HELVIS - A SMALL-SCALE AGRICULTURAL MOBILE ROBOT PROTOTYPE FOR PRECISION AGRICULTURE

Velasquez A. E. B[+], Higuti V. A. H[+], Guerrero H. B[+], Milori D. M[*], Magalhães D. V[+], Becker M[+].

\+ São Carlos Engineering School, São Paulo University, São Paulo, Brazil.

*Embrapa Instrumentation, São Carlos, São Paulo, Brazil

A paper from the Proceedings of the
13th International Conference on Precision Agriculture
July 31 – August 4, 2016
St. Louis, Missouri, USA

Abstract. *The use of agricultural robots is emerging in a complex scenario where it is necessary to produce more food to feed a crescent population, decrease production costs, fight plagues and diseases, and preserve nature. Around the world, there are many research institutes and companies trying to apply mobile robotics techniques in agricultural fields. Mostly, large prototypes are being used and their shapes and dimensions are very similar to tractors and trucks. In the present study, a small-scale prototype was designed, aiming to facilitate the controller development phase and the execution of experiments in the university using a farm-like scenario (before validating the controllers in a real scenario). It is important to highlight that all control parameters were parameterized to allow the control portability to other prototypes. Helvis is an electric small-scale car-like platform whose traction and steering systems are powered by Maxon motors and driven by EPOS2 boards. Its navigation system uses 2 LiDAR sensors (UTM-30LX) to scan the environment (one in the front and the other in the back) and an Inertial Navigation System (IG500N) to estimate its orientation. In this paper we present experiments carried out in rows of a corn crop field. As previously mentioned, before making the experiments in a real farm, a farm-like scenario was constructed in the lab to calibrate the controller parameters. Since each cornrow constitutes itself a discontinuous wall, a filter based on the LiDAR data was developed in order to create virtual continuous walls. So, these virtual walls were used as references for a wall-follower control system. When it is possible to create walls in both sides of the robot, the navigation problem can be simplified to moving the robot in a virtual aisle. The filter calculates the distances between robot and virtual walls, which are used as input data for the fuzzy controller responsible to keep the robot in the path between corn rows. Its output signal acts in Helvis' steering system. Real environment experiments allowed adjusting fuzzy rule set and improving robot performance.*

Keywords. *Mobile Robot, Precision Agriculture, Perception, Navigation.*

INTRODUCTION

Nowadays agriculture is facing unprecedented demands. United Nations estimate that the world population will achieve 10 billion people by 2050 (UNFPA, 2015) and that there will be globally about 795 million people undernourished in 2014-16 (FAO, 2015). In this scenario, agricultural production must grow substantially while confronting the need of reducing its environmental footprint (Foley et al., 2011). In this context, it was possible to observe in the last few years that robotics technology is becoming more present in many agriculture areas (Reina et al., 2015).

Cheein and Carelli (2013) summarize the most important abilities of automatic agricultural vehicles into four categories: guidance, detection, action, and mapping. For these activities, devices like camera modules, Real Time Kinetics Global Positioning System (RTK-GPS) and Light Detection and Ranging (LiDAR) are needed to retrieve information from environment. An example of vision based system can be found in (English et al., 2014) which describes a crop rows tracking method using vision based texture model. An automatic and accurate guidance of a four-wheel-steering mobile robot using RTK-GPS is described in (Cariou et al., 2009). Weiss and Biber (2011) used a 3D LiDAR sensor to detect single plants in crop rows in real time.

Auto-guided agricultural vehicles with GPS-based navigation systems started appearing around 1997. They were able to follow a predefined map, bringing benefits like accuracy increase and driver's fatigue reduction (Heraud and Lange, 2009). But when dealing with unmanned vehicles, which do not have an operator to manually interfere in the presence of environmental alterations, a GPS-based system is not enough to guarantee that the vehicle will not ride over the crop and unmapped elements (like other vehicles, animals, humans or recent changes) thus raising a major safety issue (Reina et al., 2015). Besides that, high-accuracy GPS systems, such as differential GPS and RTK-GPS, are expensive but even so, subject to interruptions or poor availability of the signal which leads to great position errors and possibly, navigation failure (Hiremath et al., 2009). Although great improvement can be seen in (English et al., 2014), vision based methods are sensitive to variations in environment light and atmospheric effects. Outdoor environments, such as the agricultural ones, would require frequent calibration procedures (Hiremath et al., 2009).

Regarding LiDAR-based navigation systems, Hiremath et al. (2009) proposed and showed the robustness of a LiDAR-based autonomous navigation based on a particle filter. Barawid et al. (2007) highlighted two advantages of using such systems: as it takes into account real-time local information, a pre-surveying task to generate a map can be omitted (thus saving time); and they can be used when GPS becomes non-functional. In addition, Weiss and Biber (2011) state that in contrast to most stereo vision cameras, the ranging information is calculated on the LiDAR sensor itself, thus no further time-consuming calculation has to be carried out on the computer, and the distance values' precision is also independent from the measured distances.

For the aforementioned reasons, this study is focused on the development of a LiDAR-based autonomous navigation system of a car-like mobile robot through cornfield without relying on a planned path. For this purpose, a fuzzy controller is proposed and experimentally tuned for the crop row following. Appropriate row exit and entrance maneuvering routines based on LiDAR and robot's heading readings were also implemented. Although high-level decision making would require additional resources, such as landmarks, Radio-Frequency IDentification (RFID) tags or even a GPS-based map, the developed navigation system is able to work in standalone mode ensuring crop's integrity.

HELVIS PROTOTYPE

Helvis (Fig 1) is an electric small-scale car-like platform (Length | Width | Height: 0.65m x 0.23m 0.45m) whose traction set consists in an EPOS2 50/5 board, an automotive-like differential mechanism and an EC-4Pole 30 Maxon Motor (reference 309756) which has two shafts (rear and front). The rear shaft is coupled to an incremental encoder (reference 255778) and the front shaft is

coupled to a planetary gearhead GP 32 (reference 326661) for a 23:1 reduction. Its steering set has an EPOS2 24/5 board, an Ackerman mechanism, a power screw and a DC Maxon motor REmax29 (reference 226802). The rear shaft of DC Maxon Motor is coupled to an incremental encoder (reference 225805) and its front shaft is coupled to a planetary gearhead GP 32 (reference 166938) for a 33:1 reduction. In order to scan the environment, it has two embedded LiDAR sensors (UTM-30LX) manufactured by Hokuyo (Hokuyo Automatic, 2012): the first one is located in the front part of the robot and the other one is in the rear part. Also, the robot has an Inertial Navigation System (IG500N) manufactured by SBG Systems (SBG Systems, 2013) in order to estimate its orientation and position (it will be used in future works). In this work, orientation refers to yaw orientation angle. Its main control unit is a Raspberry Pi 2 running Raspbian operation system. For monitoring some variables during tests, Helvis has a router to allow remote communication between Raspberry Pi 2 and an external computer. The interaction between its embedded devices is presented in Fig 2. Finally, characterization of sensors and description of each part of Helvis structure are more detailed in (Velasquez, 2015).

