A LIDAR Interactive Data Visualization for Ground Aircraft Detection at Small Airports

María-Jesús Lobo, Christophe Hurter, Mathieu Cousy

A LIDAR Interactive Data Visualization for Ground Aircraft Detection at Small Airports

Maria-Jesus Lobo
ENAC
Toulouse, France
Email: maria-jesus.lobo@enac.fr

Christophe Hurter
ENAC
Toulouse, France
Email: christophe.hurter@enac.fr

Mathieu Cousy
ENAC
Toulouse, France
Email: mathieu.cousy@enac.fr

***Abstract*—Light Detection and Ranging (LIDAR) sensors are nowadays widely used for multiple applications such as autonomous driving, wind estimation and robots, however their use in airports has not yet been fully explored. Nevertheless, LIDAR sensors are decreasing in cost, and there are detection algorithms that could be useful in small airports. We present a system to visualize and detect aircraft in small airfields. The system enables an interactive calibration of the algorithm, and exploration of the results. The system has proven useful for the detection and positioning of airplanes in the ENVISION Sesar project.**

***Keywords*—Visualization, LIDAR, aircraft**

I. Introduction

A-SMGCS (Advanced-Surface Movement Guidance and Control Systems) are used in airports to control and monitor aircraft and vehicles in order to optimize capacity, efficiency and safety [1]. They combine cooperative and non cooperative sensors. Their usage has grown in the last decade and they are widely used in airports control towers. However, their implementation is not currently available at small airports because of their high economic cost.

The ENVISION SESAR project aimed at implementing a low cost surveillance system using new low cost technologies and the advances in detection algorithms, such as surveillance cameras and LIDAR sensors. The advances in image processing techniques make it possible nowadays to achieve real time object detection in video streams. LIDAR sensors have been successfully used in a wide set of applications, such as autonomous driving and forest classification and their cost is decreasing. To take advantage of both technologies, a system consisting of an array of cameras, ADS-B (Automatic Dependent Surveillance-Broadcast) receptors and a LIDAR sensor was studied in two different airports. The fusion of these data is displayed to the controller in an Airport Operator A-CDM user interface. The preliminary results show that the proposed system can achieve detection of aircraft, vehicles and persons at approximately 400 meters, 200 meters and 100 meters respectively.

The system relies on machine learning techniques to classify objects in the video stream and on a segmentation algorithm [2] to detect objects in the LIDAR data. The LIDAR algorithm is complemented by a calibration phase to correctly filter out noise, such as very small objects or false detections. To ease this phase, we present a web-based interactive visualization system that displays the point cloud and the detected objects by the algorithm, and enables tunning the algorithm parameters interactively.

This paper focus on the LIDAR contribution to the project and is organized as follows. First, we present the related work that studies the use of LIDAR sensors in airports, the visualization techniques for 3D point clouds and an overview of existing systems. Then, we give an overview of the ENVISION project. Then, we present the implemented system and features. Finally, because one of the main advantages of using a LIDAR sensor is its capability to work in low visibility conditions, we present the results obtained during a field test in light rainy weather. The results suggest that the LIDAR detection algorithm provides high precision positioning for a vehicle in this scenario. We finish with a discussion about the limitations of the system and some perspectives for future work.

II. Related Work

A. Lidar usage in airports

The usage of LIDAR sensors in airports has gained interest recently. For example, Toth et al. [3], [4] use four Velodyne VLP-16 sensors to detect aircraft in the taxiing lane, resulting in motion and position estimation. Mund et al.[5] explored the use of a LIDAR sensor to detect foreign objects on airport aprons. They were focused on detecting small objects that might be hazardous, using a high resolution LIDAR. Their results suggest that the size of the reflective area of the target objects highly influences the resulting detection, and that reflectivity might be more important than distance for object detection. They achieved good detection rates for medium and big size standardized objects.

