EDGE COMPUTING CAR PARK

Project Report

5th year - ISS - T. Monteil

Ilhame Hadouch
Njord Løkebø
Teo Geneau
Kilian Desportes
Yanis Imekraz
Catherine Vaugelade Berg

National Institute of Applied Science (INSA), Toulouse, France
Department of Electrical and Computer Engineering
January 22, 2021

Table of Contents

1 Introduction

A common problem among people arriving at INSA by car is finding a free parking space. During this semester we have carried out a project to solve this problem by utilizing a connected camera and developing the associated software.

To solve this problem we set up a mobile application which can be installed on any phone, which will allow users to navigate to a free parking space. The goal is to make traffic more fluid and reduce the delay due to the search for free places. This application will get its information from a camera system installed on the roofs of the INSA campus, which will detect available spaces and smartphone application will guide the user to the most appropriate location, as close as possible to the building they wish to go to.

Firstly, an empty space is filmed with the camera placed on a roof. This image is then processed locally on a small laptop, an NVIDIA Jetson Nano, which is connected to the camera. This computer will use image recognition to detect whether or not a place is free and will then send the information about all the places to a server. The mobile application will, in a second step, interrogate the server so that it can access the data collected by the map, to allow the application to access the free or not free spaces, and then to guide the user to a parking space.

To do that, we have separated the whole project in 3 parts : Image processing, to go from camera video to place detection, the Server part, to make the data available, and the Application part, to transfer this data and make them user-friendly through an application.

2 Specifications /Limitations

2.1 Overall method

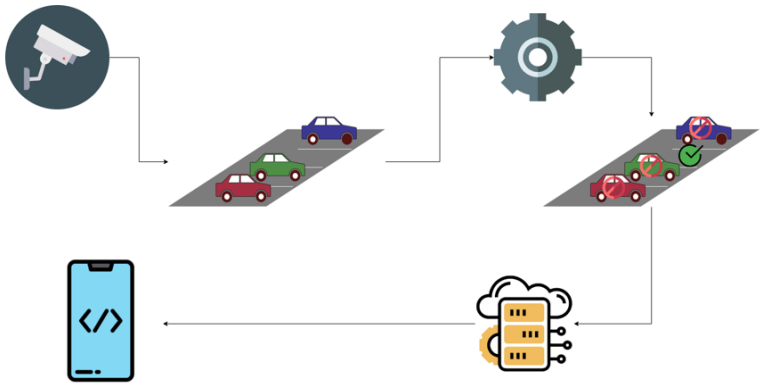

Figure 1: Overview of system

The previous image above shows the overall method of our project. First, we collect images of parking slots using cameras installed on INSA's roofs. Then, we use an algorithm for image processing in order to have the best image resolution possible with less noise. These new images allow us to detect free slots. This information is gathered in the server. When a user needs to park, the server sends the position of a free slot.

2.2 Application specifications

For this application, the 3 main parts are data acquisition, image processing and user interface.

- Data acquisition from the Jetson
  - Position of the camera inside the campus
  - Rotating or fixed camera
- Image processing of spaces
  - Free or not
  - Handicapped
  - Which department is closest to this place
- User interface
  - Department choice
  - Map with different colors for each zone (red/orange/green)
  - Possible to choose only Handicap parking slots
  - Button to start the "Search"

2.3 Limitation of project

The biggest challenge for the project is the visual coverage of the cameras, which is not optimal to cover all the areas with a low number of cameras. This will therefore force us to make a choice between low coverage with few cameras or a wider coverage (like the whole campus) but at much higher costs.

The use of the application is also complicated by the fact that not all motorists looking for a parking space will necessarily use the application. There will therefore be a problem if the person is guided to a parking slot that is already taken by someone, as no one was detected because they are not using the application. This can potentially waste a little time because the person may have to be redirected to another place that may be quite far away.

3 Project Management

This project was completed using the Agile Method and was a request from the client, Prof. Thierry Monteil. The goal of this method is to be close to the client to ensure that our project follows his needs and specifications.

To ensure that this team-client proximity was respected despite the various problems of proximity encountered (lockdown, no face-to-face meetings, etc.), we held weekly meetings throughout the project.

