S3 Project : Defense Report

Beyond the words

Andilath AMOUSSA
Oana Alexandra PATRU
Karthikeyan MADOUSSOUDANANE
Natsuki GAZEL

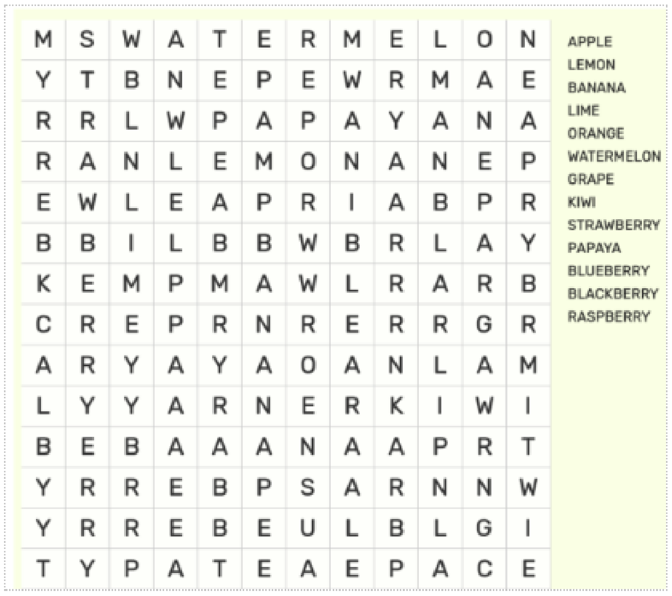
EPITA
(Undergraduate (2nd year))

Contents

1 Project and Group Presentation

1.1 Project presentation

The aim of the project is to make an OCR Software which solves a grid composed of hidden words. The application provides a graphical interface that allows users to load an image in a standard format and display the fully completed and solved grid. The solved grid should also be able to be saved. This is an example of a grid to be solved :



M	S	W	A	T	E	R	M	E	L	O	N
Y	T	B	N	E	P	E	W	R	M	A	E
R	R	L	W	P	A	P	A	Y	A	N	A
R	A	N	L	E	M	O	N	A	N	E	P
E	W	L	E	A	P	R	I	A	B	P	R
B	B	I	L	B	B	W	B	R	L	A	Y
K	E	M	P	M	A	W	L	R	A	R	B
C	R	E	P	R	N	R	E	R	R	G	R
A	R	Y	A	Y	A	O	A	N	L	A	M
L	Y	Y	A	R	N	E	R	K	I	W	I
B	E	B	A	A	A	N	A	A	P	R	T
Y	R	R	E	B	P	S	A	R	N	N	W
Y	R	R	E	B	E	U	L	B	L	G	I
T	Y	P	A	T	E	A	E	P	A	C	E

APPLE
LEMON
BANANA
LIME
ORANGE
WATERMELON
GRAPE
KIWI
STRAWBERRY
PAPAYA
BLUEBERRY
BLACKBERRY
RASPBERRY

1.2 Group presentation

1.2.1 Andilath

The fields that interest me most in computer science are computer architecture, as well as programming and algorithmics. These disciplines offer a fascinating perspective on data structures and our digital environment, which captivates me greatly. During the implementation of this project, I discovered that creating a crossword puzzle is actually more complex than it appears.

This experience allowed me to adopt a different perspective and become more precise in my coding efforts, regardless of the subject. Indeed, every project presents its own set of challenges that require reflection and rigor. Furthermore, working in a group opened my eyes to the crucial importance of organization and communication.

These two elements are essential for successfully carrying out a collaborative project. I was pleasantly surprised by the strong cohesion within our team, which greatly facilitated our exchanges of ideas and collaboration. This positive dynamic enriched my experience and made me appreciate the idea of working together even more. In short, this adventure not only allowed me to develop my technical skills but also reinforced my belief that teamwork is a key factor in the success of any project in computer science.

1.2.2 Oana -Alexandra

My interest in computer science started at a very young age, encouraged by my mother, who motivated me to attend Scratch programming lessons on Saturday mornings. This early exposure sparked a curiosity that quickly evolved into a deeper passion. Since then, I haven't stopped seeking out information on the latest technological innovations, reading about advancements, and exploring new fields within computer science. Over time, my fascination grew specifically towards artificial intelligence and the sophisticated models developed to identify and classify similar images. I found this area of study intriguing, as it combined both creativity and complex algorithms to solve real-world problems.