Our approach is similar to the one presented by Hoch et al. [3], but instead of using multiple LIDAR sensors, we use only one as we focus on detecting only aircraft position, and not trajectory. Also, the configuration of the system is different because the goal is to reproduce the ground radar information, and not to calculate precise trajectories. Furthermore, the contribution of this paper lies more in the interactive visualization of the point cloud, than in the detection algorithm and configuration of the system.

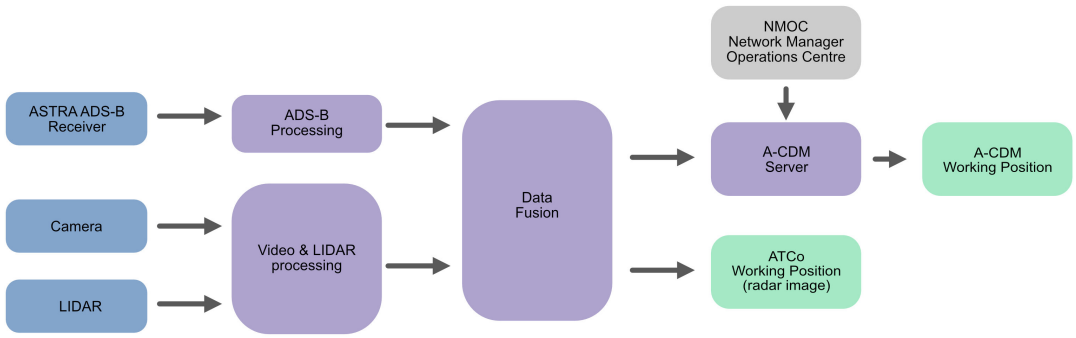

Figure 1. Overview of the ENVISION platform. Sensors are depicted in blue, the processing units in purple, outputs in green.

B. Point cloud visualization

Visualizing large point clouds, such as the ones produced by LIDAR sensors, to gather terrain data is challenging because of the big number of points captured. Several systems have been proposed to visualize LIDAR data [6], [7], [8]. Kuder et al. [6] propose a web based system that enables users to retrieve LIDAR point clouds remotely and visualize them through point based rendering. Maravelakis et al. [7] propose also a web based system to display and navigate in 3D point clouds. A more recent approach, proposed by Kuder et al. [9], create textures from LIDAR data and use them to render the final surfaces, achieving a faster visualization. Deibe et al. [10] propose a web based Lidar Visualization and they provide tools to make geospatial measurements as well.

These works focus mostly on efficient visualizations because they are targeted to display big point clouds. In our case, the point clouds generated by the LIDAR, are sparse, so using traditional rendering techniques is enough. Our system rather focuses on performing the visualization and detection in real time, and providing visual cues to help understanding the algorithm results.

There are some commercial and open source existing software to visualize LIDAR data in real time, such as LAStools [11], Fugro Viewer [12], Lidar View [13], Potree [14] and Veloview [15]. VeloView, for example, is a multi-plataform solution that can be connected to a Velodyne LIDAR and provide a visual representation of the live cloud point. It provides several color encodings for the points, and some basic filtering operations, such as cropping the data on the x, y and z axis. It also supports some basic navigation such as panning and zooming. The system described in this paper also provides these options, but it adds the capability to apply real time detection to the data and visualize different algorithm outputs.

C. Interaction techniques

Interaction techniques further enhance these visualization, by making it possible for the users to navigate and select subsets of the data. Selection techniques for 3D visualization have been widely investigated, such as ray casting [16], alternative spaces [17] and 3D volumes [18]. Some techniques leverage the data context and characteristics to facilitate selection and exploration. The CAST techniques are designed specifically for selection in 3D point clouds and use context information to infer the desired user selection. Prouzeau et al. [19] introduce new techniques for scatterplot exploration that maps the data density to the vibration of virtual reality controllers, providing a haptic feedback.

Immersive technologies have also been used to visualize LIDAR data. Burwell et al. [20] propose an virtual reality environment to explore LIDAR data. An evaluation with expert users suggest that this visualization can be useful to have a more realistic perception of the data. Ave [21] uses LIDAR data to extract buildings and then project imagery on top of them to create a virtual environment.

