Università degli Studi di Padova
ICT for Industrial Application
Final Project
Academic Year: 2022-2023

Autonomous Driving Scenarios

Maddalena Boscaro - ID: 2090560
Giorgio Povegliano - ID: 2089066
Giada Zuccolo - ID: 2055702

1 Introduction

Autonomous Driving is one of the most popular and discussed technologies nowadays, as it's expected to revolutionize the way of transportation in the future. A lot of progress has been made in the last few years, especially thanks to the constant research activity and the huge advancements in artificial intelligence and sensor and communication technology. Currently there are 5 generally accepted levels of Autonomous Driving (developed by the Society of Automotive Engineers (SAE)), divided by the amount of human intervention required while operating the vehicle, they are presented below.

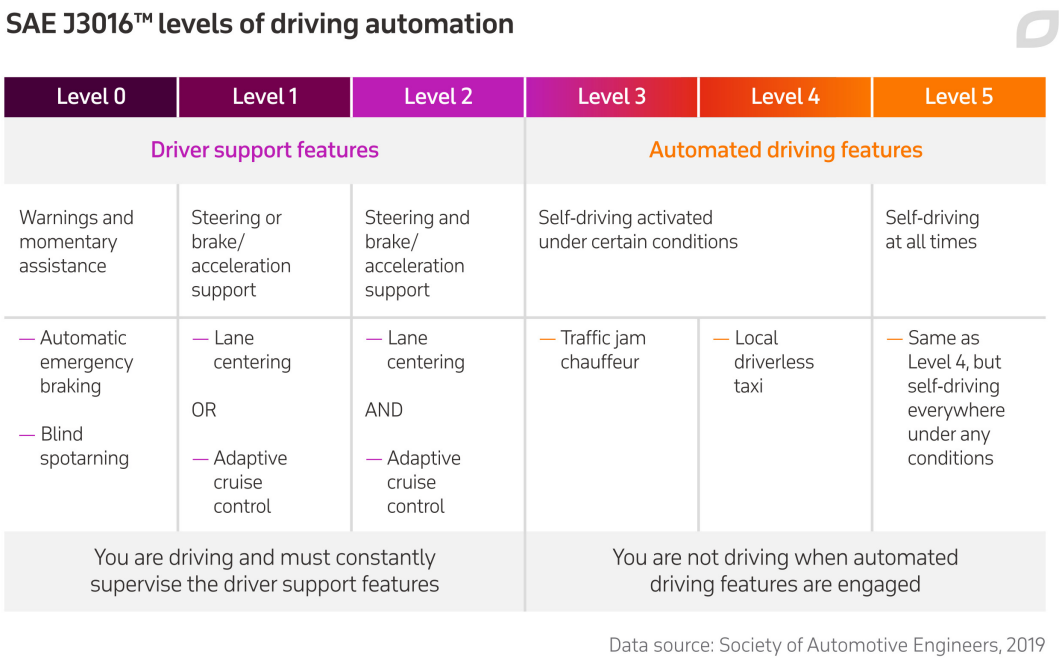

Figure 1: Five levels of Autonomous Driving [3]

The advantages brought by automated driving features (Level 3-4-5) are numerous, as are the challenges that must be overcome to implement this technology on public roads. One of the primary objectives of autonomous driving is to enhance safety on the streets. The majority of road accidents are the result of human error, including distracted driving, drunk driving, and

fatigue. By eliminating human error from the equation, we can significantly reduce the likelihood of accidents.
Autonomous systems have the potential to react faster and make more accurate decisions. However, achieving this requires fast detection and recognition of the objects in the vehicle's surroundings. Relying solely on self-acquired information, especially in critical situations where time is limited for the system to react, would be challenging, if not impossible. In such cases, intelligent vehicles may need to share their data with computing platforms and/or other vehicles to identify obstacles that are outside the range of their own sensors. Nevertheless, the transmission of large volumes of data resulting from this can be difficult to handle, particularly in areas with dense traffic, such as main intersections, even for modern communication technologies.
In this study, we aim to address the issue of exchanging data safely and efficiently to meet latency requirements. The objective is to develop an analytical method that filters and transmits only meaningful data, prioritizing transmissions that provide the highest value of information for the target application. Our approach is generalised, which means that we assume there is a central entity that receives and processes data. The aim of the controller is to minimize the number of transmissions from vehicles within the intersection and at the same time ensure the most complete view of the environment.
The following chapters of this study are organised as follows. Chapter 2 provides a brief summary of the existing literature on autonomous driving scenarios, addressing the significant challenges related to the enormous volume of data required for their functioning. Chapter 3 describes in detail the method used to simplify the view which the external controller has of a certain road intersection. Chapter 4 explains and compares three possible solutions to the problem, two of these use a linear programming algorithm. Chapter 5 presents the numerical results of the performance. Chapter 6 discusses possible future developments of this research.

2 State of the art

2.1 Different type of Sensors in Autonomous Driving

To gather information about their surroundings, autonomous vehicles heavily depend on their sensor system. As mentioned above, it plays a crucial role in detecting the presence of other vehicles, pedestrians, and other relevant objects. In recent years, the field of autonomous driving has experienced notable advancements, characterized by substantial progress in various technological aspects. Moreover, considerable effort has been dedicated to the development of rapid and precise sensors in order to address safety concerns and achieve accurate estimations. Autonomous vehicle navigation relies on a perception system comprising a combination of active and passive sensors, including cameras, RaDARs, and LiDARs [7].
The choice of sensor depends on the specific requirements of the autonomous driving system. Cameras are extensively utilized in numerous applications, including traffic sign recognition, lane detection, and obstacle detection [8], primarily due to their exceptional versatility and cost-effectiveness. Their ability to capture visual information, coupled with advancements in image processing algorithms, makes cameras a valuable choice for these tasks. However, these types of sensors heavily rely on visual cues, making them susceptible to challenging lighting conditions, occlusions, and variations in object appearances. In scenarios with low visibility or adverse weather conditions, cameras may struggle to provide accurate and reliable data.

