TrimBot2020 Deliverable D4.3

Drivable garden regions

Principal Author: University of Amsterdam (UvA)
Contributors:
Dissemination: RE

Abstract. The aim of Deliverable 4.3 is to identify the drivability of the region immediately in front of the robot, including recognizing the various terrain types and static obstacles.

Deliverable due: Month 40

1 Drivable Regions

In the scope of the project, the drivable regions are defined to be either grass or pavement regions which are not blocked by any of the static garden objects such as trees, bushes, poles, etc.

The framework to identify drivable regions is implemented as part of the path planning module. It captures changes that have happened in the garden, e.g. re-positioned objects or new occurrences, terrains that prevent the robot from traversing, etc. To that end, it provides the changes to the sketch map module that in return will be updated accordingly.

The drivable regions module receives as input the semantic segmentation map (provided at `uva_semseg`) and fused disparity map (provided at `local_fusion_disparity`), and outputs a semantic point cloud indicating obstacles and non-traversible terrains.

2 Recognition of Drivable Regions

2.1 Garden Semantic Information

The input semantic map provides a semantic label for each pixel in an image captured by the robot's cameras. In particular, the semantic classes are defined by the reduced annotation set from TrimBot2020 workshop challenge 3DRMS [1] as follows:

0 Unknown

1 Grass

2 Ground

3 Pavement

4 Hedge

5 Topiary

6 Rose

7 Obstacle

8 Tree

9 Background

Among the defined classes, we consider the Grass and Pavement regions (labels 1 and 3) drivable, and the others non-drivable or obstacles.

2.2 Geometric Information

To prevent the robot from depending solely on the output of a single predicted module, which is error prone to less visible objects or intricate lighting conditions, we include an obstacle detection module using geometry information. To that end, we obtain depth maps for each image from the fused disparity module using the formula:

$$depth = \frac{baseline}{disparity}, \tag{1}$$

where baseline is the distance between 2 cameras in a stereo pair. We reconstruct the 3D coordinates for each pixel from the obtained depth maps and camera intrinsic parameters:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} . \tag{2}$$

To overcome the possible inaccuracies due to noise in depth maps, we fit a parametric surface to the ground plane, i.e. utilizing only pixels labeled as grass, pavement, and ground from semantic segmentation. A 2nd ordered quadratic function is employed to model the hill-valley shape of a usual ground as follows:

$$y = a_1x^2 + a_2z^2 + a_3xz + a_4x + a_5z + a_6, \tag{3}$$

where $a_i, i \in \{1..6\}$ are the surface coefficients, x, y, z are 3D coordinates of each point in the camera coordinate system, i.e. z-axis coincides with camera looking direction, x-axis to the right, and y-axis downward.

Note that the surface function is computed solely from the ground regions indicated in the semantic map. It models a surface expanding infinitely to both sides without taking into account the garden objects. Illustrations of the surface together with input RGB, semantic segmentation and depth map are shown in Figure 1.

Having the parametric surface fit to the ground, the obstacles can be identified as those higher than a given threshold to the fit surface. We set the threshold to 10 centimeters and mark the objects according to their heights. The results are shown in Figure 2.

2.3 Quantitative Evaluation

We measure the system performance based on pixel-wise semantic accuracy using 3 metrics: global accuracy, class-average, and mean intersection-over-union. The results are provided in Table 1 for both synthetic dataset and TrimBot2020's test images. The class-wise detailed results are shown in Table 2. We use TrimBot2020's WUR2 dataset for finetuning and testing purposes, following the convention provided in [1]. In general, the system recognizes drivable regions at 90.95% accuracy and 87.04% for non-drivable regions.