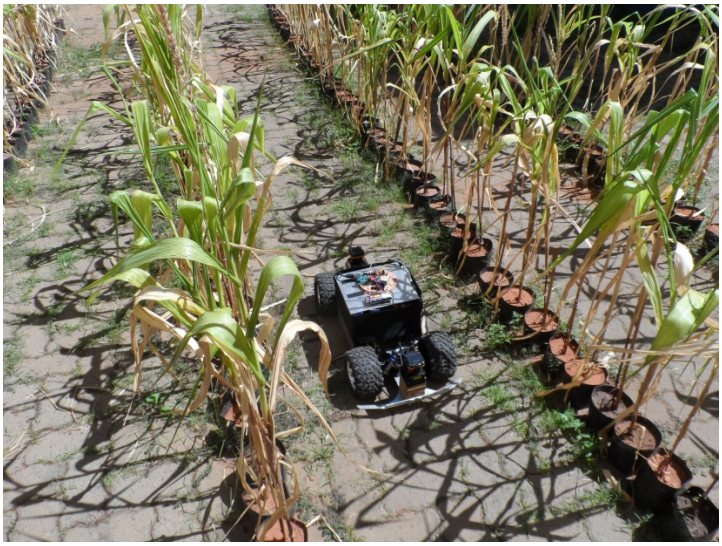

Fig 1. Helvis in middle of cornrow crop.

The steering system described in (Velasquez, 2015) was changed in order to increase its robustness and steering angle ranges. Its new steering and traction systems are showed in Fig 2.

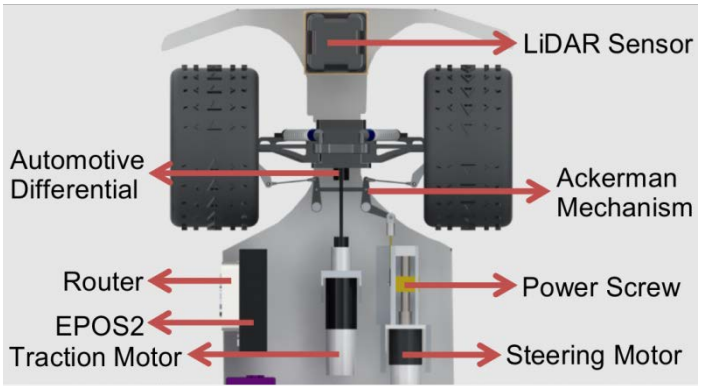

Fig 2. Steering and traction sets of Helvis.

CONTROL SYSTEM OF HELVIS

Helvis' control system is divided in two levels: High and Low control level systems (Fig 3). High-level system (also called wall-follower control system) consists of a fuzzy controller that keeps the robot on the path between the cornrows. Its input is the difference between distances from the robot to each

wall of corn rows and its output is the desired angle of the robot's front wheels. As previously mentioned in abstract, each corn row constitutes itself a discontinuous wall, then a filter was developed to generate two continuous virtual walls based on the sensor readings (vehicle's orientation measured by IG500N and distances measured by UTM-30LX). The controllers and filters are programmed with C++ language.

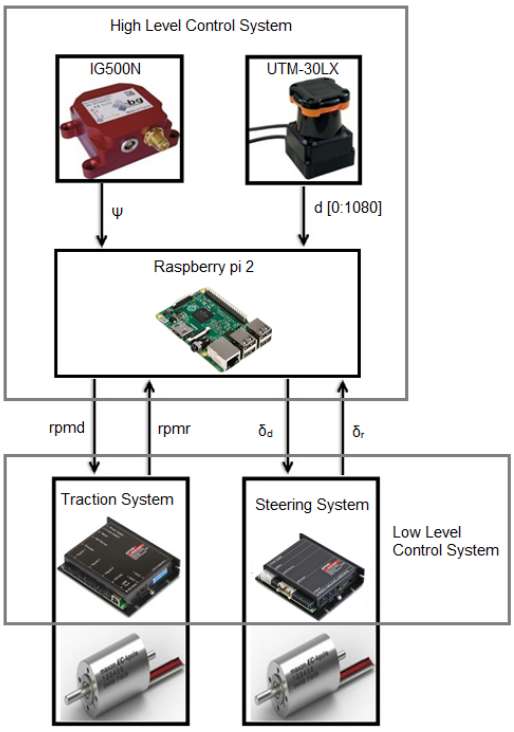

Fig 3. Interaction of Helvis' embedded devices.

The high-level control system output is front wheels' desired steering. This output and vehicle's desired speed, which maximum value is fixed at 0.2 m/s, are sent to low-level control system through serial communication. Low-level control system consists of two EPOS2 boards in charge of controlling traction and steering motors' speed and position. For this purpose, the board EPOS2 has embedded Proportional and Integral (PI) controllers (their description is available in Maxon Motors, 2013). While EPOS2 24/5 gets desired steering angle and makes DC Maxon Motor's position control, EPOS2 50/5 gets desired robot speed and makes EC Maxon Motor's velocity control.

Filter to create the virtual walls

As previously mentioned in abstract, all experiments presented in this paper were carried out in a farm-like scenario representing rows of corn crop field. Each row has about 40 corn plants organized in a straight line (Fig 4). Separation between each corn plant is 0.2m and distance between each row crop is approximately 1m.

Fig 4. Farm-like scenario created in the lab.

The robot uses a LiDAR sensor to scan the environment. LiDAR's scan range is 270° and its angular resolution is 0.25° (270°/1080 steps). Every 0.1s, the sensor updates a vector with 1080 positions with distances (in mm) between the sensor and any detected object for each step of its scan range. When it comes to the filter, three situations are defined: the first situation is when robot is moving without orientation error ($e\psi$) (Fig 5a); the second one is when it is moving with a positive orientation error (Fig 5b); and the last one is when it is moving with a negative orientation error (Fig 5c). Orientation error ($e\psi$) is the angular difference between path orientation (red arrow) and actual robot orientation (blue arrow). Path orientation is the obtained orientation when the robot enters the crop row.

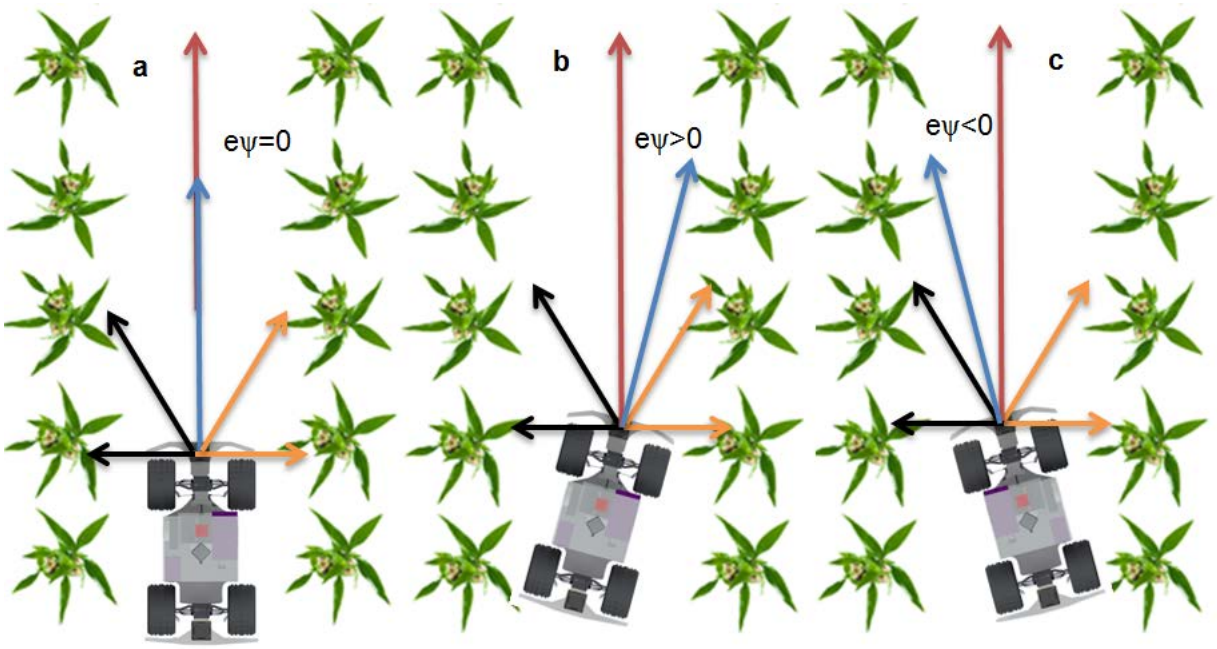

Fig 5. a- The robot is running without orientation error. b- The robot is running with positive orientation error. c- The robot is running with negative orientation error.

