Design of a Sensor-based Controller Performing U-turn to Navigate in Orchards

A. Durand-Petiteville[1], E. Le Flecher[2,3], V. Cadenat[2,3], T. Sentenac[2] and S. Vougioukas[1]
[1]*Department of Biological and Agricultural Engineering, University of California, Davis, CA, 95616, U.S.A.*
[2]*CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France*
[3]*Univ. de Toulouse, UPS, LAAS, F-31400, Toulouse, France*

Keywords: Mobile Robots, Sensor-based Control, Orchard Navigation, Spiral Following.

Abstract: In this work, the problem of designing sensor-based controllers allowing to navigate in orchards is considered. The navigation techniques classically used in the literature rely on path following using metric maps and metric localization obtained from onboard sensors. However, it appears promising to use sensor-based approaches together with topological maps for two main reasons: first, the environment nature is rather changing and second, only high-level information are sufficient to describe it. One of the key maneuver when navigating through an orchard is the u-turn which must be performed at the end of each row to reach the next one. This maneuver is generally performed using only dead reckoning because of the lack of dedicated sensory data. In this paper, we propose two sensor-based control laws allowing to perform u-turns, improving the performance quality. They allow following particular spirals which are defined from laser rangefinder data and adapted to realize the desired maneuver. Their stability is studied and their performances are thoroughly examined. Finally, they are embedded in a complete navigation strategy to show their efficiency in our agricultural context.

1 INTRODUCTION

To meet the demands of nine billion people in 2050, scientists predict that the agricultural production has to double (Foley et al., 2011). Robotics has been identified as one of the solutions with the highest potential to achieve this goal (Reid, 2011). Indeed, robots can improve the efficiency of each process of the crops production: field preparation, seeding/breeding, transplanting, planting, growing, maintenance, harvesting, sorting and packing. To successfully perform most of those tasks, the navigation is one of the key challenges. Indeed the robot has to safely drive up along one row, turn at the end of the row and enter the next one (Siciliano and Khatib, 2016). For open field crops, *e.g.* wheat or lettuce, the navigation strategies mostly rely on path following using GPS-based localization, such as the works presented in (Bak and Jakobsen, 2004), (Fang et al., 2006), (Eaton et al., 2009) and (Johnson et al., 2009). However, these approaches are no more suitable for orchards crops, *e.g.* apples or pears, because of poor satellite reception under thick canopies. It is then required to localize and/or control robots thanks to embedded sensors such as inertial units, cameras, lidars, radars, etc. The work presented in (Barawid et al., 2007) focused solely on straight line recognition of the tree rows using a laser scanner as a navigation sensor. A Hough transform is applied to the points cloud to recognize the tree row, which is then used to autonomously drive the robot in the orchard. In (Hansen et al., 2011), a metric localization is performed thanks to a 2D laser scanner, an odometer and a gyroscope. The collected data are then processed in three derivative free filters in order to localize the robot. In (Zhang et al., 2014), a mapping method takes as input readings from the lidar and encoders as the vehicle is manually driven around the orchard for at least two full rounds. Then, the collected data are processed off-line to generate a metric map, which is then permanently stored on the vehicles on-board computer in order to navigate. In (Bayar et al., 2015), the navigation is performed by following metric paths. To do so, the localization and control are based on a laser range-finder and wheels and steering encoders. All the mentioned approaches rely on a path following using a metric localization. This latter has to deal with slopped and slippery terrains, branches sticking out of the canopy, tall grass, and missing trees. Moreover, the u-turns are usually partially performed using dead-reckoning because of

Durand-Petiteville, A., Flecher, E., Cadenat, V., Sentenac, T. and Vougioukas, S.
Design of a Sensor-based Controller Performing U-turn to Navigate in Orchards.
DOI: 10.5220/0006478601720181
In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2017) - Volume 2*, pages 172-181
ISBN: Not Available

the absence of trees in the field of view of the sensors while following outside-the-row trajectories.

To overcome these limitations, we propose, based on the work presented in (Durand-Petiteville et al., 2015), to design an approach relying on non-metric maps and sensor-based controllers in order to navigate in an orchard. First, having an up to date metric map of the orchards seems challenging because of the changing nature of the environment. Indeed, over the years, the trees grow and are pruned; over the seasons, the leaves grow and disappear; over the days, the fruits grow, bend the branches and finally fall. Moreover, to identify when the robot has to drive through the row or switch from one row to the next one, only high level localization data are required. Topological maps being barely sensitive to local environment modifications (*i.e.* size of the trees, presence or not of leaves, position of the branches, ...) and providing high level localization, they seem relevant to model the orchard. Then, when navigating in an orchard, the robot main goal is not to reach a destination, but to locally behave appropriately. Indeed, it has to drive through a row while staying at a defined distance with respect to the trees when picking fruits, or to follow a local path when switching from one row to the other. For this reason, it looks appropriate to use sensor-based controllers[1].