This allowed us, as a team, to get feedback on the progress of the application and whether what we were doing was in line with the customer's requirements. It also allowed us, with the customer as our technical tutor, to have feedback on what we were developing at a technical level and the choices we were making in terms of technology and practices, for example.

Concerning the software development method, we did not really work following the Agile method, which consists in developing a software by successive iteration by adding functionalities.

As our software only needed one and the same functionality, the client asked us to share the workload of the whole software chain (Image Processing - Server - Application) to move forward each on our side so that we could integrate the whole software before the test phase.

We therefore split into two groups of three people, three on the image processing part and three on the mobile application part, with each of the teams also assigned to the server/communication part related to its side of the final architecture. The group assigned to Image Processing was:

- Kilian Desportes
- Teo Geneau
- Yanis Imekraz

and the group assigned to Mobile Application was:

- Catherine Berg
- Ilhame Hadouch
- Njord Løkebø

3.1 Risk Management

We had to assess the various risks associated with the project to see how they could be reduced and managed. We identified several types of risks to which our project was subject, material risks, concerning the material we use, risks related to the organization of the project itself, and risks related to technical difficulties that could be encountered.

Concerning the material risks, the main risk was at the camera level. It could be defective and/or produce a poor quality image. However, we evaluated this risk as being low, the camera we are using being a 'general public' camera (the Raspberry Pi camera), and the impact on the project being low, it could be easily changed without having to change the algorithm much.

The risk related to the project organization was mainly concerning time management. At the beginning of this project, we were not familiar with any of the technologies we were going to use during the whole development chain, from image processing to the mobile application. So we assessed that the risk of the time management could be possible. We considered this risk to be 'green' because we were trained in similar technologies during our degree course, and, being a group of 6, we were quite confident that we would have the time to learn and develop the project during this semester.

The last type of risk identified was the difficulty of implementing and developing of the different stages of the project itself. We thought it was possible to have difficulties with the image recognition or the development of the mobile application. Even more important, the risk of the integration of the two parts (these two parts being developed in parallel by the two sub-teams), which could go wrong and make us waste time, because of problems we hadn't foreseen, technologies that are incompatible, for example.

We can therefore summarize the risks according to their probability and potential impact using the table below.

Impact | Probability	1	2	3	4	5
1	Defective Camera				
2		Time management			
3		Development issue			
4					
5		Integration			

Figure 2: Risks

However, we did not foresee all of the risks, and the absence of Teo, for example, at the end of the project delayed us enormously.

In the end, the risk that affected us the most was that of time management. At the end of the project, with various unforeseen events, we lacked the time to finish the application and make it usable in real conditions.

We were still able to make a prototype and a demonstration to show that the project was completed and functional even if it was not implemented on campus as planned.

3.2 GANTT Chart

At the beginning of the project, we made a GANTT chart, visible below, to give us a guideline during this project, knowing that it would not be respected precisely, because of our lack of

technical experience, the Agile method we used, or the different external constraints that could impact the project (other project reports,etc.). We therefore used this diagram as a guideline to help us position ourselves in relation to our objectives.