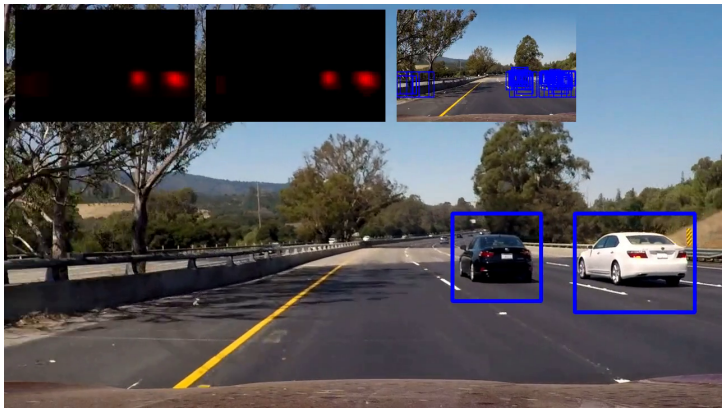

Figure 2: Sample image from the front and rear camera of an autonomous vehicle

In contrast, LiDARs and RaDARs are considered the most precise and effective sensors for representing the environment in autonomous driving applications, with LiDAR often regarded as superior in providing highly accurate and detailed measurements of the environment. LiDARs are active sensors that emit laser beams to illuminate the surroundings. By processing the received laser returns from reflecting surfaces, they can precisely measure ranges and create detailed 3D representations of the environment [9]. This allows for accurate mapping, obstacle detection, localization, and speed detection. RaDAR sensors, on the other hand, use radio waves to detect objects and estimate their distances, speeds, and angles. Radars are particularly advantageous in providing long-range detection and are less affected by adverse weather conditions compared to cameras. However they typically offer lower spatial resolution and less detailed measurements as opposed to LiDARs. The main disadvantage associated with these two type of sensors is their cost. LiDARs and RaDARs tend to be more expensive, which can limit their widespread adoption in certain applications. The cost of LiDAR systems, in particular, has been a significant factor hindering their widespread deployment in the automotive industry. In addition, the 3D pointcloud generated by LiDARs are quite difficult to compress with low latency, are sensitive to diffraction effects (especially in closed environments like tunnels), and offer a sparse representation of the surroundings, particularly when considering faraway objects [10].
Sensor fusion plays a crucial role in the field of autonomous driving. By combining data from above mentioned sensors, the system can leverage the complementary information they provide. For example, fusing camera and LiDAR data enhances object detection and tracking capabilities. LiDAR pointcloud provides accurate object positions and dimensions, while cameras offer more detailed visual information. By combining these two sources, the system can obtain a more comprehensive understanding of the environment. Similarly, RaDAR data can be fused to improve object detection in challenging conditions like low visibility or adverse weather.

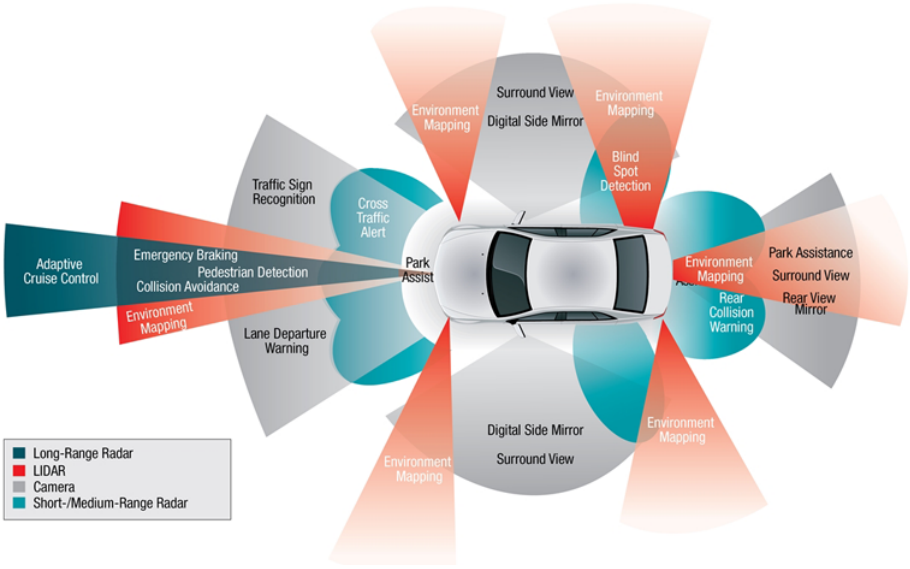

Figure 3: Example of sensors layout in an intelligent vehicle

2.2 Data Compression for Automotive Sensors

As explained before, the data acquired by LiDAR sensors are encoded as a 3D point cloud, which in the end generates high data rates that can be challenging to handle with standard Vehicle-to-Everything (V2X) technologies. An effective approach to addressing capacity issues is to exploit the millimeter-wave spectrum, as supported by the recent IEEE and 3GPP standardisation initiatives for future vehicular networks. Regardless, data rates could be decreased even more if the point clouds generated by LiDAR were compressed efficiently before transmitting the data.
The primary challenge is associated with the unordered and sparse structure of point clouds, which conventional storage methods struggle to handle efficiently. In this discussion we will investigate two main techniques: Voxel Grid (VG) and Octrees.

Voxel Grid: The Voxel Grid is a technique used in point cloud processing to organize and partition the data into a three-dimensional grid structure. It operates by dividing the point cloud

space into small volumetric units called voxels, which are essentially 3D cubes. For each cube a centroid is chosen as the representative of all the points that lie on it. VG structure helps in managing the sparsity of the point cloud data. It allows for more efficient memory utilization and enables faster processing by focusing computations on occupied voxels instead of individual points. However, the sparsity of the 3D LiDAR data resulted in many empty voxel cells, which can make the voxel grid representation inefficient [1].

