S3 Project : Defense Report

Beyond the words

Andilath AMOUSSA
Oana Alexandra PATRU
Karthikeyan MADOUSSOUDANANE
Natsuki GAZEL

EPITA
(Undergraduate (2nd year))

Contents

Page

1 Project and Group Presentation **3**
1.1 Project presentation . . . 3
1.2 Group presentation . . . 4
1.2.1 Andilath . . . 4
1.2.2 Oana -Alexandra . . . 4
1.2.3 Karthikeyan . . . 5
1.2.4 Natsuki . . . 6

2 Task Distribution **7**
2.1 Loading an image . . . 9
2.2 Removal of colors . . . 10
2.3 Manual rotation of the image . . . 10
2.4 Detection of positions . . . 11
2.5 Saving each letter as an image . . . 11
2.6 Solver . . . 11
2.7 Neural network for XOR . . . 12
2.8 Neural network for OCR . . . 13

3 General Progression **16**

4 Technical Aspects **18**
4.1 Manual Rotation . . . 18
4.2 Filter . . . 19
4.3 Detection . . . 22
4.4 Decoupage . . . 23
4.5 Automatic Rotation . . . 26

4.6 Neural network for XOR 27
4.7 Neural Network 28
4.8 Interface 30
4.8.1 Utility Functions 30
4.8.2 Callback Functions 31
4.8.3 Image Processing Functions 33
4.8.4 The Main 34
4.9 Grid Solving, Display, Save 35
4.10 Website 37

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 :

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

First defence

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

Second defence

Tasks	**Members**
Complete treatment of the image	Oana and Andilath
Neural Network	Karthikeyan and Natsuki
Building the grid and word list	Karthikeyan and Natsuki
Grid solving	Karthikeyan and Natsuki
Displaying the grid	Oana-Alexandra
Saving the result	Karthikeyan and Natsuki
Graphical interface	Oana and Andilath
Website	Andilath
Automatic rotation	Andilath

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 :

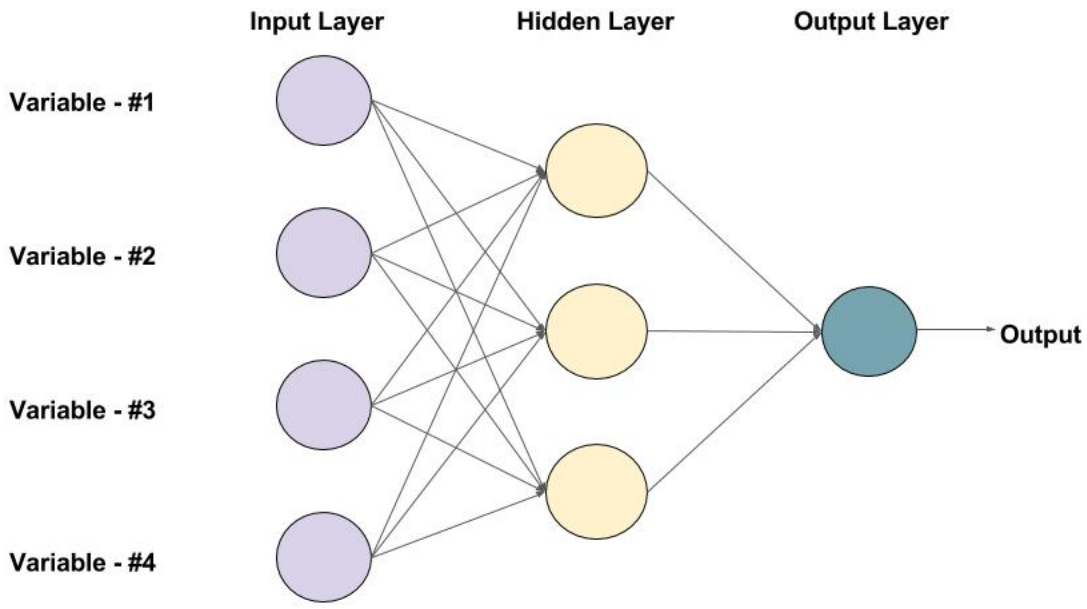
2.7 Neural network for XOR

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.