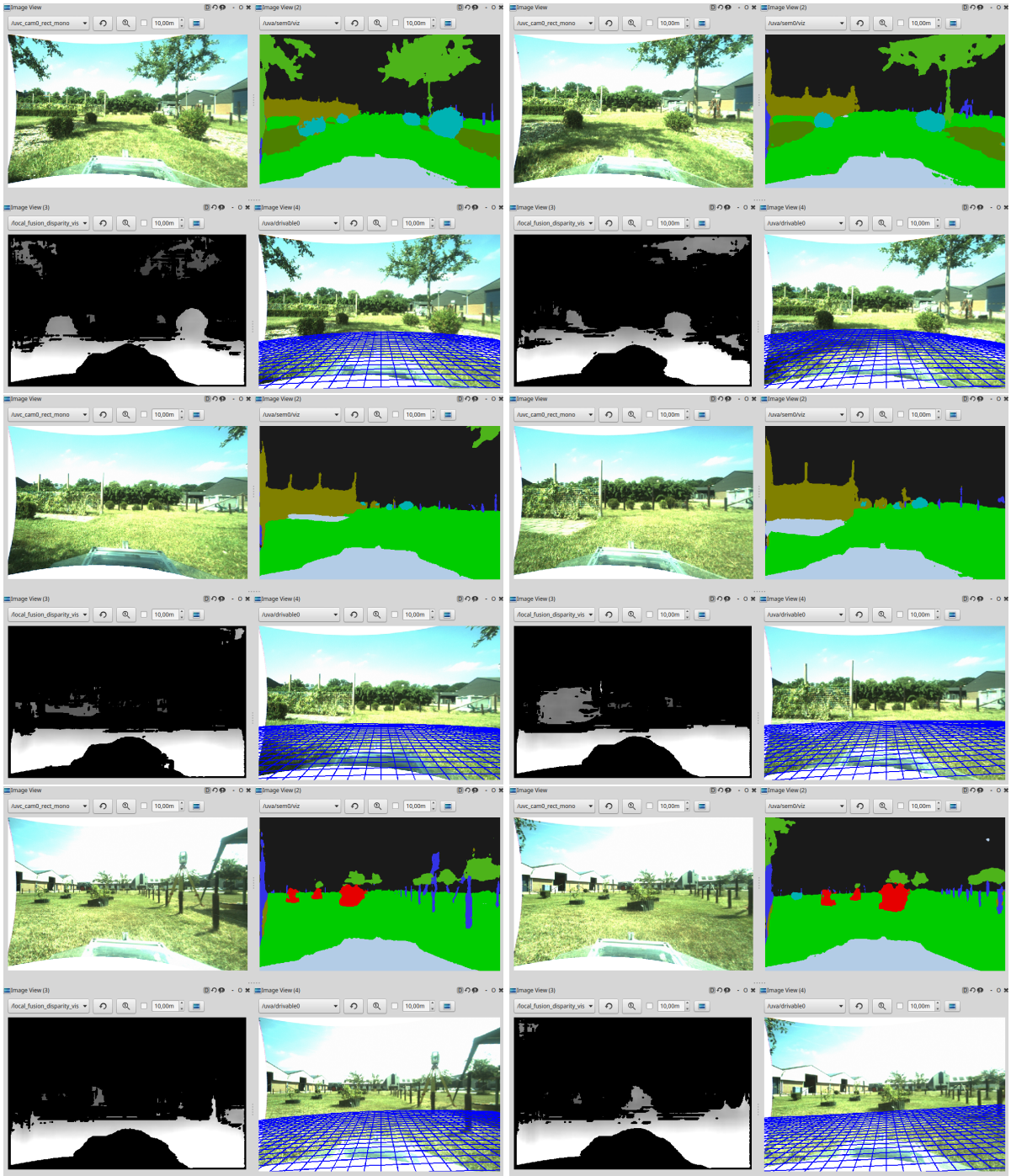

Figure 1: Screenshots from the surface-fitting module that runs as a ROS package. Each group composes of 4 screens, which are, from top-left corner in clockwise order, an RGB image captured from a robot's camera, semantic segmentation, a fit surface overlaying on the RGB image, and a depth map.