In this project, I chose to focus primarily on image processing because I see it as the foundation of effective computer vision tasks. Understanding how an image is "preprocessed" is crucial to me, especially because of a past experience with a project where the aim was to recognize the faces of different actors from popular TV series. In that project, I realized how essential preprocessing was; unfortunately, the preprocessing steps were incredibly complex and time-consuming, which ultimately cost me the top spot in an international competition. That experience taught me just how much the quality of preprocessing can impact the entire workflow and accuracy of the results.

For this current project, I've taken that lesson to heart. I am dedicating extra time to fully comprehend each aspect of image preprocessing, ensuring that every image is processed with the utmost precision. My goal is to master each stage, from noise reduction and binarization to more advanced segmentation and feature extraction techniques. By paying close attention to these foundational steps, I am determined to achieve highly refined images that will maximize the accuracy and reliability of the final outputs. I believe this level of thoroughness is key to reaching the best results in any project related to image recognition and classification.

1.2.3 Karthikeyan

In this project, for this defense, I have done the solver, I was also in charge of the neural network together with Natsuki.

My interest in computer science appeared in high school when I chose to specialize in that subject. From the start I had difficulties understanding algorithms and their intricate mechanism as it was all new to me. Throughout high school I deepened my understanding of computer science, yet I was not confident in my abilities and always felt lacking. Nevertheless, my interest in programming and its very logical and mathematical aspect has always drawn me to continue making efforts to improve, now at EPITA I feel confident and able to be an active member in this project.

All the members in this project are people I have never worked with in the past. I knew that working on this project together would involve a lot of communication with will be needed to grasp the way everyone in this group operates.

Heading into this project, I was very motivated to do the solver as it seemed very interesting to me, I was already thinking about ways to approach the functions before starting the project.

The neural network was also a task which caught my attention, it is something very new and does not have anything to do with anything that we had done before, it is very different from the solver, doing both of those tasks seemed like it would be a very interesting learning experience. This also allowed me to focus on elements that are not related to images which need much knowledge to handle. The workload would be more evenly distributed that way.

1.2.4 Natsuki

I'm interested in computer science because it plays a fundamental role in our daily lives and surrounds us everywhere. I believe understanding the modern world is hard without a grasp on computer science.

The fields that I am particularly drawn to are fields like game design, UI/UX and software development, as they combine creativity with technical skills.

This semester's project on building a word search puzzle solver is especially engaging because it introduced me to topics like image processing, which has applications in game design and animation, and machine learning, a tool that is used in many fields. Learning these skills is a great foundation for future projects.

Working on this project as a team has also been a good experience. This allowed us to share ideas and track our progress together. I believe group work in computer science can be especially beneficial, as it brings together different perspectives and ideas which brings innovation, which is the foundation of computer science.

2 Task Distribution

Tasks	**Members**
Loading an image	Andilath
Removal of colors	Oana-Alexandra
Manual rotation of the image	Andilath
Detection of positions	Oana-Alexandra and Andilath
Saving each letter as an image	Oana-Alexandra
Solver	Karthikeyan and Natsuki
Neural network	Natsuki and Karthikeyan

We decided to structure the project in this way to better align with the unique interests and strengths of each team member, so that we could ensure both expertise and engagement. Karthikeyan and Natsuki have shown a strong motivation to focus on the solver implementation and the design of the neural network, two areas they find particularly interesting. By focusing on these aspects, they will delve into the challenge of developing efficient and innovative solutions, as well as machine learning which will most likely benefit them greatly in the future.

Meanwhile, Oana and Andilath showed a preference for working on the image processing components, where they can leverage their knowledge and skills in visual analysis. This area allows them to explore advanced image processing techniques and apply complex visual analysis methods, which will play a crucial role in making the project's outputs more accurate and insightful.

This distribution of tasks not only enables each team member to make the most of their individual strengths but also provides an opportunity to further develop their expertise within specialized domains. It creates a balanced workflow where everyone can contribute effectively to distinct but interconnected parts of the project, ultimately strengthening the overall quality and cohesiveness of our work.