III. Envision

A. Overview

The ENVISION system couples multiple sensors and processing modules to calculate the position of aircraft, vehicles and people. For now, the ENVISION system uses multiple cameras, one LIDAR and one ADS-B sensor. Each one of these sensor feeds a different processing module, that decode and process its data. The processing results are then consolidated in a Data Fusion module that track targets, calculate target positions and generate aircraft movement events. The platform overview is displayed in Figure 1.

The ENVISION project was tested on Muret Airfield, France. The LIDAR was positioned on a mast at a height of approximately 8 meters. From there, it covered part of the parking area and the closest taxiing.

B. LIDAR processing

The LIDAR processing module is in charge of processing the raw data coming from the sensor and performing real time object detection in the point cloud. The LIDAR sensor measures the distance to objects by using laser lights. In our case, the LIDAR generates point clouds, where each point is characterized by its distance, and its horizontal and vertical angles. This is then used to calculate the point positions in 3D. To perform object detection, we use the algorithm presented by Bogoslavsky and Stachniss [2]. This algorithm is specially

ISSN 0770-1268

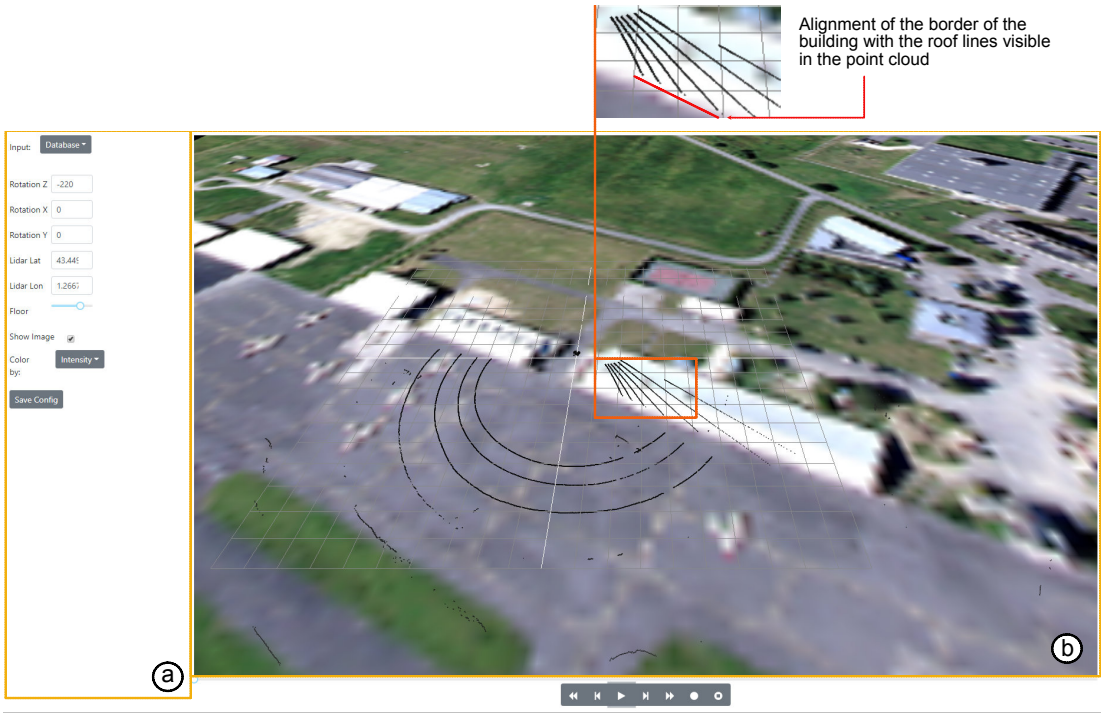

Figure 2. Interface of the system. (a) Inspector area (b) Point cloud visualization. The background image can be used to find the rotation and position of the LIDAR.

suited to our use case because it performs fast detection on sparse point clouds.