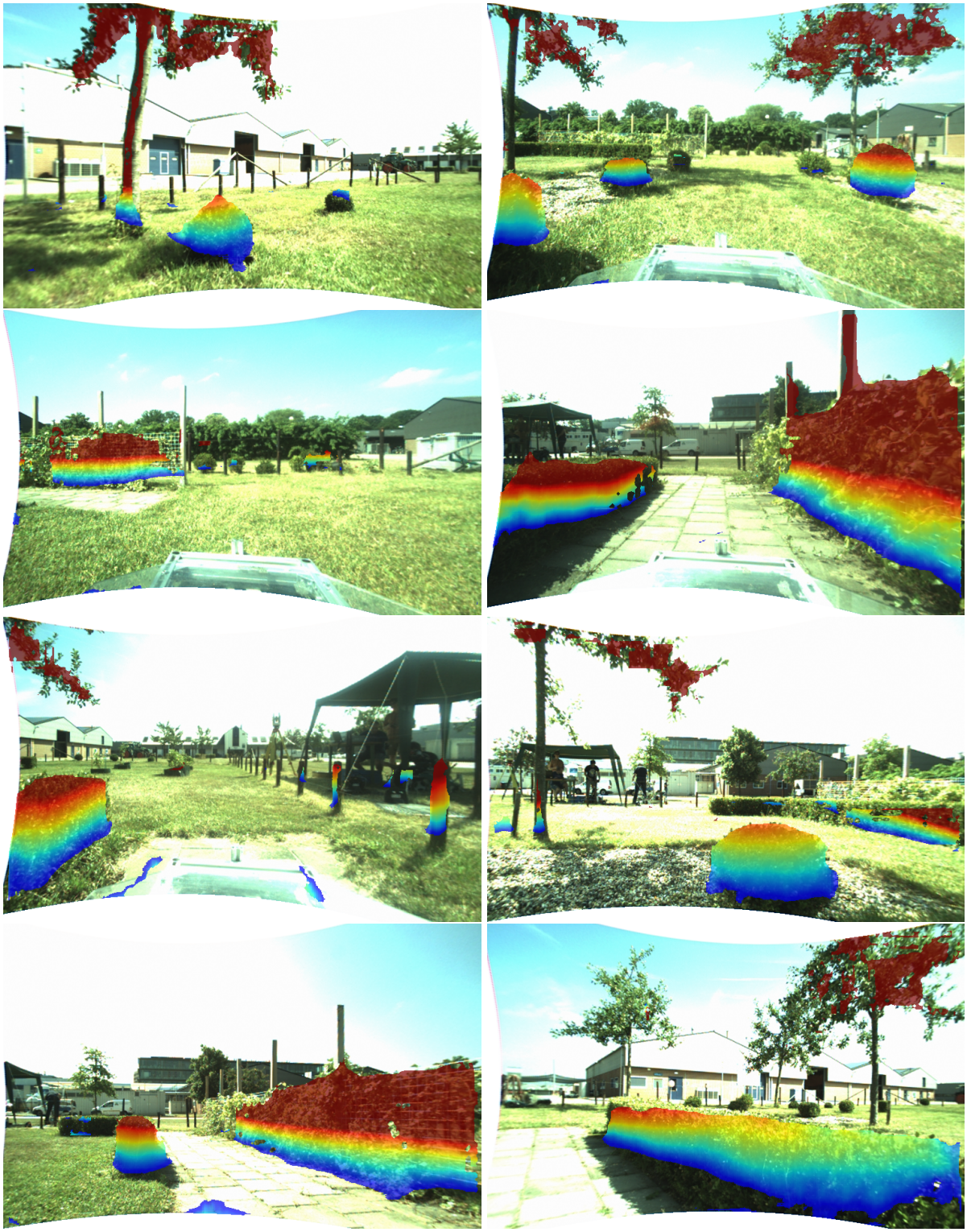

Figure 2: Obstacle detection by plane fitting: colors encode objects' heights above the ground.

	global	class-average	mIOU
synthetic data	97.07%	94.18 %	89.58%
real data	94.28%	88.01 %	81.54%

Table 1: Accuracy of semantic labels on both synthetic images and TrimBot2020's recorded images (test_around_garden sequence of the WUR2 dataset)

drivable		non-drivable					
grass	pavement	ground	topiary	hedge	rose	trees	other obstacles
96.33%	85.56%	90.25%	91.38%	91.27%	75.36%	81.58%	92.40%

Table 2: Detailed accuracy for each class for real data test. The mean accuracy for drivable and non-drivable regions are 90.95% and 87.04%, respectively

2.4 Non-drivable Point Cloud

We merge the obstacles and non-drivable terrains detected from semantic segmentation module into one single point cloud of the scene. The non-drivable point clouds are published and passed to the sketch map module for updating. Some examples are provided in Figure 3. Note the absence of the drivable regions, such as grass and pavement, and the presence of only garden objects and non-drivable terrains, such as pebble stones and gravels.

