UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"A programmatic approach to finding document edges in images"

Relatore: Prof. Migliardi Mauro

Laureando: Cescon Francesco

ANNO ACCADEMICO 2021 – 2022

Data di laurea 20 luglio 2022

SOMMARIO

Abstract 3

Sommario 3

Introduction 5

The current approach 5

Context and requirements 6

Document detection 7

Image manipulation and pre-processing 7

Canny edge detection 10

Hough transformation 11

Quadrilateral detection 12

Calculating corners and clustering 12

Quadrilateral construction and filtering 13

Scoring and detection 16

Integration and cloud deployment 17

Results and considerations 19

Future adjustments 19

Bibliografia 21

ABSTRACT

It is very common that a valid ID is required when signing up for websites, services and apps. As more and more of our daily tasks move to smartphones, users would find convenient it having the ability to simply take a picture of their form of identification, regardless of the background and orientation.

The proposed algorithm allows the user to process images coming from cameras, identifying the borders of paper documents, to then crop and straighten the selected area to fit a rectangular container.

The proposed solution would not use artificial intelligence models, as it would be extremely difficult to gather a large enough dataset of valid IDs, even after considering data augmentation techniques.

SOMMARIO

È molto comune che un documento d'identità valido sia richiesto per iscriversi a siti, servizi o app. Nel frattempo, molte delle nostre attività quotidiane si sono trasferite sugli smartphone,

L'algoritmo proposto permette di processare immagini provenienti da fotocamere, identificando i bordi di un documento cartaceo, per poi ritagliare e allineare l'area interessata per poter essere contenuta in un rettangolo.

La soluzione proposta non fa uso di intelligenza artificiale, in quanto sarebbe estremamente difficile comporre un dataset sufficientemente grande di documenti d'identità, anche dopo aver preso in considerazione tecniche di generazione sintetica dei dati.

INTRODUCTION

Between 2014 and 2019 in Europe the share of people aged 16 to 74 who used mobile devices connected to the internet grew from 48% to 73%, while in 2021 90% of EU households had access to the internet via a broadband connection. [1]

This socioeconomic context has allowed for a fast and prominent shift to online digital services, accompanied by an increased level of digital skills in the population. [2]

The digital nature of these services is in opposition to the physical form of identification and official documents necessary to start using them. The usual process for providing such documents is to simply take a picture or upload a scan of them. In mobile applications, the user could also be asked to align them to an outline before taking the picture, in desktop solutions handles can be provided to trace the outline.

The current approach

The solutions above, while functional, are certainly not as intuitive or as easy as handing the document to a person. Despite the increasing number of people becoming familiar with online services, some of them might still find current approaches hard to understand, leading to incorrect or partial data and frustration for the final user.

Artificial intelligence is one of the go-to solutions for problems involving computer vision and understanding, thanks to how a machine learning algorithm can autonomously understand and achieve the required output by a process of trial an error.

This training period requires a large enough dataset of valid and correctly tagged documents, which would need to be as sizeable as the precision required. Official documents, however, tend not to be public, making the process of gathering enough of them very hard. Moreover, each country's document has a unique format and design, requiring a more general solution to the problem, to avoid having to train multiple models with numerous types of documents.

On top of these considerations, privacy also needs to be considered when collecting and processing any number of personal and potentially confidential information.

The algorithm proposed in the next chapter takes advantage of multiple mathematical concepts to extract, from a generic image, the rectangular shape that most probably contains the intended document.

Context and requirements

The software described in the following chapters was commissioned by a public institution for higher education in Italy in an effort to update and modernize the admission and enrollment processes.

The school noted that past students, as part of the information required, had to provide a photocopy of their ID document. Although the ultimate goal was to digitize the whole process for students, they were still legally obligated to provide such documents, prompting the development team to look for a convenient solution.

While considering the possible approaches to solving this problem, uploading a photo or scan of such documents appeared to still be the most convenient option. A list of the school's requirements was subsequently collected and extended with requirements for the students and final users of the website.

One of the first characteristics mentioned by the school was the need for accessibility and ease of use of the software, especially knowing that the target audience of such a system is, theoretically, a vast group of people with various types of internet connectivity methods and devices. It also needed to be reasonably fast, while not sacrificing on accuracy in order to limit the amount of human intervention necessary to fix possible miss-detections. Lastly, it needed to be flexible enough to accept and understand different types of file formats, image framing and backgrounds. On top of these functional requirements, the solution needed to be ready and functional before the beginning of the admission window for the new academic year.

