppedImage \leftarrow$ fourPointTransform$(image, fourPoints)$

Figure 5. Finding outline of business card

By cropping out the image the output of text has less noise, is more accurate and reliable.
The angles are calculated using high school math:

$$\cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\| \cdot \|\mathbf{v_2}\|}$$

INPUT	OUTPUT
Input four points: $approx$	True or false
Explanation	Pseudo-code expression

Find four angles from four points. Two vectors are taken that go into the same point. Dot product and Magnitude are found and using them calculates the angle	$angles \leftarrow []$
	for $i \in \text{len}(approx)$ **do**
	  $vector1 \leftarrow approx[i-1]$
	  $center \leftarrow approx[i]$
	  $vector2 \leftarrow approx[(i+1) \mod \text{len}(approx)]$
	  $dotProduct \leftarrow \text{dotProduct}(vector1, vector2)$
	  $magnitudeProduct \leftarrow$ magnitude$(vector1) \times$ magnitude$(vector2)$
	  $\theta \leftarrow \arccos\left(\frac{dotProduct}{magnitudeProduct}\right)$
	  $angle \leftarrow \text{degrees}(\theta)$
	  $angles.append(angle)$

Check if either the minimum and maximum of the angles are within the 80 to 100 angle range	**if** $80 < min(angles) < 100 \ \ \& \ \ 80 < max(angles) < 100$ **then**
	  return True
	else
	  return False

Figure 6. Angle calculation

Angle checking verifies that the shape is a rectangle with minimal angle distortions. After all the filtering image is sent to Tesseract-OCR where it scans text from given image.

4.2 OCR

To scan text from the image Tesseract-OCR is used[2]. It internally filters the image, that is why only cropped out image is given without any additional filters. The character whitelist that is given are the following:

```
0123456789abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ąčęėįšųūĄČĘĖĮŠŲŪ@-+.,/\"
```

With a whitelist redundant characters `!@#$%^&()[]{}`, or emoji, or other symbols are ignored. This allows for better filtering of scanned text.

4.3 NER

OCR output generates a lot of characters, some random, some with meaning. To combat this a NER model is employed [6][9]. Spacy is the library that is used for recognizing entities such as

organization, full name, email, etc. It is a statistical language model that is trained with business card examples[12]. Lithuanian cards have been found by asking around acquaintances and some on the internet.

> Language modeling is the task of assigning a probability to sentences in a language. […] Besides assigning a probability to each sequence of words, the language models also assigns a probability for the likelihood of a given word (or a sequence of words) to follow a sequence of words[13]

4.3.1 Statistical language model

Spacy already has many trained language models but every model has a different domain. There were no business card open source models, that is why it was trained from the few examples that had been gathered. For example, there is a spacy pretrained Lithuanian model that was trained on news and it includes full name (PERSON), organization (ORG), Countries (GPE), Locations (LOC) that are useful, but it does not include job title, phone number, email, url. Example training data can be found in appendice A. Training the model can best be described by Spacy documentation:

> Training is an iterative process in which the model's predictions are compared against the reference annotations in order to estimate the gradient of the loss. The gradient of the loss is then used to calculate the gradient of the weights through back propagation. The gradients indicate how the weight values should be changed so that the model's predictions become more similar to the reference labels over time.[13]

Spacy also provides small language models that can quickly check in what language is the text in. The program does not use it because there is only an LT model trained, but it can be added quickly.

4.3.2 Telephone number formatting

Statistical language models work best when data is similar or the same as it has been given in training. To enhance the accuracy, every 8-20 digits that may have "() -+." in between are checked and substituted with a '+' if there was in the beginning, following digits.
For example: (+370) 6 55 18736 would be +37065518736

4.3.3 Validation

To validate whether a scanned entity is correct or not is hard to do for data such as name, job title, address because it is just regular text without a strict format. The opposite can be said for email, telephone and website. They can be checked with the following regular expressions:

EMAIL

```
[\w.+-]+@([\w_-]+(?:(?:\.[\w_-]+)+))
```

Explanation

1. `[\w.+-]+` - Match any word or character '.+-' one or more times

2. `@` - Match '@' character
3. `[\w.-]+` - Match word one or more times
4. `(?:(?:\.[\w_-]+)+)` - Match '.' then word or character '.-' one or more times

