Evaluation of parking space detection systems using wireless cameras

David H. S. Lima[1], **Eliana S. Almeida**[2], **Andre L. L. Aquino**[2]

[1]Federal Institute of Alagoas (IFAL)
Viçosa – AL – Brazil

[2]Computer Institute – Federal University of Alagoas
Maceió – AL – Brazil

david.lima@ifal.edu.br, {alla.lins, eliana.almeida}@gmail.com

Abstract. *This work presents a deep analysis of an embedded system to detect* ***free on-street parking slots*** *by using wireless cameras. To allow distributed processing and the communication, intelligent boards were embedded in the cameras. Three different system architectures are considered: centralized, hybrid and embedded. Each architecture was simulated considering the variation of the communication radius size, the amount of cameras and the amount of concurrent system queries. The results of simulation revealed that the performance of embedded proposal is better than the hybrid one in all scenarios. Additionaly, the embedded proposal was evaluated considering eventual cameras failure. It was observed that these failures influence directly the answer time of system.*

1. Introduction

The task of looking for free on-street parking slots in urban areas demands a significant time of drivers. This occurs, generally, due to a large number of vehicles and few available parking spaces (on-street or off-street). For driver's convenience, parking spaces detection systems may be responsible for notify to drivers the availability of free-space. Therefore, this search is performed manually, these parking systems would need a significant amount of human resources to work properly. Different cities around the world have installed sensing technologies to suggest free parking spaces. The use of these technologies is always motivated by detailed statistical studies about the available parking spaces or the number of vehicle in a specific area. For instance, cities as San Francisco[1] in USA and Santander [Sanchez 2010] in Spain, use sensor systems to suggest free parking space based a smart city model.

This work performs the analysis of an ad-hoc and embedded system to suggest **free on-street parking slots** by using intelligent board embedded to cameras. It is expected the use of existing cameras around the city as, for example, surveillance or traffic cameras. Our hypothesis considers the use of the intelligent board embedded to the cameras to suggest free parking spots for drivers. This board allows wireless communication and execution capability to the cameras. In this case, no additional equipment is required. The slots suggestion is performed using image processing techniques. Legal, privacy and security issues by using existing cameras is out of scope.

[1]http://sfpark.org/

Three different system architectures were evaluated to detect free on-street parking slots: centralized, hybrid and embedded. The *Centralized* uses a central server to be responsible for all image processing. The client-server and server-cameras communication are performed through a wired infrastructure. The *Hybrid* uses intelligent boards associated to the video camera to process the image. The client-server and server-cameras communication are performed through a wired infrastructure. The *Embedded* (Ad-hoc), uses only intelligent boards associated to the video camera, to process the image and to perform all the client-cameras communication through a wireless infrastructure.

General solutions use only off-street parking detection (shopping centers or parking spaces), the main contribution of this work is to allow the free parking space detection on-street. The *Embedded* architecture, proposed by Lima et al. [Lima et al. 2014], was adapted to be used in large scale ad-hoc scenarios considering: i. an ad-hoc and distributed system; ii. a routing algorithm suitable for this application in smart cities; and iii. the analysis of the system in a large scale scenario.

The response time of each architecture was evaluated including the variation of cameras available and concurrent system queries. As expected, the *Centralized* architecture outperforms the other ones. The *Embedded* architecture presented a reduced response time when compared with the hybrid one. In the *Embedded* case, the data traffic was reduced because the images were processed in the cameras in a distributed fashion. Consequently, the network traffic was reduced when simultaneous requests were considered. Furthermore, we evaluate the *Embedded* architecture in the presence of the camera failure. In this case we variate the communication range, cameras available and concurrent system queries. The results revealed that the *Embedded* architecture keeps an acceptable response time in the presence of different failures. This occurs because the routing strategy used is robust and ensures the data delivery.

The work is organized as following: Section 2 presents the related work. Section 3 describes the *Embedded* architecture. Section 4 shows the simulation and the corresponding results. Finally, Section 5 presents the remarks and future directions.

2. Related work

There are different proposals to perform parking space detection.