2.1 Loading an image

At first, I found it difficult to understand how to display an image using SDL. I wasn't certain which functions to call or how they worked together to create the graphical output I wanted. To get started, I examined the main file of one of the SDL functions from our assignment, hoping it would provide some guidance. While this helped me identify certain elements, there were still parts of the code that remained unclear, particularly the exact roles and relationships between some functions involved in rendering. To improve my understanding, I decided to do further research online. I reviewed articles and explored several specialized forums where others had shared their experiences and insights with SDL. Additionally, I watched multiple YouTube videos that broke down the basics of graphical rendering with SDL, covering key concepts like creating windows, loading images, and managing textures on the screen. With the help of these resources, I gained a clearer understanding of how SDL's graphical elements interact, and I was able to grasp the steps needed to load and display an image properly on the screen. This combination of research and video tutorials was invaluable in helping me build a foundational understanding of SDL's display mechanisms.

2.2 Removal of colors

To remove colors from the images, it's essential to follow several precise steps. First, we need to apply a grayscale filter to all images, which removes most of the colors while preserving variations in light intensity. Once this step is complete, we apply a contrast filter to further enhance the differences between shades. This helps bring out image details by increasing contrast, ensuring better visual quality before the final black-and-white conversion. After these adjustments, we then apply the black-and-white filter to fully eliminate any trace of color, resulting in a clean monochrome image. Initially, we also thought we needed to apply a noise reduction filter to remove unnecessary details and smooth out the image. However, after half a day of intense work, we finally realized that this step wasn't required for the first defense but would only be needed for the final one. This misunderstanding led to several extra hours of work on these functions, as we had already started implementing the noise reduction filter before understanding it wasn't necessary at this stage.

2.3 Manual rotation of the image

It took me nearly half a day to complete this task. At the beginning, I struggled to understand how certain elements worked, and I frequently confused textures with surfaces, which caused me to lose quite a bit of time. Initially, I had estimated that three hours would be enough to accomplish this function, but in the end, I had to dedicate much more time than expected. This discrepancy from my original estimate is mainly due to interpretation difficulties and the mistakes I made along the way.

With the experience I gained during this task, I hope to spend much less time on similar functions in the future and to better anticipate the technical aspects to come.

2.4 Detection of positions

The objective of this module is to analyze an input image, detect the initial black pixel region, and overlay a grid based on estimated cell dimensions. This grid provides structure, allowing for the segmentation of characters or symbols within a predefined cell size for further image analysis or OCR tasks. By aligning grid cells with specific regions of interest, we ensure that the image is optimized for subsequent recognition steps.

2.5 Saving each letter as an image

The module consists of three main functions: collect_black_pixels, group_adjacent_pixels, and create_images_from_groups. Each function plays a specific role in preparing and exporting the data, ensuring effective segmentation and preservation of character integrity.

2.6 Solver

The solver is a very important part of the project, in this defense, there were many tasks involving the usage of images and working around them. However, the solver does not incorporate any element involving images as this one takes as an input a text file and returns positions of letters in the grid being a matrix. The solver is composed of different functions with each of them having a different role, first we have a function which takes a text file and reads it, taking every letter one by one, transferring them into a matrix for each line. Next, we can start working with the elements of the matrix, to do this we create a new function which goes through each element of the matrix, thus we get a double loop, in this loop we will have to call another function which looks for the word with the given letter as an entry, this will allow us to look for the word from each element of the matrix. The function which checks for the word is long and tedious, but it is not particularly difficult to say. It will go from the given letter in the matrix and increment by one to go through the matrix, at the same time we must use that index to go through the word and check if the characters are identical. This method will be used 8 times (at most) to go through each direction of the matrix being: up, down, left, right and the diagonals. Finally, the main function will call all the others in order to get a return value, it is important to make a to upper () function which gets the input and makes it capital letters to match with the grid, we also verify that the word given as input is only composed of letters and also that the matrix is at least composed of 5 columns and 5 rows.