TELEPHONE

```
^\+?[\d]{6,12}$
```

Explanation

1. `^\+?[\d]{6,12}$` - Begin optionally with '+' and end with with a digit. Digits count is in the range between 6-12 characters

WEBSITE

```
(?:http(?:s)?:\/\/)?(?:www\.)?([\w_-]+
(?:(?:\.[\w_-]+)+))([\w.,@?^=%&:\/~+#-]*[\w@?^=%&\/~+#-])
```

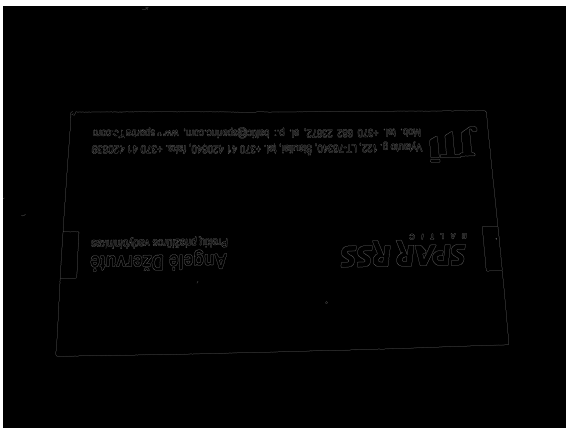
Explanation

1. `(?:http(?:s)?:\/\/)?(?:www\.)?` - Can optionally begin with "http://" or "https://" "www."
2. `([\w_-]+(?:(?:\.[\w_-]+)+))` - Match word one or more times if it has dot and word one or more times later in the structure
3. `([\w.,@?^=%&:\/~+#-]*[\w@?^=%&\/~+#-])` Match query parameters such as /index in website.com/index

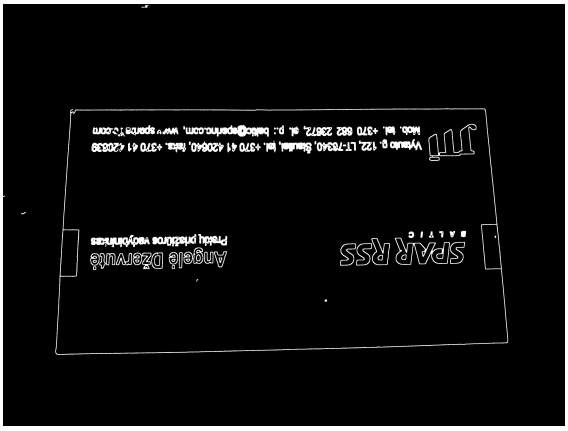
4.4 Real world example

Figure 7. original

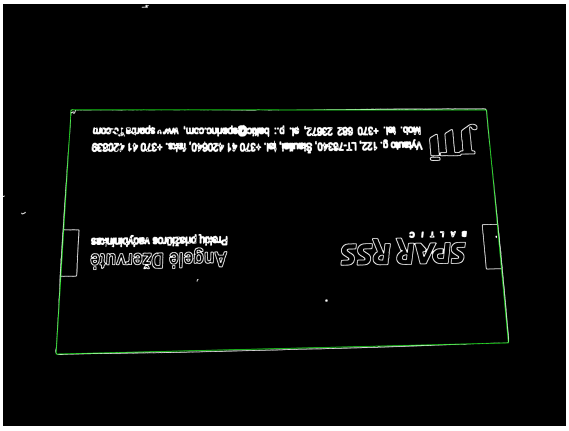
(a) Gaussian blur



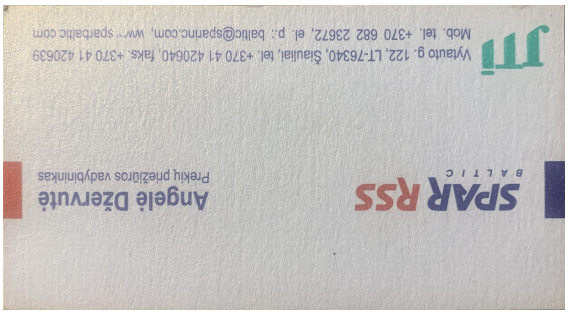
(b) Canny edge algorithm detection



(a) Dilated image



(b) Contoured image



(a) Four point transform

(b) Rotated image

Scanned text:

```
BFALOA LC "ti 00 AM Luis Rrekių prieziūros vadybininkas
i Vytauto g. 122, LT 76340 Šiauliai, tel +37041420640 faks +37041420639 Mob.
r "T. M ĮOrarą i Wi ša F ,
```