- Image processing approaches [Al-Kharusi and Al-Bahadly 2014, Reddy et al. 2013] apply techniques into images from cameras. After the processing phase, some methods are used for verify if there is any parking spot in the image. For this it is possible to use classifiers, image patterns or any other technique. In particular, the work of [Al-Kharusi and Al-Bahadly 2014] presents an intelligent system for parking space detection based on image processing technique. The proposed system performs the parking space detection identifying a green rounded image drawn at each parking spot. A common drawback in proposals based on image processing is that they fails when there are objects covering parts of the space or when the vehicle color is similar to the road color. Also, in general, they need some kind of marks on the road to distinguish the parking space.
- Wireless sensor networks approaches [Zhang et al. 2013, Geng and Cassandras 2012] work using sensor nodes to verify the availabil-

ity of the parking spots. In most solutions, it is necessary a sensor node for each parking spot. These sensor nodes are responsible to sense the parking spot and, if any change occurs, they have to inform their base station. These changes are, basically, a variation in the parking spot status (free or occupied). In particular, the work of [Zhang et al. 2013] proposes a street parking system to monitor the state of every parking space using a sensor node on each parking spot. They deployed the system with several sensor nodes and it has been running for more than one year. The results of the experiment show that the system detection accuracy is better than 98%, and it is energy efficient. A drawback of these proposal is that the infrastructure used is expensive and impracticable in on-street cases.

- In VANETs with attached sensors approaches [Thornton et al. 2014, Alhammad et al. 2012], some sensors are attached to the vehicles. These sensors identifies the available parking spot near the vehicle or they request some parking spot to a server. The idea is similar to the wireless sensors networks. The difference is that the communication and routing are performed among the vehicles. In particular, the work of [Alhammad et al. 2012] presents a VANET based on-street parking system that exploits the concept of InfoStations and context-aware systems to locate and reserve a parking space. All parking zones have an assigned InfoStations that provides wireless coverage to that parking zone. The system is divided in three tiers: the first tier is the vehicles; the second tier is the InfoStations; and the third tier is the InfoStation Center, which is responsible to monitor and control all InfoStations in the system. A drawback of these proposal is when the vehicles cannot communicate to perform the server request.

In addition, there are several commercial systems to detect free parking spaces (off-street) without human supervision, as for example, Fastprk (`www.fastprk.co`), ParkHelp (`www.parkhelp.com`) and StreetLine (`www.streetline.com`).

3. Parking space detection system

Three architectures were evaluated to detect free on-street parking using cameras. The *Centralized* architecture consists of the use of a client-server architecture where the client communicates directly with the server. When a request occurs, the server processes the images of parking area and sends the response to the client. The client-server communication is performed through the Internet. The images are always available to the server. The *Hybrid* architecture uses the server to manage the connections with clients and to forward the requests to the cameras. Intelligent boards are embedded in wireless cameras to process the image of parking area. The result of image processing returns to server to be delivered to the client. All client-server and server-camera communications are performed through the Internet. The *Embedded* architecture allows the client to send a request directly to the cameras, i.e., all cameras reached by the client will receive the request. The cameras propagate the request until reach the cameras in target area chosen by the clients. When the cameras in the target area receive the message, they will capture and process the parking area images. After that, the cameras will send the result to the client.

In all architectures, an Android client application is used in the vehicles (embedded directly or a smart phone App). In this application, the client request for free on-street parking in a target area chosen in a city map. The system execution is illustrated in Figure 1, with the following steps 1. The client selects the target area. The Android application creates a list with existing parking spaces in the selected area. 2. The Android application sends the list to manager module which identifies the cameras in parking lot. This manager module will be the server in *Centralized* or *Hybrid* proposals; or the intelligent board in *Embedded* proposal. 3. The manager module requests images from the cameras. 4. The cameras obtain the image and send it to the manager module. The manager module processes the selected images to detect free on-street parking. 5. After the image processing, the manager module sends the result to the client that will be able to visualize it.

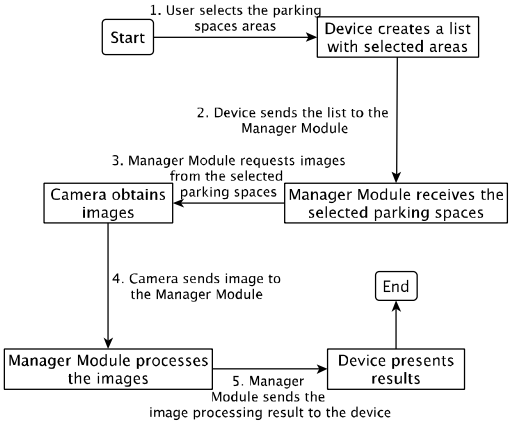


Figure 1. Client Android execution

We have multiple requests and only one free parking space is available, all clients will be informed about the free space. This scenario is not treated by our system. Additionally, we do not address the parking space reservation problem. Since we deal with on-street parking spaces, we can not guarantee the reservation of public spaces on street.