The filter's main goal is to generate virtual walls based on information of two interest regions. The first interest region, further referred as right-side interest region, comprehends the obtained readings of steps located between the two orange arrows showed in Fig 5a, 5b, and 5c. The second region, further referred as left-side interest region, contains the obtained readings of steps located between the two black arrows showed in Fig 5a, 5b, and 5c. Angular separation between black arrows and between orange arrows are both 60° (240 steps). For all cases presented in Fig 5, definition of an initial and a final step for each interest region is needed. For the case A (right-side interest region - Fig 5a), initial step is 180 (0°) and final step is given by equation 1.

$$final\_step_{right-side\_region} = initial\_step_{right-side\_region} + \left(\frac{60°}{0.25°}\right) \quad (1)$$

For the case of left-side interest region (Fig 5a), initial and final steps are defined by equations 3 and 2, respectively.

$$final\_step_{left-side\_region} = initial\_step_{right-side\_region} + \left(\frac{180°}{0.25°}\right) \quad (2)$$

$$initial\_step_{left-side\_region} = final\_step_{left-side\_region} - \left(\frac{60°}{0.25°}\right) \quad (3)$$

For the cases B and C (Fig 5b and Fig 5c), initial and final steps of right-side interest region are defined by equations 4 and 5, respectively, while initial and final steps of the left-side interest region are defined by equations 6 and 7, respectively.

$$initial\_step_{right-side\_region} = 180 + floor\left(\frac{e\psi}{0.25°}\right) \quad (4)$$

$$final\_step_{right-side\_region} = inital\_step_{right-side\_region} + \left(\frac{60°}{0.25°}\right) \quad (5)$$

$$final\_step_{left-side\_region} = initial\_step_{right-side\_region} + \left(\frac{180°}{0.25°}\right) \quad (6)$$

$$initial\_step_{left-side\_region} = final\_step_{left-side\_region} - \left(\frac{60°}{0.25°}\right) \quad (7)$$

Each region has readings of 240 steps. Then, two vectors (each one with 240 positions) were created in order to save information about interest regions. The first vector is called *points_right_region* and contains information about right-side interest region. The second one is called *points_left_region* and contains information about left-side interest region.

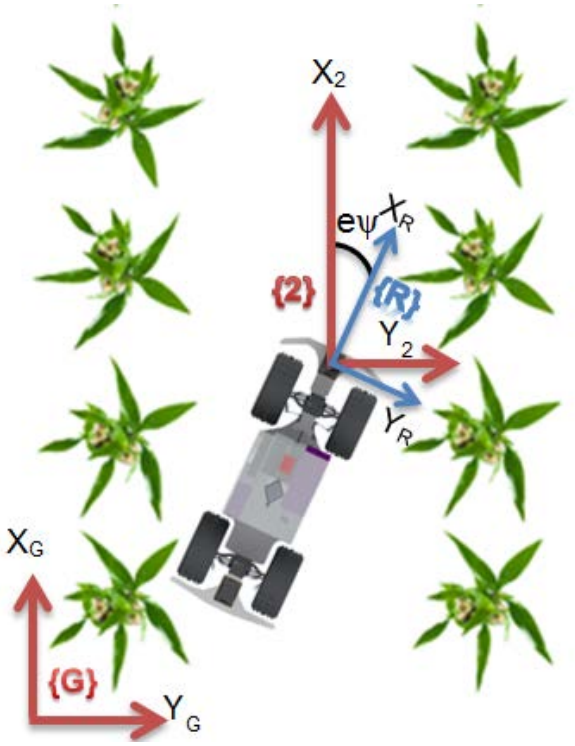

Fig 6. Global and Local Coordinate frames.

The main issue with information saved in *points_right_region* and *points_left_region* vectors is getting their *x* and *y* coordinates (in meters) relative to coordinate frame *{2}*, shown by X_2 and Y_2 axes in Fig 6. This coordinate frame is a translation of global coordinate frame *{G}* (X_G,Y_G axes in Fig 6) to the origin of robot's local coordinate frame *{R}* (X_R,Y_R in Fig 6). While *{G}*'s origin is fixed at one corner of the corn field, *{R}* is attached to the robot, with its origin in the center of front LiDAR sensor and X_R axis coincident with robot's longitudinal axis.

As it can be seen in Fig 6, orientation error (e_ψ) is the angle between X_2 and X_R. Thus in order to obtain projections in frame {2} from *points_right_region* and *points_left_region* vectors, equations 8, 9, 10 and 11 uses e_ψ to compensate robot's orientation deviations. Using these equations, four vectors are generated, and they contain *x* and *y* coordinates of *points_right_region* and *points_left_region* vectors. Their names are *X_right_region*, *Y_right_region*, *X_left_region* and *Y_left_region*.

$$X\_right\_region(posrw) = points\_right\_region(posrw) * \sin\left((steprw * 0.25°) - 45° + e\psi\right)/1000 \quad (8)$$

$$Y\_right\_region(posrw) = points\_right\_region(posrw) * \cos\big((steprw * 0.25) - 45° + e\psi\big)/1000 \quad (9)$$

$$X\_left\_region(poslw) = points\_left\_region(poslw) * \sin\big((steplw * 0.25°) - 45° + e\psi\big)/1000 \quad (10)$$

$$Y\_left\_region(poslw) = points\_left\_region(poslw) * \cos\big((steplw * 0.25) - 45° + e\psi\big)/1000 \quad (11)$$

where *steprw* is an incremental variable (with increments equal to 1) which starts at $initial\_step_{right\text{-}side\_region}$ and ends at $final\_step_{right\text{-}side\_region}$. The *steplw* is another incremental variable (also with increments equal to 1) which starts at $initial\_step_{left\text{-}side\_region}$ and ends at $final\_step_{left\text{-}side\_region}$. Finally, *posrw* and *poslw* are described by equations 12 and 13, respectively.

$$poslw = steplw - initial\_step_{left-side\_region} \quad (12)$$

$$posrw = steprw - initial\_step_{right-side\_region} \quad (13)$$

Those projection vectors are filtered to generate virtual walls. The filter for virtual left wall is described by the routine presented in equation 14.

```
if ((Y_left_region (stepfilter) > -1.2) && ((Y_left_region (stepfilter) < -0.2)
{
    Yvirtual_left_wall (stepfilter) = Y_left_region (stepfilter);
    Xvirtual_left_wall (stepfilter) = X_left_region (stepfilter);
}
```
(14)

And the filter for right wall is presented in equation 15.

```
if ((Y_right_region (stepfilter) < 1.2) && ((Y_right_region (stepfilter) > 0.2)
{
    Yvirtual_right_wall (stepfilter) = Y_right_region (stepfilter);
    Xvirtual_right_wall (stepfilter) = X_right_region (stepfilter);
}
```
(15)

Where *stepfilter* is an incremental variable (with increments equal to 1) that starts at 0 and ends at 239. Filtered vectors have *X* and *Y* coordinates of virtual walls.