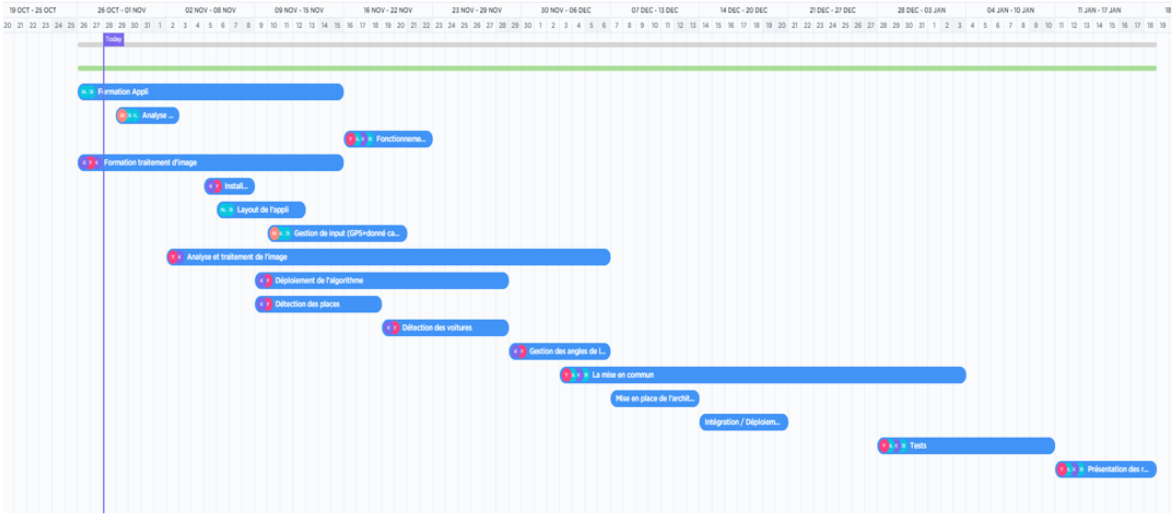

Figure 3: GANTT Chart

With the end of the project, we are now able to analyze our deviations from this diagram. For example, we can cite the fact that the technical parts of each one were undersized, as they took much longer, and therefore the integration/pooling part, which was supposed to start around December, was only realized at the very end of the project, in January. In addition, some tasks were not carried out, such as the tests in real conditions, because of technical parts were not yet developed to ensure this part, or because the priority of this task was low compared to others.

3.3 Tools used

Concerning the tools used during the project, we mainly used 3 types of tools, those for communication, those for document and source management, and those for organization. For the communication, it was mainly done with email with the client for example, and sometimes even within the group, even if the use of Facebook Messenger was what was used first.

For document and source sharing, we used the GitLab provided by Mr. Monteil and a Google Drive for everything related to non-technical documents (photos, power points, etc). Finally, the only project management tool we used was ClickUp, a site that allows us to manage and organize different tasks. It is notably on this site that we created Gantt chart.

Figure 4: Tools used

4 Image Processing - Realisation

4.1 Image Processing - Detection of place coordinates

The purpose of this part was to detect, starting from an image of a parking lot, the different spaces available in this parking lot, and to give the coordinates of these spaces. These coordinates will then be used as a 'region of interest' for the detection of the spaces.

To do this, the detection is carried out in several main steps. The first one is the general processing, in particular to reduce noise, to change the image to gray shades and to detect edges. Once the image is ready and all the shapes present in the image are detected, the second step will isolate the squares representing the squares at INSA then store them according to their coordinates, to deduce 2 or 4 lines of squares in the image (for 1 line of squares, or two). Once these squares are detected, a step of interpolation of the squares allows us to simulate the presence of potentially undetected or visible squares (for multiple reasons). Once this interpolation is completed, the algorithm will define the regions of interest by simply linking the different squares representing the squares.

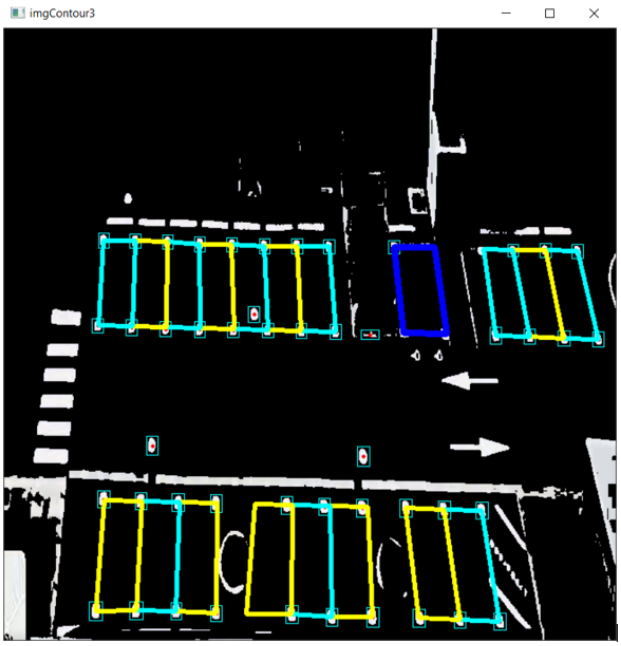

Figure 5: Visual output of detected places.

In order to detect handicapped parking slots, the user has to set some parameters, as the algorithm doesn't detect them, as well as unusable slots. The user must configure the algorithm to determine which slots are occupied by a tree, or if there is a pedestrian passage.