All in all, the solver does not incorporate many technical elements, with the most technical one being reading the file in C which I had not done before, it was interesting to make use of the knowledge about elements such as malloc that we have used during the bimester. Here is an example of the output shown with the handling of different cases :

2.7 Neural network

Learning about making a neural network was an incredibly interesting experience. It deepened my understanding of how machines can learn and make predictions. I discovered how using training data can help the algorithm learn the relationship between the inputs and the outputs. For this defense, we had to make a simple neural network to learn the XOR function. There were many new things to understand and it took a lot of research to be able to begin working on it. The XOR problem is particularly interesting because it's not linearly separable. Therefore a single-layer perceptron couldn't have solved it. Throughout this process, I also learned the importance of tuning hyperparameters, such as the learning rate and the number of epochs, to optimize performance. Overall, creating a neural network to solve the XOR function was a challenging and fulfilling experience. It not only reinforced my understanding of machine learning but also made me more interested about the broader applications of neural networks in many fields, from image recognition to natural language processing.

3 General Progression

TASKS	**EXPECTED**	**COMPLETED**
Loading an image	100%	100%
Removal of colors	100%	100%
Manual rotation of the image	100%	100%
Detection of positions	100%	100%
Saving each letter as an image	100%	100%
Solver	100%	100%
Neural network	100%	100%

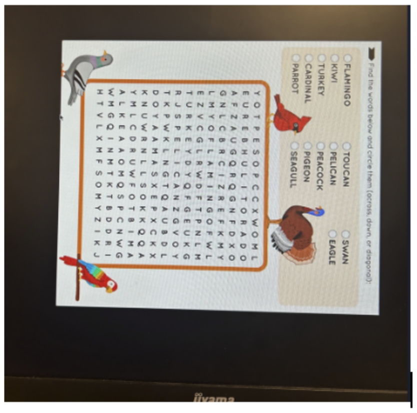
4 Technical Aspects

First, it is essential to load an image, which is a fundamental step for the smooth progress of the project. To achieve this, we use the SDL (Simple DirectMedia Layer) library, which offers a powerful and versatile range of media management features. SDL makes it easy to create a window where the image will be displayed, as well as to generate a graphical rendering. With its multiple advantages, this library also simplifies the creation of a texture, which is used to handle and manipulate the image in memory. Additionally, SDL facilitates the loading of the image in an appropriate format, ensuring maximum compatibility and quick access to graphic data. This approach enables effective management of image display and processing in our program.

To implement manual image rotation, we simply needed to enlarge the window so it could contain the entire image after rotation. This step ensures that even if the image is rotated to a significant angle, it remains visible without being cut off by the window's edges. Next, we used the SDL_RenderCopyEx function, which offers advanced manipulation capabilities, especially for rotation, thanks to its numerous parameters. This function takes as inputs the renderer, the image texture, and a rectangle defining the area where the rotation will be applied. This rectangle allows for precise control over the portion of the image affected by the transformation. One of the key parameters is the rotation angle, which can be specified in degrees and enables the image to rotate to any degree, offering great flexibility. The SDL_RenderCopyEx function also allows specifying a central point around which the rotation will occur. This parameter is particularly useful for adjusting the center of rotation, making it possible to control whether the image rotates around its center, one of its corners, or another specific point. Thanks to this combination of parameters, SDL_RenderCopyEx facilitates image rotation along customized axes and angles, providing great versatility for the needs of our project.

Here is an example of a rotation of 90 degrees:

Find the words below and circle them (across, down, or diagonal):
FLAMINGO
KIWI
TURKEY
CARDINAL
PARROT
TOUCAN
PELICAN
PEACOCK
PIGEON
SEAGULL
SWAN
EAGLE
Y O T P E S O P C C X W O M L
E U R E B H U L I T O R A D O
A F Z A U G Q R Q G N F D X O
G N L C B R C I Z R E F K M Y
L M Y O F L A M I N G O F W L
E Z V C G L R W D F T P N L M
T U R K E Y D Y Q F G E U K G
R J S P E L I C A N Z G V O Y
T K P W K L N G T Q A U B D L
D O Y A A M A S X E P X C K X
K N U W R N L L S O K K Q Q A
Y M L C D R U W F O T B I M A
A L K E A A O M Q S P C N W G
W M G Q I N M T K T B D D R I
H T V L X M F S O M Y Z I K J



4.1 Filter

Basically for the filtering part of this project, each filter that is applied to an image involves iterating through each pixel, extracting its RGB values, and applying a formula to the pixel to change its color. Since we must use SDL for this projet we have to :

- Lock the SDL surface to ensure safe pixel access
- Loop through each pixel, retrieve its color, calculate value of the new color of the pixel, and set the new color
- Unlock the surface after processing.

