

Combining Clustering Algorithms to Extract Symmetric Clusters from Noisy Data, Applied to Parking Lots

Gyulai-Nagy Zoltán-Valentin ^a

Department of Computer Science, Babeş-Bolyai University,
1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

Keywords: Clustering, SOM Clustering, HDBSCAN, K-Means, Real-Time Application, Automation, Parking Lot Space Detection, Feature Extraction.

Abstract: The paper presents an approach for detecting symmetrical clusters in noisy data, using parking space detections as a real-world example. The paper proposes a plug-and-play solution that uses camera systems to automatically detect parking spaces and provide metrics about availability and accuracy. The approach uses clustering algorithms and image detection for data acquisition and mapping, which can be easily adapted to any application that requires geometrical data extraction. The paper also presents the different phases involved in mapping parking spaces and the challenges that need to be addressed. Overall, the proposed approach can benefit both parking lot administrators and drivers by providing real-time information on available parking spaces and reducing emissions, fuel costs, traffic, and time spent searching for a spot.


1 INTRODUCTION

Detecting where there are free spaces in a parking lot has always been a complex problem. In previous years, as cars play a great role in our day-to-day means of transportation, cities got agglomerated and therefore, it is harder and harder to find free spaces in parking lots. One might spend more time searching for an available space in big cities than traveling to the destination. We need to also take into account the pricing of leaving your car at a lot. When in high demand due to having a preferential placement in the central area, the price might be a bit higher when compared to other options. One might park farther from their destination to avoid paying higher fees. Lately, there have been a few initiatives to incentivize people to use public transportation for daily activities, but this does not seem to solve the problem of parking lot availability, as space is limited in crowded cities. (Cooke and Behrens, 2017) It would be helpful to just open an application and see in real-time where we might have the chance to find available spaces. This would also contribute to reduced emissions, therefore a greener environment, cost savings on fuel, reduced traffic, and time efficiency, due to not being required to search for a space.

This paper will focus on combining clustering algorithms to detect parking spaces based on positional data. Unlike other approaches, we provide a plug-and-play solution that does not require user-provided information at all, provides metrics and insights, relies on potentially existing equipment, can automatically adapt to any topology, and does not require complex setup or hardware.

2 LITERATURE REVIEW

In the past, the topic of detecting the occupancy of parking lots has been answered with different kinds of approaches. One of the most straightforward ones is manually marking the spaces and using simple image recognition on cars to predict occupancy. This has evolved into a race where applications try to detect the spaces automatically using the available data from images. This approach uses features that might delimit a space, like markers placed on the ground (Acharya et al., 2018). Other similar approaches try to optimize the images further to reduce the impact of lightning conditions or glare (Huang et al., 2022). Further research has been conducted to rely only on the position of the vehicles. Although a camera is still required, using positional data also has advantages when considering unmarked spots. This involves using cluster-

^a  <https://orcid.org/0000-0001-6506-4679>

ing and predefined information about the lot (Grbić and Koch, 2023). Another more popular approach, due to its applicability to a large variety of topologies, is to use sensors placed in the ground or above the cars to detect occupancy. This approach has the advantage of not requiring a camera. Therefore, information that might be private to drivers but embedded in the images, like car plates, doesn't need to be trimmed off, and potential security breaches are excluded. It requires more manual setup, but once done, it provides accurate data (Zhang et al., 2015). Modern vehicles are being equipped with multiple types of sensors and cameras. Some approaches try to use them to provide information about parking lots. This has the advantage of not requiring the installation of new sensors (Luo et al., 2017).

Previous research has explored the combination of clustering methods, revealing that leveraging results from multiple algorithms can indeed enhance accuracy in distinguishing members across multiple clusters. (Khedairia and Khadir, 2022) Another paper delves into the effectiveness of amalgamating K-means with Genetic Algorithms, leading to processes that are both more accurate and efficient. (Zeebaree et al., 2017) Additionally, research has demonstrated that combining weak clustering algorithms can further enhance the accuracy of the outcomes. (Topchy et al., 2003). In conclusion we can state that there can be benefits in terms of performance and accuracy when well known algorithms are combined.