3 Qualitative on Integrated System

Figure 4 and 5 show some screenshots from the obstacle detection module in the integrated real-time system. An illustration of the front grayscale camera is shown together with the resulting semantic segmentation on the lower left windows, while the point cloud of obstacles is shown on the right window. Note that the grayscale image is only for visual inspection and not being used as input for the semantic segmentation module, the real input is rectified RGB images. Note also the absence of drivable regions such as grass and pavement in the obstacle point clouds and the presence of only obstacles and non-drivable regions such as gravel or pebble stones.

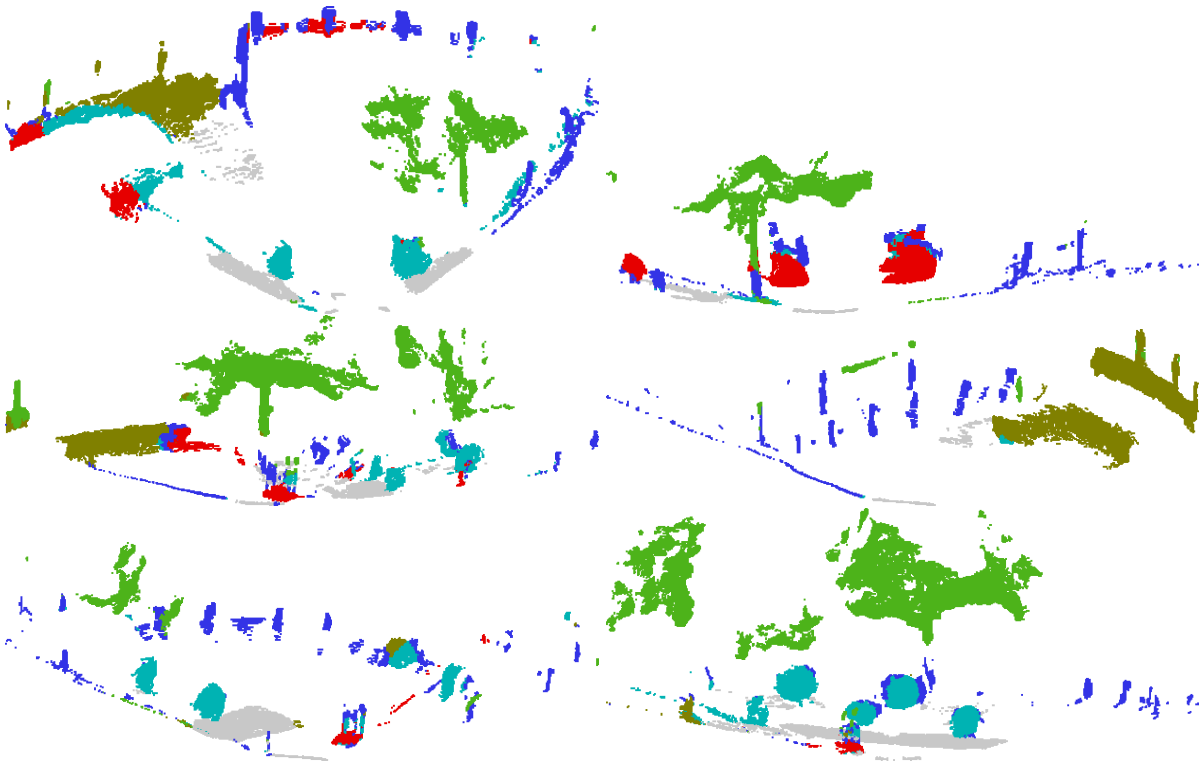

Figure 3: Non-drivable point clouds using TrimBot2020 ground truth depth images, only obstacles and non-drivable terrains are visible: pebble stones (grey), rose (red), fence (blue), topiary (cyan), hedge (yellow), tree (green)