To achieve fast segmentation, Bogoslavsky and Stachniss propose to perform the detection on the 2D image generated by the LIDAR rather than in the 3D point cloud. For this, the algorithm starts by performing a classification to ignore the points that are on the ground. In our case, we use a plane fitting approach. Then, for each point that is not in the ground, four neighbors are considered: the two closest points coming from the same laser (horizontal neighbors) and the closest points with the same azimuth coming from the top and down lasers (vertical neighbors). With this, it is possible to build a graph where each point will have a maximum of four neighbors, depending of the distance between the points. For two points to be considered neighbors, the distance between them has to be inferior to a *distance threshold*.

Once the graph is built, for each two neighbors, the *beta* angle is calculated, as the angle between the segments that goes through the LIDAR origin and the farthest point among the two neighbors, and the line going through the two points. This angle reflects the depth difference between the two points, if it is small the points are probably at different depths and thus probably belong to different objects. A *beta threshold* is determined to decide if two points belong to the same object or not.

Because this algorithm requires a calibration of the *beta threshold* and the *distance threshold* parameters, we designed and implemented an interactive visualization that enables users to calibrate the algorithm. This system is described in the following section.

IV. Interactive Visualization

The system was designed to support the design and implementation of the LIDAR detection algorithm. Because of this, we draw four design requirements to enable users to explore the LIDAR data and the algorithm results. These requirements are inspired by existing visualization systems.

A. Design requirements

- Support navigation in the point cloud through basic operations like pan and zoom.
- Support manipulation of the algorithm parameters and present the results directly in the visualization
- Support exploration of live and recorded data
- Present visually cues to enable users to understand how the algorithm work

B. Features

The final interface is depicted in Figure 2, and consists of an inspector, so that the user can calibrate the algorithm, and a point cloud visualization, so that the user can explore the data and the algorithm results. The point cloud visualization supports panning and zooming operations. Through the inspector, the user can tune the algorithm calibration and the visualization. The specific features addressing the design requirements are detailed next.

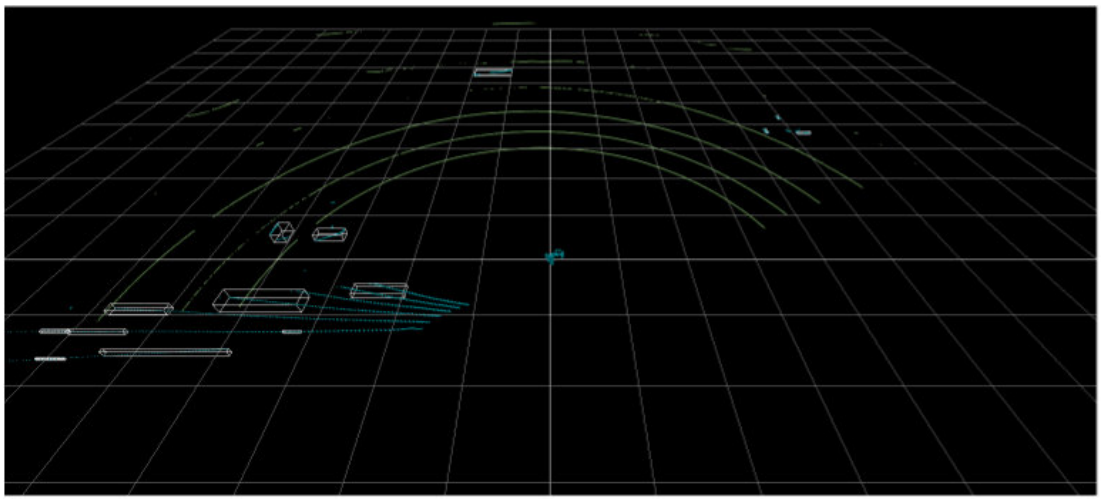

Figure 3. Detection results using the algorithm presented in [2], depicted as white wireframe boxes.