Our proposal implements a reactive approach. However, the *Centralized* proposal could perform a proactive approach, i.e., when the system detects some event in the parking area, the images are sent to server and processed. This approach was not considered here because the *Embedded* proposal will waste resources when nobody is looking for free parking spaces.

4. Evaluation and results

Two evaluations were considered: a single client request scenario by experimentation; and a multiple client and concurrent requests scenario by a large scale simulation.

The work of [Lima et al. 2014] evaluated three different image processing techniques to detect free parking spaces. The evaluation was performed based on a confusion matrix [Stehman 1997]. An image repository with about 3.300 images was created and applied to each technique. The best results was performed by the *Dilation after Border Detection* technique, reaching an accuracy of 93%. The value of accuracy is the proportion of true results (both true positives and true negatives) in the population. An accuracy of 100% means that the measured values are exactly the same as the given values. Based on this result, the *Dilation after Border Detection* technique was used in this work to perform the evaluations.

4.1. Single client request scenario

For a more realistic simulation, some individual experiments were performed to estimate the time processing of each technique used. Thus, each architecture was evaluated con-

sidering the average time spent to execute a complete client request (communication + processing). It was used an image processing approach based on border detection followed by thresholding and dilation. Border detection refers to the process of identifying and locating sharp discontinuities in an image. The border detection adopted here was proposed by Prewitt [Prewitt 1970]. Thresholding is one of the simplest methods of image segmentation. From a gray scale image, thresholding can be used to create binary images. The thresholding used here was proposed by Otsu et al. [Otsu 1979]. Dilation is one of the basic operators in the area of mathematical morphology. It is typically applied to binary images, but there are versions which work on gray scale images. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels. An advantage of this approach is that it does not require an empty on-street parking image. It runs over the image from left side to the right side. If it finds a free parking space, it will finish the processing [Lima et al. 2014].

For the experiments, seven cameras DCS-2130[2] from DLINK were used to compose an image repository. The intelligent boards embedded in the cameras are Panda Boards[3], with processor ARM dual-core 1.2 GHz, 1 GB of memory, and Wi-Fi capability. The operational system used was Ubuntu-arm 12.04. The server used in *Centralized* and *Hybrid* architecture was the Intel Core2Duo P8600 processor at 2.4 GHz, 4 GB of memory RAM and Ubuntu 12.04 64 bits operational system. All of the image processing techniques were implemented using the OpenCV library under C++ language.

The images were captured from two separated parking areas during 4 hours (8 a.m. to 12 a.m.). The number of images captured for each camera was, respectively, 267, 496, 1,294, 527, 314, 236, and 186 from the total of 2,557. The weather was sunny with a few clouds in the sky. The parking areas are rectangular, one area has about $100\,m^2$ and the other one has $500\,m^2$. Each camera monitors different spots from the parking area. All cameras were installed in the same height (4 m). The distance between each camera is about 3 m.

Table 1 presents the average time spent to execute a complete client request in each image captured (a total of 2,557 images). The *Centralized* architecture has the best response time. This occurs due to the advantage of server processing power when compared with the intelligent board. The *Embedded* architecture is faster than *Hybrid* one because it has one communication stage less. They spent basically the same time in the processing stage since both use the same intelligent board to perform this task.

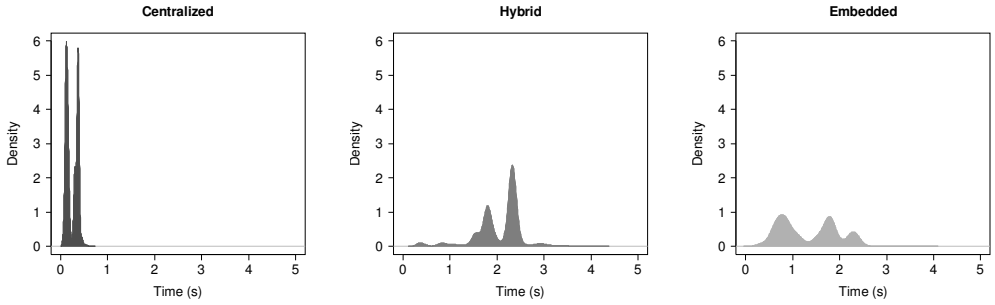
Table 1. Delay (s) in a request

Architecture	Average	CI	Std dev
Centralized	0.24	[0.23, 0.24]	0.12
Hybrid	2.03	[2.00, 2.06]	0.52
Embedded	1.35	[1.33, 1.38]	0.62

