Hardware-software implementation of car detection system based on LiDAR sensor data - a demo

Joanna Stanisz
AGH University of Science and Technology Krakow, Poland
E-mail: stanisz@student.agh.edu.pl

Konrad Lis
AGH University of Science and Technology Krakow, Poland
E-mail: lis@student.agh.edu.pl

Tomasz Kryjak, Senior Member IEEE
AGH University of Science and Technology Krakow, Poland
E-mail: tomasz.kryjak@agh.edu.pl

Marek Gorgoń, Senior Member IEEE
AGH University of Science and Technology Krakow, Poland
E-mail: mago@agh.edu.pl

Abstract—The article presents an object detection system based on LiDAR sensor data. The solution allows real-time detection of cars using information about the distance and intensity of the reflection obtained from LiDAR. The module has been implemented in programmable logic resources of the Xilinx Zynq SoC (System on Chip) device. The detected objects were projected onto the camera image for visualization. The system is dedicated to Velodyne HDL-64E LiDAR.

Index Terms—car detection, point cloud, LiDAR, FPGA, Zynq SoC, real-time processing, signal processing

I. INTRODUCTION

The LiDAR sensor (Light Detection and Ranging) allows to obtain a point cloud describing the space around the vehicle. By properly processing this data, object can be localized and classified. This information is then necessary for the operation of autonomous cars. It allows, among others, for object tracking, speed estimation and thus also for detection of dangerous situations. In order to provide fast vehicle response, the delay between data acquisition and detection results should be as small as possible.

Until now, data from LiDAR were mainly processed on platforms with general purpose processors [1], [2] and in one case on the heterogeneous Zynq SoC [3]. In the paper [2], the authors describe the pedestrian detection method based on data from the commercially available 3D Flash LiDAR sensor. They use algorithms such as RANSAC and k-means clustering. The authors of article [1] proposed an algorithm with main steps similar to those described in Section II, as well as filtration of high objects such as buildings. In their work [3], the authors used the ZedBoard development board with a Zynq device. However, the proposed algorithm has been implemented in large part on the ARM processor using the OpenCV library. In this paper, in order to meet real-time requirements and maintain energy efficiency, it was decided to implement the car detection system using a hardware-software approach.

The work presented in this paper was supported by AGH University of Science and Technology project.

II. THE PROPOSED ALGORITHM

The first step of the proposed algorithm is data acquisition from the LiDAR sensor. Due to the lack of access to a LiDAR, we used the KITTI database as a source of data [4]. It provides data sequences recorded via a multi-sensor setup: video streams and point clouds in Cartesian coordinates. The LiDAR data includes three basic coordinates of a point: X, Y (position on the plane) and Z (height). Each point also has a fourth component, the reflection intensity, which is a measure of the scattering of the reflected light beam.

In the next stage a preliminary processing is applied for the point cloud. It consists of ground removal (i.e. points that lie on the road surface), filtration and background removal (i.e. points that belong to e.g. buildings). These operations are carried out in 1m x 1m square cells in the XY plane. In each, the number of points and the average value of their z components are computed. If the second value is smaller than a pre-set threshold, then the cell is considered to belong to ground. Moreover, at the filtration stage, cells in which the number of points is less than a threshold are also removed. During the background removal stage, cells in which the maximum z component is too large are rejected. In result, buildings and other high objects are not included in further processing.

The next stage is segmentation. Its purpose is to separate areas where potential objects are located from the point cloud. It is applied to cells marked as correct during pre-processing, i.e. not being a ground, background or noise. These cells are aggregated into larger segments of 3m x 3m. In addition, the segments overlap in both directions on the XY plane.

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>f_1</td>
<td>Real cell sizes [m] (height, length, width)</td>
<td>3</td>
</tr>
<tr>
<td>f_2</td>
<td>Number of points in a cell</td>
<td>1</td>
</tr>
<tr>
<td>f_3</td>
<td>Mean intensity of reflectance in a cell</td>
<td>1</td>
</tr>
<tr>
<td>f_4</td>
<td>Histogram of reflectance intensity with 25 bins</td>
<td>25</td>
</tr>
</tbody>
</table>

TABLE I

Features used in vehicle detection based on LiDAR

The authors of article [1] proposed an algorithm with main steps similar to those described in Section II, as well as filtration of high objects such as buildings. In their work [3], the authors used the ZedBoard development board with a Zynq device. However, the proposed algorithm has been implemented in large part on the ARM processor using the OpenCV library.
A vector of thirty features is calculated for each segment (Table I). The real size of a cell is the difference between the minimum and maximum value of a particular $x$, $y$, $z$ component in the cell.

The computed feature vector is the input to a linear SVM classifier. Logistic regression was used to convert its output from the distance from the hyperplane to the probability of belonging to the “car” or “not car” class. The complete flow of the proposed algorithm is depicted in Figure 1.

III. HARDWARE – SOFTWARE IMPLEMENTATION

A scheme of the proposed system is shown in Figure 2. The first stage is data acquisition. It was assumed that data from the LiDAR sensor is send from a PC and contains three coordinates: $x$, $y$, $z$, as well as reflection intensity. The data is divided into 1m x 1m cells. This is accomplished by setting a valid flag for points belonging to a given cell. Between successive cells a time interval is added and during the period the flag is not set. In addition to LiDAR data, the corresponding video data is transmitted. In the current version only for visualization purposes and in the future as the second data source for data fusion. Both data streams are combined into a single superimposed image and then the one is sent to the target platform via HDMI. There, the separation takes place again into video and LiDAR data streams.

The pre-processing stage has two main purposes. The first is to set a flag indicating whether a given cell should be deleted. It takes place when any of the cell is classified as ground, noise or background. The second is to determine the feature vector described in Table I for each cell.

At the segmentation stage, a cell is described by a flag indicating whether it is deleted and the feature vector listed in Table I.

Data aggregation into 3x3 segments is performed using BRAM memory. Cells are stored in a set order, so that the entire segment can be read easily. To calculate features for a segment, the set of values calculated for each not removed cell are used.

Finally the features are sent in a fixed order to the linear SVM classifier module. Its implementation is very simple – feature values multiplied by appropriate weights are accumulated and $bias$ is added to the result. SVM coefficients were obtained while training the classifier on a PC. The result is then transferred to the logistic regression module. This was done to convert the SVM output value into a probability measure. A sigmoidal function, whose coefficients were determined by logistic regression, was used. Its values were pre-computed and stored in a LUT. For visualization, segments classified as cars are projected onto the video stream. The appropriate projection matrix was obtained from the KITTI calibration data.

The system was implemented on a Zybo board with Zynq 7z010clg400-1 device. It uses 10012 LUTs, 18677 FF (about 50% of the available), 11 BRAM (18%) and 14 DSP (17%). So, there is still space left for adding more features and additional data processing.

IV. SUMMARY

Implementing the described algorithm in programmable logic allowed to obtain real-time processing, assuming that the LiDAR data is transmitted in Cartesian coordinates and pre-divided into cells. Currently the system processes 60 frames with included LiDAR data per second. Ultimately, the system should receive data in spherical coordinates and without cell division (in accordance with the typical data format from the LiDAR sensor). The classification accuracy is about 60-70 %. In order to improve it, additional features could be added, such as the $z$ coordinate histogram or moments of inertia. In addition, LiDAR data fusion with video data could also be used to improve detection efficiency. Another extension may include cluster aggregation so that each detected car is only displayed once. The solution can be used as part of a advanced driver assistance system or autonomous vehicle.

REFERENCES