2.8 Neural network for OCR

Development of the Neural Network Project



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

When we started working on our neural network, we understood the basic structure: the input layer, hidden layer, and output layer like the basics components you can see on the image above . This part seemed simple and clear. However, as we moved forward with building and training the network, we faced many unexpected challenges. What appeared to be a straightforward idea turned into something much more complicated, with many new parts and settings to figure out. This process required us to not only learn new concepts but also solve problems as they came up.

Debugging and Fixing Problems

As we worked on the project, we ran into many technical problems that slowed us down. For example, at one point, the program would only load images of the letter "A" and ignore all other letters. Fixing this problem was frustrating and took time, but it also taught us a lot. By carefully checking how the program was working, we found the issue and fixed it. This experience helped us understand the program better and prepared us to solve similar problems later.
Debugging became a big part of our work. Each time something went wrong, we used debugging tools to figure out what was causing the issue. While this could be very slow and tiring, it helped us make steady progress. It also made us more confident in solving problems as they came up.

Even with debugging, some parts of the training process were very tricky. There were many settings to adjust, like the number of epochs (how many times the network sees the dataset), the learning rate (how fast the network updates itself), and the number of hidden nodes. If any of these settings were wrong, the network would not work properly. It felt like trying to fit all the pieces of a puzzle together perfectly, which could be stressful at times.

Dealing with Hardware Limitations:

One of the biggest issues we faced was the limited power of our laptops. Train-

ing a neural network takes a lot of computing power, and our laptops were not strong enough to handle large workloads. This was frustrating because it kept us from fully testing what the network could do. We often had to make compromises to work within the limits of our hardware.

To deal with this, we tried two solutions:

Batch Training: We divided the dataset into smaller batches to make the training process easier for our laptops to handle. This allowed us to run a few training sessions, but it was not enough to train the network fully.

Reducing Hidden Nodes: We also tried using fewer hidden nodes to reduce the workload. However, this made the network less effective at recognizing patterns, so we abandoned this idea quickly.

Teamwork Made It Easier

One thing that really helped us during this project was working as a team. Unlike some groups that gave the neural network task to one person, we worked on it together. This made a big difference. When one of us felt stuck or frustrated, the other could offer new ideas or solutions. This teamwork kept us motivated and made the process less overwhelming. It also allowed us to split up some of the tasks, which helped us move faster.

Working on this neural network was both challenging and rewarding. From figuring out hidden nodes to building the dataset and dealing with hardware limits, we faced many obstacles. However, each challenge also taught us something new and gave us a better understanding of neural networks. By working together and staying determined, we were able to make progress and gain valuable experience in this field.

3 General Progression

First defence

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%

Second defence

TASKS	EXPECTED	COMPLETED
Complete treatment of the image	100%	100%
Neural Network	100%	100%
Building the grid and word list	100%	100%
Grid solving	100%	100%
Displaying the grid	100%	100%
Saving the result	100%	100%
Graphical interface	100%	100%
Website	100%	100%
Automatic rotation	100%	100%

4 Technical Aspects

4.1 Manual Rotation

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.

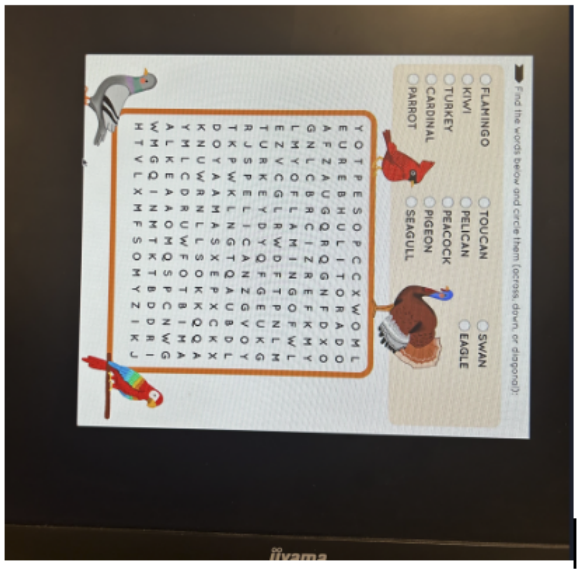
Additionally, SDL facilitates the loading of the image in an appropriate format, ensuring maximum compatibility and quick access to graphic data. This approach enables effective management of image display and processing in our program.