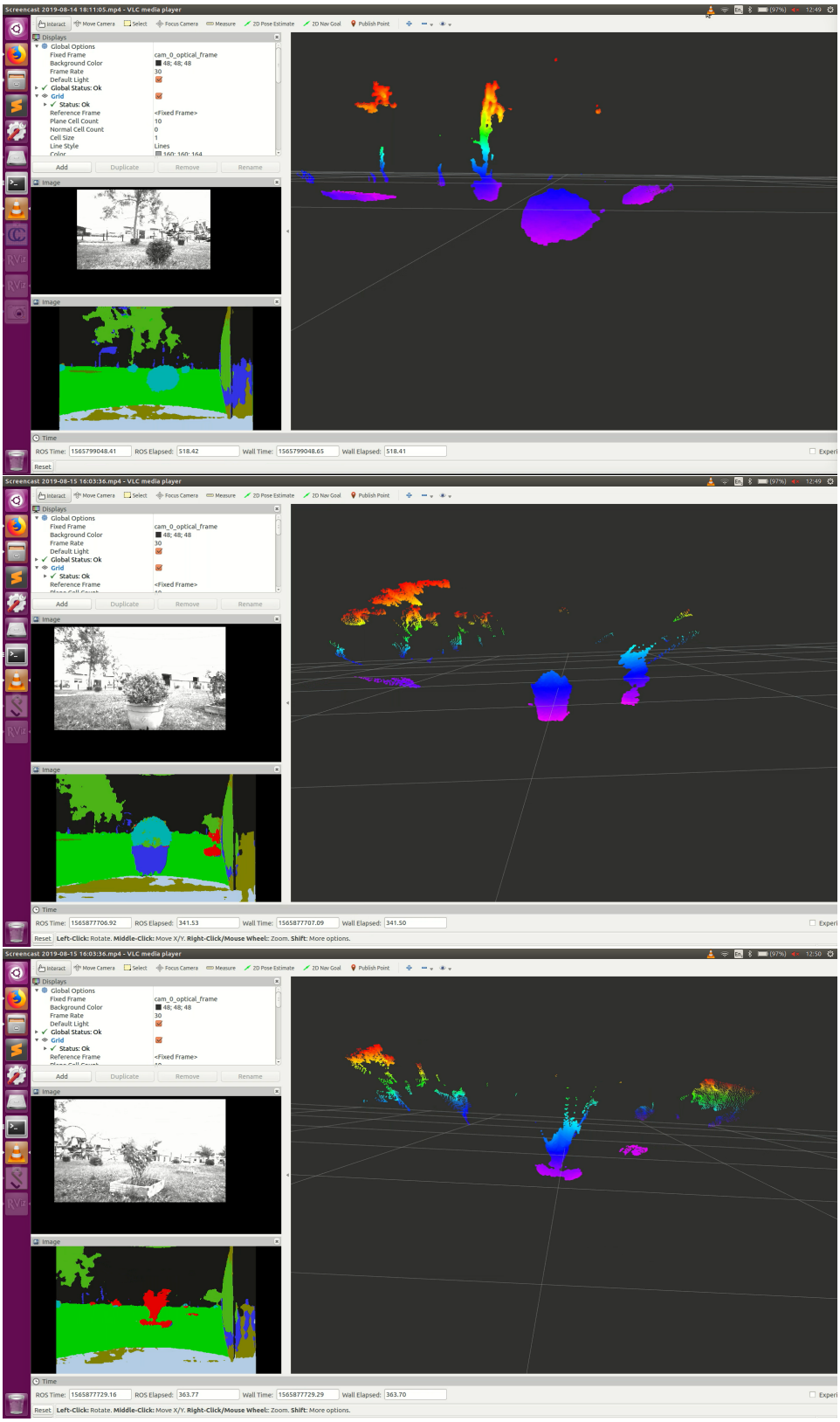

Figure 4: Obstacle detection module in the integrated system. Top-down left-right: system information, output of grayscale front camera, semantic segmentation, obstacle point-clouds. Note the absence of drivable regions such as grass and pavement