In this paper, we focus on the sensor-based controllers required to navigate in an orchard. First, driving through the row consists in following the line in the middle of the trees. It exists a large collection of controllers performing such a task, *e.g.* simple PID controllers. To complete the navigation system, the challenge consists in designing a sensor-based controller allowing to perform a u-turn to switch from one row to the next one. In this paper, we propose to address this issue by designing two sensor-based controllers following spirals around a point of interest. The first one allows to follow a spiral whose distance to the center is not known whereas the second one follows a spiral at a known distance from it. The design of the controllers relies on the work presented in (Boyadzhiev, 1999) which describes how insects fly around a point of interest by performing a spiral. In other words, it shows how to follow a path, a spiral in this particular case, by simply knowing the coordinates of a point of interest in the moving object frame. This work has already been used on a UAV (Mcfadyen et al., 2014) and a ground robot (Futterlieb et al., 2014) to design sensor-based controllers avoiding obstacles. The relative position of the obstacle with respect to the robot is used to push the robot away from danger. Thus, the robots modify their behavior without following a path/spiral. The work presented in this paper aims at designing controllers following spirals. Thus, by tracking a point of interest, a tree in our case, it will become possible for the robot to actually follow a path ending in the next row.

The work is presented as follows. In the next section the spiral model is briefly summarized. Then two controllers allowing to perform a spiral following are designed in section 3. Next, some simulations results showing the efficiency of our approach are presented. In a last section, an example of sensor-based navigation architecture is first presented, then simulations of a navigation in an orchard are provided.

[1]Sensor-based controllers only use the data regarding the surrounding environment and provided by the embedded sensors.

2 SPIRAL

Spirals have been studied in (Boyadzhiev, 1999), where the author presents a large variety of equations to model them. The present work focuses on some ideas extracted from (Boyadzhiev, 1999) and more especially on the one claiming that a spiral can be seen as the path described by a point O_p moving on a plane with respect to a fixed point O_s. From now on this point will be considered as the center of the spiral. $\vec{v^*}$ is the velocity vector applied to O_p and its norm is denoted $v^*(t)$. Moreover $\vec{d}^*$ is the vector connecting O_s to O_p whose norm is $d^*(t)$. Finally $\alpha^*(t)$ is defined as the oriented angle between $\vec{v^*}$ and $\vec{d}^*$.

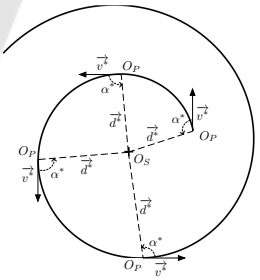
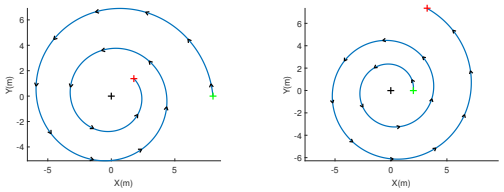


Figure 1: Spiral model.

In (Boyadzhiev, 1999) it is shown that if both $v^*(t)$ and $\alpha^*(t)$ are constant then O_p describes a spiral whose center is O_s. For this reason they are respectively denoted v^* and α^* from now on. Moreover the author provides an important equation regarding $d^*(t)$:

$$\dot{d^*}(t) = -v^* \cos(\alpha^*) \quad (1)$$

As it can be seen in this equation, the type of spiral performed depends on the sole parameter α^*. First if $0 < \alpha^* < \pi$, O_p turns counter-clockwise with re-



(a) Inward spiral - $\alpha^* = \frac{15\pi}{32}$ (b) Outward spiral - $\alpha^* = \frac{17\pi}{32}$

Figure 2: Example of spirals - Black cross: center of the spiral - Green cross: initial position - Red cross: final position.

spect to O_s otherwise if $-\pi < \alpha^* < 0$ it turns clockwise. Then if $0 \leq \alpha^* < \pi/2$ or $-\pi/2 \leq \alpha^* < 0$, $d^*(t)$ decreases with time (see figure 2(a)). In other words, O_p is describing an inward spiral around O_s. If $\pi/2 < \alpha^* \leq \pi$ or $-\pi \leq \alpha^* < -\pi/2$ then $d^*(t)$ increases with time (see figure 2(b)) which means O_p is describing an outward spiral around O_s. Finally, if $\alpha^* = \pi/2$ or $\alpha^* = -\pi/2$, $d^*(t) = d^*(0)$. O_p then describes a circle of radius $d^*(0)$ around O_s.
Equation (1) and its analysis highlight our interest in spirals. Indeed, by adapting the spiral model to a robot and then making converging α, angle between its linear velocity vector and the point of interest O_s, towards α^*, it is possible to make the vehicle follow a spiral defined by α^*. Thus the sensor space is used to control the robot path.