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:



4.2 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

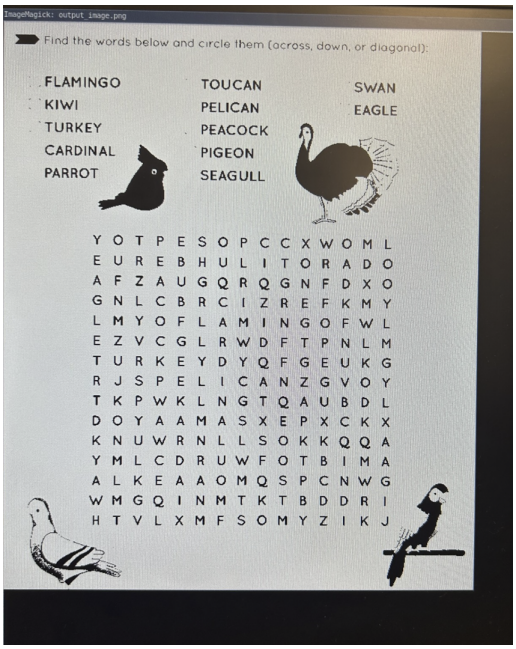
between these regions giving the image a a sharper, more dramatic appearance. Higher contrast values stretch the intensity range of pixels, pushing low-intensity (dark) values lower and high-intensity (bright) values higher. This amplification brings out edges and details, especially useful for OCR and 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.



4.3 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.

4.4 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.

