ARTIFICIAL VISION ROVER

TEAM SPAZIO

ALLEGRETTI GIOVANNI
DI DOMENICO FABRIZIO
DI PRISCO GIOVANNI
SCHETTINI MARCO

Syllabus

1. Project requirements

The dictated requirements of the project concern both the hardware and the software, in fact it was expressly indicated not to modify the physical structure of the Rover as much as a software structure that included the programming interface necessary to move the machine. Other specifications have been provided on the route to be followed with the related competition regulation.

1.1 Rover hardware

The Rover uses:

- The **Intel® RealSense™ D435i** offers the widest field of view of all our cameras, along with a global shutter on the depth sensor that is ideal for fast moving applications. The Intel® RealSense™ depth camera D435i is a stereo tracking solution, offering quality depth for a variety of applications. It's wide field of view is perfect for applications such as robotics or augmented and virtual reality, where seeing as much of the scene as possible is vitally important. With a range up to 10m, this small form factor camera can be integrated into any solution with ease and comes complete with our Intel RealSense SDK 2.0 and cross-platform support.
- **NVIDIA® Jetson Nano™** is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts.
- **STM32 Nucleo-64 development board with STM32F401RE MCU** provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller.
- 2 x **Sabertooth dual 5A motor driver:** Sabertooth 2X5 is the motor driver of choice for small differential-drive robots. It is ideal for smaller robots- up to 3lbs in combat or 25 lbs for general purpose use. It is also a great choice in any application where you need to control two brushed motors.

- 4 x **19:1 Metal Gearmotor 37Dx52L mm with 64 CPR Encoder**. This gearmotor is a powerful 12V brushed DC motor with a 18.75:1 metal gearbox and an integrated quadrature encoder that provides a resolution of 64 counts per revolution of the motor shaft, which corresponds to 1200 counts per revolution of the gearbox's output shaft. These units have a 0.61"-long, 6 mm-diameter D-shaped output shaft.
- Complement tools: **craft shield for STM32F401RE**, **2200mAh 12v Li-ion battery**, **12V a 5V / 4A step-down voltage regulator**, **Noctua 5v fan**, **switch** and **button**.

1.2 Rover software structure

The provided programming interface includes the primitives needed to set the linear speed and angular speed of the rover. This interface is present in Python scripts processed on Nvidia Jetson Nano which communicates with the Nucleo-64 board on which an ad-hoc developed firmware is present.
The structure of the pre-existing software includes a three-node ROS architecture:

- **realsense2_camera**: proprietary node of the SDK for Intel® RealSense ™ which publishes on the topics defined by documentation to which it is possible to subscribe to obtain, among much information, image and depth.
- **mivia_rover**: node responsible for publishing the information necessary for movement.
- **my_controller**: node containing the business logic of the Rover movement control.

The project specifications require that only a script be modified in which it is possible to use the information coming from **Intel® RealSense™ D435i**, a classification algorithm for artificial vision and therefore manage the movement of the Rover.

1.3 Rules

The route to be followed requires the Rover to pass into numbered doors, thus respecting their order. Between the doors there are totems accompanied by mandatory signs and a sign indicating the degrees in which the next door will be present.

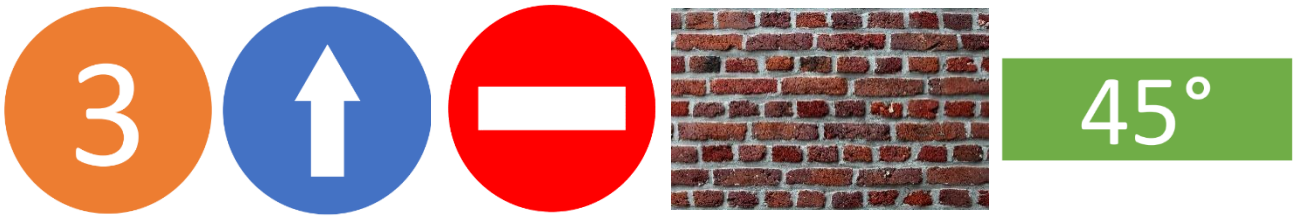

Figure 1 : Types of signs

Alternatively, there may be an obstacle with the texture of a wall. In addition, minimum measurements were provided between the various components of the route.