DOCUMENT DETECTION

The document detection phase is responsible for identifying the borders of the document within an image. The results can then be used to extract the shape from the image and straightening it to fit within a rectangle.

The input image used in the following paragraphs is image 1, depicting a sample Italian ID document on an irregular background.

Image 1: Original image

Image manipulation and pre-processing

Considering the possible variety in the size and quality of images that an online service can receive, a pre-processing step is necessary to get a clear and usable input.

The first operation performed is resizing the given image. This is to reduce the amount of data that needs to be processed and, also, to have more consistent results, as certain operations might have different outputs based on the image dimensions.

An example of possible output differences is visible in the edge detection step described below, where more edge details are exposed when using the full resolution image. These details are

not necessary to correctly detect the document edges and might even result in less accurate predictions in subsequent operations.

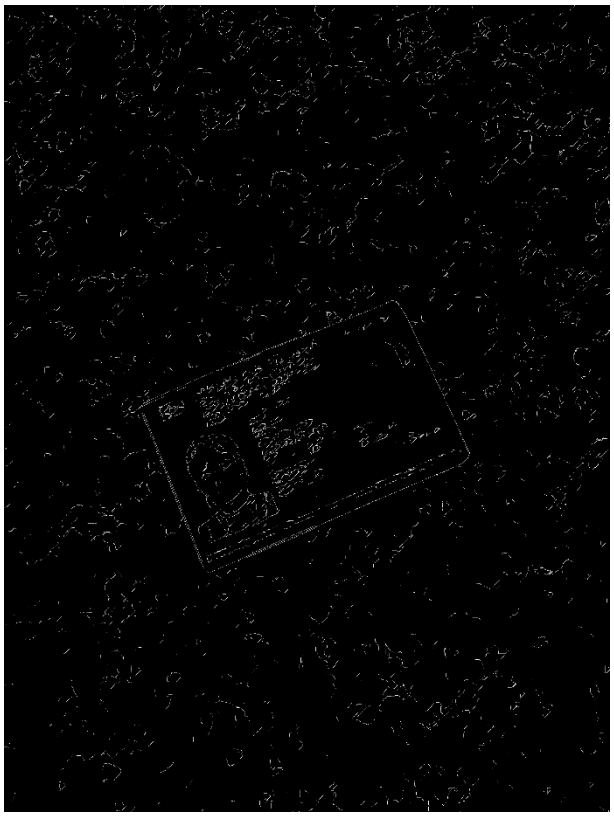

Image 2: Edges detected in the full-resolution image

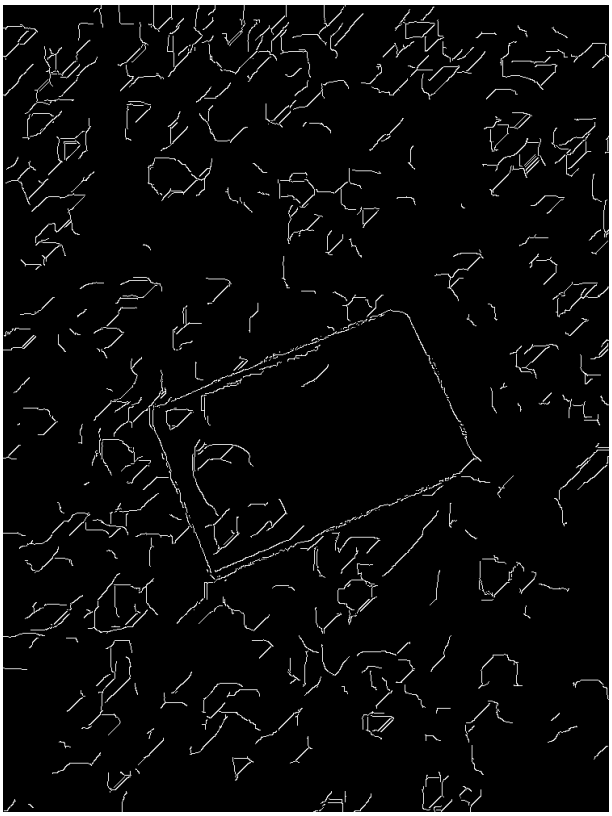

Image 3: Edges detected in the scaled image

For this implementation, the longest side is fixed at 1024px, while the shortest side is scaled accordingly to preserve the original ratio.

To reduce noise in the image, a light Gaussian blur is applied, follow by a set of linear morphological operations on the image pixels.