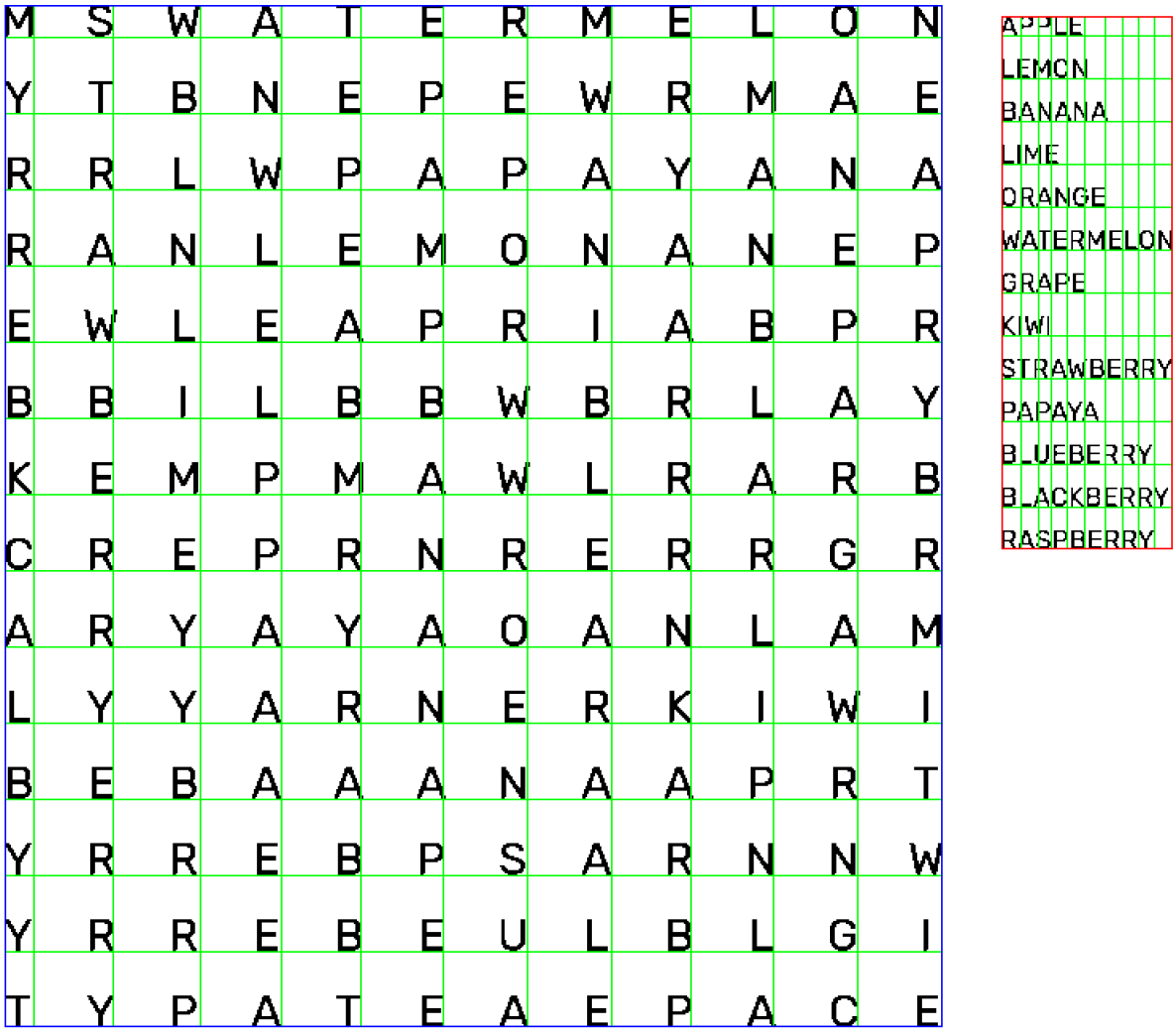
- **Function**: collect_black_pixels : The primary function of this step is to detect all black pixels within the input image surface. Black pixels are identified based on a predefined threshold, which determines the RGB intensity level below which a pixel is considered "black." The function iterates 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.

4.5 Automatic Rotation

Andilath was in charge of the automatic rotation, as it represents the logical continuation of the manual rotation process previously developed. This task played a crucial role in improving the workflow because it aimed to resolve a recurring and time-consuming issue: dealing with images that were initially misaligned, skewed, or improperly oriented. Misaligned images can pose significant challenges, particularly when they need to be processed, analyzed, or presented in a structured and professional manner. Therefore, automating this task was an essential step in enhancing both the accuracy and the efficiency of image alignment.

To accomplish this, Andilath worked on developing a robust and intelligent function capable of detecting whether the content of an image was properly aligned. The function systematically analyzed the image and assessed its orientation. If the content was determined to be already straight and aligned, the program would confirm that no rotation was needed, thus preserving processing resources and maintaining efficiency. On the other hand, if the content was found to be skewed or tilted, the function would then calculate, with high precision, the exact angle of rotation required to restore the image to its proper orientation.

This automated approach significantly reduced the need for manual intervention, which can often be tedious, error-prone, and time-consuming. Moreover, it ensured that the final output image was not only perfectly straight but also optimized for subsequent tasks. For instance, aligned images can be far more effective for further analysis, such as data extraction, pattern recognition, or visual presentation. By minimizing visual noise and inconsistencies caused by misalignment, Andilath's solution also improved the aesthetic quality of the images, making them appear cleaner, sharper, and more professional.

Through her work, Andilath demonstrated a keen understanding of both the technical challenges involved and the practical outcomes required. By implementing this innovative solution, she contributed to a seamless workflow, ensuring that

the images met the highest standards of quality and usability. Her efforts not only enhanced the overall efficiency of the system but also showcased the importance of automation in achieving reliable and consistent results.

4.6 Neural network for XOR

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.