Furthermore, a noteworthy aspect pertinent to our methodology is the intriguing research where it was proven that by inputting initial recognition data from image processing into a neural network with the intention of retraining it, we can attain improved classification performance (Doulamis et al., 2000). All these findings contribute to our methodology and provide the basis for our contribution.

3 PROBLEM STATEMENT AND RESEARCH QUESTIONS

Mapping of parking spaces is a complex operation and requires multiple phases to achieve adequate results without relying on any kind of user input. The adaptability to changes in the topology and providing metrics also means that we need to do continuous processing, and the results need to be as accurate as they can get between each mapping phase to provide value in real-world applications. In the following subsections, we are going to provide details on each phase and the problem that they need to solve.

3.1 Data Acquisition and Preprocessing

First of all, we need to acquire the data that will be used further in the next phases. In our application, we want to rely on camera images as lots of parking lots already have them placed due to security measures. This also means that images should cover most of the available area. In this paper, we will exclusively focus on scenarios where a single camera constructs the topology of a parking lot. This choice doesn't compromise the plug-and-play functionality of the system; rather, it maintains it for single-camera setups. However, future enhancements can extend this capability to derive topology from multiple camera angles, resulting in more precise data.

Most cameras are placed higher above ground to cover as much space as they can. The angle of view and lens distortion are factors that we need to take into consideration. We can work with non-uniform data as it is going to be presented in the next sections, but transforming the data is going to provide further benefits in distinguishing symmetrical features. Therefore the first research question is phrased as follows:

RQ1 What is the most appropriate transformation for the data to illustrate the symmetrical features?

3.2 Data Denoising

By using the detected cars in the image, we are going to have lots of noise. This is because even if we process the images from a feed at a given time interval, there is going to be movement of departing and arriving cars.

We also need to take into consideration that we might have such areas in a parking lot where illegal parking occurs. By illegal parking, we don't mean it is punishable by law to leave the car there; that's a factor that might be unreliable to be extracted purely from images, and it's not in our scope. We actually mean that space is not designated to be used as a parking space, and even so, people leave their cars there as it won't impact other drivers' access to other spaces.

This would provide valuable insight for drivers to determine by their arrival how likely it is to find their chosen space occupied. Given the previous information, we can conclude our interests:

RQ2. What algorithm can denoise and regularize the data?

3.3 Determining the Number of Parking Spaces

We need to take into consideration lots of variables in the detected data when we want to determine the num-

ber of available parking spaces. Depending on how people park their cars, in the more accessible spaces, we are going to have much denser data, whereas in areas that are like a fallback to if the main places are occupied, we are going to have much fewer data points. Therefore if a parking lot is frequently at its peak capacity, we have a greater uniformity in data, but we can't solely rely on that as we want our approach to work on less frequented lots too. If the detected data is not dense enough, we might want to consider them as noise, but in time as the samples increase, it is desired that they get to be detected as valid points.

A small memory-like approximation is also desired. As the topology of a parking space does not really change too often and the camera is also kept in the same place, remembering on a short term where there were the previous detections it's a helpful feature. We can also extract further data from these detections and we can make sure that if, for instance, the camera deteriorates and it's noticed in a short period of time, we still have stable estimations based on past data.

RQ3. What algorithm can delimit clusters via data frequency and can keep them in memory?

3.4 Parking Space Assignments and Real-Time Detection

As a final phase, we want to detect a lot's occupancy in real time. For this, we need an estimate for the size of each parking space and their center point. We also need to detect the center points of the clusters in such a way that they're placed in the most dense area. It is important to delimit the parking spaces by prioritizing the Euclidean distances between them to avoid putting more clusters than needed over areas that are highly dense in data.

RQ4. What algorithm is capable of accurately positioning centroids based on Euclidean distance?

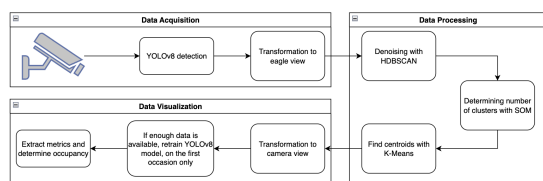


Figure 1: Steps of data processing.