2. High-level design

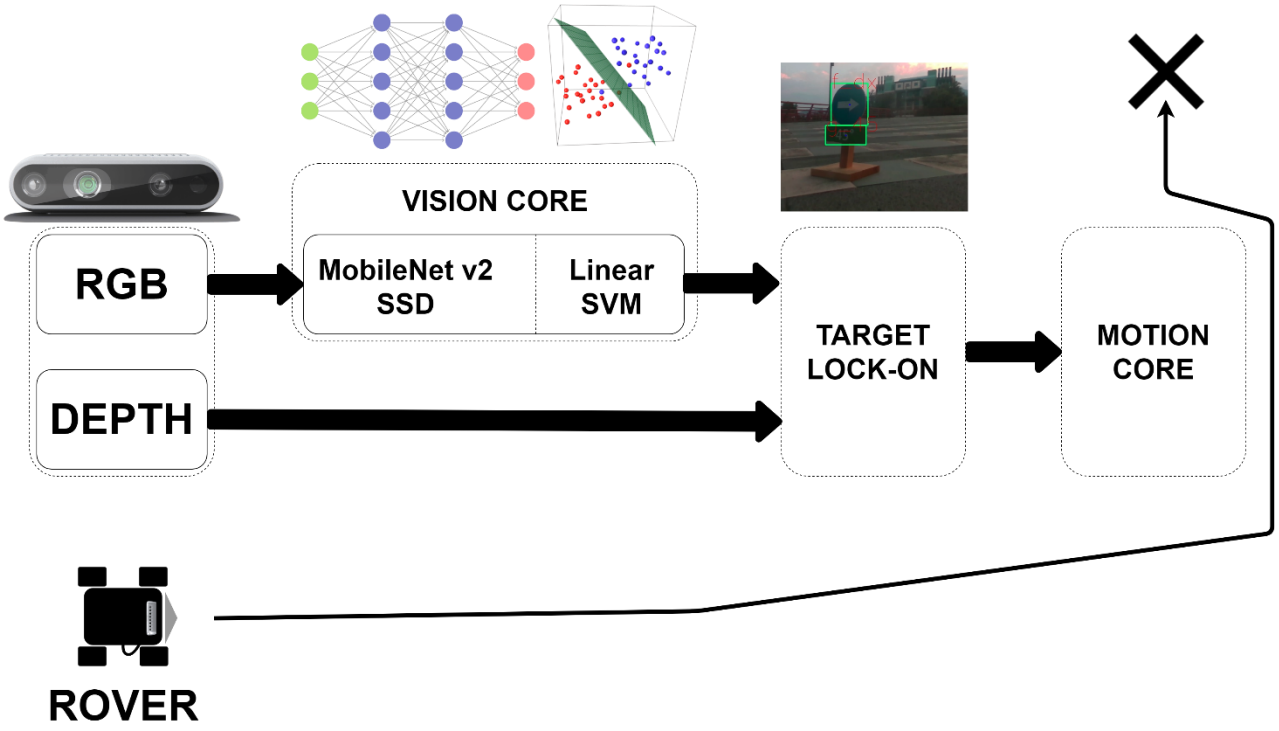

Figure 2 : High-level design

The high-level architecture can be described through separate modules:

- **Vision core.** The module takes care of converting the acquired camera image into useful information, identifying the target of interest through a deep neural network (MobileNet v2 SSD - COCO pre-trained dataset) and a linear SVM in cascade. The reasons for this choice will be explained in the next chapter.
- **Target lock-on.** The module basically deals with choosing, among the objects identified by the upstream module, the target of interest. To make the choice we also use depth information in order to discriminate objects based on their distance. It also uses a logic that has been implemented ad-hoc from the specifications of the demonstration path presented for this project.
- **Motion core.** The module deals with the movement of the Rover according to the chosen objective. It also uses the depth information obtained from the **Intel® RealSense™ D435i** to make, depending on the type of target, the appropriate corrections based on the position and inclination of the object within the visual frame.