4.7 Neural Network

The Challenge of Hidden Nodes

One of the first challenges we faced was deciding how many hidden nodes to use in the network. We ended up using 64 hidden nodes, but this was not something we expected or understood at first. It took us a while to realize how important this number was and why it needed to be just right.

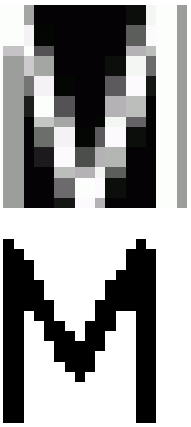
Hidden nodes are key to the network's ability to learn patterns. If there are too few, the network will not be able to handle complex tasks. On the other hand, if there are too many, the network becomes too focused on the training data and does not work well with new data. This is called "overfitting." We had to experiment a lot to find the right balance, which was not an easy process.

Building the Training Dataset

Another major challenge was creating the dataset to train our neural network. A neural network learns by studying many examples, so we needed a large collection of images of letters. However, gathering and preparing this dataset turned out to be much harder than we had imagined.

At first, we tried to manually crop letter images from grids and save them. This method worked, but it was very slow and took a lot of time. Even after spending hours on this, we did not have enough images to train the network properly. We realized we needed another way to get more data.

We decided to look for letter images online. While this gave us more options, it also brought new problems. Many of the images had inverted colors, with white letters on black backgrounds instead of black letters on white backgrounds. At first, we tried training the network with these images, but the results were not good. We then tried flipping the colors of these images to make them match our original dataset. This helped, but mixing the online images with the ones we cropped from grids still caused inconsistencies that made training less effective.



Here are some components we used during the course of the project :

Forward Propagation :

Forward propagation is the process through which data flows through a neural network from the input layer to the output layer. This is the prediction phase, where the network processes input data and generates outputs.

Each neuron in a layer receives input values either from the dataset (input layer) or from the previous layer (hidden layers). These inputs are associated with weights, which determine the importance of each input. Each neuron calculates a weighted sum of its inputs:

Activation Functions :

After computing the weighted sum, each neuron applies an activation function to introduce non-linearity. Without activation functions, the neural network would behave like a simple linear model, no matter how many layers it has.

The sigmoid function which was already used on the previous network.

The softmax.

Training Process

Once forward propagation is complete and predictions are made, the training process updates the network's weights to improve its performance.

Cost Function

A cost function quantifies the difference between the predicted outputs and the actual targets.

Backpropagation :

Backpropagation computes the gradient of the cost function with respect to each weight in the network using the chain rule of calculus. These gradients are used to update the weights.

Gradient Descent

Gradient descent optimizes the weights to minimize the cost function

Loss Functions in Neural Networks

The loss function is a critical component in training neural networks. It measures how far the predicted outputs of the network are from the actual target values, providing a way to quantify the performance of the model. During training, the objective is to minimize this loss, thereby improving the model's accuracy.

4.8 Interface

4.8.1 Utility Functions

Utility functions provide auxiliary functionalities such as displaying dialogs and managing errors. These functions enhance user interaction by providing feedback and notifications.

- show_error_dialog : Displays an error dialog with a specified message to inform the user of issues or failures within the application. Used throughout the application to notify users of errors, such as failed image loading, processing issues, or invalid operations. Ensures users are informed about problems that occur during their interactions.

- show_info_dialog : Displays an informational dialog with a specified message to provide feedback or guidance to the user. Used to inform users about successful operations, provide instructions, or convey general information. For example, notifying users when an image has been successfully loaded or when a processing step is complete.

4.8.2 Callback Functions

Callback functions respond to user interactions with the GUI, such as button clicks. They handle events and execute corresponding actions within the application.

- on_solve_clicked : Triggered when the "Solve" button on Step 4 is clicked. It processes the image based on a matrix file and updates the GUI with the processed image or displays an error if processing fails. Executes the core image solving functionality when the user initiates it via the "Solve" button. It ensures that the processed image is displayed to the user, providing visual feedback on the operation's outcome.