Figure 2 details the results presented in Table 1. This figure shows the density plot for each proposed architecture, i.e., the time to execute (x-axis) by the approximation of the number of images processed (y-axis). We note two separated modes in each architecture, this occurs because the images presents two distinguished characteristics: free parking spaces at (i) beginning or at (ii) ending in the parking lot. The algorithm process the image from left to right, at beginning means free parking spaces locate at left

[2] http://www.dlink.com.br/produtos-detalhes/items/DCS-2130.html
[3] http://pandaboard.org/

of image, consequently at ending corresponds free parking space locate at right of image. Assuming that these parking spaces are at beginning, the algorithm takes less time to perform the processing, because when a parking space is found the algorithm stop the processing. When the parking spaces are at ending of image the algorithm take more time because it processes the complete image.



(a) Centralized architecture (b) Hybrid architecture (c) Embedded architecture

Figure 2. Density plot for each proposed architecture

The *Centralized* proposal is the most homogeneous and faster than other. Its shapes have the same density (Figure 2(a)). The *Hybrid* proposal is the most affected by the communication process, since it has one more stage than the others (Figure 2(b)). In this case, the communication justifies the increase in the values of the second mode. Finally, the *Embedded* proposal (Figure 2(c)) is more homogeneous than *Hybrid*. However, it has more influence of communication when compared to the *Centralized* one.

4.2. Concurrent requests scenario

The development of a large scale **experiment** is a difficult task. As an alternative, a large scale **simulation** with different scenarios was developed in this work. The idea is to analyze the behavior of each architecture in a stress situation. Our simulation use the network simulator Sinalgo[4], in a computer Intel Xenon E5 32 cores with 2,6 Ghz processor, 128 GB memory `RAM`, and Fedora 20 64 bits operational system.

In the *Centralized* and *Hybrid* architectures all communication is directly through Internet. In the *Embedded* one, the ad-hoc behavior is simulated through wireless communication. In this case, the cameras forward the messages to targets areas. In the previous experiment we consider a direct communication, so the routing strategy was not used. In the concurrent scenarios, the routing algorithm, proposed by Maia et al. [Maia et al. 2013], was used. The motivation is the usage of complex network as a model for data propagation. This algorithm gives the advantage to decrease the amount of message sent in the network, due to inserted logic links among devices with greater processing capabilities. These links are used as shortcuts in message forwarding process and avoid network disconnection, i.e., if there is not a path from a camera to another, the request could be sent to a device that has a logical link with another more close to the target area.

Figure 3 presents the response time (y-axis) of each architecture (lines in the graphic), considering the variation of cameras available (x-axis with 100, 200 and 400)

[4]`http://www.disco.ethz.ch/projects/sinalgo/`

and concurrent system queries (each box with 100, 500 and 1000). The communication range is 400 m The average of request time (s) is presented with CI of 95%. As expected, the *Centralized* architecture presents the best performance in all scenarios. The *Embedded* architecture has a better performance than the *Hybrid* one. This occurs because the *Hybrid* architecture has one more communication stage. In the *Embedded* architecture, the response time increases proportionally to amount of cameras used. This occurs because there are more cameras participating of the routing phase. So, more messages must be generated in the network. The *Embedded* architecture outperforms the *Hybrid* one because it takes advantages of ad-hoc structure. In this case there is a high availability, i.e., if any camera is not working properly then other camera can be used. Additionally, different of *Hybrid* architecture, the *Embedded* architecture does not have to wait for the server response neither to queue multiple requests.

