A lightweight Steering Algorithm for Smart Scanning

Aurélien Fresneau, Tiana Rakotovao, Diego Puschini

A lightweight Steering Algorithm for Smart Scanning

A. Fresneau[1], T. Rakotovao[1], D. Puschini[1]

1: Univ. Grenoble Alpes, CEA, List, F-38000 Grenoble, France

Abstract: This work presents a lightweight steering algorithm specifically designed for active perception sensors (Lidars, Radars or Sonars) that support online configurable scanning beam. The algorithm aims to improve the sensor capabilities in terms of frame rate and accuracy in some regions of interests (ROIs) with a limited power budget typically less than 2 Watts.

Keywords: Smart Sensors, Lidar, Radar, Low Power, Smart Scanning, Smart Steering

1. Introduction

Active perception sensors such as Lidars, Radars or Sonars scan the environment by sending out signals in the form of beam shots, and processing the reflected echoes in order to locate obstacles. Traditional scanning policy consists in sending preconfigured beams with constant properties toward predefined directions. This approach is well illustrated by the scanning mechanism of rotating Lidars, which send light beams with the same properties toward predefined azimuth and elevation angles at a constant rate.

Sensors which have online configurable steering capabilities, for example with regards to beam direction and beam divergence, already exist on the market for some technologies – Multiple Input Multiple Output (MIMO) radars for instance – and are in development for other technologies (Sonar, Lidar). By allowing the beam parameters to be configured in real time, smart scanning policies, more complex compared to traditional ones, can be implemented in order to improve the sensor capabilities in terms of frame rate and energy consumption.

A smart scanning policy aims to provide higher rate and higher precision measurements in some regions of interests (ROIs), while maintaining constant or reducing rates and precision elsewhere. In this work, ROIs are determined using a lightweight steering algorithm integrated into the device and which analyses the scan history. The algorithm can also take into account user-defined ROIs provided by external applications through an Application Programming Interface (API).

Modern sensors are equipped with micro-controller units (MCUs) in order to perform advanced edge signal processing. Our smart scanning algorithm is designed to be integrated in such MCUs and is thus expected to have a low impact on the final bill of material (BOM) of the sensor device.

The steering algorithm relies on innovative approaches [1, 2, 5] to build an environment model from successive sensor scans with a power budget typically less than 2 Watts. It subsequently generates a list of commands that controls the beam divergence and direction in order to optimize the amount of beam shots required for covering the ROIs [3], hence decreasing the number of beam shots necessary to cover the environment. This optimization translates into either a faster possible frame rate or a lower power consumption. It additionally allows for a higher accuracy in the detection and definition of obstacles within the ROIs.

2. Principle of the steering algorithm

We assume that the active perception sensor being used can be controlled with at least one of the following parameters:

- The polar angle θ and elevation φ of the sensor beam
- The angular opening α of the sensor beam

Several modules are running together and communicate with each other in order to perform the real time steering process (Figure 1).

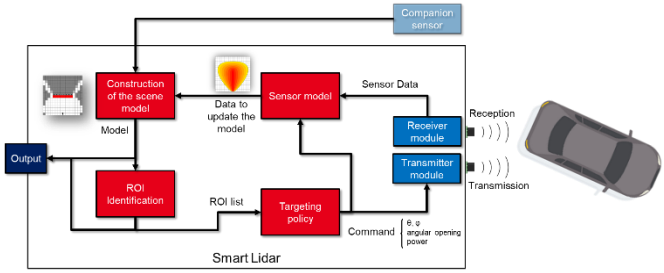

Figure 1: Diagram describing the principle of the steering algorithm

2.1 Building the environment model

The first module receives the data produced by the sensor, and processes it in order to build the environment model of the scene. This is performed either in conjunction with another perception

sensor's data – called companion sensor – or solely with the current sensors data. Either way, the environment model chosen is a probabilistic Occupancy Grid (OG). In a probabilistic OG, space is divided into squares in 2D or cubes in 3D, and each of these cells contains the probability of this space being occupied based on the sensors data and prior knowledge. The Bayesian fusion performed to build the OG utilizes a patented method that uses integer arithmetic in order to save computing power [4].