3. Vision core

3.1 MobileNet v2 SSD - COCO dataset pre-trained

Since the object detector used is MobileNet v2 SSD, this means that the backbone used as feature extractor is the deep neural network-based classifier MobileNet v2. The power of this object detector is that the same features extracted to perform the classification are also used to perform the detection of the objects in the image. Below is the list of the various layers that make up this feature-extractor:

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 3 : MobileNet v2 SSD layers

About the Single-Shot-Detection technique, the image is divided into a grid with a certain number of cells, on each cell different portions of the image are considered, at different scales and with different aspect-ratio.
Below is an example architecture of deep neural network-based detector, which use the Single-Shot-Detection technique:

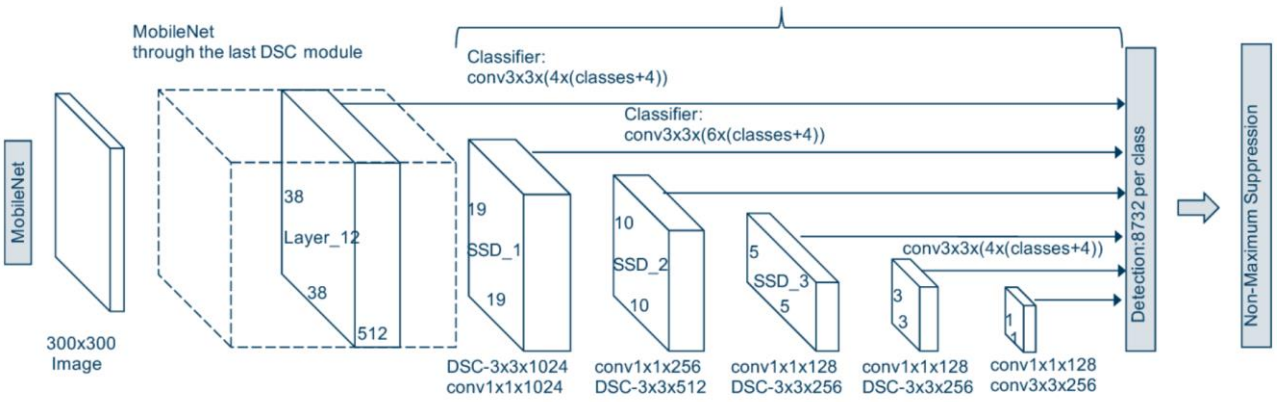

Figure 4 : Example of MobileNet v2 SSD use

To reduce the number of samples for the training data set, a pre-trained network with the COCO data set was executed. Among the same pre-trained MobileNet networks, the one with COCO was chosen for the compromise between the number of positive samples (1.5 million) and the not excessive number of classes (91 categories of products).

The fine-tuning of this deep neural network was performed using the Object Detection API of Python's TensorFlow framework. In particular, the training was performed on 12000 iterations and a batch size of 24. So, each epoch is composed of 15899/24 iterations = 662 iterations, for a total of 12000/662 ≅ 18 epochs, with a constant learning rate equal to 0.004.

The samples are subjected to a data augmentation automatically during the first phase of fine-tuning. The exploited properties are:

- Random adjust brightness (max delta: 0.3)
- Random adjust contrast (min delta: 0.8, max delta: 1.25)
- Random adjust hue (max delta: 0.01)
- Random adjust saturation (min delta: 0.8, max delta: 1.25)

Obviously, the option concerning the rotation is not used in order to discriminate the direction of the arrows on mandatory signs.

So, during the performed fine-tuning, 6 different grid layers were used for which to create the anchors, with a scale factor for the anchors ranging from 0.2 (which corresponds to the finest resolution) to 0.95 (which corresponds to the coarsest resolution). In addition, the following 5 different aspect ratios were used for each cell: 1:1, 2:1, 3:1, 1:2, 1:3.
Finally, the network takes images at 300x300 pixels resolution, then an image resize operation is previously performed, to bring the image back to the desired resolution, inserting a uniform background for images with an aspect ratio different from 1.