Morphological operations are used to remove imperfections and unnecessary details from the image by looking at the shape and structure of the image itself. They are based purely on the relative order of pixels, not their specific content. These operations are not performed on the whole image, but are instead applied to smaller portions of a specific shape and size.

The operation used in this implementation is *close*, which is a concatenation of erode and dilate as follows:

$$\text{close}(A, B) = \text{erode}\big(\text{dilate}(A, B)\big)$$

“Dilation is the morphological transformation which combines two sets using vector addition of set elements.” [3] and is formally defined as:

Let A and B be subsets of E^N. The dilation of A by B is denoted by A $\oplus$ B and is defined by

$$A \ominus B = x \in E^N \mid x + b \in A \text{ for every } b \in B$$

Dilatation, represented by the symbol $\oplus$, acts as a local maximum filter, creating a new image where each pixel contains the maximum value within a region in the original image of specified shape and size centered at that pixel.

Erosion, represented by the symbol $\ominus$, is the dual operation to dilation and it acts as a local minimum filter, creating a new image where each pixel contains the minimum value within a region in the original image of specified shape and size centered at that pixel.

The closing operation can be redefined as

$$\text{close}(A, B) = (A \oplus B) \ominus B$$

Its most useful application is to remove small gaps and imperfections in an otherwise solid shape.

These linear operations are iterated for 15 times, to obtain image 4.

Image 4: Pre-processing result

Canny edge detection

Edge detection is a fundamental step in numerous computer vision processes, as it “serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries.” [4]

The resulting image is going to be the same sizes as the original, but it will have only 2 different variants of pixels: non edges and edges, usually saved as black and white respectively.

The Canny edge algorithm was chosen as it is optimal according to three edge detection parameters: low error rate, minimal distance of the detection to the actual edge and minimal number of detected edges for each real one, ideally just one; additionally, the algorithm allows a certain degree of customization using parameters.

The resulting image can be seen in image 5.

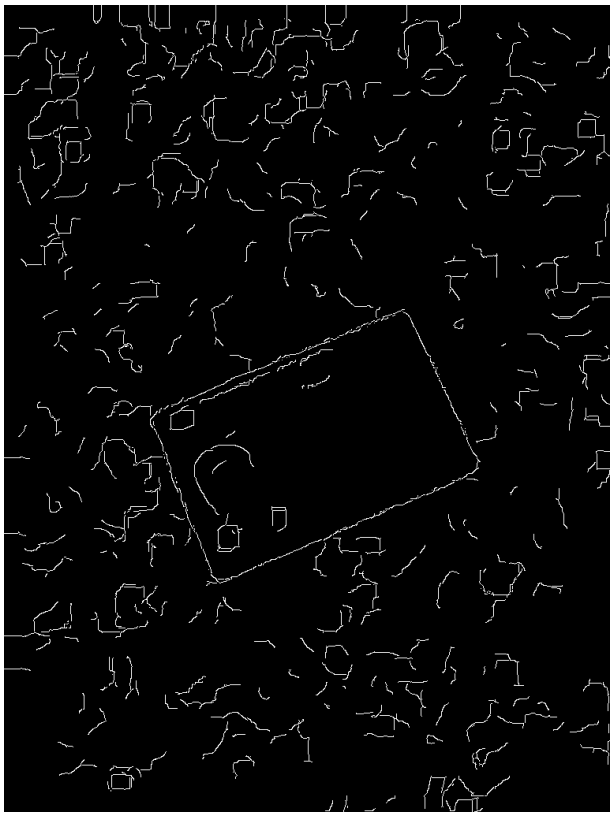

Image 5: Edges

Hough transformation

The Hough transformation is a method for estimating a shape characteristic from its boundary points.

To achieve this, the Hough transformation creates a parallel domain in which to represent the edge image. To detect straight lines a two-dimensional space $\mathbb{R}^2$ is needed that has ρ as one component and θ on the other.

Each edge point (x, y) detected using the Canny algorithm is converted as follows

$$\text{Hough}(x, y) = (\rho, \theta) \in \mathbb{R}^2 \mid \rho = x \cdot \cos\theta + y \cdot \sin\theta$$

Essentially, each (ρ, θ) pair represents the line perpendicular to the segment of length ρ and angle θ, which means that a given (x, y) pair is converted to all the possible (ρ, θ) lines that contain(x, y).