This algorithm is not optimal, particularly the interpolation step, as it requires an initialization phase. That is why we have decided to use a deep learning library using a Haar cascade algorithm, which is a sort of artificial neural network, to ensure automatic slot detection. We created our

own database from car park and car images; it is however not large enough to train our algorithm properly, we will therefore not explore this direction any further.

```
id: 0
handicap_list: 1
coordinates: [[484,628],[526,630],[507,512],[467,511]]

id: 1
handicap_list: 1
coordinates: [[440,626],[484,628],[467,511],[426,509]]

id: 2
handicap_list: 1
coordinates: [[348,625],[391,625],[386,508],[346,507]]

id: 3
handicap_list: 1
coordinates: [[305,622],[348,625],[346,507],[306,506]]

id: 4
handicap_list: 1
coordinates: [[257,622],[305,622],[306,506],[266,505]]

id: 5
handicap_list: 1
coordinates: [[182,619],[225,621],[226,504],[186,503]]

id: 6
handicap_list: 1
coordinates: [[140,618],[182,619],[186,503],[146,502]]
```

Figure 6: Output of the places detection algorithm.

4.2 Image Processing - Detection of free places

For this part, the goal is to take as input the different coordinates of the places given by the detection of place coordinates, and to output, in real time, the different states of places, if they are free or taken.

To detect this state of the places, we had to analyze if a car is or is not in a specific place. To do this, we tried several methods to perform this detection of cars on the squares.

Our first idea was to use a neural network with the TensorFlow library. This idea was quickly abandoned because training a neural network with a data set that we had to build ourselves (not having found one on the internet) seemed a cumbersome and not necessarily obvious approach.

In a second step, we found on GitHub an algorithm based on image recognition with OpenCv. This recognition consisted in applying several successive functions (reduction of noise and detail, switching in grey mode, find the edges, etc) on each square to detect if a car was present on the square.

We simply had to change the different inputs and outputs of the code (the inputs being a manual place detection, and the output a simple visual output on the video). This solution worked well during the tests we did, but it proved to be deeply inefficient during tests with low quality images like the one we had for INSA (which was probably closer to the image we would have with this kind of camera).

Finally, we used a method that was quite restrictive at first glance but very effective, background subtraction. This method allows, from an image of the "empty" parking lot, to detect the difference with another image (with a car) and thus to say whether or not a car is in this place.

Visually, we can determine which places are free or not by the color of the contours, as shown in the next Figure.

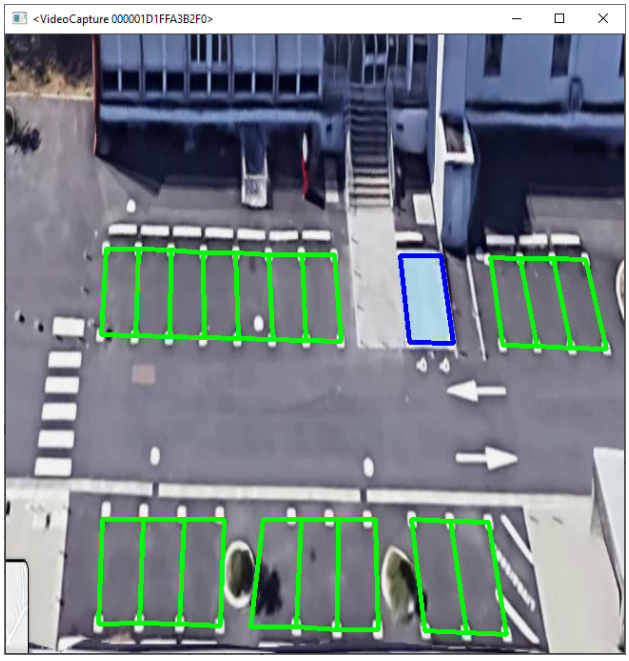

Figure 7: Visual output of free places detection

The algorithm gives, for each place, and in real time, the status of each place.

When cars are detected in a place, the place will turn red.