Octrees: Octrees are a method of organizing space by dividing it into smaller octants, creating a hierarchical structure. The entire space is initially represented by a root node, which is then split into eight equal-sized octants. These octants, or nodes, represent smaller regions of space. The subdivision process continues recursively, generating additional octants, until the desired level of detail is reached. Octrees utilize voxels to organize and represent spatial information hierarchically. The main concept behind octrees is adaptive division of space based on the presence of data points. If an octant contains points, it is further subdivided into smaller levels. Conversely, if an octant is empty, it remains at a lower level to avoid unnecessary subdivisions and conserve memory.

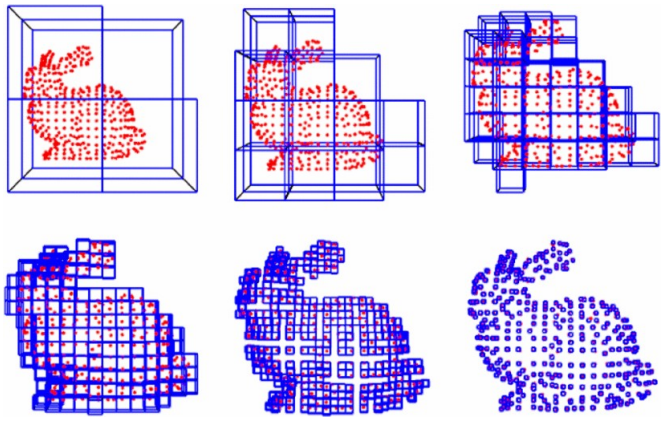

Figure 4: Six levels of a sample octree [3]

The absence of a widely accepted standard for point cloud compression has stimulated extensive research in this field. As a result, various compression algorithms have been developed, depending on the specific representations of point cloud data. Some examples are presented below.

- **3D-Compression methods** An example of 3D-compression is GPP (Geometry Point Picking). It is a 3D-compression technique that uses the octree representation of point cloud data. Each cube in the octree stores information about whether it is empty or occupied. Instead of encoding all the points within the occupied cubes, GPP selectively chooses representative points to be encoded and stored for compression.

  During decompression, the octree structure is reconstructed, and the representative points are distributed back to their corresponding cubes.

- **2D-Compression methods** A completely different solution is presented in [2]. The main idea is to apply two-dimensional (2D) transformations to the point cloud data using graph algorithms. Following this, image-oriented compression techniques can be employed, such as JPEG and PNG. Additionally, video-oriented compression techniques can be utilized to further improve compression efficiency.

- **Deep Compression Algorithms** The recent success of deep neural networks in image and video compression brings a new paradigm towards structured data compression for point clouds. Novel techniques based on deep learning have made it possible to further improve point cloud compressibility by extracting features from individual points and predicting voxel occurrence in 3D scenes.

  The approach employed in this research [4] utilizes the efficient octree structure to obtain an initial encoding of the raw point cloud (Figure 5).

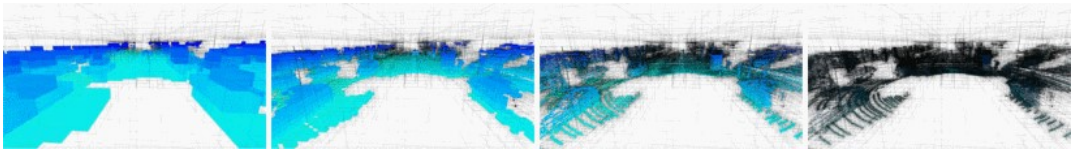

Figure 5: Four levels of depth (from left to right: 8, 10, 12, 14) in a octree structure representing a point cloud [4]

- **Hybrid Semantic Compression (HSC)** In paper [5] a different strategy is used. The compression of LiDAR point clouds is achieved by combining the compression power of Google's Draco software with the semantic segmentation capabilities provided by RangeNet++ (a Deep Neural Network (DNN) architecture designed for semantic segmentation of LiDAR points). The first step in this process is to take the raw LiDAR point as input and perform a series of operations to extract meaningful features. In this way, each point is classified into a semantic class, such as: roads, vehicles, pedestrian, buldings, etc (Figure 6). In scenarios where capacity is limited, by selectively choosing which data acquisitions to transmit, the system optimises the use of limited resources and ensures that critical LiDAR frames are given higher priority.

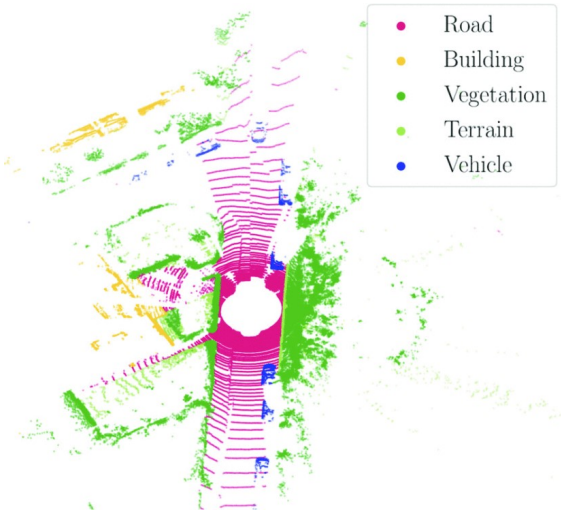

Figure 6: Example of point cloud semantic segmentation performed by RangeNet++ [5]

It may be possible, for example, to transmit only moving objects such as vehicles or pedestrians prioritising those elements that are crucial for ensuring safe and efficient autonomous driving operations.