If the angular opening α of the sensor beam is controllable, it is modelled with the help of the patent [2] before being fused in the OG. In order to take into account the temporal aspect of the fusion between the previous knowledge of the environment and the new data, a forgetting factor is introduced. Indeed, before adding the new data to the OG, each cell receives a treatment that uniformly shifts the occupation probability of the cell towards 0.5. This gives more importance to the most recent data.

2.2 ROI identification

The second module uses the environment model built by the first one as an input in order to calculate ROIs. The ROIs are described as a list of positions and sizes weighted by the importance of each region. The way they are calculated can vary greatly based on the targeted application. They can for instance be based on identified specific objects, edges, least observed areas, moving targets, closest targets, etc… Since the form that this module can take is so diverse and this is not the focus of this article, the ROI module used as an example will only be briefly discussed in the result section.

2.3 Targeting policy

The next module uses the ROIs calculated by the previous one as inputs to determinate the list of commands to send to the controllable sensor for its next scan. Based on their size and importance, different profiles can be chosen: higher or lower density of shots, wider or narrower beam.

2.4 Output format

Depending on the needs of the user, this processing chain can provide several different outputs. First, an obvious choice is to keep the original output format of the sensor. It has the same format as what would the sensor provide without the steering algorithm, and it is thus transparent for the user. The only difference is the non-uniform distribution of shots, and the variation in the beam divergence between shots. If a more exhaustive understanding of the environment is expected, the whole model of the scene can be provided (OG), as well as the list of ROIs. Finally, virtual data can even be reconstructed from the OG in order to provide uniform pointclouds (PCL) even in areas where the sensor did not shoot, if such a data structure is required by the user.

We will now describe with more details a specific implementation of the steering algorithm developed as a proof of concept.

3. Experimental results

The steering algorithm has been tested on an NVIDIA Jetson Nano with 4GB of RAM. It should be noted that the GPU is only used for the display of the data. The algorithm itself runs exclusively on the CPU. It runs at the same framerate as the sensor, which is 10Hz. The primary sensor in mind when this algorithm was developed was a solid state FMCW Lidar relying on OPA technology for the steering. Such sensors will hit the market of perception sensors in a few years, but are not yet available for testing. In order to test the capabilities of our steering algorithm with an experimental setup, we used a non-controllable Lidar (Ouster OS1-64). It is a rotating turret spinning at 10Hz with 64 vertical layers of laser, connected with an RJ45 cable to the Jetson Nano. To mimic the behaviour of a steerable Lidar, we replaced the module that gives commands to the sensor with a module that instead filters the data of the Lidar. The filtered data points represent where a controllable Lidar would have fired. This approach conveniently allows us to calculate how many shots would be saved compared to the traditional uniform targeting of the rotating turret Lidar.

A webcam is used in the experiments uniquely to improve the understanding of the displayed data; it does not play any role in the steering algorithm. However, it could be utilized as a companion sensor with the help of AI algorithms to select ROIs from the images for instance. The ROIs chosen by the steering algorithm would then come from both the data of the camera and the Lidar.

The ROI identification module chosen to demonstrate the capabilities of the steering algorithm in the experimental setup is based on the algorithm introduced in [5]. This algorithm uses OGs as inputs in order to estimate the motion of likely occupied cells and detect dynamic cells. Groups of cells likely moving together in the same direction are then clustered. The ROIs chosen are those clusters.

The Lidar is kept immobile in the experiments conducted. Figure 2 shows a snapshot from the display of the experiment. A person is moving in the scene (towards the sensor in this specific example), and we can see the higher density of data points targeting the moving person compared to the surroundings. The targeting policy consists in using the full Lidar resolution in the ROI, and 1/8 of the points in the other areas. The ratio of the amount of points in the filtered PCL compared to the full resolution PCL thus varies between one 8^{th} and one depending on the area of the ROIs. Although this

data reduction does not save energy in the filtering setup, it would in the case of a steerable Lidar. This would thus lead to a lower power consumption of the Lidar, or a faster frame rate depending on the application targeted.