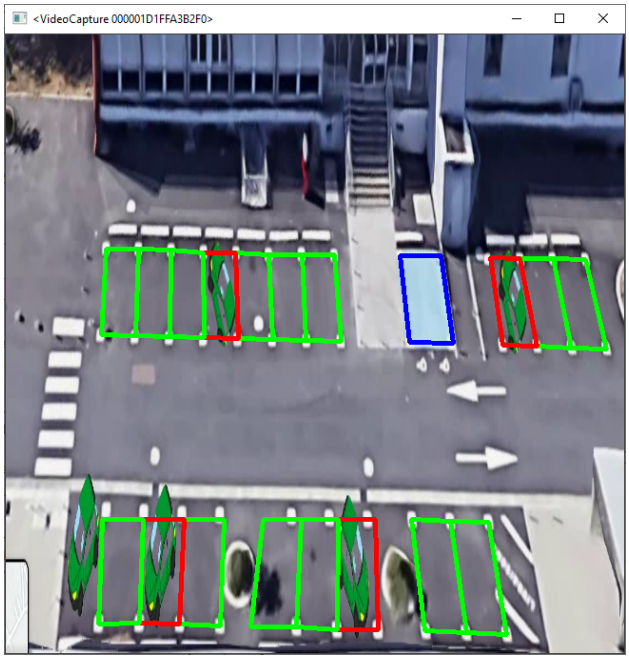

Figure 8: Visual output of free places detection with cars

For the output part we simply created a Json file and wrote these statuses into the file during the algorithm. The goal of this file is to be transmitted to the server to allow the mobile application to retrieve it and use it.


```
{
    "1": {
        "free": "False",
        "disabled": "False"
    },
    "2": {
        "free": "True",
        "disabled": "False"
    },
    "3": {
        "free": "True",
        "disabled": "False"
    },
    "4": {
        "free": "True",
        "disabled": "False"
    },
    "5": {
        "free": "True",
        "disabled": "False"
    },
    "6": {
        "free": "False",
        "disabled": "True"
    }
}
```


Figure 9: Json output

Once everything was working on a test environment (a computer), we still had to deploy the code on the NVIDIA Jetson Nano, which will be used with the camera to retrieve, process and

send the data, once placed on the top of the GEI building. We simply export the sources on the card and installed the needed libraries to make the code work.

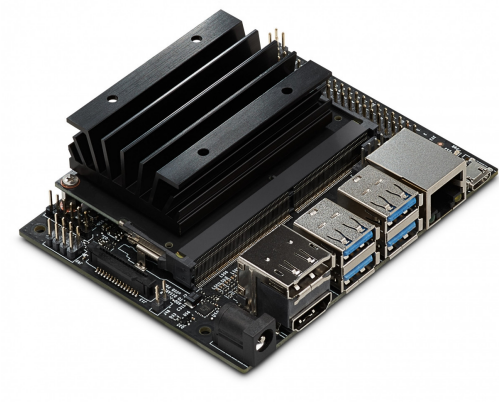

Figure 10: Jetson Nano Card

We then installed the camera on the card. Once the camera was plugged in, we were able to use it in our code thanks to different information provided by the Raspberry Pi test codes, so that we could replace our simulation video with the real video source of the camera.

Unfortunately, we didn't have time to use the Jetson camera in real conditions. Its implementation was complicated and the detection method was rather restrictive (the fact of having to require a 'blank' image imposes a kind of calibration period for the algorithm, with shots at precise moments, and then making the algorithm work in all circumstances).

However, we are quite confident that, once these other constraints are resolved, the application should be functional.

5 Application - Realisation

5.1 Functionalities

For the application we envisioned a complete, yet intuitive platform where we would present the live parking data to the user. Initially set on making a website to serve this function, as this would make it easy to show the information on a number of different platforms, we quickly steered towards making a mobile application since the processing of GPS data would be easier.

For the initial functionalities we conceptualized an app with a search button at the bottom and a built in map of the parking area as a background. On the leftmost side would be an icon that could be clicked to open a drawer with extra options for the search. The drawer would also be reachable by sliding a finger from the left part of the screen towards the middle, a so-called "swipe". When the search button was pressed, another drawer would pop up from the bottom, listing the available spaces as text while the upper part of the screen hosted the map with updated pins locating the different free parking spots. Clicking one of the options in the list would remove the drawer, showing the map in full screen once again with updated directions on how to get to the selected parking space.