3 Presentation of the problem

Linear programming is an important tool we will utilize to address the problem at hand. Our objective is to provide the external controller with a centralized view of the intersection using only data from the vehicle's sensor system, while minimizing the impact of transmissions on the network. Taking real-world scenarios into account, we will assume a predetermined maximum capacity for the network, measured by the number of concurrent transmissions that can be effectively handled by the system.
To achieve accurate perception, we will employ a sensor fusion approach that combines data from cameras and LiDAR sensors. This approach is known for its exceptional accuracy in detecting not only large objects like cars, but also vulnerable road users such as pedestrians and cyclists. In this context, the nominal rate required for a single vehicle to transmit is approximately 148 Mbps [10]. This rate serves as a benchmark for the network's capacity and will be considered in our optimization process.
Our objective is to maximize the perception of the intersection while ensuring the network load remains within its capacity limits and meets latency requirements. Another assumption is that the central controller has access to the GPS coordinates of the transmitting cars. The transmission

burden associated with transmitting GPS coordinates is considered negligible compared to the transmission of sensor data.
In a real application, we assume that the controller (on the basis of information that will be described below) decides which of the vehicles is authorised to transmit. All this has to be done in a very short time, which is why the efficiency of linear programming is exploited. Now that the controller has a comprehensive view of the intersection, it can process the data and prioritize the most significant information to be transmitted to the vehicles that require it.

3.1 Representation of intersections: occupancy matrix

Now, let's briefly discuss the challenge of representing the environment and the positions of various objects such as vehicles and pedestrians within the intersection. A common approach is to divide the intersection into different cells, each represented by an element of a so called occupancy matrix. The occupancy matrix, as a representation of the spatial occupancy status within the intersection, plays a crucial role in estimating the positions of objects. The size of the occupancy matrix is directly related to the level of precision in these estimations. With a fixed physical dimension of the intersection, a larger occupancy matrix, containing more elements, enables a more precise estimation of the position of each element within the intersection. We adopted a row-major approach for numbering the cells within the occupancy matrix. In this approach, the cells are numbered as follows:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 \\ 31 & 32 & 33 & 34 & 35 & 36 \end{bmatrix} \tag{1}$$

Matrix 1: Numbering of the occupancy matrix (6x6)

Each cell in the matrix described above can be assigned a value that represents the content of the corresponding location. In the simplest case, we consider the road layout and the presence of vehicles. In the following example, we assign these values:

- -1 to buildings or any structures that are not part of the road (highlighted in blue).
- 0 to road infrastructures (highlighted in green).
- 1 to cars (highlighted in red).

$$\begin{bmatrix} -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 & -1 \\ -1 & -1 & 0 & 0 & -1 & -1 \end{bmatrix} \tag{2}$$

Matrix 2: Occupancy matrix (6x6)

These assigned values enable us to distinguish between different elements within the intersection and facilitate subsequent analysis and decision-making processes. It is worth noting that more complex intersection layouts can be accommodated by including additional entities such as bicycles or pedestrians, and assigning appropriate values to represent them within the occupancy matrix. In this case, further analysis can be conducted as both pedestrians and bicycles are more vulnerable, and require priority in the transmission process.

3.2 Visibility matrix

Starting from the aforementioned occupancy matrix, we can now generate another matrix that will serve as a fundamental component for understanding the intersection and conducting all the

subsequent linear programming operations. This matrix, referred to as the visibility matrix (Matrix 5), will capture the line of sight of each cell represented in the occupancy matrix.
The visibility matrix is a square matrix with dimensions corresponding to the product of the rows and columns of the occupancy matrix. Each element of the visibility matrix represents whether one cell can or cannot visually perceive another cell within the intersection. Specifically, when analyzing row n of the visibility matrix, we position ourselves as an observer inside the nth cell of the occupancy matrix (following the same numbering as in Matrix 1). Each element in the row vector indicates whether the observer cell has a direct line of sight to the corresponding cell (1 for visible, 0 for not visible). For simplicity, we will assume that each car has visibility of every cell on the road where it is located, up to the point where there is a wall or obstruction. We will not consider diagonal visibility or blind spots created by corners in this analysis. To better understand what the field of view of a cell actually is, let us consider the following examples using the same configuration as Matrix 2.

$$\begin{bmatrix} -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 & -1 \\ -1 & -1 & 0 & 0 & -1 & -1 \end{bmatrix} \tag{3}$$

Matrix 3: Example of line of sight for the 9th cell of coordinates (2,3).

$$\begin{bmatrix} -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & -1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & -1 & -1 \\ -1 & -1 & 0 & 0 & -1 & -1 \end{bmatrix} \tag{4}$$

Matrix 4: Example of line of sight for the 22th cell of coordinates (4,4).

The information provided by the visibility matrix is crucial for making informed decisions regarding which vehicles should transmit their sensor data in order to maximize the view of the cells within the intersection. It plays a key role in formulating and solving the linear programming problems by ensuring that the decisions and optimizations consider the available visual information and account for potential blind spots or limited visibility between cells. By leveraging the visibility matrix, the proposed solutions can prioritize transmissions from vehicles that offer the most comprehensive and reliable visual coverage, leading to enhanced perception and decision-making capabilities for the central controller.
To facilitate comprehension, we show an example of the visibility matrix for the intersection depicted in Matrix 2. Due to space limitations, we will only illustrate the visibility for the first two rows of Matrix 2. The colors used in the visualization align with those used in the occupancy matrix, emphasizing the structural relationship between the two matrices and facilitating the interpretation of the visibility patterns within the intersection. As described above, each row contains the visibility for each single cell in the occupancy matrix.

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0 \\ 0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ \multicolumn{36}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \\ \multicolumn{36}{c}{\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots} \end{bmatrix} \tag{5}$$

Matrix 5: Visibility matrix (36x36)

Prior to applying the linear optimization algorithm to the visibility matrix, a critical step is to determine which cells are eligible for transmitting data. Since the available data for transmission is derived from the sensors of the vehicles, the criterion for selecting the transmitting cells is based on the presence of a vehicle within each cell. This step ensures that the subsequent optimization process is focused exclusively on cells containing vehicles, aligning with the data that can be provided to the central controller.
An effective approach to execute this operation involves setting every row of the visibility matrix that does not contain a vehicle to 0, indicating that it does not have visibility. In our specific case, we set all cells containing buildings (blue ones) and all cells in the road containing no vehicles (green ones) to zero. This can be achieved by generating a diagonal matrix with the same dimensions as the visibility matrix, where the diagonal elements (Matrix 6) are set to 0 except for the position corresponding to the cell containing a vehicle (red ones).