For this first defense, the filters that were implemented were the grayscale filter, the contrast filter, the gamma filter and the threshold filter. The grayscale filter main scope is to Simplifies the image by reducing color information, making it easier to process while retaining essential structural details. To do so, it reduces an image with three color channels (Red, Green, Blue) into a single intensity channel. Each pixel's grayscale intensity is calculated based on a weighted sum of its RGB values using the following formula Gray=0.2989 × R. +. 0.5870 × G. +. 0.1140 × B. This formula gives more weight to the green channel, as human eyes are more sensitive to green. The result is a single-channel image where each pixel represents the luminance of the original.

The contest filter increases or decreases the contrast, enhancing the difference between dark and light areas. Contrast adjustment typically involves stretching or compressing the pixel intensity values. This can be done linearly or non-linearly, depending on the method used. Linear contrast stretching scales the pixel values to cover a wider or narrower range given a factor contrast which redistributes pixel values to enhance contrast:

- If the factor contrast is below 1.0 it lowers the contrast making the difference between light and dark areas less pronounced. When contrast is decreased, the intensity values (RGB) of pixels are brought closer to a middle range. Dark areas become lighter, and bright areas become darker, reducing the overall range of intensity differences. This reduces sharpness, which can be detrimental if detail visibility is crucial, such as in edge detection for OCR tasks
- If the factor contrast is above 1.0 it increases the contrast making the light areas lighter and the dark areas darker, creating a more pronounced difference between these regions giving the image a a sharper, more dramatic appearance. Higher contrast values stretch the intensity range of pixels, pushing low-intensity (dark) values lower and high-intensity (bright) values higher. This amplification brings out edges and details, especially useful for OCR and feature extraction. However, overly high contrast can lead to "clipping," where details in the darkest and lightest areas are lost (i.e., dark areas turn solid black and light areas turn solid white), potentially eliminating useful information.

The gamma correction, adjusts the brightness of an image in a non-linear way, enhancing or diminishing midtones without significantly altering the darkest and lightest parts. It applies a power-law transformation to the pixel values. Lowering the gamma value makes the image brighter, while increasing it darkens the image. It take a gamma correction value that adjust the brightness of the image :

- If the correction id below 1, it brightens the image. Shadows and midtones are made lighter, making details in dark areas more visible
- If if is above 1, bright regions are preserved, but shadows and midtones become deeper.

Regarding the threshold filter, it Converts a grayscale image into a binary image, where each pixel is either black or white. This simplifies the image significantly, highlighting regions of interest while discarding unnecessary details. A global threshold value is fixed and applied to decide whether a pixel should be turned white or black. For global thresholding, a single threshold value is applied across the entire image.

4.2 Detection

This module contains three main components: the cell size estimation, grid overlay, and display functions. Each function contributes to identifying, processing, and segmenting critical parts of the image :

- **Function**: estimate_cell_size_from_first_black_region : This function detects the initial black region in the image and uses its dimensions to estimate the grid cell size. The assumption is that each black region corresponds to a cell or area of interest, allowing the function to infer grid dimensions that align with the image's content. The function calculates an estimated cell size based on the maximum dimension (width or height) of the black region. A small padding is added to ensure that each cell is slightly larger than the detected area, providing flexibility for minor variations in size. This estimation is crucial for drawing the grid, as it adapts to the specific features of the image.
- **Function**: draw_grid_from_point : Using the estimated cell size, this function overlays a grid starting from the first detected black region, with constraints to prevent overlap with unnecessary areas. The grid lines segment the image into clear, defined cells aligned with key content areas. his results in a clear grid with cells that align well with the image's structure. The grid lines are drawn in a predefined color (LINE_COLOR), making the overlay visible while preserving essential image content.

- **Functions**: load_image and main : The main function manages image loading, grid processing, and output. This part of the module enables interactive visualization and saves the processed image for further analysis.