5.2 Development

After having chosen to create an app we needed a system and work method that would ensure that the finished product would work on both Android and iOS platforms. For this purpose we used Flutter, an open-source UI software development kit created by Google. Seeing as no one in the group had much experience in programming mobile applications, this software simplified the app making process quite a bit. It sports a number of tools and pre-made functionalities with which we could make most of the basic functionalities of mobile applications. Lastly, the software design is based around widgets, which made it easy to add or remove certain functionalities of the app as the project evolved.

5.3 Integrating the free map service

For the map service we use MapBox, a mapping and location cloud platform for developers. With it we get access to the SDKs and APIs needed to integrate real time location features into our app. The service has different paid subscription plans based on, among other things, the number of users that are expected to access the service at the same time. As the free tier has a generous limit of 25 000 users it is more than sufficient for our project. The website of the company also has a number of resources to get started with the implementation in the form of explanatory documents and videos. It also has a map building tool where users can customize a map to their liking and easily create an API access token that is needed to utilize the service inside the app.

Figure 11: Excerpt of the application code detailing the map service

In the figure, above a part of the implementation of the flutter Mapbox service is depicted. The token is created for free on mapbox.com and gives access to the service. The parameters given to the LatLng class is the latitude and longitude of the INSA Toulouse parking space, marking where the map loads by default.

5.4 Communication between application and server

Once the user interface was done, we had to establish the communication between the application and the server. For that part, we added some lines and packages in the application code. First, we used the http package in order to allow a connection with the server. We have created new variables for the position which is given with coordinates. The first goal is to send data to the server with the user's position, including the coordinates and the chosen department, by clicking on the department chosen. This generates a request from the application to the server. This means that we want to receive the position of the best slot. The server makes a calculation in order to find the closest slot from the user and the department chosen. At the end, we receive the position of the slot. This position is given with the coordinates.

We wanted to test this new code. For that reason we created a local server with a Python script. We launched the test allowing the communication. By clicking on a department, we received the coordinates of a new position. We noticed that the coordinates were different when we chose another department, which is the intended outcome.

The simulation tests had to be done with the real server created for this project. The server uses a virtual machine. We will talk more about the server in the part 6. When we tried the first tests, we could not send packets to the server. This conducted us to change our first idea that was to take a function using UDP protocol -with a udp socket- instead of HTTP protocol. For this reason, we changed a bit the code and it finally worked as we expected. Also, between two tests sessions, we decided to change the type of the request and the response because we could not use geolocation. We were, at this stage, sending the department choice and receiving a list with all parking slots available around this department. This solution showed us that the communication between the server and the application was running.