This network, for each input image, outputs a tensor containing 4 elements:
• The first element indicates the number of objects detected in the image.
• The second element indicates, for each object detected in the image, the *objectness* relative to the detected object (it indicates how likely it is to be an object).
• The third element indicates, for each object detected in the image, the bounding box relative to the detected object (therefore it contains the 4 elements that will indicate the coordinates of the Top-Left and Bottom-Right points of the bounding box).
• The fourth element indicates, for each object detected in the image, the class identifier relative to the detected object.

The neural network is used to classify 7 objects:

- Number of door sign
- Straight direction mandatory sign
- Right direction mandatory sign
- Left direction mandatory sign
- Warp angle sign
- Wall
- No-entry sign

3.2 Linear SVM

Downstream of the deep neural network that is able to recognize generic classes such as the signal of a door number and the curve angle signal, a linear SVM was used to

discriminate the minor classes within the same *Door number sign* macro-class (door 1, 2, 3, 4, 5, 6) and the *Warp angle sign* macro-class (0°, 45°, 90° and 180°).

Linear SVM (with 12.5 as penalty parameter) uses HOG descriptor. Its characteristics are listed below:

winSize = (50, 50)	*nbins = 9*	*L2HysThreshold = 0.2*
blockSize = (20, 20)	*derivAperture = 1*	*gammaCorrection = 1*
blockStride = (10, 10)	*winSigma = -1.*	*nlevels = 64*
cellSize = (10, 10)	*histogramNormType = 0*	*signedGradients = True*

3.3 Why not use a single detector?

Initially the only neural network was used to carry out the detection of the signals and therefore their recognition. It was noted that the network was particularly confused about the recognition of numbers (number of door signs and degrees of curvature).
A possible solution would have been to increase the training dataset paying attention to the photographic capture of the signals in question.
In any case, having already collected a huge dataset of 15900 samples (almost uniformly distributed among all the signals) for training only while another 2000 for validation, it was thought to act differently.
The intent was to use the part of the dataset with the samples for the port numbers to carry out the detection on the *port number* class, thus increasing by six the samples since the subclasses would have been the numbers from 1 to 6. The same consideration was made for the macro-class *degrees* which contains the classes *degrees 0*, *degrees 45*, *degrees 90*, *degrees 180*. Once the macro-class was detected, it was submitted to the linear SVM able to classify the single number between the port numbers and the single cartel of the degrees in the corresponding macro-class.
The following is a comparative overview for the evaluation of the solution that provides the only neural network against the ***deep neural network plus linear SVM*** solution.

ONLY DEEP NEURAL NETWORK SCORES

CLASS	PRECISION	RECALL	F1_SCORE
divieto	1.0	0.671	0.803
f_dx	0.941	0.829	0.881
f_sx	0.913	0.901	0.907
f_up	0.676	0.548	0.605
g_0	0.909	0.526	0.667
g_180	0.632	0.456	0.529
g_45	0.939	0.733	0.823
g_90	0.936	0.791	0.857
n_1	0.9	0.2	0.327
n_2	0.86	0.343	0.49
n_3	0.577	0.254	0.353
n_4	0.556	0.435	0.488
n_5	0.63	0.582	0.605
n_6	0.788	0.588	0.673
wall	0.911	0.887	0.899

DEEP NEURAL NETWORK + LINEARSVM SCORES

CLASS	PRECISION	RECALL	F1_SCORE
divieto	1.0	0.658	0.794
f_dx	0.755	0.77	0.762
f_sx	0.851	0.781	0.815
f_up	0.694	0.595	0.641
g_0	0.521	0.658	0.581
g_180	0.867	0.494	0.629
g_45	0.88	0.651	0.748
g_90	0.961	0.794	0.87
n_1	0.975	0.867	0.918
n_2	0.972	0.734	0.837
n_3	0.833	0.339	0.482
n_4	0.357	0.435	0.392
n_5	0.841	0.873	0.857
n_6	0.961	0.868	0.912
wall	0.965	0.726	0.829

The implemented solution (on the right) requires that the neural network is trained for the seven classes described above. The dataset that was therefore balanced for 15 original classes is now unbalanced if we group [n_1, n_2, n_3, n_4, n_5, n_6] into a single class and [g_0, g_180, g_45, g_90] in a single class. It is this unbalancing of the network that will most likely result in a minimal decrease in performance on the other classes.