4 METHODOLOGY

We will use a multi-step process to provide real-time data about the occupancy of parking spaces. We can differentiate two streams in this process: one is re-

sponsible for creating the parking lot model, and the second is providing real-time metrics and data. In the following subsections, we are going to describe them. An overview of the following steps can be found in Figure 1.

4.1 Data Acquisition and Preprocessing

For this section, we are going to use the PKLot dataset (de Almeida et al., 2015). It provides images of a parking lot over a longer period of time, and it covers multiple scenarios like daytime images, nighttime images, and different types of weather conditions. The angle of the camera is positioned in such a way that it covers the lot. It contains scenarios where the cars are not only in preferred positions, like stationed in a parking lot, but also when they are moving in and out of the parking spaces. It also contains images where people leave cars in unmarked spots. Basically, it contains most scenarios that are going to be covered when we use a camera stream and pick a few images at predefined time distances to be processed. As we want to cover real-world scenarios, this is a viable database.

To detect the cars in the images, we are going to use YOLOv8 (8th version of YOLO model). In the past few years, it has been one of the best solutions when talking about real-time detection of different types of objects (Terven and Córdova-Esparza, 2023). The accuracy of detections is good, the lighting conditions do not have such a significant impact on the results, and it has configurable models depending on what accuracy we want to achieve. It is a good candidate to be used on small devices like Raspberry Pi's that can be extended with camera modules. This is important because, as previously mentioned, data privacy is an essential factor when images are involved. Using a device that can do the processing on the spot reduces the risk of leaking additional, potentially private, information via internet connections. We assume that having the flexibility to choose between different types of models can bring further advantages in the cost optimization domain. Going forward we are going to use the YOLOv8n (8th version of YOLO model with 3.2 million parameters) model, but we are going to include in the tests other models too.

Once the cars are detected in the images, we are going to store the center points of the bounding boxes. These points need to be processed further to better indicate the symmetry between the parking spaces. Due to a lack of raw dept information, the eagle view seems to be the desired way of viewing the data, as the dimensionality of the points is reduced from 3D to 2D by trimming out the depth of field. We can achieve this by various methods, but via recent research, we

can easily do so with geometric approaches (Abbas and Zisserman, 2020). We can use the vertical vanishing point and ground plane vanishing lines to compute a homography matrix that will transform the data. These can be computed by using a Convolutional Neural Network (CNN) trained in this scope. The network is trained on a large amount of synthetic images with known camera configurations.

After the perspective transformation is applied, we are going to pass the data to the next phase.

In the next step, we are going to use HDBSCAN clustering. It extends DBSCAN into a hierarchical clustering algorithm that is able to find clusters with varying densities. The only parameter that we are going to use as input is the number of minimum cluster sizes. As we don't have an exact number for that, we are going to iterate over a few values to remove the noise. If our data is entirely evenly distributed in the parking lot, we could use 2882 as the starting value, and the resulting cluster number should be 24. This won't be the case as the spaces on the right side of the image are less frequented, and the YOLOv8n model will have fewer detections due to image distortion. We are going to use this value as a starting point, and we are going to decrease it until we get to at least 24 detected clusters. Once we have that value, we consider a simple rule to eliminate clusters that are too large and present irregularities. For each iteration going forward, we calculate the average width of the clusters. Everything that is larger than the average times two is going to be flagged. We chose this metric because we assume that the spaces are symmetric, meaning that if the width of a cluster is double compared to the average, it can be split into two additional clusters. We decrease the minimum cluster size by 100 and clusterize the data again. The 100 is a number that provides enough granularity in the iterations. A smaller number can be used, but it would result in more iterations. In case a larger cluster breaks into smaller ones, and the total number of flagged clusters also decreases, having noise points increasing, we remove the previously flagged clusters. We aim to find scenarios where clusters are not separated and symmetrical enough, and they are connected in some grade. This ensures we remove clusters that are too large and can't be clearly classified as parking spaces. To better understand the iterations, the following bullet points summarize the steps: (1) Determine the average max width of clusters. (2) Flag the clusters that are larger than the double of average width. (3) If there are fewer flagged clusters compared to the previous step and there is additional noise, remove the previously flagged clusters. (4) Decrease min cluster size with 100 (or any smaller number).

The process stops once we get to a minimum cluster size smaller or equal to the decrease value (100) or when the number of detected clusters on the original data is closest to the maximum number of detected cars in an image, which is 64 in this case.

