Vehicle navigation: transverse a corridor with a computer vision system.

Alejandro D. J. Gomez F.
ajgomezf@unal.edu.co

Motivation

A development platform based on an RC-Car, a Jetson Nano, and a web camera, has been built to create computer vision applications with a self-driving approach. It is expected with this project to be able to create applications based on computer vision that allows processing images captured by the camera at a $20Hz$ frame rate with a resolution of 320x480 pixels to allow small cars scale of 1:10 to navigate indoor environments. A picture of the platform built is shown in **Figure 1**. The car's computer supports computer vision applications thanks to the *Ubuntu OS*, which allows runs programs made in Python with the *OpenCV* library and interacts with the steering and throttle actuators in the car, through *I2C* commands.

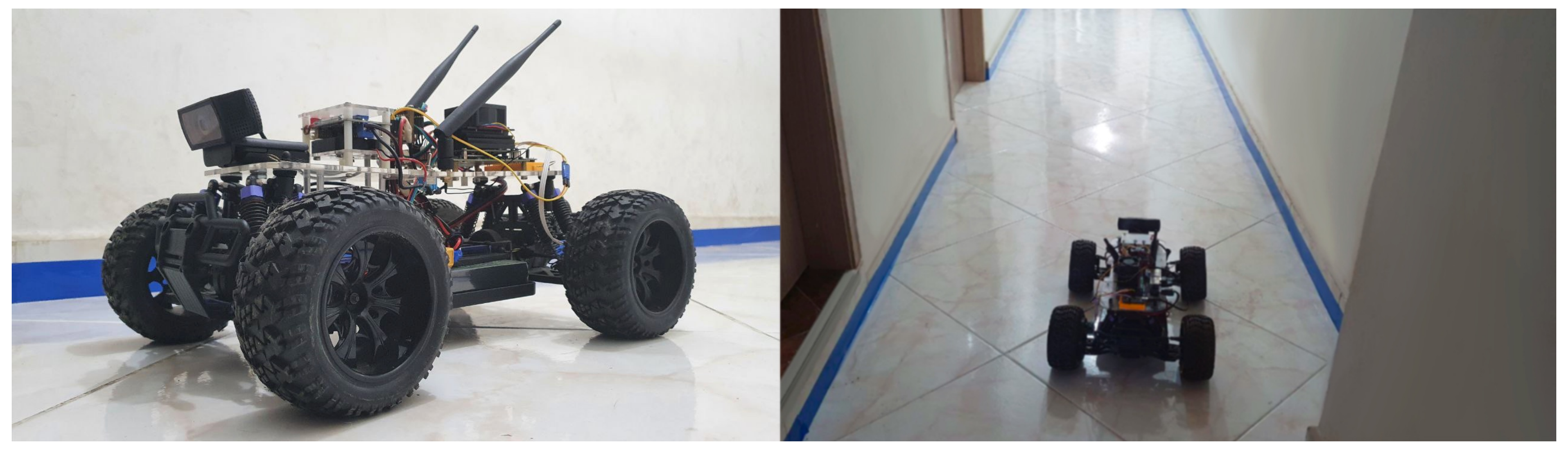


Figure 1. Self-driving car platform.

To validate the vehicle functionality, a computer vision application running inside of the car should be able to identify obstacles, and take actions. The environment selected for this problem is the corridor inside of the author's house.

General problem

The steering of the chosen car is not centered precisely. For that reason, it never moves in a direct line. Therefore, the vehicle requires a feedback system that allows moving safely and avoiding an obstacle for autonomous navigation. Such feedback system is also defined as the perception system, and for this context, it refers to the camera that provides images about the environment where the car moves.

For a self-driving challenge for indoor navigation, a corridor provides a path where the car can move. The car starts at one end, and must arrive at the other end without human intervention and without crash with the walls. **Figure 2** shows the environment dimensions and the car steering issue with a random direction.