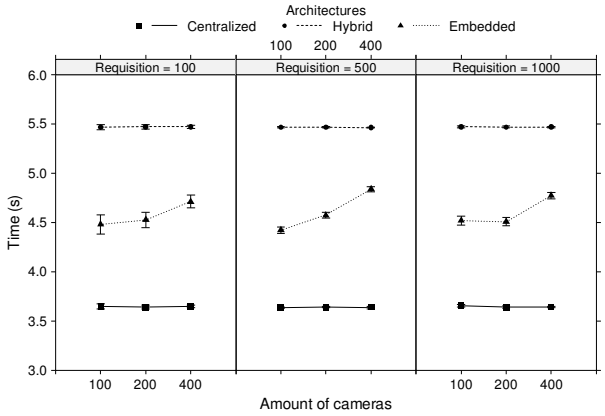


Figure 3. Simulation results for all architectures

These evaluations show that the amount of requests does not affect the result. When the amount of simultaneous requests increases the behaviour' architectures continuous the same. These results corroborate the results presented in Section 4.1. In smart cities there are several kinds of network connections with different speeds. For example, we could have a slow bit-rate connection where we assume that *centralized* approach would not be necessarily an easy winner.

Figures 4-6 evaluate the response time (y-axis) of *Embedded* architecture when some cameras fail (lines in the graphic). The average time of request (s) is presented with CI of 95%. In each evaluation we fix, respectively, the amount of cameras, the communication radius, and the amount of simultaneous requests. **Amount of cameras** fixed in 200 (Figure 4) was evaluated considering the variation of simultaneous requests (100, 500, and 1000 in x-axis) and the communication radius (300 m, 600 m, and 900 m in each box). If a communication radius of 300 is used, the percentage of failure does not affect the average of request times. When the communication radius is increased further delay in the presence of failures are observed. This occurs because some routing nodes fail and consequently increasing the message trajectory.

Communication radius fixed in 600 m (Figure 5) was evaluated considering the variation of simultaneous requests (100, 500, and 1000 in x-axis) and amount of cameras (100, 200, and 400 in each box). In this case the amount of neighbors do not increase.

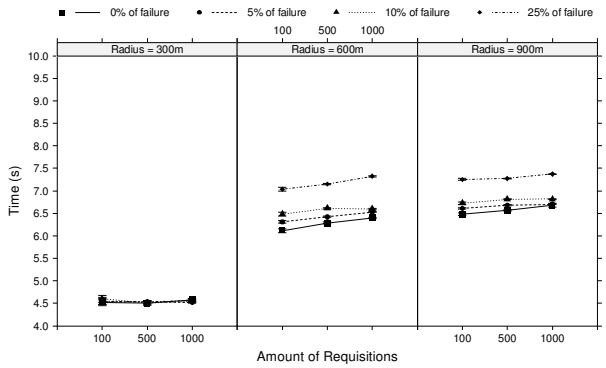


Figure 4. Simulation results with 200 cameras

So the delay variation is observed only in the presence of failure and is similar in all scenarios. This occurs because more routing nodes fail and consequently increases the message trajectory and delay.

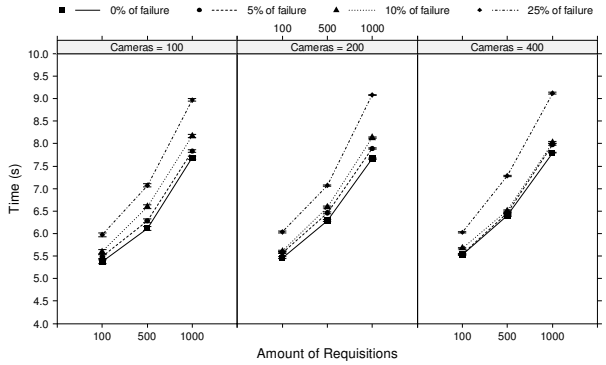


Figure 5. Simulation results with 600 m of radius

Amount of simultaneous requests fixed 500 (Figure 6) was evaluated considering the variation of amount of cameras (100, 200, and 400 in x-axis) and the communication radius (300 m, 600 m, and 900 m in each box). For the radius equals to 300, the variation between the amount of the cameras and the failure percentage is minimum but, if we increase the size of the communication radius the request time also increases. When the size of the communication radius increase, the amount of neighbors cameras and the amount of messages in the network also increase. Again, this occurs because more routing nodes fail increasing the message trajectory and delay.