Scanned Entities:

```
'CITY': 'Šiauliai',
'FAX': '+37041420639',
'POSTCODE': '76340',
'STREET': 'Vytauto g. 122',
```

```
'TEL': '+37068223672',
'TITLE': 'Prieziūros Vadybininkas',
'URL': 'sparbaltic.com'
```

5 CardDav

CardDav[8] is the protocol used by most contact applications to adhere to a standardized way of communicating contact information between client and server. The server exists for the purpose of synchronizing multiple client contact information. CardDav is an extension of WebDav, and WebDav is an extension of HTTP. Using CardDav protocol we have methods such as PROPFIND, REPORT, MKCOL, LOCK, etc., as well as a set of standardized XML payloads for the exchange of contact information. It also specifies properties that the server needs to have.

Pullout client is the CardDav client and I have configured a CardDav server implementation of SabreDav [3]. The prototype is only tested on this specific server, which has a predefined user with no way to create new ones. It might not be the same as other CardDav servers, but with a more robust implementation, it should work with any kind of CardDav server.

5.1 vCards

CardDav protocol works with vCards [3] seen in Figure 11. The information used by Pullout client is very basic, but there are more possibilities to define types, custom variables and more. More features can be added such as: contact time of creation, home or work email, phone, address.

```
1  BEGIN:VCARD
2  VERSION:3.0
3  FN:Renaldas Narbutas
4  EMAIL:renaldas.narbutas@mif.stud.vu.lt
5  TEL:+370699187366
6  ADR:;;Didlaukio g. 59;Vilnius;;;
7  TITLE:Student
8  ORG:Vilnius University
9  URL:google.com
10 END:VCARD
```

Figure 11. vCard example

5.2 Discovery

By providing a general URL of a CardDav server the program should be able to find out the users addressbook URI. The process goes as follows:

1. Send request with login info to find principal URI
   (e.g. https://myserver.com)

2. Send request to URL + PRINCIPAL-URI to find user addressbook URI
   (e.g. https://myserver.com/principals/johndoe)

3. Request for all addressbook data in URL + ADDRESSBOOK-URI
   (e.g. https://myserver.com/addressbooks/johndoe)

If all three steps were completed successfully, then connection is good.

5.3 Synchronization

Each addressbook contains a CTag (Collection Tag). Each time an addressbook changes, the CTag changes as well. It tracks changes within a collection without having to retrieve the entire addressbook. If CTag is different or locally it does not exist, the program requests vCard metadata called ETags (Entity Tags) in that addressbook. Their purpose is the same as CTags - to track modifications made to a vCard. The program checks if remote is the same as local and if that vCard does not exist locally, requests for the contact information.

The addressbooks and vCards are associated using HRefs (Hyperlinks References or URIs), which are a way to identify where in the CardDav server each resource exists. In this program, each vCard name stored in server is the vCard table primary key.

6 Conclusions and Recommendations

In conclusion, the pullout program allows easy conversion from a physical business card to a digital version. It is an efficient tool for extracting contact information from business cards. The use of the CardDav protocol extends the functionality of the program by facilitating synchronization and integration with contact management systems.

References

[1] Shazia Akram, Mehraj-Ud-Din Dar, and Aasia Quyoum. Document image processing- a review. International Journal of Computer Applications, 10(5):35--40, 2010.

[2] Vernon Estrada Bugayong, Jocelyn Flores Villaverde, and Noel B. Linsangan. Google tesseract: Optical character recognition (ocr) on hdd / ssd labels using machine vision. In 2022 14th International Conference on Computer and Automation Engineering (ICCAE), pages 56--60, 2022.

[3] Cyrus Daboo. Carddav: vcard extensions to web distributed authoring and versioning (webdav). RFC, 6352:1--48, 2011.

[4] Bello Ahmed Dangiwa and Smitha S Kumar. A business card reader application for ios devices based on tesseract. In 2018 International Conference on Signal Processing and Information Security (ICSPIS), pages 1--4, 2018.