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6)$$

Matrix 6: Diagonal of the (36x36) diagonal matrix for car selection

Subsequently, the visibility matrix is left-multiplied by the generated diagonal matrix. The resulting matrix is illustrated in Matrix 7, which reflects the updated visibility information considering only the cells with vehicles. We can see that only line 9 (highlighted in red) remained non-zero while the other green rows were set to zero as the corresponding cells contained no vehicles.

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\multicolumn{36}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\
\multicolumn{36}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots}
\end{bmatrix} \quad (7)$$

Matrix 7: Actual visibility matrix (36x36) used in the linear programming algorithm

4 Linear Programming Approach

Having outlined the problem, our objective is to find an efficient solver to address the previously presented task. As mentioned earlier, the goal is to provide the central controller with the best visibility of the intersection. This means prioritizing the transmission of sensor data from vehicles that contribute to the highest number of non-zero elements when considering the sum of their rows in the visibility matrix (Matrix 7).
If the vehicles can collectively provide a complete visibility of the intersection, it means that every cell in the intersection is visible from at least one vehicle. In this case, the vector formed by the sum of their rows, that we will refer to as "controller vector", will have all of its elements greater than 0. We can say that each element of controller vector represents the cumulative visibility of a particular cell based on the contributions of multiple vehicles.
The constraint variable in this context is referred to as the "vector of transmissions". It consists of a set of elements, with each element corresponding to a cell in the occupancy matrix. The value of each element is determined by the decision of the controller: 1 indicates that the vehicle within that cell is transmitting, while 0 indicates that it is not transmitting. The vector of transmissions serves as a constraint in the linear programming problem, guiding the optimization process to determine the most effective allocation of transmission resources within the intersection. Given the predetermined maximum capacity of the network, it is essential to ensure that the total bandwidth consumed by the sum of the transmissions does not exceed this limit. Consequently, a constraint is imposed on the number of ones present in the vector of transmissions. This constraint guarantees

that the simultaneous transmission of data from vehicles within the intersection does not lead to congestion and allows for efficient data transfer. After examining the problem's structure, we can observe that it falls under the category of the well-known knapsack 0-1 problem.

4.1 Binary Knapsack Problem

The 0-1 knapsack problem is a maximization problem that has attracted significant research interest due to its broad applicability. It is named after the idea of packing a knapsack with a limited capacity, where each item has a specific weight and value. The goal is to determine the best selection of items to maximize the total value while ensuring that the combined weight does not exceed the knapsack's capacity.
In the 0-1 knapsack problem, each item is either included in the knapsack (assigned a value of 1) or excluded from it (assigned a value of 0). This constraint makes it a binary decision problem, where only whole items can be selected. In our problem, we can establish an analogy with the 0-1 knapsack by considering each vehicle as an item that can either transmit (value 1) or not transmit (value 0). The objective is to maximize the overall benefit or utility obtained from each transmission.
Similar to the knapsack problem, we have a constraint on the total number of vehicles that can transmit simultaneously, which limits the capacity of our "knapsack". In our specific case, although it may not reflect a real-world scenario, we simplify the problem by assuming that the weight of each transmission is uniform for all vehicles and equal to 1. This means that each transmission consumes an equal amount of resources or capacity within the network. In our case, the benefit of each transmission is determined by the amount of visibility it can provide to the central controller. However, it is important to note that the benefit of a transmission does not depend only on that individual transmission. Each transmission contributes to the overall visibility of the intersection, and the benefit is derived from the collective visibility provided by all the transmissions.
Due to the interdependence of visibility among the cells, a serial approach where we transmit cells with the best visibility one by one until we reach capacity limit is not suitable. This is because each cell can provide the best visibility in isolation, but there may be overlapping information with other transmissions. In this case there would be redundancy in the data received by the central controller. This would result in the central controller seeing the same portion of the intersection multiple times, while potentially missing out on other portions.

4.2 A Simple Example

Before going into the more technical part of the study, to facilitate the understanding of the problem and the variables involved, we provide a simple example of the proposed algorithm.

4.2.1 1° step: occupancy matrix

As usual, we number the cells of the matrix from left to right:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{8}$$

Matrix 8: Numbering of the occupancy matrix (3x3)

Let's assume that the intersection has a cross-shaped road layout, with buildings on either side. Within this intersection, we have two vehicles located in cell 5 and cell 8, respectively.

$$\begin{bmatrix} -1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 1 & -1 \end{bmatrix} \tag{9}$$

Matrix 9: Occupancy matrix (3x3)

4.2.2 2° step: visibility matrix

In our program, the visibility matrix is calculated immediately after the occupancy matrix to determine the line of sight between cells in the intersection. However, in a real-world scenario, the external controller does not need to recalculate the visibility matrix every time. The latter is based on the intrinsic characteristics of the intersection, which remain fixed unless there are changes to its physical structure.

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Matrix 10: Visibility matrix (9x9)

The blue lines corresponding to the buildings are always set to 0 in the visibility matrix because buildings do not contain vehicles and cannot provide any meaningful information to the controller.

4.2.3 3° step: compute the available visibility matrix

To determine the number of rows in the visibility matrix that are relevant for data transmission, the controller needs to identify the positions where vehicles are located. This can be done by creating a diagonal matrix denoted as D (Matrix 11) where the diagonal entries are set to 1 only in the positions corresponding to cells containing a vehicle. In the provided example, the diagonal elements at positions (5,5) and (8,8) (highlighted in red) have value 1 because cells 5 and 8 of the occupancy matrix (Matrix 8) contain vehicles.

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

Matrix 11: Diagonal matrix (9x9)

Then, the complete visibility matrix U is left-multiplied by the matrix D just computed (Matrix 11).

$$V = D \cdot U \tag{12}$$

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

Matrix 13: Actual visibility matrix (9x9)

The resulting matrix V (Matrix 14) will then have non-zero values only in the rows corresponding to the positions of vehicles that can transmit data (highlighted in red in the figure).