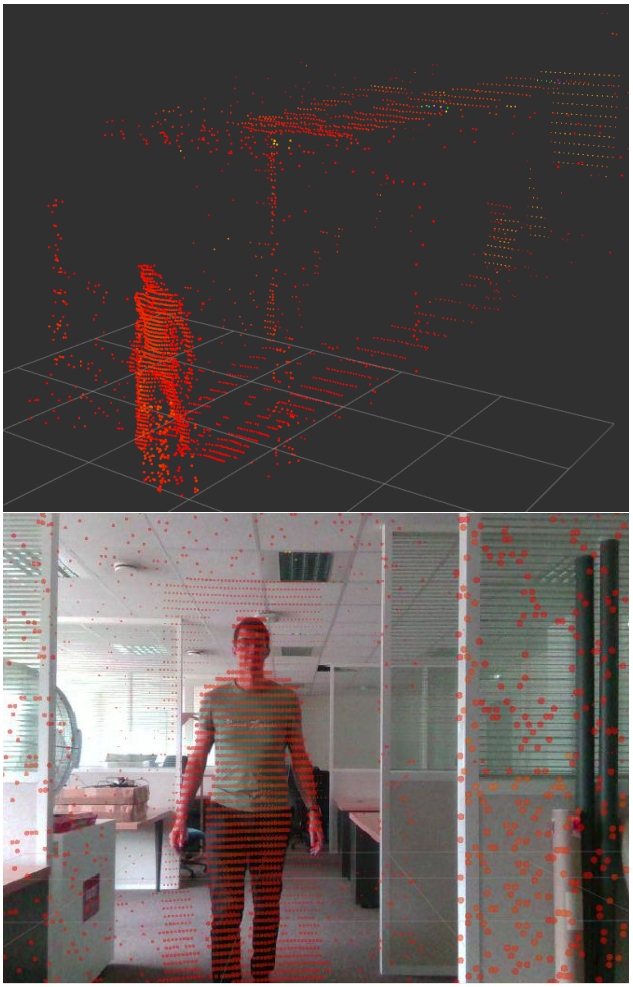

Figure 2: Snapshot of the display of the experimental setup. The top panel shows the filtered PCL representing the data that a real steerable Lidar could provide. The bottom panel shows the filtered PCL superimposed with the image of a webcam for a better understanding of the scene. The difference in the density of points is clearly visible in this representation.

4. Conclusion

We presented in this article a versatile steering algorithm with a modular architecture that makes it adaptable to many use cases. Thanks to the use of OGs, the method is agnostic to the presence or absence of companion sensors as long as their data can be fused in an OG. Both the output format and the ROI calculation can be adapted for each situation. The lightweight approach of the algorithm coupled with the fact that it can work in closed loop only with the steerable sensor's data makes it ideal to be integrated in the sensor MCU itself, enhancing its capabilities without increasing its BOM.
The experimental results shown are encouraging and pave the way for when steerable Lidars will be available. The control of the beam divergence was not investigated with the current setup due to the nature of the available Lidar but will be in future experiments, as it is expected to bring even more flexibility to the system.

5. Acknowledgement

The authors thank the members of the LIIM laboratory at CEA for their constructive comments in the process of developing the steering algorithm.

6. References

[1] J. Mottin, D. Puschini, T. Rakotovao – Patent EP3353720 – Method and system for perceiving physical bodies.

[2] R. Dia, F. Heitzmann, S. Lesecq, J. Mottin, D. Puschini, T. Rakotovao – Patent EP3594719 – Environment sensing method and apparatus using a wide-angle distance sensor.

[3] A. Fresneau, D. Puschini – Patent Pending – Procédé et système de perception de corps matériels à balayage optimisé.

[4] T. Rakotovao, J. Mottin, D. Puschini, and C. Laugier, "Multi-sensor fusion of occupancy grids based on integer arithmetic," in Proceedings - IEEE International Conference on Robotics and Automation, vol. 2016-June, 2016.

[5] T. Rakotovao – Patent EP4002274 – Iterative method for estimating the movement of a material body by generating a filtered movement grid.

7. Glossary

OG: Occupancy Grid
ROI: Region Of Interest
MIMO: Multiple Input Multiple Output (Radar)
API: Application Programming Interface
MCU: Micro Controller Unit
BOM: Bill Of Material
PCL: Pointcloud
GPU: Graphics Processing Unit
CPU: Central Processing Unit
RAM: Random Access Memory
FMCW: Frequency-Modulated Continuous-Wave
OPA: Optical Phased Array
AI: Artificial Intelligence