Novel Implementation of a Line Follower with a Smartphone for the RoboCupJunior Rescue Secondary Category

Pedro Sales, Guillermo Trinidad

Montevideo, Uruguay

Abstract. RoboCupJunior is an international robotics event which brings students from all ages to compete in different challenges, one of which is Rescue Line. To solve the task, a line following robot is needed which is normally implemented using discrete light sensors and a microcontroller. We made a radical change to this by adding a smartphone to our robot, using the camera to follow the line allowing for more robust and efficient line following.

1 Task

The RoboCupJunior 2015 rescue line secondary challenge consisted of a line-following rescue robot, able to navigate a complex line featuring gaps, intersections, ramps and obstacles. An evacuation room is placed at the end of the line, where the robot has to rescue victims, represented by a 5cm diameter ball, by taking them to an evacuation zone situated at one of the room's corners. [5]. In figure 1 one example of arena is shown.

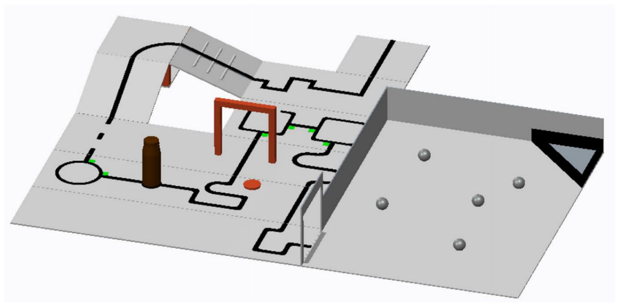

Fig. 1. Rescue line example arena.

Making a traditional line-following robot was the initial idea for solving this task, however this approach had various disadvantages. Firstly, the robot would require an array of line sensors, which added to the mechanical and electronic complexity of the robot, making it prone to breaking. These sensors would not be enough, as green dots next to the line have to be detected in order to negotiate intersections. Secondly, the information gathered from the line will be limited to the number of sensors employed, limiting the robot's capability anticipate the path it will have to follow next. Finally, a traditional line-following robot was used by the team for a similar task the previous year, so merely making a variation in 2015 would not be as enriching as developing a new solution from scratch.

2 Proposed Solution

Our solution consists of a novel line-following robot which tracks the line with a camera rather than an array of line sensors. This allows to sense the upcoming section of the line, making the robot's path at intersections and 90 degree turns smoother and shorter, thereby more efficient.

Since image processing requires great amounts of computational power, using only a microcontroller was not possible. Instead, an Android phone was chosen as it provides a high resolution camera together with a powerful processor and wireless communication. In terms of software, an android version of the open source computer vision library OpenCV [4], was used. The library provides implementations of image filters which can be used to clean the image and process it, allowing to extract the information needed to follow the line. The image processing line following algorithm was packed into an app which then communicated with the robot via bluetooth, sending the necessary motor speeds to stay in the line. Low level communication with the robot's hardware was done by an Arduino microcontroller, which merely relayed the motor speeds to the Dynamixel AX12 [6] motors when following the line. In the figure 2 the final configuration of the robot is shown.

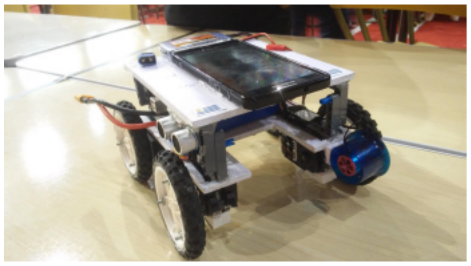

Fig. 2. Robot used in the competition.

One of the biggest physical challenges this approaRoboCupch had was uniformly illuminating the line, eliminating all reflections and outside sources. The phone's flashlight was used but it created reflections on the black line which made it impossible to distinguish parts of the black line from the white background. To solve this, cross light polarization was used. A light polarizing filter was placed on the flashlight and another one rotated 90 degrees in the camera.

This is based on the fact that light can not go through polarizing filters placed 90 degrees to each other, consequently all specular reflections, which maintain polarization, are eliminated. Diffuse reflections however are conserved, so light from other parts of the floor can go through and are perceived by the camera. [1]

2.1 Image processing3

The image processing algorithm consisted of three main steps: filtering the image to have a clean representation of the line, analyzing the resulting image to find the line's edges and direction, and calculating the required motor speeds required to correct for the current deviation.