Image 6: Hough transformation of a single point

Image 7: Hough transformation of a horizontal line

Image 7 represents the Hough transformation field of a horizontal line. There clearly is a point at the center of the image where the lines intersect: that is the $(\rho,\ \theta)$ pair that represents the original line. Although there are infinite lines going through each single pixel that composed the original line (each one represented by one point in the field), only one traverses them all.

By detecting the peaks in the Hough transformation plain, we can get a list of all the most probable lines to be present in the original input image.

Image 8: Hough detected lines

Quadrilateral detection

Calculating corners and clustering

To detect where the corners of the document are in the image, the first step is to identify every intersection between two lines. This task is quite trivial for a computer and it gives the opportunity to exclude relatively acute angles, as rectangular documents have right angles which, at most, can be affected by distortion caused by the camera perspective or lenses.

The Hough transformation will often detect multiple lines along the same edge of the document due to noise and imperfections in the original image (image 8), resulting in a number of neighboring intersection points. These are dealt with by combining together multiple points into their average position based on their relative distance, in a process called clustering.

Image 9: Lines intersections and clusters

In image 9 the intersections are marked as green circles and the cluster average position is the red circle. It is also notable how the diagonal line near the top left corner does not have any intersections with other lines, as their angle was outside the acceptable range.

Quadrilateral construction and filtering

Given a set of possible corners and knowing that we are searching for convex shapes, it is possible to compute all valid convex quadrilateral by identifying every set of 4 points having intersecting diagonals.

As the number of corner points increases, so does the number of quadrilaterals. To decrease the number of candidates, each side of the quadrilateral needs to fall within a range of acceptable dimensions, based on the sides of all the other sides of that same rectangle, that take into consideration how perspective may distort the shape of a rectangular document.

The first condition imposed is based on the average length of two opposite segments: this needs to be, at most, double the average length of the other two segments. This prerequisite is based on the assumptions that identity documents are generally not elongated.

The quadrilaterals considered valid based on this first condition, however, may still be outside the subset of possibilities resulting from perspective distortion of a rectangle.

To further refine the list of viable candidates, the maximum difference between the dimensions of two opposing edges needs to be within 5%. If this is not true for both pairs of edges, the tolerance is increased to 10% for the most diverse pair, and left at 5% for the other. This second criteria allows to select only shapes comparable to a square, rectangle, parallelogram or trapezoid.

Analyzing an image that has more than 4 detected corner points, it is possible to see how these two conditions on the quadrilateral sides can greatly reduce the number of candidates passed to the next step.

Considering $c = 9$ the number of corner points detected, the algorithm computes the $d = \binom{c}{2} = \binom{9}{2} = 36$ possible diagonals. Each one of the $q'' = \binom{d}{2} = \binom{36}{2} = 630$ pairs of diagonals is a possible convex quadrilateral.

Analyzing image 10 as an example, this number quickly goes down to $q' = 378$ when excluding quadrilaterals made of the same corners in different orders, and is further decreased to $q = 85$ convex quadrilaterals where the intersecting point of the diagonals is inside the quadrilateral itself.

The first condition, regarding the ratio of two pairs of sides of the quadrilateral, removes 13 candidates, like the one visible in image 11, while the second condition regarding the length of opposite sides eliminates a further 83 candidates, similar to what is show in image 12. The resulting list of valid candidates is, ultimately, composed of only 2 trapezoids.

Image 10: Detected lines, intersections and clusters

Image 11: Quadrilateral excluded because of elongation

Image 12: Trapezium excluded because sides are not within the tolerance ranges

Scoring and detection

Whenever the previous steps leave only one quadrilateral, it is easy to see how that would be the final output, as show in image 13.

Image 13: Detected document in image 1

Image 14: Detected candidates with more than 4 corner points

In every other case, like in image 14, we are left with a list of multiple candidates from which a single output needs to be picked.

To decide which quadrilateral is the final detection, each one is assigned a score based on two criteria: position and size.

To begin, it is safe to assume that the document will be placed roughly near the center of the image, allowing the algorithm to award less points further away from the center. This criterium alone could lead to inexact results if there are multiple valid candidates overlapping, a problem that can relatively easily be solved by taking area into account. This implies that the most probable candidate is the one closest to the center and with the largest area.

These two points are then added together in a weighted sum to give the final score of each candidate. The one with the highest value is picked.

INTEGRATION AND CLOUD DEPLOYMENT

The output of the software is nothing more than a series of coordinates, which is anything but useful to the end user of the service.