4.2.4 4° step: find the vector of transmissions

The ultimate goal is to determine and communicate to the vehicles which ones are authorized to transmit their sensor data. This can be achieved by defining a vector of transmissions. Recall that it has a length equal to the total number of cells in the intersection (in our case 9) and each element in the vector corresponds to a specific cell of the intersection. By setting the value of an element to 1, it indicates that the vehicle in that cell is authorized to transmit its sensor data. The other positions set to 0 can correspond to three different categories:

1. There is a building in that position that cannot transmit anything.
2. There is no vehicle in that position that can possibly transmit.
3. There is a vehicle in that position, but the central controller does not authorize the transmission of its data because considered wasteful or redundant for obtaining a complete view of the intersection.

$$vector\_tx = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{bmatrix} \tag{14}$$

Matrix 14: Vector of transmissions

The optimization problem in this context involves selecting the optimal vector of transmissions for the controller to maximize its visibility of the cells in the occupancy matrix. The visual field of the controller can be represented by the so called controller vector, which is calculated as the product of the vector of transmissions and the visibility matrix V.

$$controller\_vector = vector\_tx \cdot V \tag{15}$$

Let us consider two examples of vectors of transmissions and study their performances:

- **Brute-force vector_tx**

  In the brute force case, where all transmissions are authorized, the vector of transmissions will consist of only 1s, indicating that all vehicles are allowed to transmit. In this scenario, the resulting controller vector will contain the maximum possible visibility, as the controller can receive data from all vehicles.

$$controller\_vector = \begin{bmatrix} 0 & 2 & 0 & 1 & 2 & 1 & 0 & 2 & 0 \end{bmatrix} \tag{16}$$

  We can also denote that vector_tx represents the sum of the rows of matrix V.

- **Optimal vector_tx**

  In the optimal case, the goal is to maximize the number of non-zero elements in the controller vector, which represents the visibility of the controller. In a situation like the one in Matrix 8 where there is a bandwidth constraint and only one of the two vehicles can transmit, only two possible solutions are feasible: either send the data from vehicle in position 5 or send the data from vehicle in position 8. By examining the occupancy matrix (Matrix 9), it is clear that it is more advantageous for the vehicle in position 5 to transmit. In this scenario, the vector of transmissions will have a single 1 in position 5, and the resulting controller vector will be:

$$controller\_vector = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{17}$$

  By selecting the vehicle in position 5 to transmit, the controller achieves the same level of visibility as the brute-force case while respecting the capacity constraint.

4.3 Implementation

When it comes to finding a solver for the 0-1 knapsack problem, there are several different approaches and algorithms available. In this section, we will present three methods that can be used to solve the problem and compare their results. These methods vary in terms of performance and efficiency. By examining their outcomes, we can determine which method is more effective in finding the most efficient vector of transmissions in a limited bandwidth scenario.

4.3.1 Random Solver

The first approach, which is also the most naive, involves a random selection process to determine which vehicle is authorized to transmit. This method randomly selects one vehicle at a time until the maximum capacity is reached. While it is simple to implement, this approach does not consider any optimization criteria and relies solely on chance. As a result, we do not expect this approach to provide good performance or produce optimal solutions. Since the selection is based on random chance, the resulting vector of transmissions may not effectively maximize visibility or meet the desired objectives, in addition to the capacity constraint.
To implement this approach, we followed the following steps:

1. Generate a vector containing all the positions of all the cars located in the intersection.
2. Iteratively extract random indices that represent the vehicles that will be transmitting, and store their position.
3. Sum the rows of the visibility matrix corresponding to the position where the selected vehicles are placed.

The only contribution to the visibility of the controller in this approach will be the sensor data provided by the randomly selected vehicles.

4.3.2 Non Optimal Solver: Maximize Elements Sum

The second approach involves applying a classic formulation of the binary knapsack problem. The objective is to select transmissions that maximize the reward, which in our case corresponds to the number of cells that vehicles offer visibility to the controller. This approach considers the isolated contribution of each vehicle and seeks to optimize the selection of transmissions based on this criterion, while still considering the capacity constraint.
We implemented this approach by following these steps:

1. Define vector of transmissions as a binary decision variable.
2. Add constraint on the capacity: the total number of transmissions (sum of elements of the vector of transmissions) must not exceed the maximum permitted capacity.
3. Compute the controller vector by multiplying the row vector of transmissions with the visibility matrix.
4. Define the objective function as the maximization of the sum of the elements in the controller vector.

These set of steps aim to select the optimal vector of transmissions by maximizing the sum of all the elements in the controller vector. However, it's important to note that this approach does not consider the redundancy of transmissions. Maximizing the sum of the elements in the controller vector may result in multiple vehicles transmitting the same cell (this is the case when elements of the controller vector assume values greater than 1). This redundancy is non-optimal as it wastes bandwidth by providing the controller with data that it already possesses. To optimize the solution, it is important to consider the uniqueness of the information provided by each transmission, rather than simply maximizing the overall sum. This is the case we analyzed in the previous chapter, where transmitting the data of both vehicles (16) would have provided the same information to the controller as transmitting only one (17).

4.3.3 Optimal Solver: Maximize Non-Zero Elements

As specified in the previous chapter, the goal of the optimal solver is to maximize the number of non-zero elements in the controller vector. In this scenario, a transmission of a cell that is new to the controller needs to have a higher reward value compared to one that is already visible. This ensures that the selection of transmissions prioritizes providing new and valuable information to the controller, rather than redundant data. In practice, the reward assigned to visible cells can be standardized to a value of 1, regardless of how many times they are received by the controller. This approach ensures that each visible cell contributes equally to the overall reward, regardless of redundancy in the transmitted data. To implement this approach in linear programming we followed these steps:

1. Define vector of transmissions as a binary decision variable.
2. Add constraint on the capacity: the total number of transmissions (sum of elements of the vector of transmissions) must not exceed the maximum permitted capacity.
3. Compute the controller vector by multiplying the row vector of transmissions with the visibility matrix.
4. Define an auxiliary binary vector called "non-zero vector". This vector takes a value of 1 when the corresponding element in the controller vector is greater than 0, and it takes a value of 0 when the corresponding element in the controller vector is equal to 0.
5. The objective function is to maximize the sum of binary variables in the non-zero vector.