- load_image_clicked : Invoked when the "Load Image" button is clicked. It opens a file chooser dialog for the user to select an image, loads and scales the image, updates the display widgets, and initializes selection mode for region selection. Facilitates the loading of images into the application. Ensures that images are appropriately scaled for display and initializes the necessary structures for subsequent image processing tasks, such as region selection. Enhances user experience by providing a straightforward interface for importing images.

- apply_filter_clicked: Called when the "Apply Filter" button is clicked. It applies a predefined filter to the current image and updates the display with the filtered image. Enables users to apply image filters to the loaded image, enhancing or altering its appearance. Integrates SDL2-based image processing with GTK's display mechanisms to provide real-time visual feedback on the applied filters.

- remove_noise_clicked: Activated when the "Remove Noise" button is clicked. It removes noise from the current image using specified parameters and updates the display with the denoised image. Provides users with the ability to reduce noise within the loaded image, enhancing image quality and preparing it for further processing tasks like filtering or region extraction.

- adjust_contrast_clicked : Triggered by clicking the "Adjust Contrast" button. It adjusts the contrast of the current image and refreshes the display with the updated image. Allows users to enhance or modify the contrast of the loaded image, improving visibility of details or preparing the image for specific analysis or processing tasks.

- rotate_image_clicked: Invoked when the "Rotate Image" button is clicked. It rotates the current image if it matches a specific filename pattern and updates the display accordingly. Allows users to rotate specific images based on filename criteria. Integrates SDL2's rendering capabilities with GTK's display mechanisms to provide visual rotation feedback. Enhances image manipulation capabilities within the application.

- next_step_clicked: Called when the "Next Step" button is clicked. It navigates the user to the next step in the notebook interface (from Step 1 to Step 4). Provides a straightforward mechanism for users to progress through the application's workflow steps. Enhances user experience by allowing easy navigation between different stages of image processing and analysis.

- on_select_region_clicked: Activated by clicking the "Select Region" button. It initiates the region selection mode, allowing the user to select specific areas of the image for further processing. Enables users to enter a mode where they can interactively select specific regions of the loaded image. These selected regions can then be subjected to further processing, such as filtering, extraction, or analysis. Enhances user control over image manipulation tasks.

- on_finish_selection_clicked: Triggered when the "Finish Selection" button is clicked. It finalizes the region selection process and notifies the user that they can proceed to the next step. Allows users to conclude the interactive region selection process. Ensures that the application transitions smoothly from selection mode to subsequent processing steps, maintaining a clear workflow and user guidance.

4.8.3 Image Processing Functions

These functions handle various image processing tasks using SDL2 and GDK Pixbuf. They bridge the gap between low-level image manipulation and high-level GUI updates.

- sdl_surface_from_pixbuf: Converts a GdkPixbuf (GTK's image representation) to an SDL$_{S}urfaceforlow-levelimageprocessingusingSDL.Facilitatesthetransitionofimagedataf$
$levelrepresentation(GdkPixbuf)toSDL'slow-levelformat(SDL_{S}urface)forprocessingtaskssuchasfilt$

- pixbuf_from_sdl_surface : Converts an SDL_Surface back to a GdkPixbuf for display within GTK widgets after processing. Enables the display of processed images by converting SDL's low-level surface data back into GTK's high-level image format. This conversion is essential for updating GUI elements with the latest image manipulations performed using SDL.

- update_image_widgets: Updates the image widgets on pages 1 and 2 with the provided GdkPixbuf, ensuring that both the control page and the display area reflect

the latest image state. Ensures that any image processing changes are immediately visible to the user across relevant GUI components. Maintains consistency in the user interface by synchronizing image displays on multiple pages.

- update_page_three: Updates the third page of the notebook with images resulting from the extraction process. It clears previous images and adds new processed images for user review. Displays the results of image region extractions, allowing users to review and analyze specific areas of the image that were selected and processed. Facilitates visualization of extracted regions, highlighting features such as words and grids, thereby enhancing the application's analytical capabilities.