The presented method defines constraints for upper and lower boundaries for the denoising algorithm, efficiently removing noise and clusters with ambiguous symmetry. As we can see in the images, the number of resulting clusters is approximately close. However, further processing needs to be done to define the exact number of parking spaces due to the wide variety of densities.

4.2 Determining the Number of Parking Spaces

This phase aims to determine the number of parking spaces that are available in the parking lot. For this purpose, we are going to use the self-organizing maps (SOM) (Miljković, 2017).

Due to other neural network properties, we will have the benefit of short-term memory. This is useful if the network is retrained with faulty data; previous information will leave its mark on the model, and we might be able to detect previously presented spaces that have been removed from the lot for a short period of time. This provides further metrics for the input data in determining the probability of detecting parking spaces.

Once we have trained the SOM model, we are going to extract the distance map. This enables us to visualize the distance of nodes compared to neighboring ones. We are going to map the values to white and black, white representing the most distant nodes and black the closest ones. To further extract information from the image, we are going to apply a few image processing algorithms to make everything more interpretable. We apply an adaptive histogram equalization to increase the contrast. This is also useful because where we have color information, they are going to be more accentuated, but where we have black shades, the new values are still going to be close to each other. (Pizer et al., 1987) This approach has a slight drawback; even though the information is easier to interpret, the noise can get overamplified. To solve this issue, we are going to apply a local mean filter with a low neighboring pixel check to the image. This ensures that noise is reduced, and we can easily extract the number of clusters shown in the image. To do so, we will use a simple watershed algorithm where the number of clusters will represent the number of parking spaces in our lot. The algorithm places the starting points for the fill process in the areas of

the image that have the least gradient change.

Now that we have the actual number of spaces and we have access to the denoised data, we can continue to the following phase.

4.3 Parking Space Assignments

At this step and also the final processing step, we have access to the denoised data and the number of available spaces. We need to define the cluster centers precisely and transform them back to the original information presented in the image to be visually inspected. For this purpose, we are going to use the KMeans algorithm. It is highly efficient in assigning each data point to the nearest cluster and minimizing distance variances in clusters based on the Euclidean distance. Even though it is a computationally demanding process, the partitioning converges quickly to the local minimum (Ahmed et al., 2020). Only one input parameter is required, that being the number of clusters, but we already know that value from previous steps. With this algorithm, we have the assurance that the centroids are placed in the most optimum position.

Once we have all the data, we store them for real-time detection and retraining in case it's needed. We store the transformed data (with nodes marked as noise, parking slot, and illegal parking), the median width of a cluster, the SOM model, including the distance map, the number of detected clusters, and the transformed centroids, including the real-world centroids.

4.4 Finetuning Our Model

Once we have all the clusters placed, it is essential that we train our YOLOv8 model to increase the effectiveness of our real-time processing. Our initial detections are going to have a below-average accuracy. To improve this, we are going to take the bounding boxes of the detected cars that overlap with the previous clusters, and we are going to retrain the YOLOv8 model based on them. We will reuse the already existing label of a car and assign it to the previous bounding boxes then feed it to the YOLOv8 training process.

This ensures that we improve the accuracy of our detections and that we customize the model to fit our parking lot, by retraining the YOLOv8 model once. Previous detections are going to be stored. This approach is suitable to increase the accuracy only if there is enough data to be processed, but in case there are not enough detections, the system will wait until enough data is collected.

4.5 Real-Time Detection

To detect whether a parking space is occupied and provide metrics about the lot, we will use a short processing of live feeds to provide real-time data. The process starts with selecting images from the feed at a predefined time distance. Depending on how accurately we want to access information, it can be defined as a shorter or longer period. We will propose time-frames of one minute as it will be accurate enough for scenarios where drivers are in search of spaces. On the selected images, we are going to apply the YOLOv8 classification with any kind of model depending on the topology of the parking lot. If there are only a few cars and they are all clearly visible, YOLOv8n would be enough. Otherwise, it makes more sense to use a more performant model. The bounding box centroids from the detected vehicles are then transformed into the eagle view. These are going to be stored in a backlog for further retraining and passed to the next step. Using the centroid positions of the parking spaces and the bounding box centers, we can easily conclude which one is the closest based on the Euclidean distance.