In any case, we believe that on average performance has improved by a linear SVM classifier cascaded to the deep neural network in the way described.

4. Target lock-on

The target locking algorithm selects the object of interest based on its type and distance. A data structure keeps all objects sorted by increasing distance from the Rover:

- If the nearest object is a mandatory sign with direction then proceed to search in the data structure of the nearest signal indicating the curve angle, within 50cm. The signal pair is then locked.
- If the nearest object is a signal indicating the curve angle then proceed to search in the data structure of the nearest mandatory signal with direction, within 50cm. The signal pair is then locked.
- If the nearest object is a mandatory signal with direction or a signal indicating the angle of the curve but the complementary signal in the direction-angle pair is not present in the data structure (or this exceeds the threshold of 50 cm) then only the signal found is locked.
- The mandatory sign with forward direction and the sign with zero curve degrees are ignored.
- If the nearest object is a door that respects the numerical order, then it is locked.
- If the nearest object is a door that does not respect the numerical order, then it is ignored and looked further.
- If the nearest object is a no-entry sign, then it is locked as a target.
- If the nearest object is a wall, then it is locked as a target.

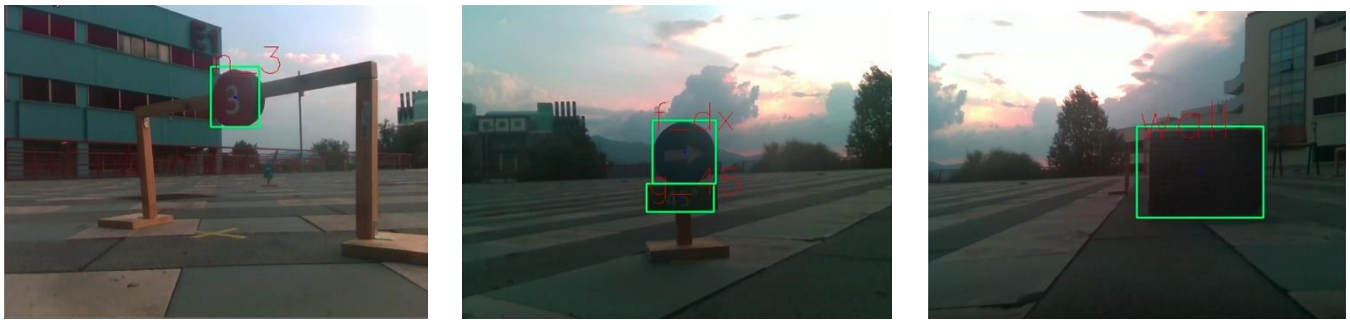

Figure 5 : Target lock-on

5. Motion core

The module responsible for controlling motion receives the target as the coordinates of the bounding box of the object on the video frame. In addition to using the image coming from the camera, it uses the depth information to best implement the curvatures and corrections on the movement at the right time.

5.1 Cruising move

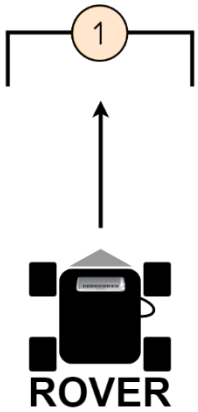

Figure 6 : Move forward

The Rover proceeds in a straight line by default at the speed of 1 m/s. In fact, it will proceed straight if it has no information from outside for 10 seconds, if the target passes to it only one of the signals in the direction-angle pair (the intent is to approach the signal so that the missing signal can be seen), if the target passed to it is the no-entry signal but the maximum distance of 0.5 meters from the signal has not been reached (the intent is to dispel the possibility that if the no-entry signal was the only one at a considerable distance, during the route other signals can be detected). If the prohibition signal is the target at an inferior distance of 0.5 meters, then the Rover is stopped.