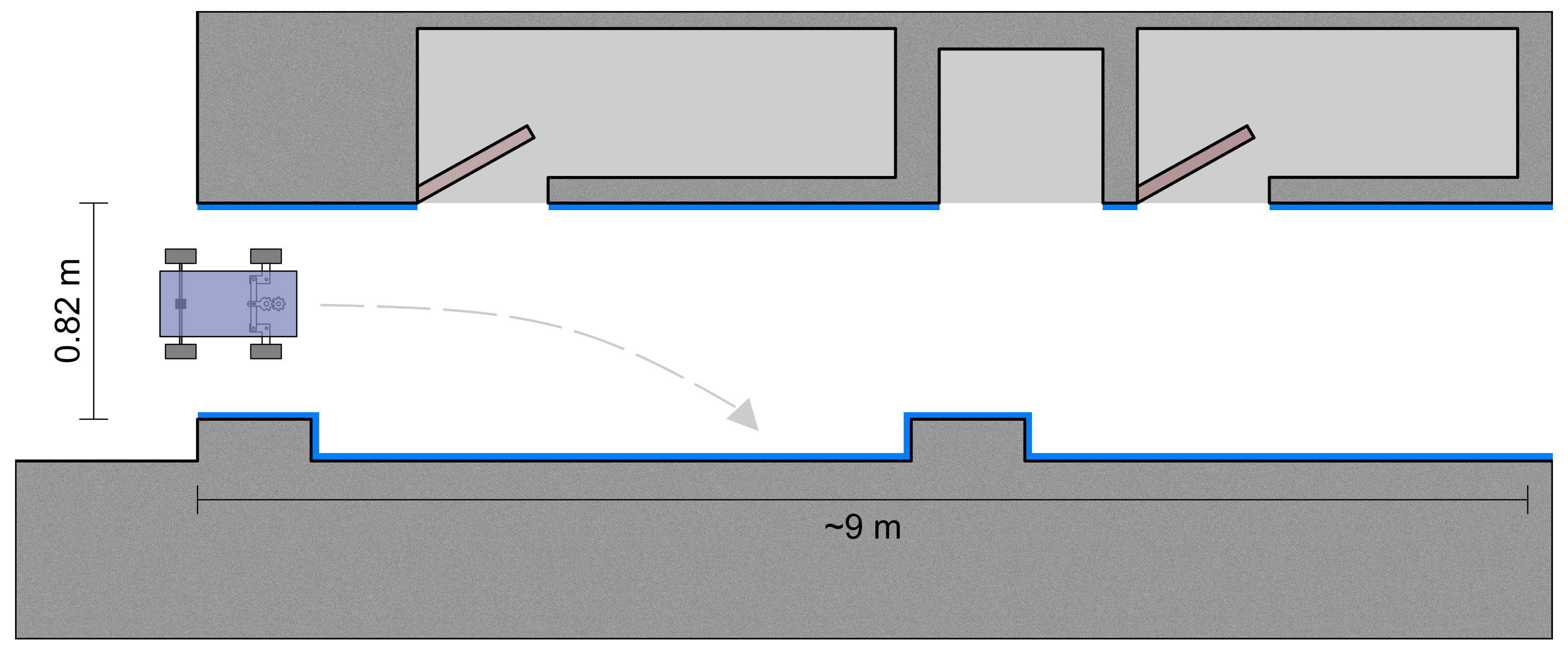


Figure 2. Car starts at one end of the corridor and must arrive at the other end without human intervention.

Specific problem

The car's computer needs a computer vision program that helps to avoid touching the walls and maintain the car in the center of the corridor as part of the perception system which helps the car navigate in autonomous mode to traverse the length of the corridor. The web camera integrated must be the main sensor. With a fixed throttle, the steering depends on the information processed from the computer vision program.

To help the car's perception system, a blue line is added to delimit the wall and the floor. **Figure 3** shows some pictures taken from the vehicle's camera.