We are going to extract further metrics from the data to evaluate the accuracy of the previous classification. The first metric is the overlap with other parking spaces. In case we detect a car centroid in the range of multiple parking spaces, where double the average cluster width is still in range for all of them, we can conclude that each space can be wrongly occupied. This is an irregularity that might still allow another car to park near the wrongly parked car. Therefore, we are going to treat it as a flag for each space. The next metric is given by the removed clusters that were not uniform. If a car is detected multiple times in the same spot, we can say it is stationary. If we extend this constraint so that the centroid of the car overlaps with the area of a removed cluster, we can extract a flag that indicates a car parked in an illegal spot. The total number of such flags helps to detect the overcrowdedness of a parking lot.

The last metric is the confidence factor for all previous classifications and flags. By using the SOM distance map translated into the pixels of the image, we can generate a confidence factor based on the distance of the node that is being triggered inside the model.

The red boxes mark positions where the filtered parking spaces have been detected, and the dots represent by color the mapped distance of the SOM node. If it is red, it means it is a very distant node, whereas if it's blue, it means it's from a crowded area. We will use another variable, whether the node is a winner node or not. We say that a node is a winner if it is

Table 1: Accuracy of detected spaces.

Subset	Official spaces	Detected Spaces	Actual spaces	Accuracy
UFPR04	28	30	26	95.56%
UFPR05	45	45	43	92.86%
PUCPR	100	excluded	excluded	excluded

triggered for the training data.

For the parking spot classification values, we can state that in case a spot is occupied, and the triggered node is a winner node with a low distance value, we are confident that the detection is accurate; otherwise, it is not.

In case a parking space or more of them are wrongly occupied, and the triggered node is either not a winner node or the distance value of it is higher than 0.5, we are confident that the detection is accurate; otherwise, it is not.

In the last scenario where a car is marked as illegally parked, we can state that we are confident about the detection in case the activated node has a lower distance value than 0.5. Unfortunately, due to the fact that SOM requires higher and lower boundaries for the input data to be segmented, we can not conclude that we have no confidence in our detection. Therefore, we are only left with a positive confidence value if applicable.

Lastly, we are going to define the cases where the system detects a change in the topology and will reiterate the training steps to gain improved accuracy. If a car can not be classified in any of the previously mentioned categories and is detected to be stationary for a longer time, on multiple occasions, the system will grab the historical data from the last training process and retrain itself. Another case is going to trigger the same behavior, and that is when, on multiple occasions in the same spots, cars are detected to occupy more than one space by being wrongly parked.

Whenever such a scenario occurs, until the predefined amount of repetitions is not reached, the system is going to show a flag that states that lot maintenance or topology change has been detected. The retraining process ensures future proofing and automated maintenance for the system.

To achieve optimal results for the system, it is necessary to accumulate data over a period of at least one month. This ensures that an adequate number of samples are collected from both high and low traffic times and the YOLOv8 model can be retrained. Results with sub optimal precision can be observed within the first week from deployment.

5 RESULTS

Our results, by using the previously mentioned approach, can make use of a short interpretation before we dive into the testing values and comparison. As illustrated in the previous images, the system is able to automatically recognize parking spaces and detect the occupancy of spaces in real-time while also providing additional metrics. Interestingly, even though we detected 45 parking spaces, which is the exact number of spaces seen on the camera, we missed two spots but added two unmarked spots. The two spaces that have been left out are at the top of the image. The first one has been removed due to the constraints of the HDBSCAN algorithm. This means the distance and density of that cluster regarding the symmetry between clusters were not met. The SOM algorithm removes the second space. This is due to the fact that we do not have enough samples in that cluster to be classified as a completely different cluster. This can be because we used the YOLOv8n model, which adds further noise to the data by having less accurate bounding box placements and fewer detected cars on the images. But this also demonstrates that the process works as designed and has a high flexibility.

The two spaces that were added satisfy the constraints for each step. Now the question would be: would it be accurate to classify those two spaces as an error if they look and act as regular parking spaces and no regulation is applied to use them? These detections could also provide insights into further parking topologies that can be implemented to enable more usable spaces; therefore, for now, we are not going to treat it as an error but as an insight.