5.2 Warp moves

At 1 meter from the signal indicating the curvature, a warp is performed at linear speed 1 m/s and angular velocity 2 rad/s with a 50 cm radius. The curves provided by the project specification are 45°, 90° and 180°, therefore the movement to the target as angle-direction is optimized for multiple angles of 45 °. In fact, the algorithm required for the warp uses fixed parameters for the set of three angles, parameters that have been calibrated according to the surface of the location indicated as specific for demonstrating the operation of the Rover, so as to exploit the friction present with the purpose to make a warp as close to that indicated by the target signal.

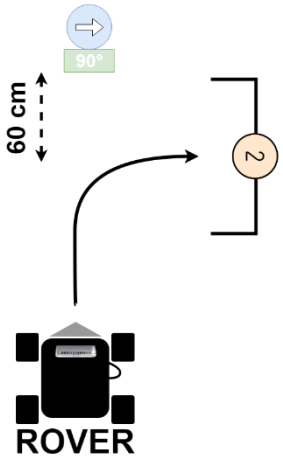

Figure 7 : Curve to the right

5.3 Alignment correction

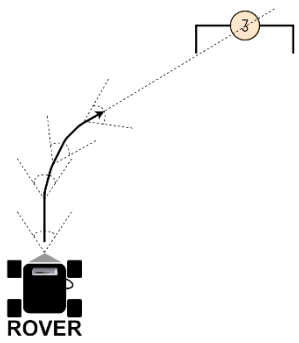

Figure 8 : Alignment correction

A first correction is performed by centering the target within the camera's visual frame. The procedure takes into account that the visual field of the camera is about 60° and counted the number of pixels that separates the center of the target from the center of the frame it is estimated the angle for which it would be necessary to rotate in order to bring the target exactly to the center of the frame.

Then we curve for this angle by calculating the time required for the maneuver since the linear and angular speeds have been set.

5.4 Distortion correction

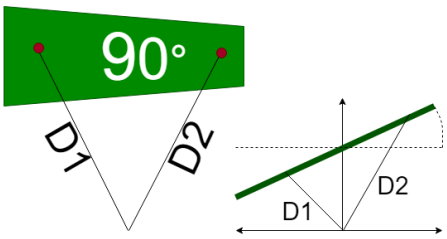

Figure 9 : Rotation calculation

Before the target of interest is centered within the frame, the angle at which the target is rotated with respect to its own axis orthogonal to the ground is calculated. Whether it is a signal with direction or a door, the distances of two points are detected almost at the ends of the target on the same horizontal line, so it is possible to detect the rotation angle as a function of the distance of the target.

5.4.1 Distortion correction on direction sign

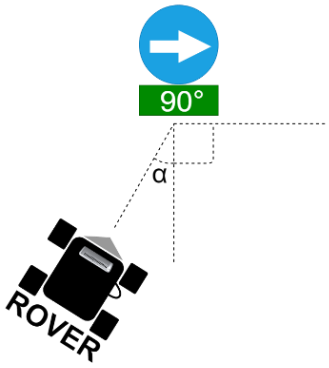

Figure 10 : Performing a steering of 90+α degrees

If the target is the direction-degree pair, it should be noted that the indicated degrees are those to be respected if the sign is encountered perpendicularly. Therefore, according to the degrees indicated by the sign, the direction to follow and the distortion measured, it is calculated how many degrees it is necessary to turn.

Once the distortion has been calculated we will turn with the linear and angular velocities previously defined for the generic curve. The only variable depending on the degrees of

curvature is the time for which to keep the curve. This time is calculated through a parabolic curve defined using the Lagrange interpolation of the times required for the 45°, 90° and 180° curves.

