Automatic context learning based on 360 imageries triangulation and 3D LiDAR validation

Daniel Amigo, David Sánchez Pedroche, Jesús García, José Manuel Molina
Group GIAA, University Carlos III of Madrid, Spain
Email: { damigo, davsanch, jgherrer } @inf.uc3m.es, molina@ia.uc3m.es

***Abstract*— Geographic data is very valuable for decision making. There are many hand-adapted datasets of roads or buildings available. However, datasets of other objects are not available, and it is very difficult to generate them manually. Remote sensing can help us to generate datasets of specific objects. This work introduces the main components for an automatic dataset generation process using any kind of sensors. To validate this process, an implementation using an open-source dataset is developed, geolocating traffic barriers using 360-degrees images captured from a car. Its results are validated with the positions extracted from a 3D LiDAR, solving the same problem at a much lower cost, providing an acceptable error for some use cases.**

Keywords — GIS data, triangulation, object geolocation, context learning, 360 imagery, 3D LiDAR

I. Introduction

Technology has reached the point where there are electronic devices everywhere on the planet capable of recording information from every part of the world massively. Satellites, surveillance cameras, sensors in autonomous vehicles, smartphones, ... are devices that allow the capture of data from different points of view, dynamics, and data forms. This information, besides its main use, can be reused and exploited for other applications. One of them is the capacity to detect and collect additional information from the environment they are capturing, generating georeferenced information useful for other processes.

The well-known Geographic Information Systems (GIS) allow the visualization of geolocalised data. These systems feature multiple algorithms implemented to provide complex analysis and decision-making. Geolocated information is very useful, but it is limited and hard to access. Road networks, buildings, terrain type, etc. are examples of accessible and hand-generated georeferenced information. In contrast, datasets of other objects are neither georeferenced nor accessible to the general public. Knowing the precise location of less relevant real-world objects is also necessary for several use cases. For example, a simplified representation of the world is used to run simulations, but the more information you have, the better solutions will be obtained. It can also be useful information for autonomous vehicles, better knowing and adapting to the context in which they move. It is even possible to reach a hive mind, generating and iteratively improving GIS detections from open-source projects such as OpenStreetMaps (OSM), processing the detections provided by different users.

These non-mapped and static objects, such as bins, trees, electricity poles or fountains, due to their minor relevance and large amount of them, make a manual mapping unfeasible. Artificial intelligence techniques can overcome this problem. Using algorithms that process real-world detections, detecting and estimating the position and orientation of objects, automatically generating these datasets. This approach allows the generation of datasets at little cost, requiring only a dataset of detections of an area and a computing system with such algorithms. For example, a video captured by a smartphone together with its GPS position and orientation records the state of a specific place and time. This also applies, in a different shape form, with detections from a LiDAR sensor of an autonomous vehicle. This information represents environmental objects of different characteristics: size, movement, or relevance. Moreover, all the detections, even from different sensors, can be fused, obtaining better results as more captures are taken.

This work aims at introducing the general components that an automated process must have to identify and georeference objects nearby using real-world detections, regardless of the sensor that captures it. It starts with the object detection on the sensor data, continues with the association of same object across all sensor measurements and ends with the tridimensional estimation of its position and orientation. The advantages and drawbacks that the fusion of different sensors detecting the same object can provide are also introduced, with respect to processes that only exploit a single sensor.

After the process is introduced, its feasibility is shown for real-world data detecting traffic barriers with an approach that exploits several sensors. Specifically, a dataset with 360-degree images is used, which also provides the detection and classification of the objects in each image. In addition, it has a ground truth of the position of the objects, extracted by an 3D LiDAR, which enables the validity of the estimations to be evaluated.

As several components of the process are already provided by the dataset, this paper focuses on the geolocalisation component through images. The implemented algorithm is based on triangulation, through cameras' orientation, their field of view (FOV) and the position of the bounding box representing the object detection. For each frame, a line is extracted estimating the orientation of the object, although it cannot geolocate it, as the distance to the object is unknown. Applied to all the frames, dozens of lines are generated.

These lines are the ones in charge of generating the estimation of the position, by means of their intersections. These lines are the ones in charge of generating the estimation of the position, by means of their intersections. This dataset has many detections per object, some closer and some further away from the object. Depending on the object detection frame, each line is more or less accurate. Two further analyses are performed to discard the lines intersections that insert more error into the estimation, thus obtaining an accurate estimation of the object's position and orientation.

Finally, the 55 barriers detected in the dataset are compared to the dataset barriers' ground truth to analyse the results of the process. This analysis is performed by using several metrics covering the error of the estimated three-dimensional object. The results demonstrate the feasibility of the problem, with less than two metres of error in position and 15 degrees of orientation. Even so, this research field has a lot of potential to be studied. Among other possibilities, the feasibility of the process with other sensors and even their fusion, or the improvement of the proposed triangulation-based system, could be explored.

This paper is organized as follows: Section II introduces similar works in the literature, which geolocate objects through different sensors detections. Then, Section III describes the proposed workflow, highlighting the main components and providing a few sensor-specific adjustments. In Section IV, the workflow with 360-degree imagery is presented. Its geolocation component is thoroughly detailed providing challenging scenarios and evaluating the accuracy of the estimations against the ground truth of the dataset. Lastly, conclusions and future works perspectives are presented in Section V.

II. State of the art

Several researchers have studied the ability to exploit sensing data to geolocate objects, generating or refining GIS datasets. All of them have made specific approaches to their solution and sensors. Generally speaking, there are several works that partially summarise this line of research.

Osco et al[1] study reviews remote sensing applied to UAV flights, in a very specific perspective. The main components, sensors, and applications that these flights use, highlighting the process of mapping objects, are analysed. Alternatively, Li et al[2] overview mapping techniques applied to automated driving, using different sensors and techniques. This case analyses another type of perspective, more focused on a Street view level.

Below, several specific approaches from the literature are explained, focusing on vision approaches.