The implementation described until now was developed in Python using primarily the OpenCV library, "an open source computer vision and machine learning software library […] built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products." [5] This library allowed to quickly prototype an initial working draft of the algorithm and was also fundamental in the final version, as it contains numerous image manipulation algorithms like resizing, Canny edge detection and Hough transformation.

Integration between this script and the online admission platform was achieved on Amazon AWS using AWS Lambda functions, a service that allows for serverless and event-driven code execution in the cloud[1]. The choice of using AWS as the cloud provider was quite straight-forward as most of the workloads and storage for the school's online services were already hosted there. The Lambda function was deployed in the Milan datacenter, where the website's database and object storage are located, to eliminate latency.

When a user uploads an image to the website, that file is placed by the web server in a specific folder on AWS S3, a very popular object storage service offered by AWS[2]. From here, thanks to the deep integration between S3 and Lambda, an event is sent to the Python script indicating which image needs to be processed and where to find it.

At this point the image has already been uploaded successfully to the website by the user, which does not need to wait for any additional processing steps to be completed. The AWS Lambda function works asynchronously in the background, taking the processing workload off the server, which allows for increased traffic capacity. These characteristics of AWS Lambda also allow to fulfill two functional requirements laid down during the initial phases of development:

[1] aws.amazon.com/lambda/
[2] aws.amazon.com/s3/

accessibility and speed. The former since it's able to accept many different file formats and convert them to something the rest of the script can understand; the latter since the user is not aware that additional processing is taking place and can continue to navigate the website without any additional delays.

Once the processing is complete and the four corners are detected, these are saved alongside the image information in the student's database entry and a straightened version of the cutout is generated and stored back in AWS S3.

In case of an error during the process or a miss-detection in the image, school staff reviewing each student's admission sees the document cutout and is able to manually adjust the corner points should that be needed to improve the legibility or completeness of the image.

Considering the current load of the servers, each AWS Lambda execution is initiated by an upload event. If the workloads become more intense it is possible to limit the number of concurrent AWS Lambda executions, for which each AWS user has a hard limit, by utilizing AWS SQS, a message queuing service[3]. In that case, each invocation is kept in a queue until it is gradually processed.

[3] aws.amazon.com/sqs/

RESULTS AND CONSIDERATIONS

The proposed algorithm allows automated systems to autonomously process document images, extracting the interesting portion for further analysis and validation.

The implementation, as described in the preceding paragraphs and deployed in the cloud, was utilized to process numerous photos during the last admission period and allowed to reduce the amount of manual work needed on each candidate, while operating silently in the background.

Although most document pictures have been successfully processed, some of them required manual adjustments, especially when presenting complex backgrounds or hard to make out edges. The amount of corrections needed was, however, limited and the admission team reported feeling confident that the system, overall, saved time.

Moreover, this time saving was achieved while the school could neglect the cost for the raw processing and storage needs of this script; the relatively small number of cloud execution and the limited size of each image account for less than $1 per month.

Future adjustments

Despite all the positive aspects listed above, the current version of this software still needs some adjustments, specifically in the parameters used by the different steps of the process.

The most challenging images came out to be primarily ones with repeating or geometric backgrounds, which can lead to spurious lines being detected which are not part of the document; this can be especially problematic if they are detected multiple times, creating additional corner points where there shouldn’t be any.

The opposite problem occurs when the image has a solid background that blends in easily with the document edges, making it extremely difficult for the algorithm to distinguish between edges and not-edges. This issue becomes even more challenging when the image gets preprocessed to intentionally remove very fine details, which usually do not contribute in finding the edges.

Finding a more suitable set of parameters will certainly make the algorithm more flexible and adaptive to complex or unclear inputs and could also further limit the amount of manual adjustment required.

BIBLIOGRAFIA

[1] “Digital economy and society statistics - households and individuals”. ec.europa.eu. December 2021. Web. Accessed May 2022. https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Digital_economy_and_society_statistics_-_households_and_individuals

[2] “Digital Economy and Society Index 2021”. digital-strategy.ec.europa.eu. November 12, 2021. Web. Accessed May 2022. https://digital-strategy.ec.europa.eu/en/library/digital-economy-and-society-index-desi-2021

[3] Haralick, R.M., Stenberg, S.R. and Zhuang, X. (1987) “Image analysis using mathematical morphology”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9, 532-550. doi:10.1109/TPAMI.1987.4767941

[4] J. Canny. “A computational approach to edge detection”. IEEE (1986) 679-698

[5] “About OpenCV”. opencv.org. Web. Accessed May 2022. https://opencv.org/about