A final evaluation (Figure 7) considers the response time (y-axis) of *Embedded* architecture when amount of cameras varies (lines in the graphic). The failure percentage is fixed in 5%, the amount of simultaneous requests varies in 100, 500, and 1000 (x-axis) and the communication radius varies in 300 m, 600 m, and 900 m (each box). The average of request time (s) is presented, with CI of 95%. When we increase the communication radius, the time spent in the requests will also increase. Again, this occurs because some

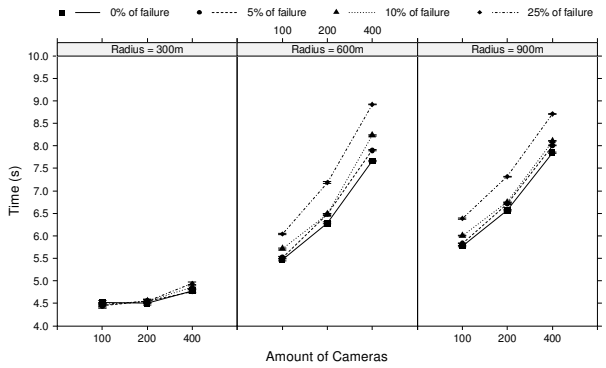


Figure 6. Simulation results with 500 simultaneous requests

routing nodes fail and, consequently, increasing the message trajectory.

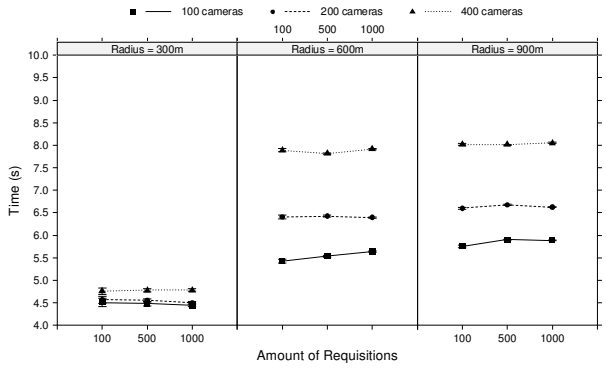


Figure 7. Simulation results with 5% of failure

5. Conclusion

This work presented an analysis of an embedded and ad-hoc system to suggest free on-street parking slots using cameras. Different architectures to perform on-street free parking space detection, *Centralized*, *Hybrid* and *Embedded* were evaluated. Two different scenarios were evaluated. The first one considered a single client request by experiments. This scenario reveal that the use of *Embedded* approach is possible. This can be corroborated due the times from the *Centralized* approach are similar to the times from the *Embedded* approach. Following, we performed an evaluation considering a concurrent request scenario in a large scale simulation. The results revealed that the most influencing factors in the variation of the request times are the size of the communication range and the amount of participating cameras. As future work we envisioning the perform of a large experiment in a real environment; and the development of a proactive free parking space solution.

Acknowledgements

This work has been partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq), and the Research Foundation of the State of Alagoas (FAPEAL).

References

Al-Kharusi, H. and Al-Bahadly, I. (2014). Intelligent parking management system based on image processing. *World Journal of Engineering and Technology*, 2:55–67.

Alhammad, A., Siewe, F., and Al-Bayatti, A. H. (2012). An infostation-based context-aware on-street parking system. In *International Conference on Computer Systems and Industrial Informatics*.

Geng, Y. and Cassandras, C. G. (2012). A new "smart parking" system infrastructure and implementation. *Procedia - Social and Behavioral Sciences*, 54:1278–1287.

Lima, D. H., Aquino, A. L., Ramos, H. S., Almeida, E. S., and Rodrigues, J. J. (2014). Oasys: An opportunistic and agile system to detect free on-street parking using intelligent boards embedded in surveillance cameras. *Journal of Network and Computer Applications*, 46(0):241 – 249.

Maia, G., Aquino, A. L. L., Guidoni, D. L., and Loureiro, A. A. F. (2013). A multicast reprogramming protocol for wireless sensor networks based on small world concepts. *Journal of Parallel and Distributed Computing*, 73(9):1277–1291.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66.

Prewitt, J. M. S. (1970). Object enhancement and extraction. *Picture Processing and Psychopictorics*, 75:75 – 149.

Reddy, P. S., Naveen Kumar, G. S., Reddy, B. R., H., S. C., and Abhilash, K. B. (2013). Intelligent parking space detection system based on image segmentation. *International Journal for Scientific Research and Development*, 1(6):1310–1312.

Sanchez, L. (2010). Smartsantander: Experimenting the future internet in the city of the future. In *21th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*.

Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89.

Thornton, D. A., Redmill, K., and Coifman, B. (2014). Automated parking surveys from a LIDAR equipped vehicle. *Transportation Research Part C: Emerging Technologies*, 39(0):23–35.

Zhang, Z., Li, X., Yuan, H., and Yu, F. (2013). A street parking system using wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2013:1–10.