A. *Camera-based*

Cameras are the most common sensor for remote sensing. They can either be applied through single images or video through frame-by-frame analysis. Position estimation with cameras is a well-explored problem in several approaches. Several of the papers found in the literature perform the approach at the street view level detecting some small and medium sized objects, as others apply it to larger objects from an aerial view.

Several researchers [3]–[6] use depth estimation to estimate the distance of different detections from the camera position. A stereo camera composed of two sensors is needed to calculate the depth, although it can be estimated with high accuracy with a single lens through AI as some works do.

Other researchers [5], [7]–[10] explore the triangulation concept. Using mono-lens cameras, they take advantage of the fact that multiple camera captures detect the same object at different positions and angles. Fusing all the detections, combined with the position and orientation of the camera, the location of a specific target can be estimated trigonometrically. For this approach, 360-degree image datasets mounted on vehicles are commonly used, such as Google Street View imagery.

Another way of obtaining images for geolocation is crowdsourcing [8], [11]. Volunteers take photographs of specific areas, which are later processed to generate a geolocalised dataset. There is an added complication of adjusting the algorithm to images from different devices. Furthermore, the GPS accuracy of each device cannot be heavily trusted. This means that the geolocation accuracy might be poor in certain detections. Even so, it potentially has a higher number of detections.

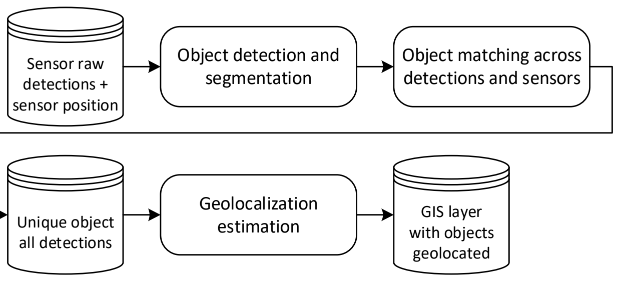
B. *Fusing vision with other sensors*

Each sensor can collect independent information about the object to be geolocated. Combining different sensors together generates greater data from the same scene, which can potentially enable a possible better detection. It is common to combine the RGB channels of a camera together with other sensors, such as LiDAR lasers, sonar or radar to detect distances of nearby objects in different directions with precision, or hyperspectral sensors that obtain the magnetic spectrum.

Kozonek et al. [12] and Chen et al.[13]] fuse LiDAR and video to detect objects and generate more accurate bounding-boxes, while Wang et al. [14] fuse video and radar to improve detections. Duarte et al[15] use UAV flight-collected data from imaging and multispectral sensors to geo-reference and analyse areas with specific vegetation parameters. Shah[16] maps an iceberg by sensing from a ship equipped with two sensors: a multibeam sonar to detect the geometry and a camera. Aubry-Kientz et al[17] fuse detections from airborne laser scanning, cameras and hyperspectral sensors to improve detection and segmentation of trees from aerial view.

III. General process

As already introduced, this work aims to establish a methodology with general components that any process of geolocation of individual objects should have. Figure 1 shows these components, for any type of sensor. Note that depending on the sensor each component is different, so this cannot be specified all in detail. However, the inputs and outputs of each component are described as well as explaining certain aspects according to the usual sensors.


Sensor raw detections + sensor position
Object detection and segmentation
Object matching across detections and sensors
Unique object all detections
Geolocalization estimation
GIS layer with objects geolocated
Figure 1 – Proposed process diagram

The first component represents the collection of all detections from the sensors used, generating the initial database. As discussed above, many sensors can be introduced, such as cameras generating images or video; point clouds from a LiDAR; direction, and distance from sonar or radar; etc. In addition, it is required to store exactly the time of measurement and the position and orientation of the sensor. It is important to remember that the object being geolocated is tridimensional, so covering the whole surface of the object ensures that its entire shape will be known in detail. If the detections do not cover it all, the object shape will be predicted in the unobserved areas.

Using each raw measurement as input, the object detection component generates a list with all the detections. Each detection is represented as a bounding box that wraps the small percentage of the entire measurement that belongs to the object. By using AI techniques such as neural networks, clustering or segmentation algorithms, it is possible to detect different objects with high accuracy. However, this step is complex and requires a lot of work and training to be able to detect all objects observed in the scene individually. Moreover, a good bounding-box fit is necessary, as the better the detection, the better the next components will perform. Once the objects have been detected, it is necessary to double-check that all detections are individual. Object detection algorithms sometimes creates a detection framing several objects if they are close to each other. Therefore, it might be necessary to apply a segmentation algorithm, which can separate the object detections in detail.

The next task, called matching or association, consists of relating those object detections of the same object in different measurements. Each measurement captures the same environment, albeit from different angles and distances, captures the same objects. This task should give each detection a unique ID that is repeated in each measurement where the same object is found. This procedure can be carried out in different ways, either online or offline. Such tasks are performed by tracking or multi-tracking algorithms, which are able to relate dynamics that are occurring over consecutive measurements. In video for example, it looks for the similar pixels in the bounding-box with slight displacements on the consecutive measurements, whereas in other sensors providing position or direction information, such position can be derived by the sensing motion and the sensor's positioning.

After this procedure, for each object, all its detections are identified per capture and per sensor. This list is used by the next process, along with the position of the sensor at each measurement to identify the three-dimensional position of the object to be geolocated. This step has variable complexity, depending on the sensor used. Sensors which measure distances facilitates the process, while image sensors as they do not generate this information needs to infer it. In both cases, these processes will estimate the position and rotation by fusing these measurements, obtaining the most accurate detection possible. This geolocation process is first carried out in local units, calculating the distance from the sensor position to the object. Later, it is necessary to apply spatial transformation equations to a global measurement, allowing it to be entered into any GIS system. It should be noted that the measurements of the detected object may not show the full geometry of the object. Therefore, if available, it is important to use the object context to correctly identify both the position and orientation of the object.

The proposed process has few general components but obtaining good results on this problem is not easy. The development of each component is complex, and there are accurate solutions for each one, but none sufficiently robust to be the problem completely solved. Therefore, such systems tend to rely on the fusion of well-calibrated and complementary sensors to facilitate and enhance each individual component. For example, LiDAR cannot easily detect an object form, but it knows the distance to the object. Alternatively, it is also possible to fuse the results afterwards, by performing the complete process for each sensor separately and joining them together to obtain a better estimate.