5.1 Testing

In the methodology section, we saw the capabilities of the approach and the benefits. To further compare the approach with other implementations and validate the benefits, we will first compare the best and worst YOLOv8 models available used in our process. From the parking spaces point of view, although YOLOv8n was able to detect exactly 45 spaces, two of them were unmarked spaces. This results in an accuracy of 95.56% for space detection in case we

count only the 43 actual spots. In our tests for detecting the number of parking spaces, there were no differences between YOLOv8n and YOLOv8x 8th version of YOLO model with 68.2 million parameters), although, in the denoising process, there were differences at which step the clusters got removed. This empowers the flexibility of the system by providing support and accuracy for heavier and lighter detection models. In 1 we can see the accuracy of detecting parking spaces. The detected spaces column gives us the total detected spaces, whereas the actual spaces column is calculated by subtracting the number of parking spaces that are found in addition to the actual number of spaces. For example, if we detect 45 spaces and two additional spaces are found that are not included in the official spaces, we are going to use only 43. The accuracy is going to be given by the percentage between the official spaces and actual spaces columns. The rows represent different subsets of the PKLot database, where UFPR04 and UFPR05 are different camera views of the same parking lot, and PUCPR is a different lot.

We need to mention that for UFPR04, the system actually detects 30, of which 4 are actual parking spaces, but trees obscure them; therefore, the official database did not include them in the visible parking spaces count. 2 spaces are not found at the edge of the image due to a lower amount of samples. In this subset there is a camera movement that results in the shift of the perspective. As a unique feature, our system automatically detects and adapts this change. For a short period of time, it will retrain itself, and after a few steps where it acquires samples, it will give the same results as before the perspective shift.

The PUCPR subset is excluded from the test due to the unreliable amount of detections provided by YOLOv8. The images can be further segmented, or YOLOv8 can be retrained (Carrasco et al., 2021) to provide more accurate detections, but this requires further development to be applicable for a general-purpose application, and we chose not to include this scenario as we want to focus in this paper on the clustering approaches and methodology.

To further provide insights we compare our approach with two other classifiers. For this, we are going to use the real-time detection feature of our application and compute the accuracy values by iterating each image as if it were a video stream. We consider it an accurate detection if a car is parked in a parking space detected by our system, and that space is also included in the official spaces. In Table 2 we can see the comparison with mAlexNet (Amato et al., 2017) and CarNet (Nurullayev and Lee, 2019).

Out of the box, the YOLOv8x model provided a

Table 2: Real-time detection accuracy.

Subset	mAlexNet	CarNet	Ours
UFPR04	99.54%	95.60%	84.97 %
UFPR05	99.49%	97.60%	86.08 %
PUCPR	99.90%	98.80%	excluded

72% accuracy when compared to the ground truth. The impact of the improved model is consistent, as seen in the previous table, and the test results were measured after training the YOLOv8n model only for two epochs with a precision coefficient above 0.7 for detections. This demonstrates the flexibility of this approach and its applicability to devices that do not have too much processing power. As seen in (Carrasco et al., 2021), the accuracy after retraining the YOLOv8 model can get up to 96.34%.

6 CONCLUSION AND FUTURE WORK

In the current research, we have developed a refined data processing framework specifically tailored to address the challenges associated with geospatial data acquired from camera systems. This new framework incorporates denoising algorithms that improve the quality of the collected data, can extract metrics, and features a short-term memory, facilitating the effective clustering of parking lot data based on transient usage patterns.

Furthermore, we have explored the capacity for performance enhancement in neural network models through a process of retraining. We observed a substantial increase in model accuracy and reliability post-retraining by initiating the process with a baseline neural network model conditioned on the initial dataset (comprising visual recognition of parking space occupancy). This approach emphasizes the practical benefits of our processing framework in optimizing neural network applications for categorizing and predicting parking lot usage.

We can conclude that the presented approach is a good fit for general-purpose applications. It is highly flexible in terms of platforms where it can be hosted, and it's also adaptable to the topology of the parking lot while providing further metrics. It can be applied to also other types of purposes like pedestrian traffic analysis or, with the change of the input stream even, seismic data analysis, as its key feature is to detect the symmetry between data clusters.