4.8.4 The Main

Orchestrates the initialization of the GTK and SDL libraries, sets up the GUI, connects signals, and enters the GTK main loop. It defines the application's structure, including the notebook interface with four steps, and manages resource cleanup upon termination. Defines the overall structure and flow of the application. Sets up the GUI components, connects user interactions to their respective functionalities, and ensures proper initialization and cleanup of resources. Acts as the entry point, orchestrating how users interact with the image processing features through a step-by-step notebook interface.

Key functionalities include:

The gtk_interface.c file implements a comprehensive image processing application with a user-friendly GTK-based interface integrated with SDL2 for low-level image manipulations. Each function within the file serves a distinct purpose, from handling user interactions and displaying dialogs to performing image processing tasks such as filtering, noise removal, contrast adjustment, and rotation. The application is structured around a notebook interface with four steps, guiding users through loading images, selecting regions, viewing extraction outputs, and execut-

ing solve operations based on matrix data.

• Image Loading and Display: Users can import images of various formats, which are then scaled appropriately for display within the application.

• Image Processing Tools: The application provides tools to apply filters, remove noise, adjust contrast, and rotate images, enhancing or altering image quality as needed.

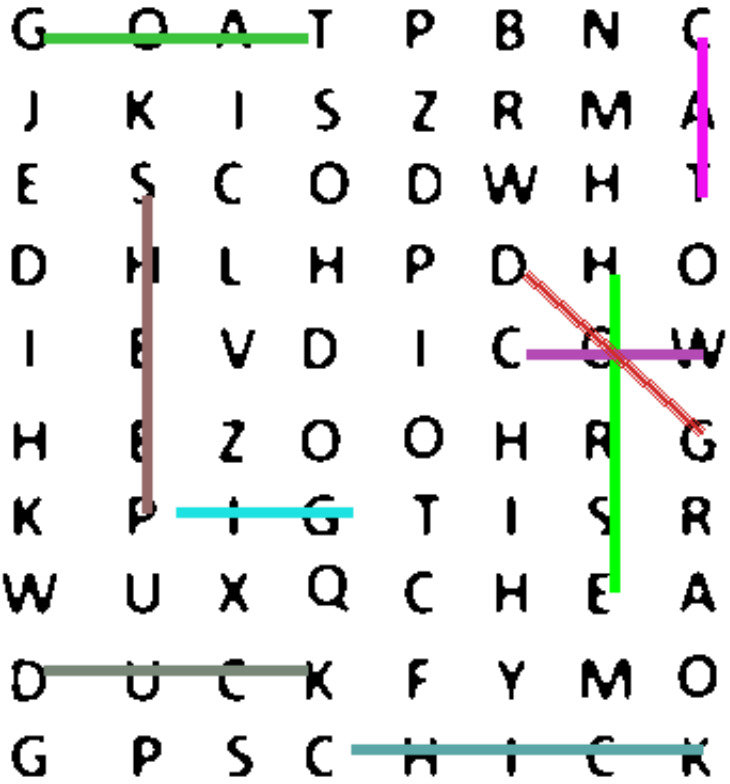
• Region Selection and Extraction: Users can interactively select specific regions within the image for focused processing or extraction, with the application managing these selections and displaying the results.

• Puzzle Solving: The "Solve" feature processes images based on predefined matrices, searching for specific patterns (e.g., animal names) and visually highlighting them within the image.

The application ensures efficient memory management and resource cleanup, providing a seamless and responsive user experience from loading to processing and finalizing images.

4.9 Grid Solving, Display, Save

The implementation of the grid solving and visualization process focuses on transforming the computational output into a clear visual representation.



To do this, we obviously had to implement SDL2 and its associated libraries.

So back to the simple solver : it provides the locations of words as start and end coordinates based on the grid matrix (for example : row 2, column 3).

However, the grid itself is an image, so these coordinates must be converted into precise pixel positions on the image. This calculation allows us to identify the exact area on the image where each word starts and ends.

Also, we take into account pixel colors to ensure the drawn annotations blend well with the image while remaining visible.