The modular design of this pipeline supports adaptive grid placement while minimizing overlap with key content areas, enabling efficient, accurate segmentation. This adaptability to diverse image layouts underscores the effectiveness of this approach for image analysis, yielding reliable outcomes in structured data environments. However, for the moment it only works for the images of level 1.

4.3 Decoupage

The module consists of three main functions: collect_black_pixels, group_adjacent_pixels, and create_images_from_groups. Each function plays a specific role in preparing and exporting the data, ensuring effective segmentation and preservation of character integrity.

- **Function**: collect_black_pixels : The primary function of this step is to detect all black pixels within the input image surface. Black pixels are identified based on a predefined threshold, which determines the RGB intensity level below which a pixel is considered "black." The function iterates over each pixel in the SDL_Surface and retrieves the RGB values. If all three RGB values fall below the BLACK_THRESHOLD (set to 50), the pixel is classified as black and its coordinates are stored. A dynamically allocated array is used to store the coordinates of each black pixel in a structure (Pixel). The memory allocation is checked for errors, ensuring robust memory handling. At the end of this process, a list of all black pixels in the image is compiled, which forms the basis for subsequent grouping.

- **Function**: group_adjacent_pixels : his function organizes them into connected groups based on their adjacency, where each group represents a distinct region or character in the image. A breadth-first search (BFS) approach is used to find and group adjacent pixels. Starting from an unvisited black pixel, the function adds neighboring pixels within an 8-pixel vicinity (to account for slight overlaps or misalignments) to a new group. Each group is stored in a dynamically allocated array, and the visited array keeps track of processed pixels to prevent redundant checks. After the group is completed, its size is recorded and reallocated to save memory. This step results in an array of groups, each containing the pixels belonging to a distinct black region. By grouping adjacent pixels, we ensure that each group is treated as a coherent character or symbol for further processing.

- **Function**: create_images_from_groups : export each group of black pixels as a separate PNG image. This isolates each character or symbol, saving them as individual files to facilitate further analysis. For each group, the function calculates the minimum and maximum x and y coordinates to determine the bounding box. This ensures that each exported image contains only the relevant pixel area, reducing file size and focusing on the character. An SDL surface is created for each group, and the background is filled with transparency to isolate the black pixels. Each pixel is mapped based on its coordinates relative to the bounding box, and the black color is applied to these coordinates. The IMG_SavePNG function from SDL_image is used to save each surface as a PNG file, with error handling for successful image saving. Each black pixel group is saved as a separate PNG file, ready for input into OCR or further pattern analysis tasks.

By thoroughly isolating each group, this approach preserves character integrity, mitigates noise from extraneous pixels, and facilitates streamlined data processing for the recognition model.

4.4 Neural network

To make this neural network, it is first necessary to prepare the training data, which is essential for teaching the network to recognize patterns. Once this data is ready, we have to set up an input layer, one or more hidden layers and an output layer. Our neural network is a multilayer perceptron (MLP). Each layer plays a unique role:

- the input layer receives the data. Here, it has two neurons.
- the hidden layers transform the data. It also has two neurons. Each neuron represents a transformation of the input data that can interact with each other to represent complex patterns.
- the outptut layer gives the network's predictions. It has one neuron.

Initializing weights randomly is essential to start the learning process without bias. If we used the same initial weight for all connections, the network wouldn't learn effectively because all neurons would perform the same operations during training. The activation function, which makes our network non linear, is a sigmoid function in our algorithm. We chose it because it outputs a value between 0 and 1 so it's ideal for interpreting probabilities.

The training process involves two main steps in each epoch: forward propagation and backpropagation.

- Forward propagation is where we input the data and calculate the network's prediction. It passes values through the layers : from the input layer to the hidden layer, and from the hidden layer to the output layer.
- Backpropagation is the process that makes the network learn from its mistakes. This is done by adjusting weights to reduce the error, moving backward from the output to the hidden and input layers and updating each weight based on its contribution to the error. These adjustments depend on the learning rate that we choose.

After the training, we can finally run the network on each input pair to test whether it learned the XOR function or not. The outputs are not going to be the exact expected values, that's because the output indicates probabilities.