Despite appearing less effective than other approaches, the method proposed in this article prioritizes flexibility and user-friendliness. Its accuracy is expected to improve over time. One of its main ad-

vantages lies in its automatic adaptability and the additional metrics it offers for assessing various conditions within a parking lot. While further research may lead to enhanced accuracy, the current methodology can represent added value when applied to business scenarios.

Based on the accuracy of up to 95.56% of our approach, we can answer the research questions in the following ways:

- **RQ1:** the eagle view is the most appropriate transformation of the data to understand the symmetry between topological data better.
- **RQ2:** HDBSCAN combined with a descending min cluster sample value can denoise data and remove irregularities.
- **RQ3:** SOM clustering can better delimit clusters based on data frequency and keep track of previous clusters for a short time.
- **RQ4:** K-means algorithm can place the cluster centers accurately based on Euclidean distance.

One of the improvements for future work might be the approximation of delimiting spacing lines between the parking spaces. As long as all spaces are marked, placing a separator line between them should be possible. This would provide further possibilities for evaluating the existence of additional parking spaces that can exist at the edge of the image.

Another topic for future work would be to evaluate the accuracy of the clustering approach after the YOLOv8 model has been trained. It should be able to provide further insights into the topology of data points.

REFERENCES

- Abbas, A. and Zisserman, A. (2020). A Geometric Approach to Obtain a Bird's Eye View from an Image.
- Acharya, D., Yan, W., and Khoshelham, K. (2018). Real-time image-based parking occupancy detection using deep learning.
- Ahmed, M., Seraj, R., and Islam, S. M. S. (2020). The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*, 9(8).
- Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., and Vairo, C. (2017). Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72:327–334.
- Carrasco, D., Rashwan, H., García, M., and Puig, D. (2021). T-YOLO: Tiny vehicle detection based on YOLO and multi-scale convolutional neural networks. *IEEE Access*, PP:1–1.
- Cooke, S. and Behrens, R. (2017). Correlation or cause? the limitations of population density as an indicator for public transport viability in the context of a rapidly growing developing city. *Transportation Research Procedia*, 25:3003–3016. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.
- de Almeida, P. R., Oliveira, L. S., Britto, A. S., Silva, E. J., and Koerich, A. L. (2015). PKLot – A robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949.
- Doulamis, A., Doulamis, N., and Kollias, S. (2000). On-line retrainable neural networks: improving the performance of neural networks in image analysis problems. *IEEE Transactions on Neural Networks*, 11(1):137–155.
- Grbić, R. and Koch, B. (2023). Automatic vision-based parking slot detection and occupancy classification. *Expert Systems with Applications*, 225:120147.
- Huang, C., Yang, S., Luo, Y., Wang, Y., and Liu, Z. (2022). Visual Detection and Image Processing of Parking Space Based on Deep Learning. *Sensors*, 22(17).
- Khedairia, S. and Khadir, M. T. (2022). A multiple clustering combination approach based on iterative voting process. *Journal of King Saud University - Computer and Information Sciences*, 34(1):1370–1380.
- Luo, Q., Saigal, R., Hampshire, R., and Wu, X. (2017). A Statistical Method for Parking Spaces Occupancy Detection via Automotive Radars. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5.
- Miljković, D. (2017). Brief review of self-organizing maps. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1061–1066.
- Nurullayev, S. and Lee, S.-W. (2019). Generalized Parking Occupancy Analysis Based on Dilated Convolutional Neural Network. *Sensors*, 19(2).
- Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., and Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368.
- Terven, J. R. and Córdova-Esparza, D.-M. (2023). A Comprehensive Review of YOLO: From YOLOv1 and Beyond.
- Topchy, A., Jain, A., and Punch, W. (2003). Combining multiple weak clusterings. pages 331–338.
- Zeebaree, D. Q., Haron, H., Abdulazeez, A. M., and Zeebaree, S. R. M. (2017). Combination of k-means clustering with genetic algorithm : A review.
- Zhang, Z., Tao, M., and Yuan, H. (2015). A Parking Occupancy Detection Algorithm Based on AMR Sensor. *IEEE Sensors Journal*, 15(2):1261–1269.