This allows us to map the matrix coordinates to the corresponding pixel locations accurately.

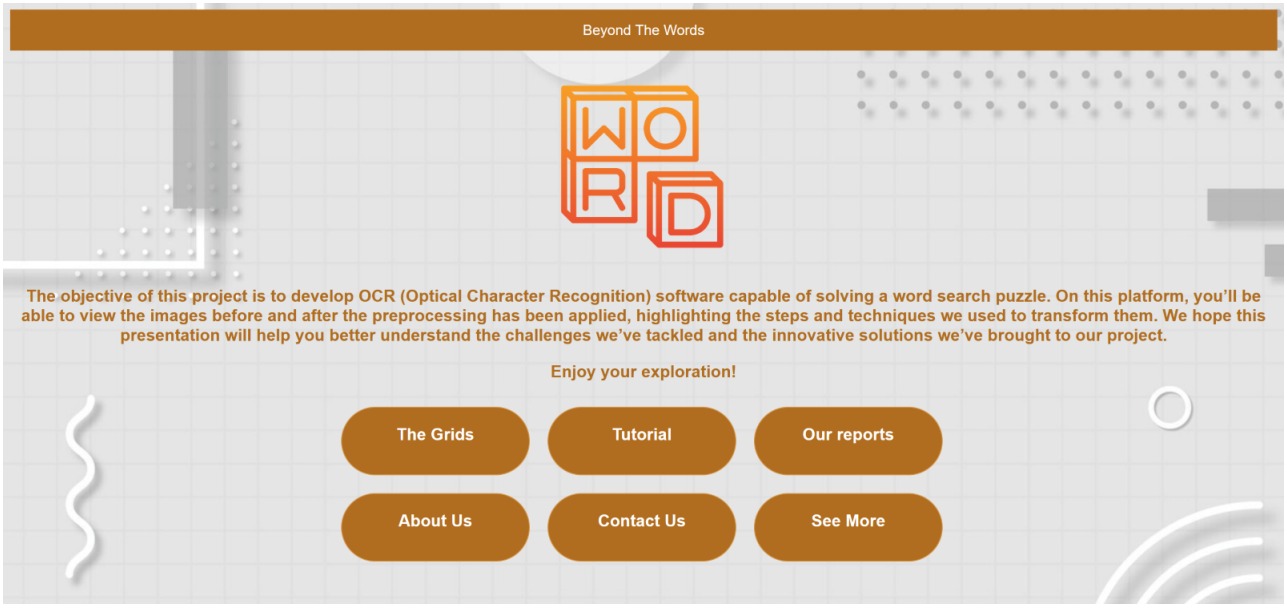
Once we have the pixel coordinates, it is pretty easy to draw the lines directly on the image. We experimented with several settings, adjusting offsets and tweaking the placement logic, until we achieved a good balance where the lines reliably highlighted the correct areas.

The lines' thickness were also nerve-wracking; it was inconsistent depending on the orientation of the lines—horizontal, vertical, or diagonal. To address this, a function changes the thickness parameter depending on the orientation of the line to have equivalent lines no matter the direction.

Finally, to make the results as clear as possible, we incorporated different colors for the lines. Each word is highlighted in a distinct color, preventing confusion when multiple words intersect or overlap in the grid.

4.10 Website

Andilath was in charge of creating the website in HTML. She decided to take on this challenge in order to apply her technical and creative skills to design the best possible website. The purpose of this site is to promote our project while presenting our team in a clear and professional manner. Thanks to her work, we were able to deliver a modern and functional platform that reflects the essence and ambition of our initiative.


Beyond The Words
WO
RD
The objective of this project is to develop OCR (Optical Character Recognition) software capable of solving a word search puzzle. On this platform, you'll be able to view the images before and after the preprocessing has been applied, highlighting the steps and techniques we used to transform them. We hope this presentation will help you better understand the challenges we've tackled and the innovative solutions we've brought to our project.
Enjoy your exploration!
The Grids
Tutorial
Our reports
About Us
Contact Us
See More