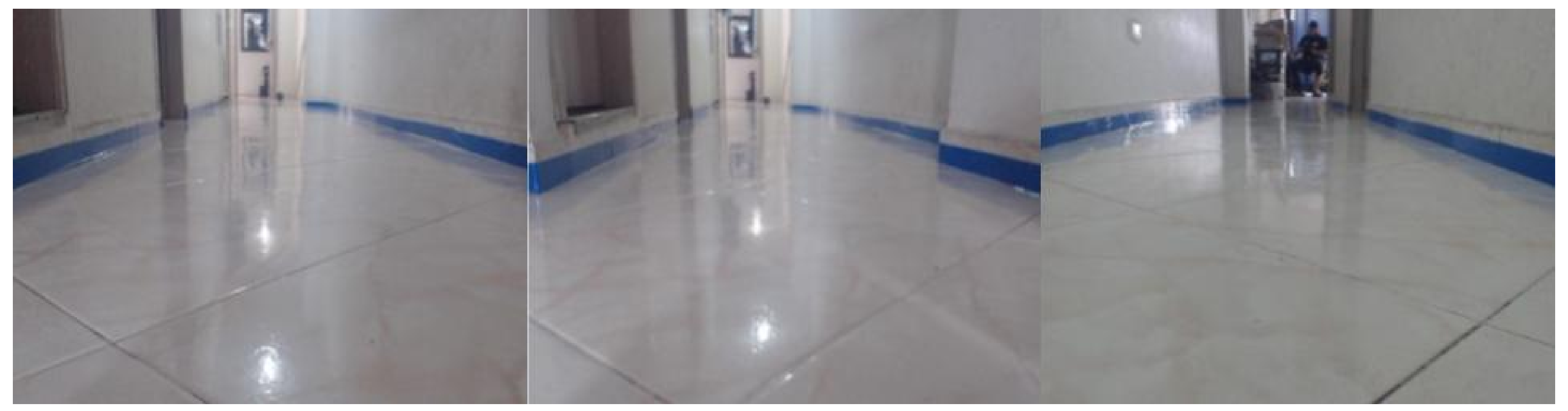


Figure 3. Pictures taken from car's camera.

Objectives

The following are the objectives set based on the environment chosen and the car's specs.

- Using the car's web camera for data acquisition.
- Creating a computer vision strategy that allows detecting the lines visible on the images and extract a measured value that allows to car take decisions to transverse the corridor.
- Using $50ms$ as the maximum processing time, according to the image frequency capture set on $20Hz$.
- Exporting the image processing pipeline to the car's computer and validate the behavior on it.

Previous work

A line detection pipeline was developed to detect the lines with the HSV color space and applying a change of perspective approach combined with a binary operation to the S channel. An example of the result is shown in **Figure 4**. The total processing time was `0.007` $\pm$`0.001` s.