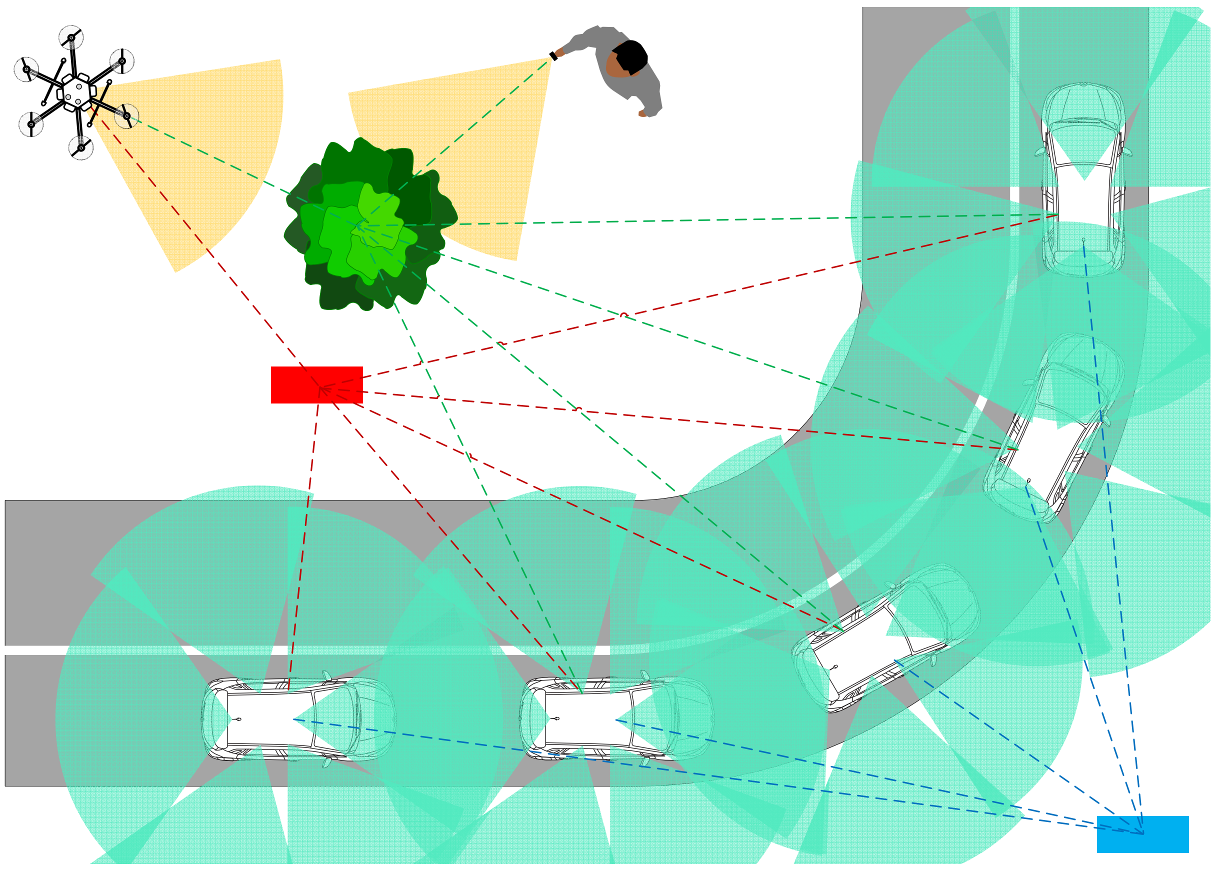


Figure 2 – Object geolocation example

IV. 360 IMAGERY BARRIER GEOLOCATION

A. General description

Once the general process has been introduced, the feasibility of this process is demonstrated by setting specific sensors, sensing type and target to be geolocated. This experiment aims to extract the position and orientation of temporary traffic barriers using a 360-degree video dataset at a street-level view.

The motive of this experiment is the study and feasibility of translating real-world objects into a three-dimensional representation, recreating reality as faithfully as possible. Such 3D environments will be used by the drone research branch of the research group, where they first develop in a simulator and then apply IA techniques on real UAVs. The more realistic the simulator and test scenario are, the better AI and results will be in the real world.

The dataset used is nuScenes[18]. Composed of detections from several sensors mounted on a vehicle, this is widely used by researchers for the planning and control of autonomous vehicles. Specifically, it has six cameras that form a 360+ image of the vehicle, five radars around the vehicle and a 3D LiDAR. In addition, it has GPS, IMU, etc. for accurate positioning and orientation of the car. In addition to raw detections, it has pre-processed information and useful algorithms to facilitate research tasks, such as object detections and object classifications across the sensors, or their precise three-dimensional positioning and orientation. An example of the dataset information is shown in Figure 3.

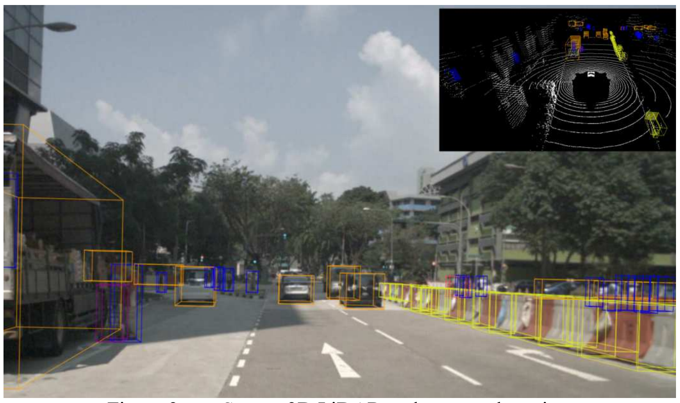


Figure 3 – nuScenes 3D LiDAR and camera detections

The dataset provides 23 different object types classified, some static such as the barrier to be detected, but most of them are dynamic, such as people or vehicles. In total, it has 1.4 million images, but for practicality as the aim is to validate that the process works, an available and reduced version of the dataset is used.