1) LIDAR calibration: Because the fusion module used in the ENVISION solution requires having absolute positions in (lat, lon) of the detected objects, the positions found in the LIDAR coordinates need to be transformed to world coordinates. This requires knowing the LIDAR geographical position and its orientation. We propose using a geo-referenced image to enable the user to infer the LIDAR position and orientation by matching the cloud point to the image. To achieve this, he can set the position and orientation in the inspector and the changes will be reflected in the interface, so he can visually match a landmark visible in both representations. Figure 2 depicts how the user can match the roof lines visible in the LIDAR point cloud to the roof on the image. The final transformation to world coordinates is done using the Turf library [22].

2) Algorithm interactive parameter setting: Using the inspector, the user can also select the information coded through the points' colors. The available parameters are height of the points, distance from the LIDAR, received intensity, azimuth and vertical laser id. These parameters match the ones available in the existing visualization solutions, such as VeloView [15].

The user can also select to depict the results of the detection algorithm, as illustrated in Figure 3. Because the results of the algorithm depend on the correct classification of points as ground or non ground, we use the color to depict this classification. Points classified as ground are depicted in green, and non ground in blue.

The detected objects are depicted as white bounding boxes. Using the inspector, the user can set the *distance threshold* and the *beta threshold* parameters and see the changes in the detection results immediately. We also incorporated some additional parameters that were proven useful to further reduce false detections:

- Minimun distance from the LIDAR: This helps reducing false detection due to very close objects, such as the cameras in the mast.
- Minimum and maximum size of the objects: Because we were interested in detecting planes, we set a minimum and maximum size to reduce false detections. This comes at the cost of potentially missing objects, if they are covered by few points. The limitations will be discussed in more detail in the Section VI.
- Minimum and maximum azimuth: As depicted in Figure 3, some objects are detected on the building roof. To avoid such detections, we can filter the minimum and maximum azimuths so that only the detections happening in the airfield parking are considered.

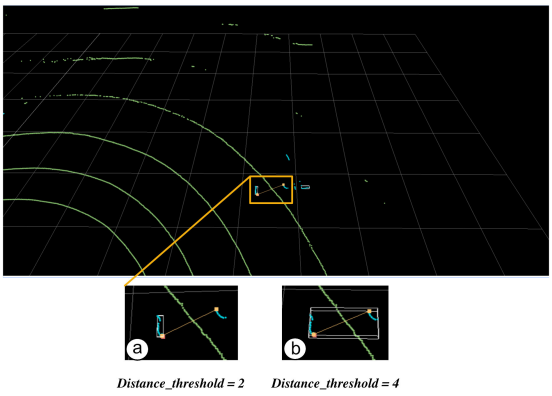

Figure 4. Example of the exploration of the algorithm results. Clicking on one node will display its neighbors as visible in (a). The distance between nodes to be considered neighbors can be modified so the identified objects are grouped as one.

Each selected object can also be selected and removed from the detected objects. This will result in filtering out all the objects at a vicinity of the removed object. This is useful to remove recurrent false detections produced by noise.

C. Interactive exploration of the algorithms' results

In order to make the algorithm processing more understandable, we first propose to enable the exploration of the graph calculated by the algorithm described in Section III. Some

Figure 5. Example of a flow generated using a 2D image. Here each laser is represented as a horizontal line, and the x coordinate represents the azimuth. The green arrows show the direction of the change between the depicted frame and the next one. Here we only display the lines covering the parking area

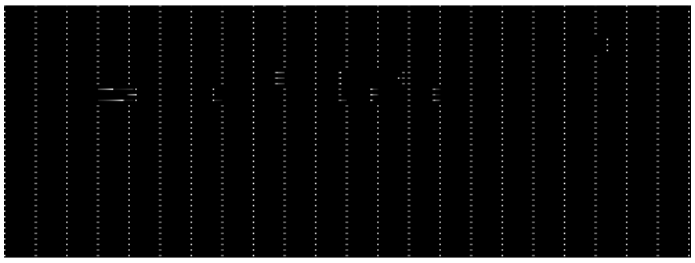

Figure 6. A flow displaying the calculated change directions from one frame to another depicted using the OLIC technique designed for flow visualization. Each LIDAR horizontal line is represented through four particle lines.

visualization techniques use propagation to explore graphs, by extending node selection to their neighbors ([23], [24], [25]). We attempt to apply this concept to explore the algorithm results. The user can select any point classified as non ground, and its neighbors will be highlighted in orange and the links between them displayed with a yellow line.