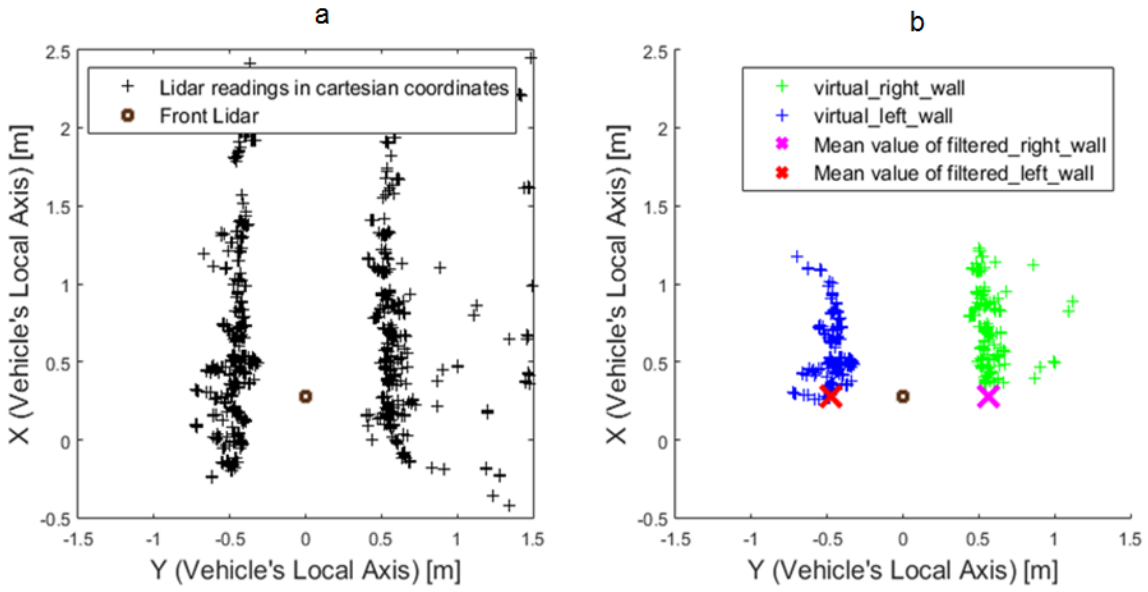

Fig 7. a- Map obtained without filter. b- Map with the virtual walls obtained with the filter.

In order to calculate distances between robot and virtual walls, mean values of *Yvirtual_left_wall* and *Yvirtual_right_wall* are obtained. *Yvirtual_left_wall*'s mean value is the distance of the robot relative

to left virtual wall (*dlw*) while *Yvirtual_right_wall*'s mean value is the distance of the robot relative to right wall (*drw*). Fig 7a shows the map obtained with LiDAR readings without the filter and Fig 7b shows the map with virtual walls obtained with filter.

Wall-follower system control

As mentioned before, the wall-follower system control has a fuzzy controller whose main objective is to keep the robot in the path between the cornrows. Block diagram of wall follower system control is shown in Fig 8.

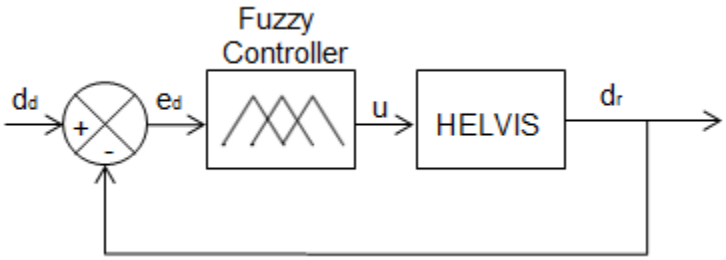

Fig 8. Block diagram of the wall-follower system control.

For this subsection, distance represents the extension of space between Helvis longitudinal axis and middle between crop rows. Wall-follower control system's input is desired distance (d_d) and its output is the actual distance (d_a). This latter value is constantly obtained by using LiDAR sensor and the filter mentioned in the previous section, and it is negative feedback to the fuzzy controller through variable e_d (distance error) which is the difference between dd and da. Value of desired distance is always zero and value of real distance is the sum of *drw* and *dlw* (described in the filter section). Fuzzy controller's output is steering angle (*u*) for Helvis. Implementation of fuzzy controller was based on the fuzzy controllers developed in (Guerrero et al., 2014; Higuti et al., 2015). It has an input fuzzy set with five triangular pertinence functions (Fig 9a) and an output fuzzy set with three triangular pertinence functions (Fig 9b). The five pertinence functions of input fuzzy set are named: *VFLW* (*Very_Far_Left_Wall*), *FLW* (*Far_Left_Wall*), *MP* (*Middle Path*), *FRW* (*Far_Right_Wall*) and *VFRW* (*Very_Far_Right_Wall*). And the three pertinence functions of the output fuzzy set are named: *LS* (*Left_Steering*), *NS* (*No_Steering*) and *RS* (*Right_steering*).

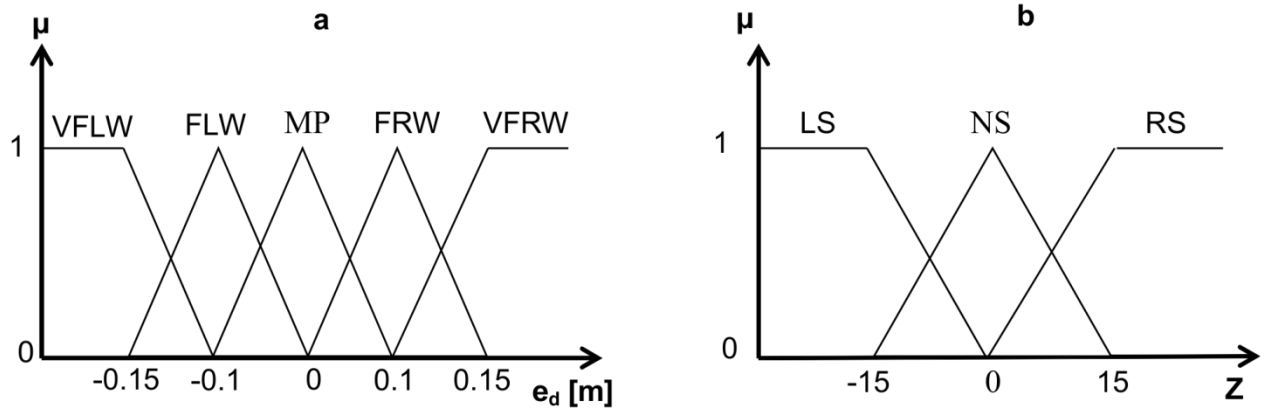

Fig 9. a- Input fuzzy sets. b- Output fuzzy sets.