3 CONTROLLER DESIGN

In the previous section, it has been explained how to choose a value for α^* in order to define a spiral. In this section, controllers allowing a robot to perform such a spiral are presented. To do so, we first present the model of the robot and how to adapt the spiral approach to it. Then a first controller following a spiral only defined by its α_B value is designed and its performance is analyzed. Finally a more advanced controller allowing to follow a specific spiral, *i.e.* defined by its α^* value and the initial distance to O_S, is introduced.

3.1 Robot Modeling

In this work we consider the differential robot presented in figure 3. Let $F_w = (O_w, \vec{x}_w, \vec{y}_w, \vec{z}_w)$ and $F_r = (O_r, \vec{x}_r, \vec{y}_r, \vec{z}_r)$ the frames respectively attached to the world and the robot. The robot state vector is defined as $\chi(t) = [x(t), y(t), \theta(t)]^T$, where $x(t)$ and $y(t)$ are the coordinates of O_r in F_w, whereas $\theta(t)$ is the angle between $\vec{x}_w$ and $\vec{x}_r$. Finally the robot is controlled using the input vector $\dot{\chi}(t) = [v(t), \omega(t)]^T$ where $v(t)$ is the linear velocity along $\vec{x}_r$, and $\omega(t)$ the angular velocity around $\vec{z}_r$.

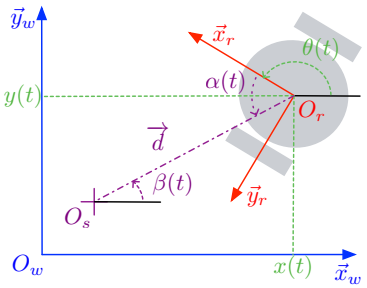


Figure 3: Robot model.

As previously mentioned, this work focuses on following a spiral whose center is denoted O_s. It then requires to adapt the spiral model to the robot. The vector connecting O_s to O_r is named $\vec{d}$ and $\alpha(t)$ is the angle between $\vec{x}_r$ and $\vec{d}$. It should be noticed that:

$$\alpha(t) = \pi - \theta(t) + \beta(t) \tag{2}$$

$$\dot{\alpha}(t) = -\dot{\theta}(t) + \dot{\beta}(t) = -\omega(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) \tag{3}$$

Finally $d(t)$ represents the distance between O_s and O_r, and it can be shown (see (Boyadzhiev, 1999)) that:

$$\dot{d}(t) = -v(t)\cos(\alpha(t)) \tag{4}$$

3.2 First Approach

A first solution to follow a spiral consists in designing a controller which makes $\alpha(t)$ converge to a desired angle named $\alpha_B = \alpha^*$. α_B determines if the spiral is an inward one, an outward one or a circle. Once the robot has converged towards the spiral, it either increases, decreases or keeps constant $d(t)$. In order to design such a controller, we impose a constant linear velocity such as $v(t) = v^*$. Then we define an error $e_B(t)$ and compute $\dot{e}_B(t)$:

$$e_B(t) = \alpha(t) - \alpha_B \tag{5}$$

$$\dot{e}_B(t) = \dot{\alpha}(t) = -\omega(t) + \frac{v}{d(t)} \sin(\alpha(t)) \tag{6}$$

As it can be seen in (6), $\dot{e}_B(t)$ depends on both v and $\omega(t)$. As it has been decided to fix a constant linear velocity, it is then required to compensate its effect while controlling the robot using solely $\omega(t)$. Moreover, to make the error $e_B(t)$ vanish, we propose to define a controller proportional to it. Thus it follows that:

$$\omega(t) = \lambda_B e_B(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) \tag{7}$$

where λ_B is a positive scalar. The controller obtained in equation (7) can be used to force the robot to follow a spiral defined by α_B. Thus it is possible to control if the robot increases, decreases or keeps constant the distance $d(t)$. However it does not allow to follow a specific spiral. In order to analyze the stability of the designed controller, we propose to define the following Lyapunov function:

$$V_B(x_B(t)) = \frac{x_B(t)^2}{2} \tag{8}$$

with $x_B(t) = e_B(t)$. The equilibrium state $x_{BE} = x_B(t) = 0$ corresponds to the robot following the spiral defined by α_B. Moreover $V_B(x_B(t)) > 0$ for $x_B(t) \neq x_{BE}$ and $V_B(x_B(t)) = 0$ when $x_B(t) = x_{BE}$. To study the evolution of $V_B(x_B(t))$, its derivative with respect to time is computed:

$$\begin{aligned} \dot{V}_B(x_B(t)) &= \dot{x}_B(t) x_B(t) \\ &= \dot{\alpha}(t)[\alpha(t) - \alpha_B] \\ &= [-\omega(t) + \tfrac{v}{d(t)} \sin(\alpha(t))][\alpha(t) - \alpha_B] \\ &= -\lambda_B e_B(t)^2 \end{aligned} \tag{9}$$