This experiment uses only 360 imageries to estimate the position and orientation of the object. The goal is to validate the feasibility of the project at a low cost, and to be able to replicate it in other environments. However, this dataset is still useful for this work, as it allows to compare the estimated position and orientation with their calculated ground truth using the 3D LiDAR. This enables the process and the system validation, so that metrics can be gathered to decide whether this process is valid under different scenarios.

B. Geolocation estimation

The implemented image-based geolocation component is exclusively based on the triangulation of different snapshots[19]. It is inspired by Zhang's work[10] with Line of Bearing (LOB) and has been extended to detect volumetric objects and their orientation rather than simple points on the map. The present work also establishes some criteria for finding the best possible geolocation, discarding noisy inputs.

The dataset used provides the precise georeferenced position and orientation of the car and its sensors at each snapshot, as well as the sensors intrinsic features. By having this information available, such as the field of view (FOV), it is possible to specify the area that each sensor covers in each sensing on a map.

In addition, the resolution of the camera is known. Specifically, it is 1600 pixels in width and 900 pixels in height. These 1600 pixels are within the camera's horizontal field of view, being endless long vertical planes gathering RGB data. These planes can be easily visualised in the well-known bird's eye view (BEV), as it is in Figure 4, drawing with a top view the car (the line represents the car's orientation), its sensors, the sensor's FOV with a triangle, and the lines representing the infinite plane.

On the other hand, we have the barrier detections, which are represented in each snapshot with a rectangular bounding-box that frames the object. These bounding boxes can be used to identify from these 1600 vertical planes the ones that correspond to the orientation detected by the barrier. In particular, the proposed algorithm selects three planes: the central plane of the bounding-box, used to accurately locate the centre of the object, and the two sides of the bounding-box, which are used to find the orientation of the object. In Figure 4 these three planes are drawn: the left ones in blue, centre in green and right in red.

Specifically, for each detection and zone of the object to be positioned, the process performs:

1. The vertical plane on the image is obtained for the area of the object to be positioned, called $Object_{Pixel}$.
2. With it, the difference in angle with respect to the image centre is obtained, by knowing the FOV of the camera.
3. This direction value is transformed to the relative coordinates, by knowing the position and orientation of both the car and the sensor, and therefore the object's direction can be obtained.

$$ObjectDiff_{Yaw}^{Sensor} = \frac{FOV}{2} - \frac{FOV}{Image_{Width}} \cdot Object_{Pixel}$$

$$Object_{Yaw} = Car_{Yaw} + Sensor_{Yaw} + ObjectDiff_{Yaw}^{Sensor}$$

The planes delimit the area where the object is located, but they are not able to estimate the distance of the object from the camera, so they cannot locate it with a single image. By exploiting all the snapshots that detect the same object, it is possible to estimate the distance to the object. Applying the same process with each detection, N lines representing the same object are generated. To calculate the intersection (x, y) of two lines i and j described as $ax + by + c = 0$ the following equation is used:

$$(x,y) = \left(\frac{b_i \cdot c_j - b_j \cdot c_i}{a_i \cdot b_j - a_j \cdot b_i}, \frac{c_i \cdot a_j - c_j \cdot a_i}{a_i \cdot b_j - a_j \cdot b_i}\right)$$

By generating the intersections of these lines, $^{N\cdot(N-1)}/_{2}$ position estimations of the object are obtained. The same process applies to the lines on each side of the bounding-box, finding intersections that represent each side. Using these intersections, centroids can be calculated and taken as a good estimate of each of the zones: centre and both sides. The estimate of the object is placed centred on the central centroid, while the orientation of the barrier can be calculated trigonometrically using the line that joins the two sides.

As each barrier has fixed dimensions, it can be drawn correctly in the BEV view. The true barrier can also be drawn to compare the result of the process and determine how good the process is. As shown in Figure 4 example, they are identical in position and slightly different in orientation.

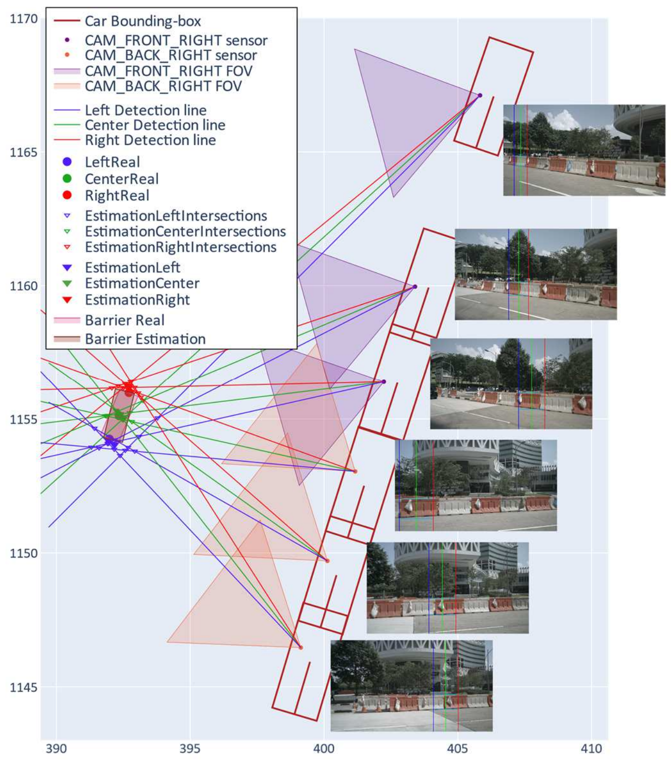


Figure 4 – Triangulation illustration process

This example's estimation is almost perfect, with multiple detections really close and covering it well. Nevertheless, A further analysis is necessary to assess whether this process is accurate enough in all scenarios. Therefore, the following dataset information and generated metrics are extracted from all detections of the mini dataset introduced above.