A value of distance error (e_d) generates two pertinence values (μ_1 and μ_2) in two different input fuzzy sets at the same time. These pertinence values are used to cut the pertinence function of the output fuzzy set according to the fuzzy rules described in equation 16.

$$\text{If } e_d = \text{VFLW or if } e_d = \text{FLW}, \quad \text{then } \mu = \text{LS}$$

$$\text{If } e_d = \text{MP}, \quad \text{then } \mu = \text{NS} \qquad (16)$$
$$\text{If } e_d = \text{FRW or if } e_d = \text{VFRW}, \quad \text{then } \mu = \text{RS}$$

When pertinence values cut output fuzzy set's pertinence functions, some output fuzzy values (*Z*) are obtained. These values are used to find the defuzzified output value (*Z**) which is obtained with the centroid method (Guerrero et al., 2014). The *Z** is fuzzy controller's output (*u*) and it is a steering angle for the robot.

SWITCHING BETWEEN TWO CROP ROWS

This section describes the maneuver process used to change crop rows. There are two maneuvers. The first maneuver is used when the robot changes to another row located on its right side (Fig 10). And the second one is a mirrored version as it is used when the following row is located to the left side. The maneuver process is divided into four steps (Fig 10).

1. End of first crop row.
2. Finding next crop row
3. Entrance to next crop row.
4. On second crop row.

First step comprehends: detection of row end, measurement of actual robot's orientation (ψ_{first_step}), calculation and subsequent movement to desired orientation for second step (equation 17).

$$\psi_{second\_step} = \psi_{first\_step} - 90° \qquad (17)$$

In order to find row end, information of *Yvirtual_left_wall* and *Yvirtual_right_wall* vectors is used. When the sum of number of elements of those two vectors is less than 220 elements, it indicates row end was found. At this point, a mean value of 100 consecutive readings of orientation angle is taken and set as desired orientation for first step. Then, ψ_{second_step} is calculated by equation 17 and the robot continues moving forward but with front wheels totally steered towards right side. About orientation signal, when robot steers to the right, the measured orientation decreases. And for the opposite movement, it increases. When actual robot's orientation is near (±10°) ψ_{second_step}, the robot motion stops and front wheels are steered back to 0° (end of first step).

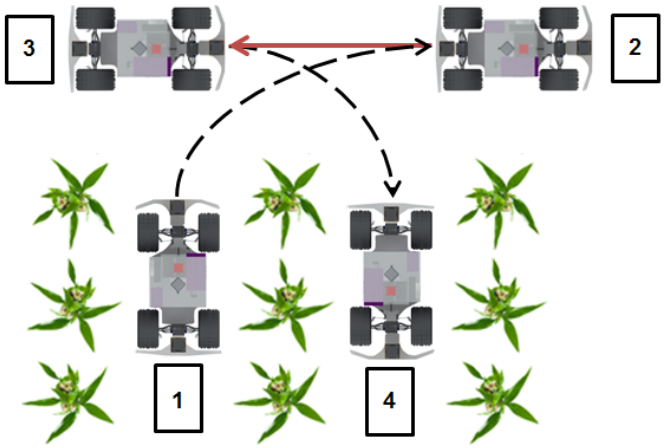

Fig 10. Maneuver to change the crop row.

In the beginning of second step, desired orientation (ψ_{second_step}) is updated. Reverse motion, crop rows' detection filter and fuzzy controller for reverse motion are activated. The condition to finish second step is the counting of two crop rows by the robot. When this condition is satisfied, robot motion, fuzzy controller and filter are stopped.

Third step consists in the entrance to the next row, which involves a 90° right or left turn. Thus, desired orientation for third step (ψ_{third_step}) is calculated by equation 18.

$$\psi_{third\_step} = \psi_{second\_step} - 90° \quad (18)$$

Subsequently, robot motion is activated and its front wheels are steered to the maximum steering value towards its right side. When the robot orientation is near (±10°) ψ_{third_step}, robot motion is stopped and front wheels return to 0° (end of third step). In the final step, ψ_{fourth_step} is given by equation 19 as the best estimative for the new crop row is that its path orientation is parallel but in the opposite direction of the previous row. Then, robot motion is activated. From this point until next turning maneuver, ψ_{fourth_step} is used in the filter (used to create the virtual walls) as the path orientation. Finally, the wall-follower control system is activated.

$$\psi_{fourth\_step} = \psi_{first\_step} - 180° \quad (19)$$

The second maneuver is used when the next crop row is located in the left side. This maneuver is divided in the same steps as first one. Orientation for second step (calculated in the first step) and fourth step (calculated in the fourth step) are different and they are defined by equations 20 and 21, respectively. In the first and third steps of this maneuver, the robot steers its front wheels to its left side. Finally, the second and fourth steps are the same.

$$\psi_{second\_step} = \psi_{first\_step} + 90° \quad (20)$$

$$\psi_{fourth\_step} = \psi_{first\_step} + 180° \quad (21)$$

Fuzzy Controller for the Reverse Motion

In order to reduce deviations while going backwards (second step of maneuver process), another appropriate version of fuzzy controller was implemented, also following (Guerrero et al., 2014; Higuti et al., 2015). Similar to wall-follower control system, a block diagram representing this control system can be seen in Fig 11.

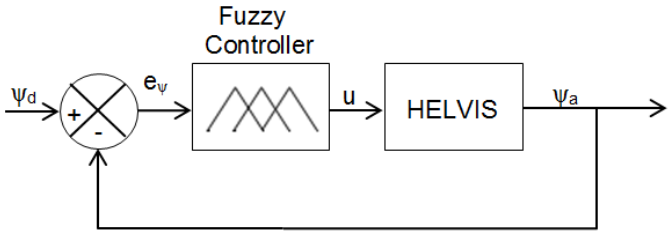

Fig 11. Block diagram of Reverse Motion Control System

Reverse motion control system's input is desired orientation (ψ_d) and its output is the actual orientation (ψ_a). This latter value is constantly obtained by using IG500N unit sensor, and it is negative feedback to the fuzzy controller through variable e_ψ (orientation error) which is the difference between ψ_d and ψ_a. As mentioned, the value of desired orientation is updated in the beginning of the second step and should be around 90 degrees off the orientation Helvis left the crop row. Fuzzy controller's output is steering angle (*u*) for Helvis. This fuzzy controller also has an input fuzzy set with five triangular pertinence functions (Fig 12a) and the same output fuzzy (Fig 12b) as it handles the steering like in the wall-follower. Here, the five pertinence functions of input fuzzy set are named: *VNOE* (*Very_Negative_Orientation_Error*), *NOE* (*Negative_Orientation_Error*), *COE* (Central_Orientation_Error), *POE* (*Positive_Orientation_Error*) and *VPOE* (*Very_Positive_Orientation_Error*).

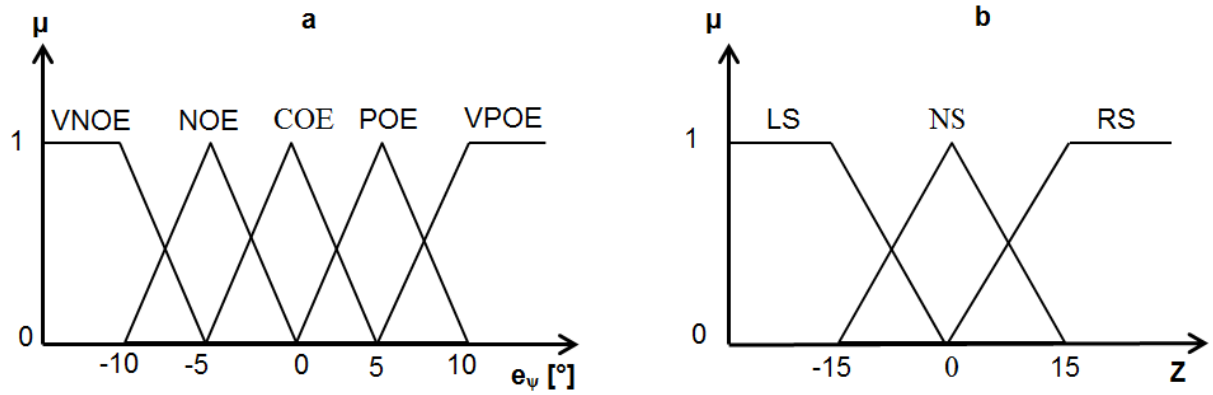

Fig 12. A- Input fuzzy sets. B- Output fuzzy sets.