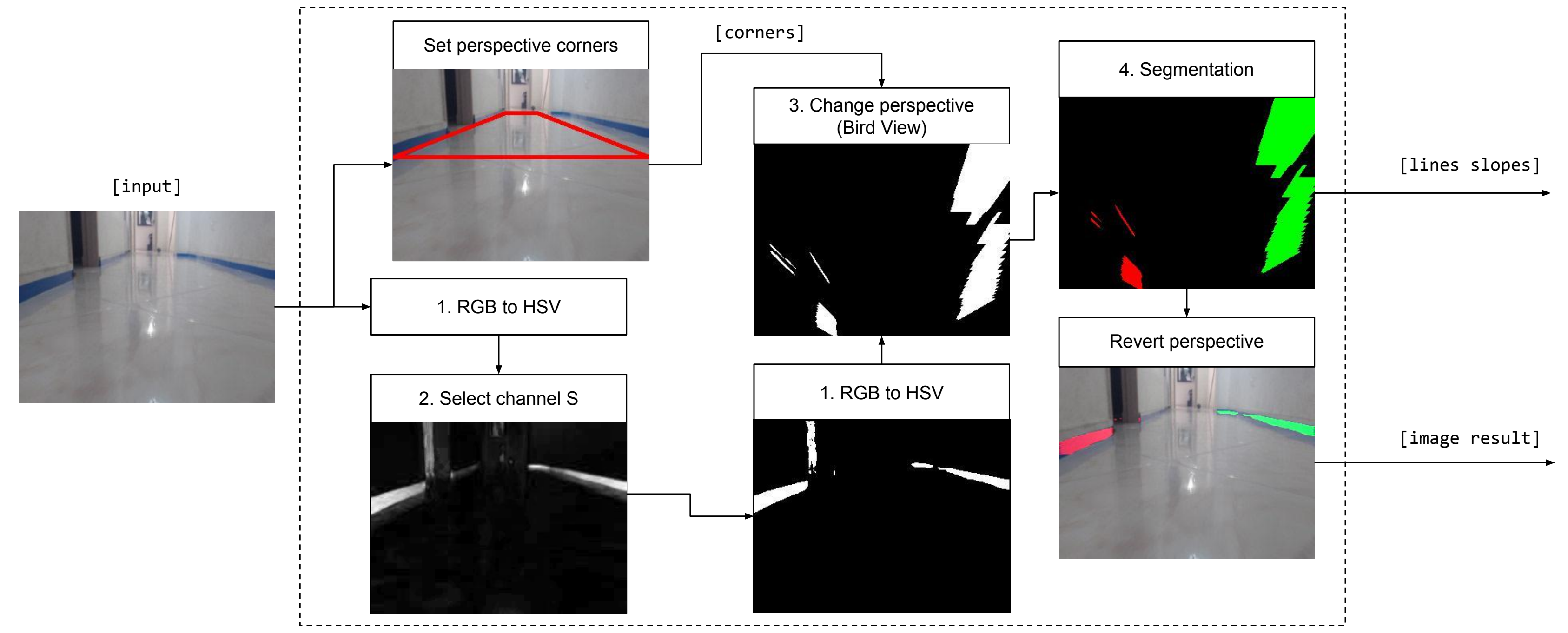


Figure 4. Line detection segmentation pipeline.

Contribution

Based on the braitenberg vehicles [1] a new computer vision pipeline was developed, to simplify the solution for line detection. The braitenberg agent reacts with the values detected by the sensors as it is illustrated in **Figure 5**. Two windows in the image will acts as separate sensors as it is shown in **Figure 6**.

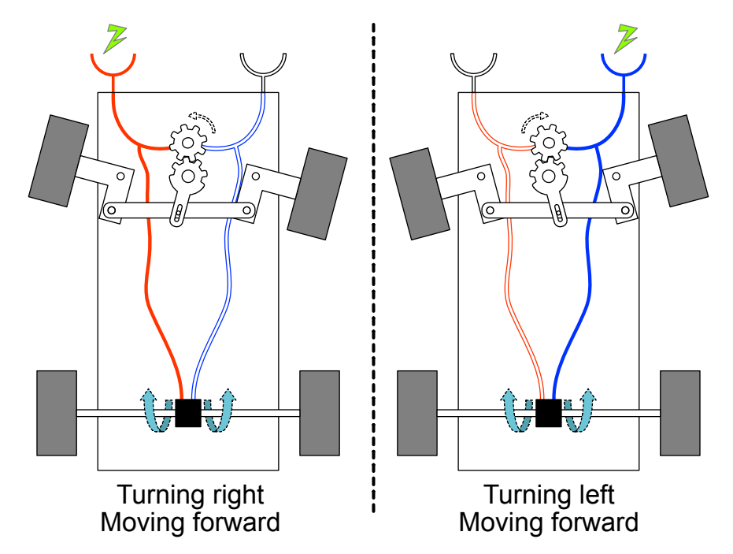


Figure 5. Braitenberg vehicle adapted to the car platform.

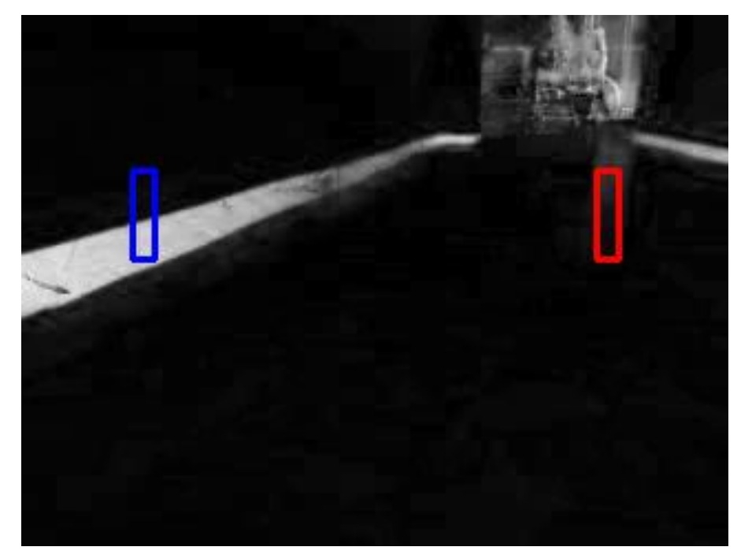


Figure 6. Chosen left and right windows on the image to be processed.