For each object, by comparing the estimated object with the real object detected by the 3D LiDAR:

- The yaw angle error in degrees.
- The overlap percentage in BEV view.
- The error of position of each side and centre.

For each snapshot and bounding box of a barrier:

- The height, width, and area of the bounding box in pixels.
- The distance from the sensor to the object.
- For each line of this frame (centre and sides) the positional error between the centroid of their intersections and the real position.

The used dataset contains 10 different scenes, of which 3 detect barriers. In total, there are 71 barriers. After applying the geolocation process on them, it has been observed that one scene is only the car stopped, detecting the barriers several times at the same spot. The triangulation algorithm requires several snapshots at different locations. For this reason, a restriction is added to the triangulation frame input: if the car has not moved more than 1 meter from an already selected frame, that frame is dropped. This process takes all 16 barriers out from the that scenario, thus having a total of 55 unique barriers. In addition, to carry out correctly the triangulation process, a minimum of 3 frames are required.

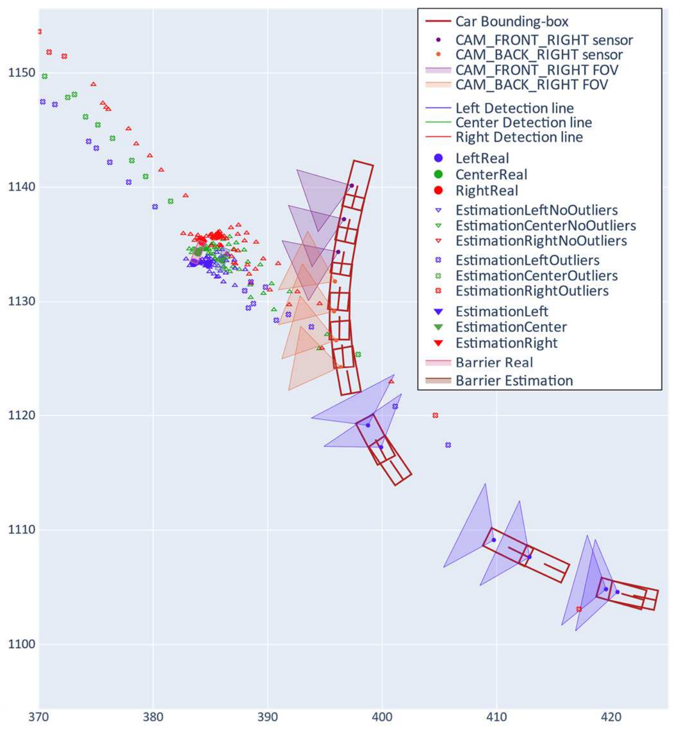


Figure 5 – Triangulation example with erroneous intersections

An analysis of the results obtained is carried out. Only 19 of the 55 overlaps with the real barrier. On average they have 2.9 metres of error and 27 degrees in orientation with respect to the real position. These results show that the process is satisfactory, but there are some issues that need to be addressed, as in Figure 5. Some intersections have a clear centroid, but many others are spread out, some of them into infinity and some even on the other side of the car. Both cases are detrimental to the centroid obtained and therefore the barrier estimation.

C. *Misplaced intersections cleaning*

All these intersections theoretically represent the same place, but they are not located in the same position. This gap is caused by the imprecision of the cameras, as well as by the bounding-box fitting at each frame. The further away the sensor is from the object, the fewer pixels it represents, and the more uncertainty is introduced to the system by its line. In addition, if the object detection process includes more pixels than it should, more error is introduced into the process.

Therefore, it is necessary to apply a system that discriminates between harmful intersections and removes them from the process, resulting in better results. In this work two potential improvements have been implemented.

It has been observed that there are intersections whose location is outside what the image should see, considering the position, orientation and FOV of the sensors capturing that object. It does not help to use such intersections, as one of the two lines is poorly aligned and generates this noise. Therefore, this first improvement calculates the intersection of FOV areas, eliminating those intersections from the triangulation process that fall outside this area. A representation of this area can be seen in Figure 6, Figure 7 and Figure 8.

This criterion eliminates many bad intersections, although in some cases several intersections have been found tending to infinity inside this FOV area. These intersections usually occur with car lines far away from the object when the object cannot be correctly enclosed. The farther away from the object, the lower the precision of the image and therefore the bounding-box is. These lines produce intersections that are far away from each the others, causing the centroid to be displaced and the result to be poor.

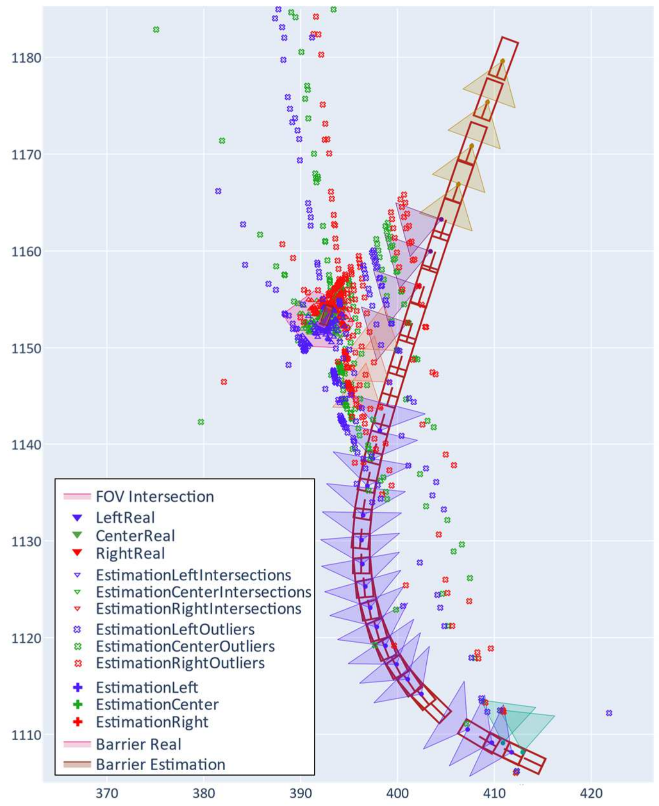


Figure 6 – Barrier intersections discarded due to FOV intersection

To overcome it, several options can be studied to limit these intersections. For example, not generating intersections between almost parallel lines, restricting further the lines to be used by the position of the car or by using only snapshots with bounding-boxes large enough implying that the sensor is closer to the object. The available GIS context could also be used to reduce intersections to only valid locations, eliminating those at buildings or other roads that are occluded from view.