5.4.2 Distortion correction on door number sign

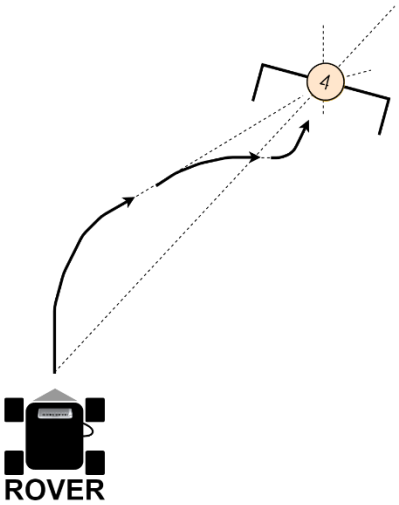

Figure 11 : Distortion correction on door number sign

If the target is the signal of a door number, then it would be necessary to make a correction that allows to enter as much as possible perpendicular to the door. As with a road vehicle, a series of corrections must be made to avoid colliding with the door supports themselves. The following procedure is applied if the distortion detected is very high:

- The linear speed is slightly decreased, leaving the angular speed unchanged;
- We curve for about 15 ° from the side opposite the path that we should follow to meet the door;
- We curve towards the door increasing the angular speed by 85%.

The angle on the opposite side and the increase in angular velocity are fixed parameters obtained empirically on the demonstration path.

5.5 Wall avoidance

The *wall sign* is the only obstacle to be avoided within the path, therefore its dimensions are known, so how much it is necessary to turn to avoid it and how much it is necessary to proceed to overcome its depth. It was decided to avoid the obstacle through four 45° curves, if the obstacle is not rotated, at a linear speed of 1 m/s and angular velocity of 2 rad/s.

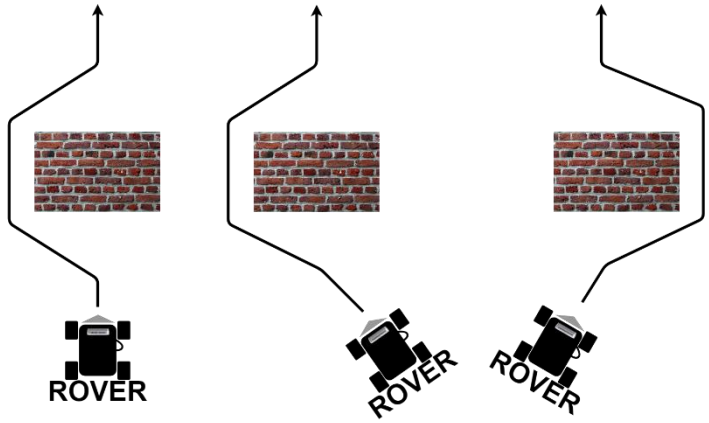

Figure 12 : Wall avoidance

The algorithm for wall avoidance is primarily concerned with correcting the alignment of the wall with the Rover through the procedure indicated above. Subsequently, depending on the angle of inclination of the obstacle with respect to the Rover, the side on which to carry out the maneuver is chosen by minimizing the first angle of curvatures: if for example the rover meets the wall rotated 15° clockwise then the algorithm choose the left side curving 30°.

The first curve is therefore based on the distortion of the wall, a 45° curve in the opposite direction, we proceed forward by 40 centimeters at the linear speed of 1 m/s and follow two other curves to go in line with the original path.

6. Notes

- **Software dependencies.** The architecture of the Rover includes pre-installed Python libraries. The only requirement that transcends pre-installed libraries is the scikit-learn library version 0.21.3 .
- **Fixed distances on shunting corrections.** The measurements are carried out starting from fixed distances from the targets of interest and are the result of the detection of multiple samples; the corrections themselves are made at fixed distances from the targets.
- **Temperature.** Although the NVIDIA® Jetson Nano ™ has been equipped with a fan on its heat sink, tests are recommended to monitor the CPU temperature so that it does not exceed 46°C after which the architecture strongly degrades its performance.

7. Index of figures