[5] Saurabh Dome and Asha P Sathe. Optical charater recognition using tesseract and classification. In 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), pages 153--158, 2021.

[6] Ahmed Hamdi, Axel Jean-Caurant, Nicolas Sidere, Mickaël Coustaty, and Antoine Doucet. An analysis of the performance of named entity recognition over ocred documents. In 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pages 333--334, 2019.

[7] ierolsen. Github of similar project. https://github.com/ierolsen/Business-Card-Reader-App. Accessed: 2023-10-10.

[8] IETF CardDAV Working Group. Carddav protocol. https://tools.ietf.org/html/rfc6352, 2011. Internet Engineering Task Force (IETF) Request for Comments (RFC) 6352.

[9] Darshita Kumar, Shambhavi Pandey, Pooja Patel, Kshitija Choudhari, Aparna Hajare, and Shubham Jante. Generalized named entity recognition framework. In 2021 Asian Conference on Innovation in Technology (ASIANCON), pages 1--4, 2021.

[10] Rahul R. Palekar, Sushant U. Parab, Dhrumil P. Parikh, and Vijaya N. Kamble. Real time license plate detection using opencv and tesseract. In 2017 International Conference on Communication and Signal Processing (ICCSP), pages 2111--2115, 2017.

[11] AS Revathi and Nishi A Modi. Comparative analysis of text extraction from color images using tesseract and opencv. In 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), pages 931--936, 2021.

[12] Xavier Schmitt, Sylvain Kubler, Jérémy Robert, Mike Papadakis, and Yves LeTraon. A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate. In 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), pages 338--343, 2019.

[13] Spacy documentation. https://spacy.io/usage/spacy-101.

[14] thucdx. Github of similar project. [Accessed 2023-10-10].

Appendices

A NER spacy training corpus

This is the type of training data NER model receives. Based on this information it trains to recognize similar word patterns.

Some of the training data contains random characters on purpose because the scanned text usually is not very clean and contains residual noise. The program tries to filter the noise, but it is not guaranteed to filter it cleanly. By giving actual data it will receive instead of the cleaned up versions it is trained to handle more cases.

Entity explanations:

- FN - Full Name;
- ORG - Organization;
- STREET, POSTCODE, REGION, CITY, COUNTRY - part of address;
- EMAIL - email;
- TEL - Phone number;
- FAX - Not used but recognized to avoid thinking fax is like phone number;
- TITLE - Job title.

Example 1

```
Grafo (H) Dvaras kavinė UAB "{Hovalta}(ORG)"
{Algirdo g. 9b}(STREET) {Maišiagala}(CITY), {Vilniaus raj.}(REGION)
Mob.: {+37065945402}(TEL), {+37065590907}(TEL)
El.paštas: {info@grafodvaras.lt}(EMAIL)
{www.grafodvaras.lt}(URL)
```

Example 2

```
{Regina Nasickienė}(FN) {ERGO}(ORG)
{Draudimo konsultantė}(TITLE) rem
- UADB ERGO Lietuva
 - 4. Kalvarijų atstovybė 3 Kalva a: A24
ies 7 {LT-08211}(POSTCODE) {Vilnius}(CITY) r
Tek iB 52777939 set - Mob.tel {86126440514}(TEL)
Faks {852685843}(FAX) Ę- ERGO draudimo grupės narė a a
iek į Leg OS
```

Example 3

```
ki. ar KA "ps En
ų m "K i - A Li 0 9 fa i
ka KIT. kas + MN d 7
```

a ų a iš ų k , 28 pt.

į 2423 Šios "a d A 3

rj .

iki ji 1 , į 4 i – i 4 –
Nkiinaas jį į i a i 4,
niai Ar Ad Ė ii
HOUSE OF PRINCE "27 a 42 alia ma i
si is 4 m ,
– 2 T
{Gintaras Paukštė}(FN) a
surastas {Prekybos plėtros vadybininkas}(TITLE) i a
UAB "{House of Prince Lietuva}(ORG) al – r 2
i {Verkių g. 29}(STREET) Tel {+37052722790}(TEL) wo a
1 {LT-09108}(POSTCODE) Vilnius Faksas {+37052722757}(FAX) ai ki
d ietuva r
į i tel {+37065256656}(TEL) pro
a ntarasOprince.lt Mob. p a