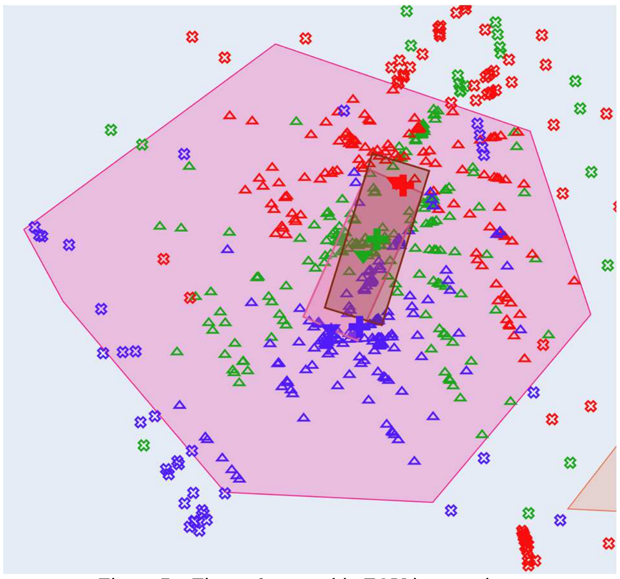


Figure 7 – Figure 6 zoomed in FOV intersection

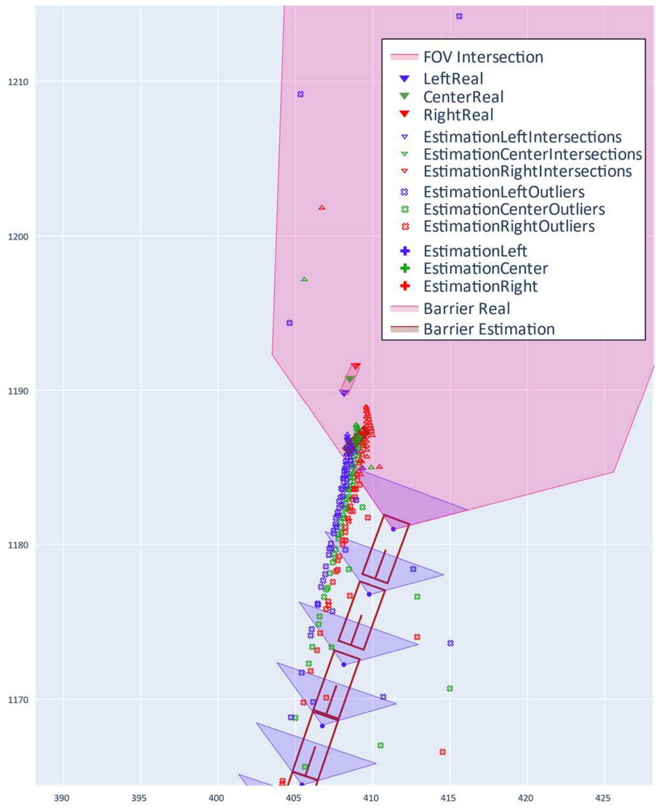


Figure 8 – Complex barrier estimation

Alternatively, the implemented approach eliminates the intersections that are considered outliers in relation to the centroid generated by all the intersections. After calculating the distance of each intersection to the centroid, the interquartile range and quartiles are extracted, eliminating the noisy intersections through a statistical analysis. Those whose distance is below Q1-1.5IQR or above Q3+1.5IQR are discarded. With the remaining intersections a new centroid is calculated which should be more accurate. With this improvement another metric can be calculated, the percentage of intersections discarded as outlier.

Both processes are complementary to each other, first by obtaining the intersections within the FOV, and then by the outlier's elimination. It is to be expected that as more and more criteria are applied to resolve found issues, the better the results will be.

Even with these improvements, there are complex cases, as shown in Figure 8. In this case, both the FOV and the outlier detection do their job, but because all the shots are from the same orientation, with the car moving away from the object, there is no correct intersection, thus the triangulation does not estimate the position correctly.

D. Metrics and results

Using the metrics introduced above, several experiments are carried out with the 55 barriers available. Specifically, the original test with all intersections (All), and two further tests, one for each proposed improvement (NoOut and FOV). Also, both improvements together are tested (FOVNoOut).

Figure 9, Figure 10 and Figure 11 show the main metrics calculated, represented with a boxplot diagram, showing quartiles, interquartile range, and outliers. With dashed lines the mean and standard deviation are shown too. Specifically, these show the difference in positioning error, from the centre and on both sides, the percentage of overlap with the real barrier and the error in barrier orientation, respectively.

It is more than evident that the results after intersection cleaning improve in all metrics, obtaining a more than reasonable accuracy for any further use.

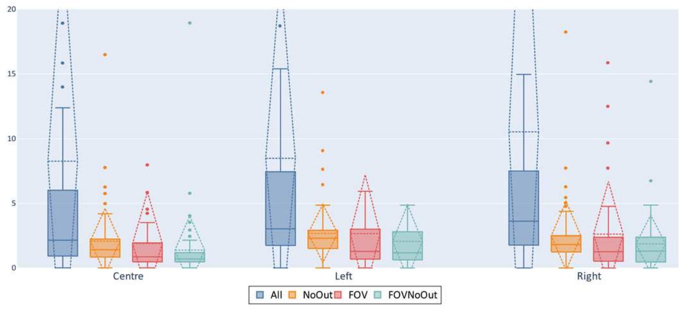


Figure 9 – Position estimation error (meters)