5.5 Park'INSA - Application

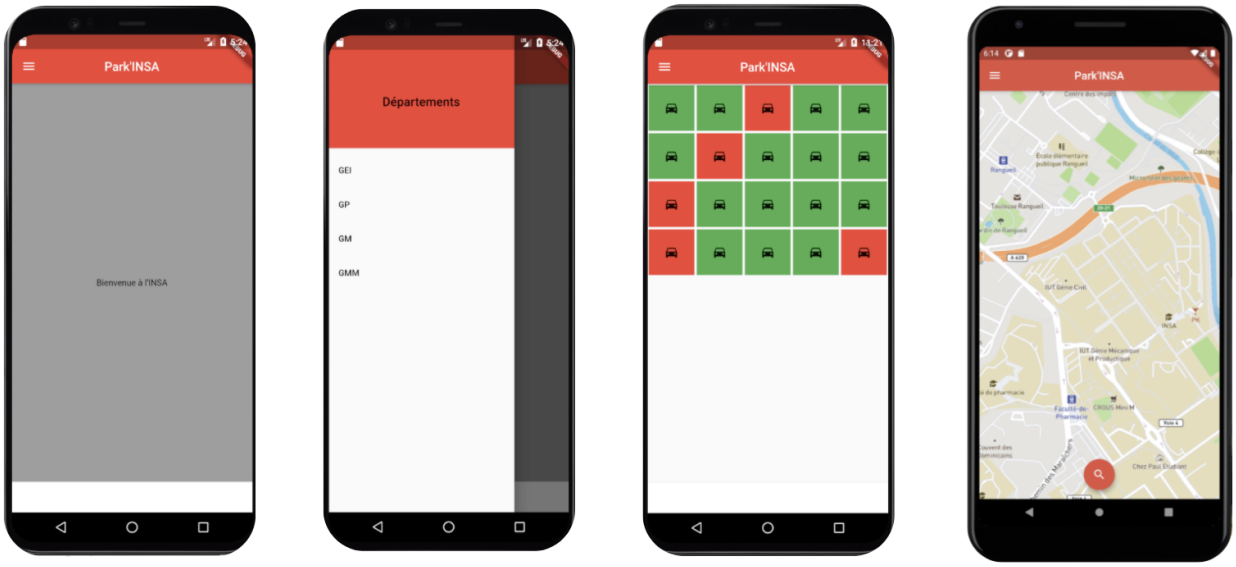

Figure 12: The functionalities of Park'INSA

The final interface of Park'INSA is shown in the figure above. As indicated, the front page is the welcoming page. The user can choose the department and then the number of available places is shown. If the top car to the left is green, that means that parking slot number 1 is free and so on. The map takes up most of the screen as we wanted to have a good overview over INSA's parking slots. To optimize the user experience, we choose to add the option of Departments in a side navigation menu.

6 Server - Realisation

Once the image processing has been completed, we need to make this information accessible. For this, we have a server hosted at INSA. To access it, one must be connected at INSA using the VPN. This solution is only temporary before getting a public IP address. The server receives the JSON files containing the information on available and unavailable parking spaces. When receiving the JSON file, it will overwrite the old one in order to keep only the most recent data. Following this, on a request from a user of the application, the server will respond by sending the JSON file on request. The requests are made in UDP. We chose UDP because we expect to send data to parking spaces every 3-5 seconds. Even if UDP QoS does not ensure 100% reception, and therefore network reliability, the very short delay between each transmission makes data always viable and makes it possible not to be limited by poor data reception.

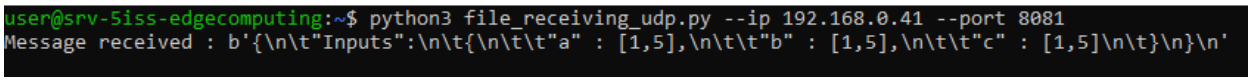

Figure 13: Reception of a JSON file on the server side

We see above the command to start the server: Port 8081 will serve as a listening port and wait for messages. If a message is received, the server overwrites the JSON file to replace it with the new one.

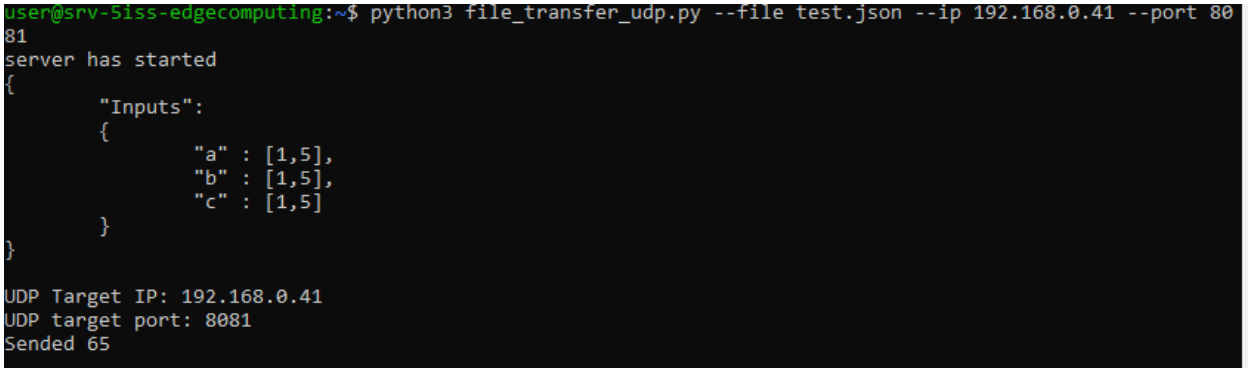

Figure 14: Sends a JSON file to the server

We see above the command to send a JSON file: Send is done by communicating the destination address, the reception port as well as the file to be sent. This script is used here to send a JSON file to the server therefore emulating the camera but will be used to respond to queries from the application requesting the available parking spaces.