This might help to understand the algorithm results and to better tune the algorithm parameters. For example, in Figure 4-(a) only a part of an object is classified as such. By clicking one of the points in the border of the object, we can see that the two points are neighbors, so they are probably not considered part of the same object because of the *distance threshold* parameter. By setting its value to 4 meters instead of 2 meters, the two objects are considered as one.

Because many changes might happen from one frame to another, it might be useful to provide to the user an overview of where the point cloud is changing, and what is the direction of the change. To achieve this, we propose to calculate a flow field that will represent the movement from one frame to another, similar to what the Optical Flow [26] algorithm does. For each point in the 2D image (azimuth and vertical angle), we use a set of neighbors to find a matching set of neighbors of the same size in the following frame, by finding the points that present the smaller difference in terms of distance from the LIDAR. Then, the vector field will display the direction from the initial point to the matched point. Figure 5 depicts such flow. We are currently investigating other techniques to visualize this data, such as flow visualization techniques. Figure 6 depicts such flow depicted using the OLIC [27] technique. This technique depicts direction as animated droplets, that are brighter following the flow direction.

D. Implementation

The system is implemented through two main components: a server and a client. The server is in charge of retrieving the point cloud (either from the database or from the sensor), processing it and running the detection algorithm. The server

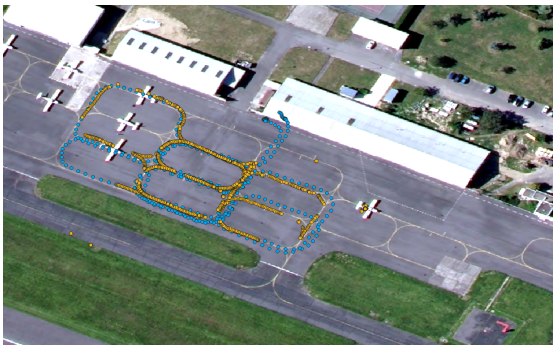

Figure 7. Points recorded by the GPS in the car (blue), and positions detected by the LIDAR algorithm (yellow). The missing detection on the left of the image can be explained by a lack of points on the car at those times. Background image just for reference.

is based on the Node.js[1] technology. The web client is based on the libraries Three.js [2], used for the 3D visualization, and React.js [3] to handle user interaction. We chose Three.js because it is a powerful toolkit to display 3D graphics that eases the use of WebGL. The system is able to perform the rendering and detection of a point cloud once per second. When performing real time detection on the airport, only the detection algorithm runs, detecting at 5 frames per second.

We use a LIDAR Velodyne VLP-16. This LIDAR has a horizontal resolution of 360 and 16 vertical angles. All the displayed images were taken on the Muret Airfield, France. The LIDAR is positioned on a mast in the airfield at a height of approximately 8 meters. Because the LIDAR is on the mast, we only can use approximately half of the 16 lines produced by the sensor.

V. Evaluation

In order to evaluate the performance and accuracy of the LIDAR detection algorithm, we conducted a test on the Muret airfield where a car equipped by a GPS drove around the parking under light rainy weather. For the 400 seconds of the recording, the LIDAR detected the car 85% of the time, with an accuracy of 6.7m 50% of the time and 10.2m 95% of the time, as depicted in Figure 8. The accuracy does not seem to depend on the distance, as depicted in Figure 9 However, the detection presents six gaps of 3s where the car is not detected. These gaps are visible in Figure 7. The gaps can be explained because we did not have points that covered the car in those moments.

VI. Discussion and Conclusion

We described a LIDAR interactive visualization system to ease the calibration phase and explore the results of a detection algorithm. The algorithm achieved good results under light rain weather. However, our approach presents several limitations.