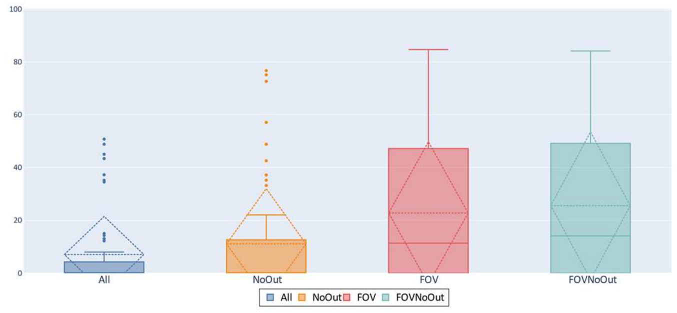


Figure 10 – Overlap percentage with real barrier

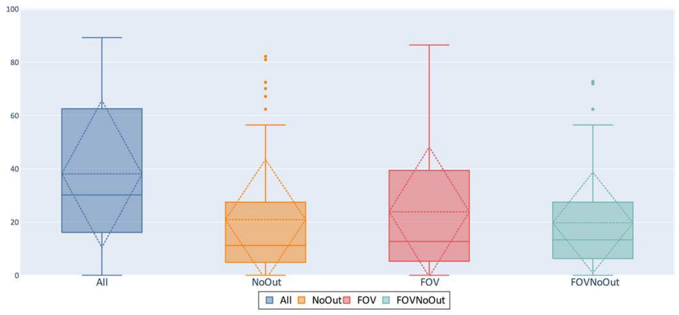
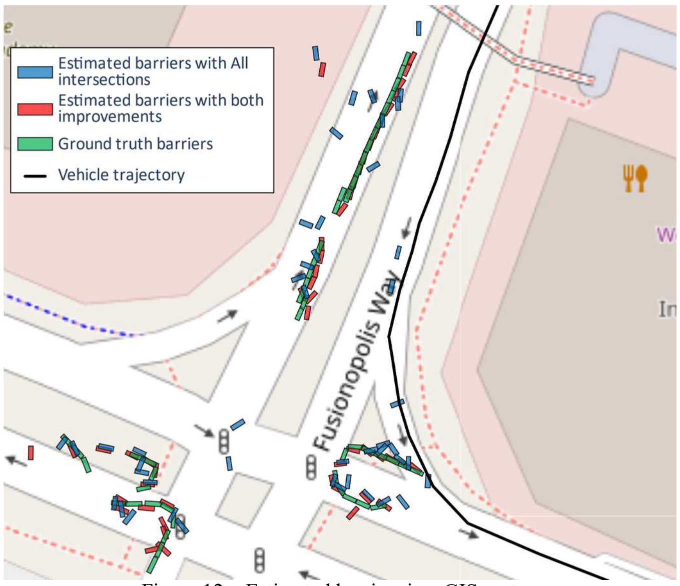


Figure 11 – Yaw orientation error in degrees



Estimated barriers with All intersections
Estimated barriers with both improvements
Ground truth barriers
Vehicle trajectory
Fusionopolis Way

Figure 12 – Estimated barriers in a GIS system

Figure 12 shows the result of switching all detections to a GIS system. Comparing the true barriers with the original and improved estimations, the effects discussed in the previous metrics are observed. By having all the barriers at once, the barriers closer to the vehicle trajectory are significantly better placed, while those further away are a bit more scattered, although they are still good enough to be a fully automatic solution. Still, in some cases the barriers do miss, requiring slight manual adjustment for more precise applications.

V. Conclusions and Perspectives

Quality GIS datasets are important for multiple applications and decision-making. This paper proposes a process to obtain them automatically for any type of sensor and detection. Furthermore, its feasibility is tested by a specific example that detects traffic barriers through 360º imaging. Although the process achieves decent results, there is scope for improvement, either with this approach, developing new ways to clean up noisy intersections, or by exploring other imaging methods.

This work sets a base line of research to be continued in future works. On the one hand, the feasibility of the process should be tested through other types of sensors, such as LiDAR. Also, whether it improves by fusing several sensors. In addition, it is necessary to explore the process operation from other perspectives as well as dealing with the third dimension, for example from an aerial shot of a UAV. On the other hand, the complete implementation of the system is of interest, including the creation of an own dataset, as well as the detection, association and position estimation of different objects close to our environment.

Finally, it is necessary to exploit this generated information. For example, by generating the above-mentioned 3D environments that represent the real world more accurately. To illustrate the capabilities enabled by this process, Figure 13 shows a comparison between a dataset snapshot and a simulator-generated one using the georeferenced information extracted with this process. It has been created using the AirSim drone simulator[20], based on Unreal Engine, and the barriers are automatically placed by the Cesium plugin[21] by means of its latitude and longitude. In addition, a 3D reconstruction of the environment generated with photogrammetry is used to provide a more realistic and context-aware environments for our simulated UAV missions.

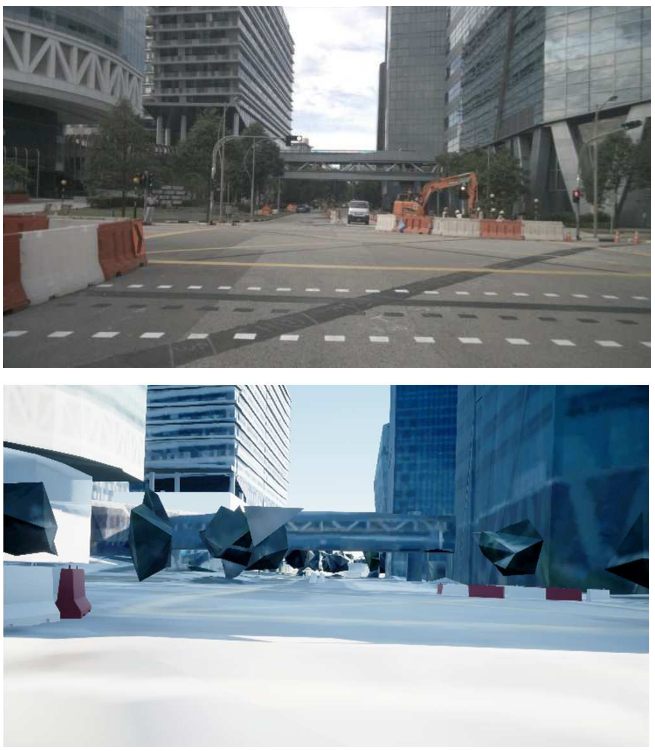


Figure 13 – Comparison between real dataset imagery and a reconstructed scenario using the georeferenced barriers

VI. Acknowledgement

This work was funded by public research projects of Spanish Ministry of Economy and Competitivity (MINECO), reference TEC2017-88048-C2-2-R.