7 Next step

7.1 Image Processing

Concerning the image processing part, several tasks remain to be done. We didn't have time to set up a real-time use of the camera, so we performed our tests with satellite images from INSA. Although our algorithms are supposed to work with the camera, this solution has not been tested in real time and it would be one of the next steps of the project. It would also be necessary to test the algorithms with the camera and to carry out a part of calibration of this one for the use of the method of background subtraction for the recognition of cars.

Another task that can be done is to improve the interpolation of places when they are not detected. For example, detecting different passes and 'half' squares (those without recognition squares on one side). This would improve the reliability of the application.

A final improvement that could be done is the recognition of disabled places. This was originally one of the goals of the project (evidenced by the 'disabled'-field of the JSON output), but unfortunately we did not have time to implement this functionality.

7.2 Application

As the communication between the app and our camera system is not yet finalized, many of the functionalities we first envisioned have not yet been coded. Our goals for further development will be to implement the ideas mentioned previously in this report.

7.3 Server

The first improvement that can be made is that currently the server is only available on the INSA network. This implies that the user must be connected to the INSA VPN in order to be able to communicate with the server. This can be solved by using a public address and not a private one so that the server is accessible from the outside via the Internet, but this is outside our jurisdiction. The second improvement would be to expand the server capabilities to manage several cameras at once. Currently the server has only one JSON file to send and does not select the data packets according to the zone desired by the user. This means that if a user wishes to have a place in front of a certain department, the server must select and send the JSON which corresponds to the place that best fits the users requirements. Another possibility is to merge the JSON files in order to have only one but this involves modifying only part of the file each time a message is received from a camera.

8 Integration of project

8.1 Jetson Nano to the server

Even though we weren't able to test the Jetson Nano in real-world conditions, the transmission method was still performed. This transmission method is based on the JSON file generated by the image processing algorithm to transmit it to the server, which will then make the link with the mobile application. This transmission is ensured by a python script that runs continuously, and that will, every 2 to 5 seconds, send a UDP frame to the server, a frame containing the contents of the JSON file.

We chose UDP protocol to ensure that transmission even though it potentially means that we will lose packets because the frame rate (2 to 5 seconds) is high enough for the parking detection application, which is a rather 'slow' application, so losing a few packets is potentially not a problem. This reduces the resources needed for the communication protocol and therefore the resources of the Jetson Nano.

8.2 Server to the Mobile Application

This part could not be tested because the server is only accessible when the phone is connected to the INSA network via the VPN. However, for the demonstration we used a server which is not at INSA in order to ensure the correct functioning of the different parts.

Conclusion

The goal of this project was to develop an application based on image processing and to allow users to easily and effectively find a parking slot on the INSA campus. The image recognition, the server, and the application were developed and tested separately. However, due to the different problems encountered on each technical part and the distance imposed by the general situation, the integration at the end of the project did not go as planned, and we were not able to create a functional integration of the cameras. Despite this we were still able to conduct a demonstration that is similar to the outlined project and that integrates all the parts. With some extra time the current project could easily be optimized and improved.

Nonetheless, this project has allowed us to develop a number of technical skills. These skills differ for each member of the group, as the tasks were divided between the members. Everyone was able to progress on their own technical environment, on specific technologies and concepts. Our overall ability to organize and manage a project has also been improved, using different management tools like risk management and Gantt charts. We have also been involved in the different phases of a software project: analysis, architecture design, development and tests. Once a week we met online to discuss the project advancement with our tutor. Having familiarized ourselves with these tools and practices is an invaluable resource that will be useful in our future professional projects.

At the beginning of the project we spent a considerable amount of time analyzing and understanding the tasks we had been given. Despite the slight setback this caused we have recognized that this was an important step that laid the groundwork for the further advancement of the project.

To conclude, during this period we have acquired skills on how to set up an innovative project from scratch, developing competences and expertise that we did not originally have. We were able to define a complete architecture and the methods required to carry out the project we had been entrusted with. Even if we did not reach the goals set in the beginning, mainly due to poorly managed risks and external constraints, we were still able to complete a 'proof of concept' for this project.