[1]https://nodejs.org/en/
[2]https://threejs.org/
[3]https://reactjs.org/

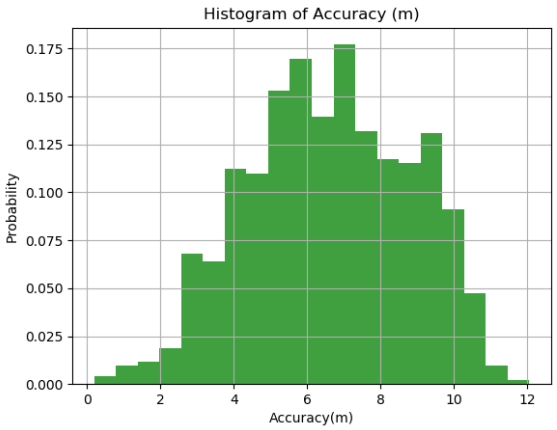

Figure 8. Histogram of LIDAR detection accuracy

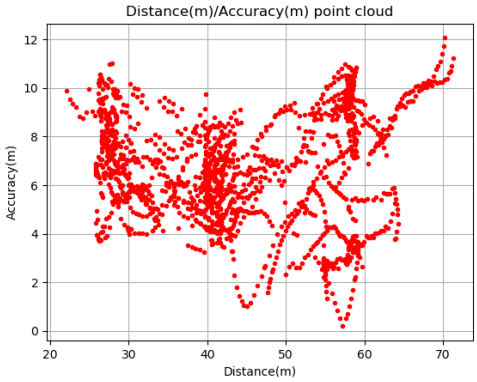

Figure 9. LIDAR detection accuracy according to the distance.

First, we have only conducted one validation in a specific context. The results are encouraging but more tests are needed to completely assess the system and detection.

Second, the tuning of the algorithm parameters and the filtering of the results can be useful to avoid false detections. However, this comes at the cost of missing objects in the detection phase. For example, we cannot detect persons or small vehicles, because we filter out object that have a bounding box where the longest diagonal is smaller than 1m. During preliminary tests, we observed that not setting this minimum size produced too many false detections, thus we decided to limit the LIDAR to airplane detection.

Third, our algorithm does not perform any kind of classification, thus we cannot know what kind of object has been detected. However, this does not hamper the ENVISION platform, because the LIDAR and the video data are complementary. The LIDAR detection provides precise depth measurements, while the video processing results in accurate classification.

These two latter limitations could be mitigated by increasing the resolution of the point cloud. This could be easily achieved by using a LIDAR with higher resolution. However, this would increase the cost of the system. Increasing the resolution could also be possible using the current LIDAR. First, the sensor could be oriented towards the floor, thus increasing the coverage of the area close to the mast. Second, we could make the LIDAR oscillate in order to make it cover the gaps between two lasers.

ACKNOWLEDGMENT

This research was partly supported by SESAR H2020 project ENVISION grant agreement No 783270.

REFERENCES

[1] E. Piazza, "A-smgcs routing and guidance functions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 15, no. 7, pp. 15–23, 2000.

[2] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 163–169.

[3] C. Toth, G. Jozkowa, Z. Koppanyi, S. Young, and D. Grejner-Brzezinska, "Monitoring aircraft motion at airports by lidar." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 3, no. 1, 2016.

[4] Z. Koppanyi and C. K. Toth, "Object tracking with lidar: Monitoring taxiing and landing aircraft," *Applied Sciences*, vol. 8, no. 2, 2018.

[5] J. Mund, A. Zouhar, L. Meyer, H. Fricke, and C. Rother, "Performance evaluation of lidar point clouds towards automated fod detection on airport aprons," in *Proceedings of the 5th International Conference on Application and Theory of Automation in Command and Control Systems*, ser. ATACCS '15. New York, NY, USA: ACM, 2015, pp. 85–94.

[6] M. Kuder and B. alik, "Web-based lidar visualization with point-based rendering," in *2011 Seventh International Conference on Signal Image Technology Internet-Based Systems*, Nov 2011, pp. 38–45.