The pipeline created for the braitenberg approach is shown in **Figure 7**. The output of this pipeline is a sum of the binary pixels on each window. Those values will be processed by a simple agent created in the car to generate the commands to move in the selected environment. Processing time for this approach was `0.001` $\pm$`0.001` s.

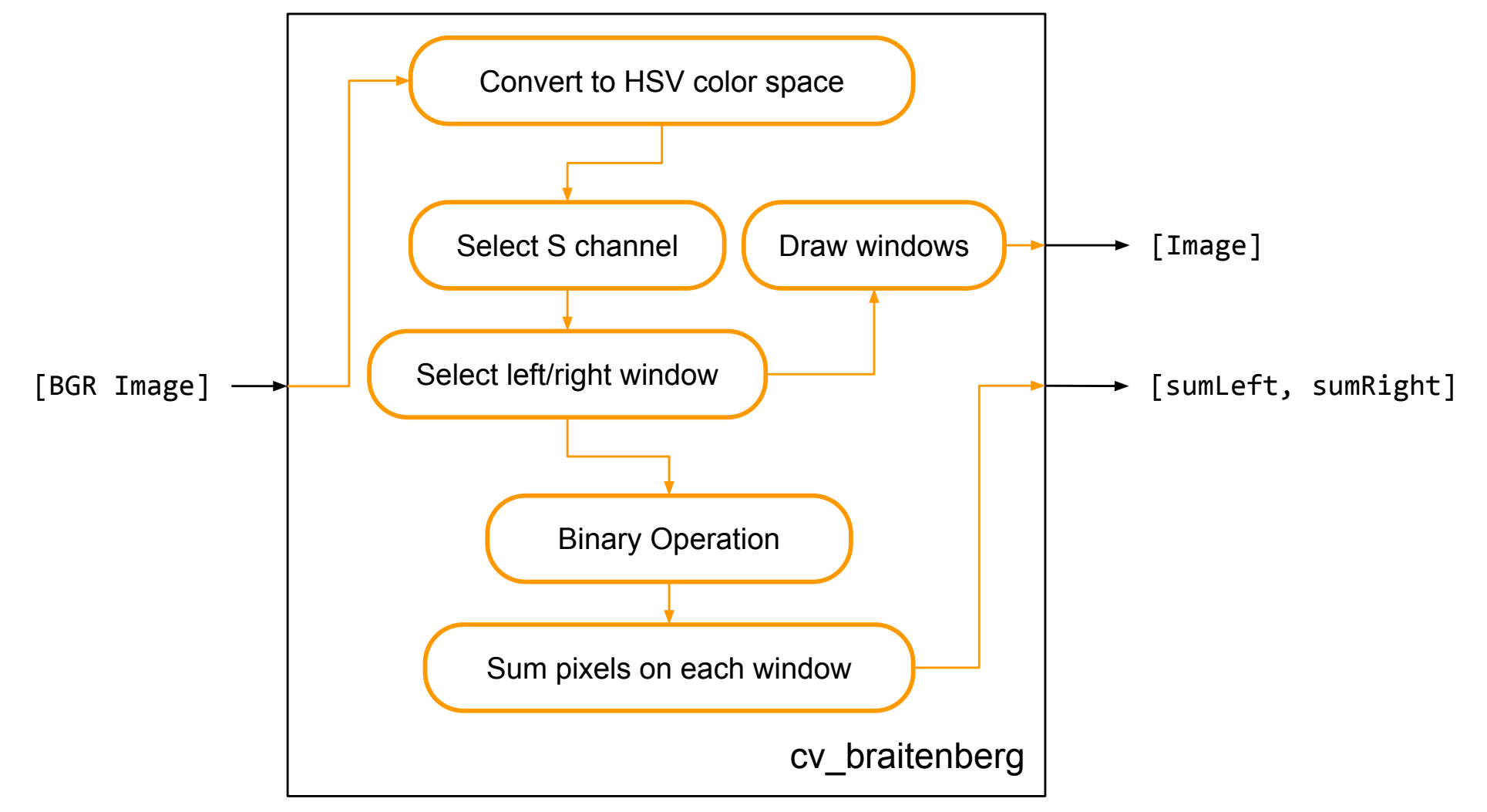
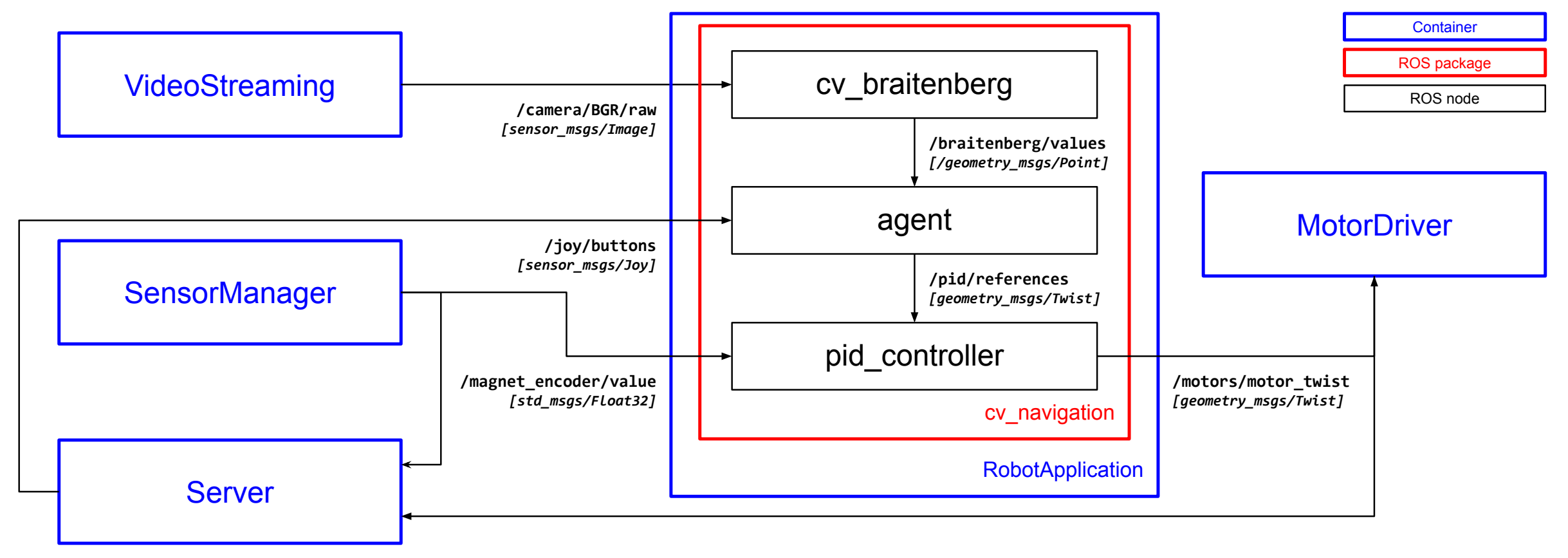


Figure 7. Pipeline created for the braitenberg approach.

A Docker-Ros architecture presented on **Figure 8** was used for its implementation.


VideoStreaming
SensorManager
Server
cv_braitenberg
agent
pid_controller
cv_navigation
RobotApplication
MotorDriver
/camera/BGR/raw [sensor_msgs/Image]
/braitenberg/values [/geometry_msgs/Point]
/joy/buttons [sensor_msgs/Joy]
/pid/references [geometry_msgs/Twist]
/magnet_encoder/value [std_msgs/Float32]
/motors/motor_twist [geometry_msgs/Twist]
Container
ROS package
ROS node
Figure 8. Application architecture implemented on the Jetson nano car's computer.

References

[1] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*, ser. Bradford Books. MIT Press, 1986. [Online]. Available: https://books.google.com.co/books?id=7KkUAT_q_sQC

[2] M. R. Endsley, "Autonomous driving systems: A preliminary naturalistic study of the tesla model s," *Journal of Cognitive Engineering and Decision Making*, vol. 11, no. 3, pp. 225–238, 2017. [Online]. Available: https://doi.org/10.1177/1555343417695197