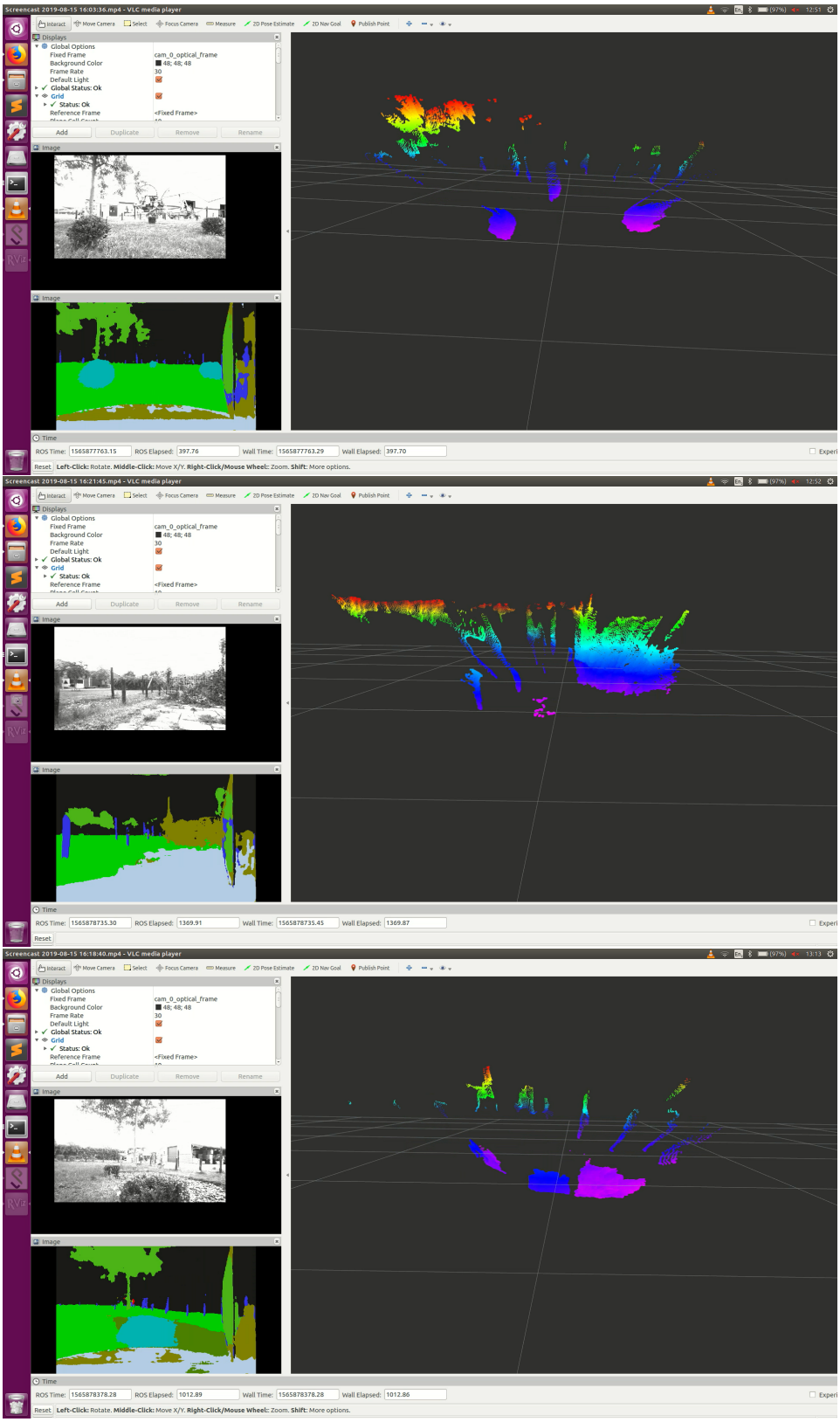

Figure 5: Obstacle detection module in the integrated system. Top-down left-right: system information, output of grayscale front camera, semantic segmentation, obstacle point-clouds. Note the absence of drivable regions such as grass and pavement

References

[1] Radim Tylecek, Torsten Sattler, Hoang-An Le, Thomas Brox, Marc Pollefeys, Robert B. Fisher, and Theo Gevers. The second workshop on 3d reconstruction meets semantics: Challenge results discussion. In *Computer Vision – ECCV 2018 Workshops*, pages 631–644. Springer International Publishing, 2019.