An important consideration needs to be made regarding the implementation of this method. Since we are using a linear programming approach, it is necessary to carefully define constraints for the non-zero vector to ensure that it assumes the correct values. In this case, we can establish the following constraints that must hold true for every index i:

- controller_vector[i] $\leq$ 10000 * non_zero_vector[i]
- non_zero_vector[i] $\leq$ controller_vector[i]

Since the non-zero vector contains binary elements, it is always true that each element at index i will be equal to $min\{1,$ controller_vector[i]$\}$.

5 Results

5.1 Random Scenarios

To assess the performance of the different solvers and compare their results, we utilized randomly generated configurations of intersections. These configurations were represented by the occupancy matrix, which allowed us to vary key parameters such as the dimensions of the intersection (number of rows and columns), the number of vehicles present, and the capacity constraint defined by the total number of allowed transmissions.
To ensure reliable results, we ran the test script for each set of parameters a total of 40 times. This iterative approach allowed us to reduce any potential bias that could arise from the randomness of the vehicle placement and the layout of the intersection. In each iteration, we generated a random configuration for the environment and placed vehicles randomly within it. This ensured that each iteration represented a unique scenario, introducing variability in the input data. For each generated configuration, we executed all three solvers, namely the random approach, the non-optimal solver, and the optimal solver presented in Chapter 4.
The use of random configurations and multiple runs helped us evaluate the performance of each solver across a wide range of scenarios, providing a comprehensive evaluation framework. By analyzing the averaged statistics, we were able to assess the effectiveness of each solver in finding the optimal vector of transmissions and determine which solver performed best under varying conditions.

5.2 Performance

The following tables depict the performance of the solvers under various operating conditions. To evaluate the efficiency of the solutions provided by the solvers, we use as a metric the percentage of visible cells transmitted, given by:

$$\text{efficiency} = \frac{\text{number of visible cells transmitted}}{\text{total number of visible cells}} \cdot 100 \tag{18}$$

This metric represents the proportion of cells in the intersection that are successfully transmitted to the controller, indicating how effectively the solvers optimize visibility. It is important to note that the capacity parameter has a direct impact on the efficiency. This is because it directly influences the balance between the number of transmissions and the total number of visible cells. Intuitively, when the capacity is decreased, the solvers have a more limited ability to transmit data from the vehicles, resulting in a lower percentage of visible cells being transmitted to the controller. This reduction in capacity inevitably leads to a decrease in the solvers' performance, as they are constrained by the available transmission resources.
The table below displays the average performance of the solvers in a specific scenario: a 15x15 intersection with a maximum of 10 simultaneous transmissions allowed. The goal is to examine how the solvers perform as the number of vehicles varies.

Table 1: Fixed size of intersection (15x15) and capacity (10)

N° of Vehicles	Random Solver	Non-Optimal Solver	Optimal Solver
15	88.02 %	90.76 %	98.08 %
25	81.11 %	78.59 %	94.12 %
35	73.59 %	74.39 %	91.39 %
45	70.92 %	69.18 %	90.50 %
55	68.01 %	66.70 %	89.10 %

As anticipated, the optimal solver demonstrates superior performance compared to the other two solvers. It achieves a higher percentage of visible cells transmitted, indicating its effectiveness in maximizing visibility within the given constraints.
Furthermore, as the number of vehicles increases, we observe a decline in the performance of all three solvers, that is more pronounced for the random and non-optimal ones. This can be attributed to the fact that with more vehicles, there is a broader view of the intersection in terms of visibility, but due to the capacity constraints, not all of this information can be transmitted.
In the case of the random solver, there is a higher likelihood of transmitting data from vehicles with limited visibility. This leads to a sub-optimal utilization of the available capacity and a decrease in overall performance.
On the other hand, the non-optimal solver may transmit data from vehicles that offer redundant information. This redundancy results in a less efficient use of the limited capacity and further contributes to the decline in performance as the number of vehicles increases.
Interestingly, as the number of cars increase, performance of the random solver becomes better than the non-optimal solver. This suggests that in the random configuration of vehicles, there is a clustering effect where vehicles with significant visibility are placed in close proximity, offering the same information to the controller. As a result, the random solver can capitalize on this clustering effect and achieve a slightly better performance compared to the non-optimal solver.
We now proceed to analyze the results for a fixed size of the intersection (15x15) and number of vehicles (35), while varying tha capacity.

Table 2: Fixed size of intersection (15x15) and n° of vehicles (35)

Capacity	Random Solver	Non-Optimal Solver	Optimal Solver
20	89.79%	90.79 %	99.76 %
15	85.03 %	83.74%	97.38 %
10	74.85 %	73.94 %	91.77 %
8	67.37 %	69.45 %	86.67 %
5	52.57 %	57.75 %	75.40 %

As anticipated, decreasing the capacity has a significant impact on the performance of the system, regardless of the approach used. When we reduce the limit on the number of transmissions from 20 to 5 the performance drops by approximately 25% in the optimal solver, while in the random and

non-optimal solvers the performance decline is around 35%. This indicates that even the capacity constraint has a more pronounced effect on the random and non-optimal solvers.
In scenarios with limited capacity, it becomes clear that the non-optimal solver outperforms the random one. This is primarily due to the decreased likelihood of randomly selecting vehicles that provide the most valuable information when the capacity is smaller. The random solver, being purely chance-based, has a higher probability of selecting cars that offer less relevant data or redundant information, leading to a decrease in performance.
Conversely, the non-optimal solver incorporates certain criteria in the selection process, even if they are incorrect. By considering these criteria, it increases the probability of selecting vehicles that provide valuable information to the controller. As a result, the non-optimal solver demonstrates better overall performance compared to the random solver in scenarios with capacity constraints.
We now take a look at the three solvers' performances for a fixed capacity (10) and number of vehicles (25) while varying the size of the intersection