Analogous to the fuzzy controller of wall-follower control system, a value of orientation error (e_ψ) generates two pertinence values (μ_1 and μ_2) in two different input fuzzy sets at the same time. These pertinence values are used to cut the output fuzzy set's pertinence function according to the fuzzy rules described in equation 22.

$$\begin{aligned} &\text{If } e_\psi = \text{VNOE or if } e_\psi = \text{NOE}, \quad \text{then } \mu = \text{LS} \\ &\text{If } e_\psi = \text{COE}, \qquad\qquad\qquad\quad \text{then } \mu = \text{NS} \\ &\text{If } e_\psi = \text{POE or if } e_\psi = \text{VPOE}, \quad \text{then } \mu = \text{RS} \end{aligned} \tag{22}$$

A better understanding of equation 22 can be achieved analyzing Fig 6: in reverse motion, negative e_ψ requires steering to the left while positive e_ψ needs steering to the right in order to correct robot's orientation. With the output fuzzy values obtained when pertinence values cut output fuzzy set's pertinence functions, the defuzzified output value (Z^*) can be found using the centroid method (Guerrero et al., 2014). Again, the Z^* is fuzzy controller's output (u) and it is a steering angle for the robot.

Detection of Crop Rows in Reverse Motion

This subsection describes how crop rows are detected in reverse motion. For this purpose, rear LiDAR readings are used. As the filter used to create virtual walls, an interest region of LiDAR readings is defined. Location of the interest region depends on the maneuver used to change the crop row. For the first maneuver, interest region comprehends the obtained readings of steps located between the two red arrows shown in Fig 13a.

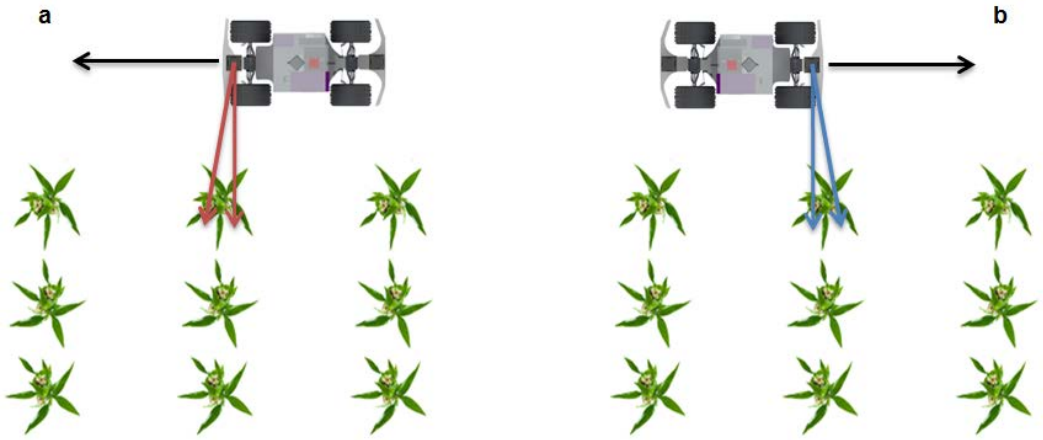

Fig 13. Interest region used to detect the crop rows.

Fig 13b shows the interest region (obtained readings of steps located between the two blue arrows) for the second maneuver. For both cases, angular separation between arrows is 5° (20 steps) and black arrow indicates the sense of robot's reverse motion. Initial and final steps of the interest region for first maneuver are defined by equations 23 and 24, respectively.

$$final\_step = 900 + \left(\frac{e\psi}{0.25°}\right) \tag{23}$$

$$initial\_step = final\_step - \left(\frac{5°}{0.25°}\right) \tag{24}$$

Initial and final steps of the interest region for second maneuver are defined by equations 25 and 26, respectively.

$$initial\_step = 180 + \left(\frac{e\psi}{0.25°}\right) \tag{25}$$

$$final\_step = initial\_step + \left(\frac{5°}{0.25°}\right) \tag{26}$$

In order to obtain vectors with projections of LiDAR readings for the first maneuver (*X_left_region* and *Y_left_region*), equations 10, 11, 23 and 24 are used. For the second maneuver, vectors with projections of LiDAR readings (*X_right_region* and *Y_right_region*) are obtained using equations 8, 9, 25 and 26. To detect a crop row for the first maneuver, a filter (equation 27) is applied to *X_left_region* and *Y_left_region* vectors.

```
if ((X_left_region (stepfilter) > 0) && (X_left_region (stepfilter) < 0.02))
{
    if ((Y_left_region (stepfilter) > -1.7) && (Y_left_region (stepfilter) < -0.6))
    {
        interest_points = interest_points + 1;
    }
}
```
(27)

Where *interest_points* is the count of filtered points. If the value of *interest_points* is greater than 5 then a possible crop row was detected. In order to consider that a real crop row was detected, three consecutives possible crop rows might be detected. For the second maneuver, the used filter is described by equation 28.

```
if ((X_right_region (stepfilter) > 0) && (X_right_region (stepfilter) < 0.02))
{
    if ((Y_right_region (stepfilter) > 0.6) && (Y_right_region (stepfilter) < 1.7))
    {
        interest_points = interest_points + 1;
    }
}
```
(28)

Fig 14. shows the moment when a crop row was detected in a test performed. The green lines represent the interest region of LiDAR readings, blue points are Cartesian projections of rear LiDAR and red crosses are the filtered points used to determine if a crop row was detected. Finally, the position of rear LiDAR is represented with a filled orange square.

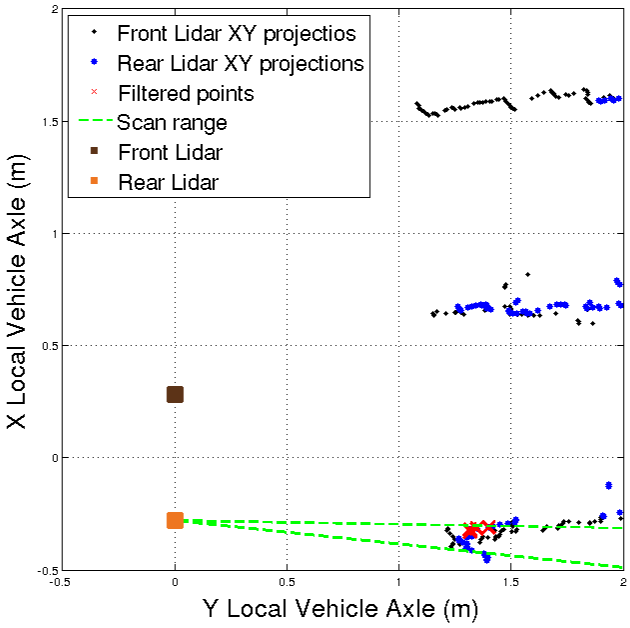

Fig 14. A crop row is detected.