Filtering First, a gaussian blur is applied to the image in order to eliminate dust particles and small black spots in the arena floor. Then, the image is binarized and inverted of the image, which first sets the pixels to either black or white and then inverts it, making the line white and the background black. This is followed by two morphological operations, an opening to eliminate small black spots (now white) outside of the line, and a closing operation to join small, unintentional gaps in the line. [3]

Analyzing After the image was cleaned, the lower edges of line are found by detecting rising and falling edges in the pixel values of the bottom row. Having found the line, circles are drawn on those points, and then the next points of the line are found by going over the perimeter of those circles, detecting the line edges. This process is repeated several times until enough points of the line are found. This is a similar process to the one another one used by another student in a simpler version of this challenge, employing a Raspberry Pi instead of a phone [2]. The figure 3 shows our application running.

Calculating With the line's points found, the change of the line's angle at each of the points is calculated. These angles are then added, giving more weight to the ones corresponding to the lower parts of the line and less to the upper parts of the line. This allows to correct the necessary amount to follow the line at the current point and also start correcting for the upcoming parts of the line. The motor speed is then calculated from the weighted sum of the angles together with the line offset from the center.

2.2 Actuating

Having calculated the desired motor speeds, they are sent to the Arduino board via bluetooth using a simple, two way communication protocol developed by

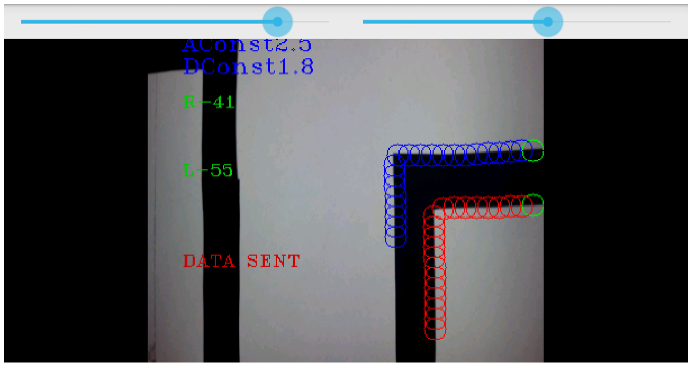
AConst2.5
DConst1.8
R -41
L -55
DATA SENT

Fig. 3. Application screenshot. Blue and red circles used to measure angles and line offset.

the team. It mainly consists of a sync byte, followed by an opcode and parameters and ended with an end byte. Thanks to the sync and end byte, when the communication is suddenly stopped or gets corrupted, it can be properly resumed once the connection gets re-established. The Arduino board then relays the information to the AX12 smart robot actuators using a half duplex serial interface

3 Results

The result was a robust line-following robot, capable of following every part of the line path in the Rescue Line category. The major advantage when compared to traditional line-following robots is the ability to see the upcoming path and make corrections in accordance to it. We ended up in 8th in the 2015 international competition in China, being the only team to use this method. [7]

This robot also serves as a proof of concept for an Android phone as a robot's main processor, opening the possibility to robots with complex behaviour. Android phones are very common and are readily available, making them a cheap, more powerful alternative to other setups. They include together with a touch-screen and a powerful processor, a variety of sensors and wireless radios, all in one small package.

4 Future Work

The algorithm could still be improved by taking into account the position of the line from the previous frames, which could help in the rare event of line loss. Another point to improve is the wireless communication, as in the competition some issues arose when connecting the phone to the Arduino's bluetooth module.

References

1. Polarization for Art Reproduction Archetype. http://www.betterlight.com/downloads/. Visited in may 2016.
2. An Image Processing Robot for Robocup Junior. https://www.raspberrypi.org/blog/an-image-processing-robot-for-robocup-junior/. Visited in june 2016.
3. morphological transformations OpenCV. http://opencv-python-tutroals.readthedocs.io/en/latest. visited june 2016.
4. OpenCV website OpenCV for Android. http://opencv.org/platforms/android.html. Visited on may 2016.
5. Robocup Junior 2015 Rescue Line rules. http://rcj.robocup.org/rcj2015/. Visited in june 2016.
6. Robotis Dynamixel AX12 smart servo. http://support.robotis.com/en/product/dynamixel/. Visited in june 2016.
7. together with video of robot in action Summary of team participation in the Robocup Event. https://www.youtube.com/channel/uc6h7vchoqckzifhfaftdlmq "video presentacion rou-bot". Visited june 2016.