Table 3: Fixed capacity (10) and n° of vehicles (25)

Size of Intersection	Random Solver	Non-Optimal Solver	Optimal Solver
8x8	89.04 %	84.69 %	99.91 %
10x10	84.87 %	81.66 %	97.91 %
20x20	75.24 %	77.36 %	91.63 %
30x30	69.89 %	78.39 %	88.22 %
40x40	69.12 %	77.79 %	87.28 %

In this case, we observe that the performance drop is relatively limited for all three solvers compared to the previous scenario. Additionally, we can highlight the remarkable robustness of the optimal solver to intersection size changes. As in the other cases, we can further investigate the difference in performance between the random solver and the non-optimal one.
As the intersection size increases, we find that the non-optimal solver tends to outperform the random one. This can be attributed to the fact that, with a fixed number of vehicles, a larger intersection tends to have lower population density. In a more sparsely populated intersection, redundant information is less prevalent as vehicles are less likely to be concentrated in the same areas. Since the primary drawback of the non-optimal solver is the repetition of transmitted data, it performs better relative to the random solver in this scenario.

6 Conclusions

In light of the various simulations and subsequent analyses, we have been able to verify the varying levels of effectiveness among the different transmission systems. As anticipated, the optimal method emerges as the most robust among the three, demonstrating its ability to maximize the visibility of the central controller even in adversary situations, such as constrained capacity. It is important to note that the solvers do not guarantee any specific lower limit to the total number of cells that the controller sees at the end of the process. This outcome is greatly influenced by the structure of the intersection and the positions of the cars.
For the purpose of the study and to facilitate the comparison of different approaches, we utilized a metric standardized over the total number of visible cells in each case. In order to ensure full safety on the roads, it is crucial to utilize complementary approaches in addition to the aforementioned transmission system. For example, it might be possible to ensure that crossings or high-risk areas are always visible to the controller.
While the optimal method proves to be highly effective in maximizing visibility and addressing constrained capacity situations, it is important to acknowledge that it might not provide full visibility of the intersection. To complement this approach and ensure comprehensive safety, the utilization of emerging technologies such as 5G and IoT (Internet of Things) can be explored.
By leveraging the capabilities of 5G networks, it becomes possible to establish fast and reliable communication between various devices and infrastructure components within the intersection. This enables real-time data exchange, allowing the central controller to gather information from not only other vehicles but also from traffic signals, crosswalks, and other infrastructure elements. Integrating IoT devices, such as sensors and cameras, with these traffic elements, valuable insights can be obtained regarding the presence and movements of pedestrians, bicyclists, and other vulnerable road users. Further studies can be conducted to expand upon the findings of this one.

In the current research, the focus was primarily on occlusions caused by buildings, and other objects such as vehicles were not taken into account. Additionally, the study's line of sight was limited to the road the vehicle was on, which may not accurately reflect real-world scenarios.

Source Code

The source code used for this report can be found in the following Git repository: https://github.com/zGiada/autonomus-driving-research-project
The Git repository contains the "classes" folder containing the "matrics" file, where the "Matrix" class and the "Occupation", and "Visibility" subclasses are defined. These classes are used to create various random scenarios representing different intersections, based on the number of rows and columns passed as parameters. The repository also includes a "linearProgramSolvers" file, which defines three different functions, one for each solver described above. Additionally, there are two scripts that can be executed: the "main" script returns the complete program with the occupation matrix, visibility matrix, and performance of the three different solvers. The "test.py" file, on the other hand, allows obtaining solver performances by iterating multiple times on multiple random scenarios.

References

[1] Y. Zhou and O. Tuzel, *Voxelnet: End-to-end learning for point cloud based 3D object detection*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.

[2] F. Nardo, D. Peressoni, P. Testolina, M. Giordani and A. Zanella, *Point Cloud Compression for Efficient Data Broadcasting: A Performance Comparison*, 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 2022, pp. 2732-2737, doi: 10.1109/WCNC51071.2022.9771764.

[3] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz and E. Steinbach, *Real-time compression of point cloud streams*, 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, pp. 778-785, doi: 10.1109/ICRA.2012.6224647.

[4] L. Huang, S. Wang, K. Wong, J. Liu and R. Urtasun, *OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 1310-1320, doi: 10.1109/CVPR42600.2020.00139.

[5] A. Varischio, F. Mandruzzato, M. Bullo, M. Giordani, P. Testolina and M. Zorzi, *Hybrid Point Cloud Semantic Compression for Automotive Sensors: A Performance Evaluation*, ICC 2021 - IEEE International Conference on Communications, Montreal, QC, Canada, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500523.

[6] B. Li and H. Holstein, *Using k-d trees for robust 3D point pattern matching*, Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings., Banff, AB, Canada, 2003, pp. 95-102, doi: 10.1109/IM.2003.1240237.

[7] J. L. Leonard, *A perception-driven autonomous urban vehicle*, J. Field Robot., vol. 25, no. 10, pp. 727-774, 2008.

[8] Cristiano Premebida, Gledson Melotti and Alireza Asvadi, *RGB-D Object Classification for Autonomous Driving Perception* in RGB-D Image Analysis and Processing, Cham:Springer, pp. 377-395, 2019.

[9] Y. Li and J. Ibanez-Guzman, *Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems* in IEEE Signal Processing Magazine, vol. 37, no. 4, pp. 50-61, July 2020, doi: 10.1109/MSP.2020.2973615.

[10] Valentina Rossi, Paolo Testolina, Marco Giordani, Michele Zorzi, *On the Role of Sensor Fusion for Object Detection in Future Vehicular Networks*