References

[1] L. P. Osco *et al.*, 'A Review on Deep Learning in UAV Remote Sensing', 2021, doi: 10.13140/RG.2.2.19456.66568/3.

[2] L. Li, M. Yang, Bing Wang, and C. Wang, 'An overview on sensor map based localization for automated driving', in *2017 Joint Urban Remote Sensing Event (JURSE)*, Dubai, United Arab Emirates, Mar. 2017, pp. 1–4. doi: 10.1109/JURSE.2017.7924575.

[3] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, 'Digging into self-supervised monocular depth estimation', in *2019 IEEE/CVF international conference on computer vision (ICCV)*, 2019, pp. 3827–3837. doi: 10.1109/ICCV.2019.00393.

[4] M. Li and W. Yao, '3D map system for tree monitoring in hong kong using google street view imagery and deep learning', *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. V-3–2020, pp. 765–772, Aug. 2020, doi: 10.5194/isprs-annals-V-3-2020-765-2020.

[5] V. Krylov, E. Kenny, and R. Dahyot, 'Automatic Discovery and Geotagging of Objects from Street View Imagery', *Remote Sens.*, vol. 10, no. 5, p. 661, Apr. 2018, doi: 10.3390/rs10050661.

[6] S. Lumnitz, T. Devisscher, J. R. Mayaud, V. Radic, N. C. Coops, and V. C. Griess, 'Mapping trees along urban street networks with deep learning and street-level imagery', *ISPRS J. Photogramm. Remote Sens.*, vol. 175, pp. 144–157, May 2021, doi: 10.1016/j.isprsjprs.2021.01.016.

[7] B. Soheilian, N. Paparoditis, and B. Vallet, 'Detection and 3D reconstruction of traffic signs from multiple view color images', *ISPRS J. Photogramm. Remote Sens.*, vol. 77, pp. 1–20, Mar. 2013, doi: 10.1016/j.isprsjprs.2012.11.009.

[8] S. Qiu, A. Psyllidis, A. Bozzon, and G.-J. Houben, 'Crowd-mapping urban objects from street-level imagery', in *The world wide web conference*, New York, NY, USA, 2019, pp. 1521–1531. doi: 10.1145/3308558.3313651.

[9] V. A. Krylov and R. Dahyot, 'Object Geolocation Using MRF Based Multi-Sensor Fusion', in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Athens, Oct. 2018, pp. 2745–2749. doi: 10.1109/ICIP.2018.8451458.

[10] W. Zhang, C. Witharana, W. Li, C. Zhang, X. Li, and J. Parent, 'Using Deep Learning to Identify Utility Poles with Crossarms and Estimate Their Locations from Google Street View Images', *Sensors*, vol. 18, no. 8, p. 2484, Aug. 2018, doi: 10.3390/s18082484.

[11] V. A. Krylov and R. Dahyot, 'Object geolocation from crowdsourced street level imagery', in *ECML PKDD 2018 workshops*, Cham, 2019, pp. 79–83.

[12] N. Kozonek, N. Zeller, H. Bock, and M. Pfeifle, 'On the fusion of camera and lidar 3d object detection and classification', *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-2/W16, pp. 149–156, Sep. 2019, doi: 10.5194/isprs-archives-XLII-2-W16-149-2019.

[13] C. Chen, L. Z. Fragonara, and A. Tsourdos, 'RoIFusion: 3D Object Detection from LiDAR and Vision', *ArXiv200904554 Cs*, Sep. 2020, Accessed: May 29, 2021. [Online]. Available: http://arxiv.org/abs/2009.04554

[14] Y. Wang, Z. Jiang, Y. Li, J.-N. Hwang, G. Xing, and H. Liu, 'RODNet: A real-time radar object detection network cross-supervised by camera-radar fused object 3D localization', *IEEE J. Sel. Top. Signal Process.*, vol. PP, pp. 1–1, Feb. 2021, doi: 10.1109/JSTSP.2021.3058895.

[15] L. Duarte, A. C. Teodoro, J. J. Sousa, and L. Pádua, 'QVigourMap: A GIS Open Source Application for the Creation of Canopy Vigour Maps', *Agronomy*, vol. 11, no. 5, p. 952, May 2021, doi: 10.3390/agronomy11050952.

[16] V. Shah *et al.*, 'Multi-Sensor Mapping for Low Contrast, Quasi-Dynamic, Large Objects', *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 470–476, Apr. 2020, doi: 10.1109/LRA.2019.2962357.

[17] M. Aubry-Kientz *et al.*, 'Multisensor Data Fusion for Improved Segmentation of Individual Tree Crowns in Dense Tropical Forests', *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 14, pp. 3927–3936, 2021, doi: 10.1109/JSTARS.2021.3069159.

[18] H. Caesar *et al.*, 'nuScenes: A multimodal dataset for autonomous driving', in *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Jun. 2020, pp. 11618–11628. doi: 10.1109/CVPR42600.2020.01164.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. West Nyack: Cambridge University Press, 2004. Accessed: Apr. 17, 2021. [Online]. Available: http://qut.eblib.com.au/patron/FullRecord.aspx?p=256634

[20] S. Shah, D. Dey, C. Lovett, and A. Kapoor, 'AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles', *ArXiv170505065 Cs*, Jul. 2017, Accessed: Aug. 29, 2021. [Online]. Available: http://arxiv.org/abs/1705.05065

[21] Cesium GS, 'Cesium for Unreal'. https://cesium.com/platform/cesium-for-unreal/ (accessed Aug. 01, 2021).