EXPERIMENTAL RESULTS

In the mentioned farm-like scenario, several tests were carried out to improve overall system algorithms, define parameters, and evaluate filters and controller systems described in previous sections. Except for rainy days, occasions when tests are cancelled as the robot is not waterproof, environment factors such as temperature and luminosity have not interfered in Helvis performance in a perceptible way.

A successful test is one in which the robot drives through five crop rows with bounded distance error and response to disturbances quick enough to avoid crashes with the surroundings. When the robot arrives in the end of crop row, it needs to turn, alternately to the right and to the left, and enter the following crop row.

Both Fig 15. and Fig 16 represent a successful complete course around the five crop rows in scenario and show the fuzzy controller output for the steering system. The first depicts distance error when the robot is moving along the crop rows (sections ◯;A◯;C◯;E◯;G◯;I), thus trying to follow the virtual walls. And the second shows the orientation error in the maneuver process (sections ◯;B◯;D◯;F◯;H).

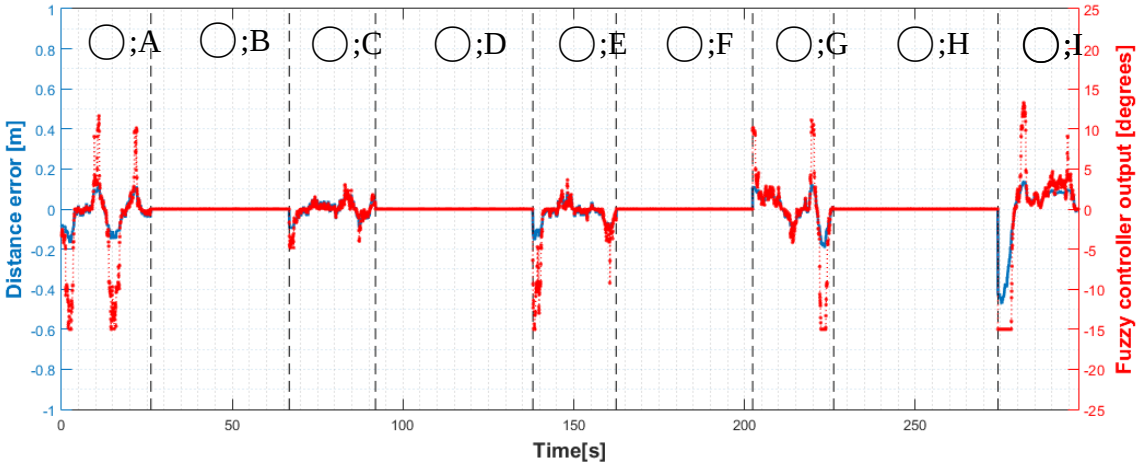

Fig 15. Distance error (blue plot) and Fuzzy controller output (red plot) through time

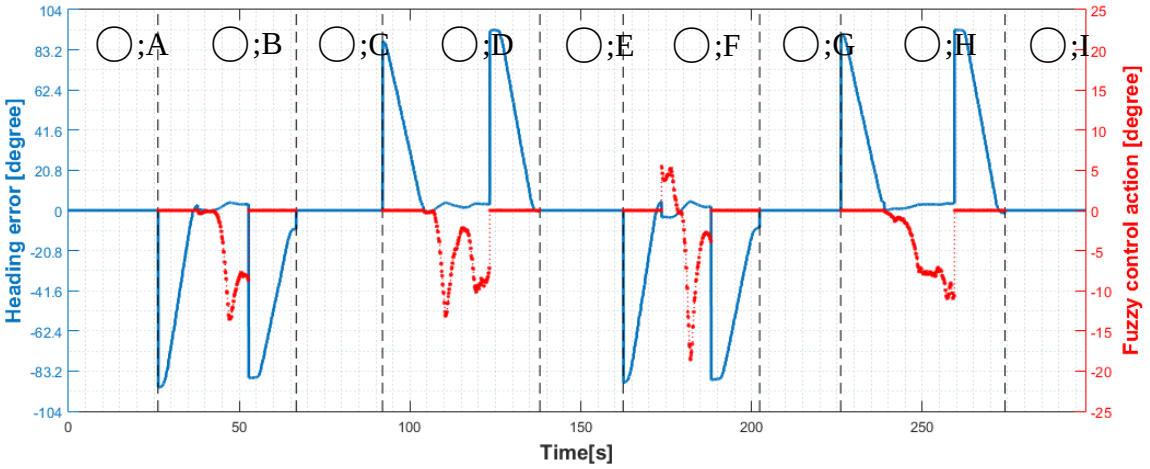

Fig 16. Orientation error (blue plot) and Fuzzy controller output (red plot) through time

As one may observe in Fig 15, the output is proportional to distance error when this value is smaller than 0.1m. There is a substantial increase in output when distance error is greater than 0.1m, which can be observed around time 281s in Fig 17a, a zoomed view of section ◯;I. This behavior is expected as a result of the chosen pertinence functions of input fuzzy set for wall-follower, which already accounts an absolute distance error of 0.1m either as *Far* (*to*) *Right Wall* or *Far* (*to*) *Left Wall* and begins to consider it as *Very Far* (*to*) *Right Wall* or *Very Far* (*to*) *Left Wall*.

The beginning of sections ◯;C, ◯;E, ◯;G, and ◯;I presents a high value of distance error because maneuvering process does not guarantee that the robot will start row in the exact middle between crop rows. Rather, because of maneuver, it is expected that the robot starts parallel to rows. And in the end of these sections, increased oscillatory behavior is expected as less information is available to create virtual walls.

Sections ◯;B, ◯;D, ◯;F, and ◯;H in Fig 16. represent the steps 1, 2, and 3 of the maneuvering process. A sawtooth shape represents the step 1 and 3 as the robot turns from its actual orientation until desired one (90° off the initial one). Between sawtooth shapes, reverse motion (step 2) is adjusted using fuzzy controller with actual orientation error as input. This part can be seen in more details in Fig 17b.

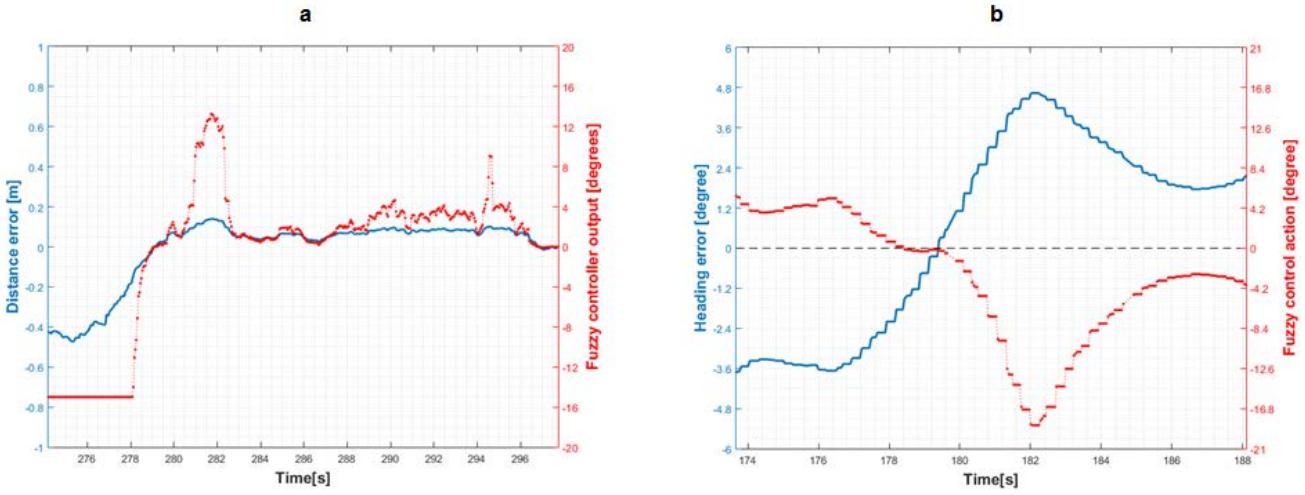

Fig 17. Zoomed views of: a – Fig 15. b – Fig 16.