[7] E. Maravelakis, A. Konstantaras, K. Kabassi, I. Chrysakis, C. Georgis, and A. Axaridou, "3dsystek web-based point cloud viewer," in *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, July 2014, pp. 262–266.

[8] M. Kulawiak and M. Kulawiak, "Application of web-gis for dissemination and 3d visualization of large-volume lidar data," in *The Rise of Big Spatial Data*, I. Ivan, A. Singleton, J. Horák, and T. Inspektor, Eds. Cham: Springer International Publishing, 2017, pp. 1–12.

[9] M. Kuder, M. terk, and B. alik, "Point-based rendering optimization with textured meshes for fast lidar visualization," *Computers Geosciences*, vol. 59, pp. 181 – 190, 2013.

[10] D. Deibe, M. Amor, R. Doallo, D. Miranda, and M. Cordero, "Gvlidar: an interactive web-based visualization framework to support geospatial measures on lidar data," *International Journal of Remote Sensing*, vol. 38, no. 3, pp. 827–849, 2017.

[11] RapidLasso. Lastools. [Online]. Available: https://rapidlasso.com/lastools/

[12] Fugro. Fugro viewer. [Online]. Available: https://www.fugro.com/about-fugro/our-expertise/technology/fugroviewer

[13] A. Krivutsenko. Lidar view. [Online]. Available: http://lidarview.com/

[14] M. Schütz, "Potree: Rendering large point clouds in web browsers," *Technische Universität Wien, Wiedeń*, 2016.

[15] Paraview. Veloview. [Online]. Available: https://www.paraview.org/veloview/

[16] G. J. K. C.-M. P. SangYoon Lee, Jinseok Seo, "Evaluation of pointing techniques for ray casting selection in virtual environments," 2003. [Online]. Available: https://doi.org/10.1117/12.497665

[17] C. Hurter, R. Taylor, S. Carpendale, and A. Telea, "Color tunneling: Interactive exploration and selection in volumetric datasets," in *2014 IEEE Pacific Visualization Symposium*, March 2014, pp. 225–232.

[18] L. Yu, K. Efstathiou, P. Isenberg, and T. Isenberg, "Efficient structure-aware selection techniques for 3d point cloud visualizations with 2dof input," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2245–2254, Dec 2012.

[19] A. Prouzeau, M. Cordeil, C. Robin, B. Ens, B. H. Thomas, and T. Dwyer, "Scaptics and highlight-planes: Immersive interaction techniques for finding occluded features in 3d scatterplots," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: ACM, 2019, pp. 325:1–325:12.

[20] C. Burwell, C. Jarvis, and K. Tansey, "The potential for using 3d visualization for data exploration, error correction and analysis of lidar point clouds," *Remote Sensing Letters*, vol. 3, no. 6, pp. 481–490, 2012.

[21] U. Neumann, S. You, J. Hu, B. Jiang, and J. Lee, "Augmented virtual environments (ave): Dynamic fusion of imagery and 3d models," in *IEEE Virtual Reality, 2003. Proceedings.* IEEE, 2003, pp. 61–67.

[22] Turfjs. Turf. [Online]. Available: https://github.com/Turfjs/turf

[23] J. Heer and D. Boyd, "Vizster: visualizing online social networks," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, Oct 2005, pp. 32–39.

[24] M. J. McGuffin and I. Jurisica, "Interaction techniques for selecting and manipulating subgraphs in network visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 937–944, Nov 2009.

[25] A. Prouzeau, A. Bezerianos, and O. Chapuis, "Evaluating multi-user selection for exploring graph topology on wall-displays," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 8, pp. 1936–1951, Aug 2017.

[26] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981.

[27] R. Wegenkittl, E. Groller, and W. Purgathofer, "Animating flow fields: rendering of oriented line integral convolution," in *Proceedings. Computer Animation '97 (Cat. No.97TB100120)*, June 1997, pp. 15–21.