Initial experiments were carried out without fuzzy controller for reverse motion. It was set that the robot should go backwards with zero steering until it counts two rows. Because of uneven surface,

there was always deviation and the robot either diverged, being too far to consider LiDAR readings of rows as valid ones, or converged, riding over the row. Although not optimized to quickly diminish the orientation error, the controller is able to counteract soil disturbances in order to keep the robot more or less orthogonal to row crops when going backwards, greatly improving chances of successful maneuver. For this experiment, orientation error was always less than 5°.

CONCLUSIONS AND FUTURE WORKS

This study focused on developing controllers and filters for an application of Helvis in agricultural context. It is a local navigation system that can be further used with a global one based on GPS data. Its purpose is, therefore, guaranteeing crop safety at all times, as GPS-based systems can suffer from signal loss. The wall-follower fuzzy controller presented an overall great response to soil disturbances, but in the borders of the rows, improvements can be made. First, a more reliable orientation estimation can be achieved using LiDAR readings and Simultaneous Localization and Mapping (SLAM) algorithms. As it will take into account surroundings in order to define path orientation, the possibility of wrong estimations because robot suffered disturbances on its orientation is mitigated. In addition, for a range of few meters, UTM30LX LiDAR sensor proved to be reliable under different luminosity conditions. Second, the presented maneuver process is a starting point for the switching row task. As one of the first maneuvers implemented in Helvis, it relies on few sensor measurements and simple calculations. However, it has the major drawback of not being able to overcome environmental changes, such as uneven soil, leading to orientation deviations, and dense foliage, complicating the correct counting of rows.

Acknowledgements

The authors would like to thank to FINEP, CNPq, CAPES, EMBRAPA and EESC-USP, for their support in this work.

References

Barawid Jr, O. C., Mizushima, A., Ishii, K., & Noguchi, N. (2007). Development of an Autonomous Navigation System using a Two-dimensional Laser Scanner in an Orchard Application. Biosystems Engineering, doi: 10.1016/j.biosystemseng.2006.10.012.

Cariou, C., Lenain, R., Thuilot, B., & Berducat, M. (2009). Automatic guidance of a four-wheel-steering mobile robot for accurate field operations. Journal of Field Robotics, doi: 10.1002/rob.20282.

Cheein, F. A., & Carelli, R. (2013). Agricultural robotics: Unmanned robotic service units in agricultural tasks. IEEE Industrial Electronics Magazine, doi: 10.1109/MIE.2013.2252957.

English, A., Ross, P., Ball, D., & Corke, P. (2014). Vision based guidance for robot navigation in agriculture. In Proceedings of EEE International Conference on Robotics and Automation (ICRA) (website). Hong Kong, Korea: IEEE. doi: 10.1109/ICRA.2014.6907079.

FAO (2015). The State of Food Insecurity in the World. Online Document. FAO. http://www.fao.org/3/a4ef2d16-70a7-460a-a9ac-2a65a533269a/i4646e.pdf. Accessed 21 April 2016.

Foley, J. A., Ramankutty, N., Brauman, K. A., Cassidy, E. S., Gerber, J. S., Johnston, M., Mueller, N. D., O'connell, C., Ray, D. K., West, P. C., Balzer, C., Bennett, E. M., Carpenter, S. R., Hill, J., Monfreda, C., Palasky, S., Rockstrom, J., Sheehan, J., Siebert, S., Tilman, D., &Zaks, D. P. M. (2011). Solutions for a cultivated planet. *Nature*, doi:10.1038/nature10452.

Guerrero, H. B., Velasquez, A. E. B., Barrero, J. F., Côco, D. Z., Risardi, J. C., Magalhães, D. V., & Becker, M. (2014). Orientation (Yaw) Fuzzy Controller Applied to a Car-Like Mobile Robot Prototype. In Proceedings of the IEEE 5th Colombian Workshop on Circuits and Systems - CWCAS2014 (website). Bogota, Colombia: IEEE. doi: 10.1109/CWCAS.2014.6994603.

Heraud, J. A., & Lange, A. F. (2009). Agricultural Automatic Vehicle Guidance from Horses to GPS: How We Got Here, and Where We Are Going. Michigan: American Society of Agricultural and Biological Engineers.

Higuti, V. A. H., Guerrero, H. B., Velasquez, A. E. B., Pinto, R. M., Tinelli, L. M., Magalhães, D. V., Becker, M., Barrero, J. F., & Milori, D. M. B. P. (2015). Low-cost embedded computer for mobile robot platform based on raspberry board. In Proceedings of 23rd ABCM International Congress of Mechanical Engineering - Cobem2015 (website). Rio de Janeiro, Brazil: ABMC. doi: 10.20906/CPS/COB-2015-0336.

Hiremath, S. A., van der Heijden, G. W. A. M., van Evert, F. K., Stein, A., & ter Braak, C. J. F. (2014). Laser range finder model

for autonomous navigation of a robot in a maize field using a particle filter. Computers and Electronics in Agriculture, doi: 10.1016/j.compag.2013.10.005.

Hokuyo Automatic (2012). Scanning Laser Range Finder UTM-30LX/LN Specifications. Product Specifications. Hokuyo Automatic co. http://www.hokuyo-aut.jp/02sensor/07scanner/download/pdf/UTM-30LX_spec_en.pdf. Accessed 20 April 2016.

Maxon Motors ag (2013).EPOS 2 Positioning controllers. Application Notes Collection. Maxon Motors ag. http://www.maxonmotor.com/medias/sys_master/8811457937438/EPOS2-Application-Notes-Collection-En.pdf. Accessed 21 April 2016.

Reina, G., Milella, A., Rouveure, R., Nielsen, M., Worst, R., & Blas, M. R. (2015). Ambient awareness for agricultural robotic vehicles. Biosystems Engineering. doi:10.1016/j.

SBG Systems (2013).IG-500N. Product Specifications. SBG Systems. http://www.sbg-systems.com/docs/IG-500N-Leaflet.pdf. Accessed 20 April 2016.

UNFPA (2015). State of world population. Online Document. ONU. http://www.unfpa.org/sites/default/files/pub-pdf/State_of_World_Population_2015_EN.pdf. Accessed 20 April 2016.

Velasquez, A. E. B. (2015). ℏelvis III – Desenvolvimento e Caracterização da Plataforma Robótica. Masters dissertation. University of Sao Paulo. http://www.teses.usp.br/teses/disponiveis/18/18149/tde-05052015-101452/es.php. Accessed 21 April 2016.

Weiss, U., & Biber, P. (2011). Plant detection and mapping for agricultural robots using a 3D LiDAR sensor. Robotics and Autonomous Systems, doi: 10.1016/j.robot.2011.02.011.