Thanks to equation (9), it can be seen that $\dot{V}_B(x_B(t)) < 0$ for $x_B(t) \neq x_{BE}$ and $\dot{V}_B(x_B(t)) = 0$ when $x_B(t) = x_{BE}$ as $\lambda_B > 0$. Thus we can conclude that the designed controller has a global exponential stability.

3.3 Second Approach

The controller in equation (7) allows to control the robot behavior, *i.e.* to increase, decrease or keep constant the distance $d(t)$. Nevertheless it does not allow to follow a specific spiral. Indeed it exists an infinity of spirals defined by α_B and in order to define a specific spiral it is mandatory to select both the angle α_B and the initial distance $d^*(0)$. When using the controller in equation (7), $d^*(0)$ is not chosen and it corresponds to $d(t_c)$, where t_c is the time when $\alpha(t)$ has converged to α_B. Thus, in order to follow a specific spiral, we once again impose a constant linear velocity such as $v(t) = v^*$. Then we propose to define the following error:

$$e_S(t) = \alpha(t) - \alpha_S(t) = \alpha(t) - \alpha_B - \alpha_D \varepsilon(t) \tag{10}$$

In order to make the new error $e_S(t)$ vanish, the constant reference α_B has been replaced by a non-constant one, $\alpha_S(t) = \alpha_B + \alpha_D \varepsilon(t)$. The vanishing error in equation (10) should lead to two successive robot behaviors. First while $d^*(t) \neq d(t)$, the robot has to navigate towards the spiral in order to make $d^*(t) = d(t)$. To do so the new angle of reference $\alpha_S(t)$ is equal to α_B modified by the amount $\alpha_D * \varepsilon(t)$. α_D is the maximal value that can be added/subtracted to α_B in order to make the robot converge towards the spiral without changing its sense of rotation with respect to the center of the spiral (clockwise or counter-clockwise). Then $\varepsilon(t)$ should have its norm equal to 1 when $||d(t) - d^*(t)|| \geq ||d(0) - d^*(0)||$, and equal to 0 when $d(t) = d^*(t)$. Thus, when the robot is far from the spiral, it converges towards it using the greater angle possible, *i.e.* $\alpha_S(t) = 0$ or $\alpha_S(t) = \pm\pi$ depending on the initial conditions. Moreover, when $d(t) = d^*(t)$, then $\alpha_S(t) = \alpha_B$. Thus the robot follows the spiral while keeping the desired distance. This last part corresponds to the second behavior expected when controlling the robot by vanishing $e_S(t)$. In order to obtain the previously explained behaviors, we propose to define $\varepsilon(t)$ and α_D as:

$$\varepsilon(t) = \text{sign}(d^*(t) - d(t)) \min\left(|| \frac{d^*(t) - d(t)}{|d^*(0) - d(0)|} ||, 1 \right) \tag{11}$$

and

$$\alpha_D = \begin{cases} sign(\alpha_B) * \pi - \alpha_B & \text{if } d^*(0) > d(0) \\ \alpha_B & \text{if } d^*(0) < d(0) \end{cases} \tag{12}$$

$\varepsilon(t)$ is the normalized error between $d^*(t)$ and $d(t)$. It has been saturated to ± 1. Thus its value belongs to the domain $[0,1]$ if $d^*(0) > d(0)$ or $[-1,0]$ if $d^*(0) < d(0)$. Thus, $\alpha_S \neq \alpha_B$ when $d^*(t) \neq d(t)$, whereas $\alpha_S = \alpha_B$ when $d^*(t) = d(t)$. Regarding equation (12), when $\alpha_B \in [0,\pi]$ then it is mandatory that $\alpha_S \in [0,\pi]$. Otherwise when $\alpha_B \in [0,-\pi]$ then $\alpha_S \in [0,-\pi]$. Equations (11) and (12) guarantee that the robot rotation direction is not modified by $\alpha_D \varepsilon(t)$, while allowing the robot to converge towards $d^*(t)$. Now that the new error has been defined, we propose to compute its derivative with respect to time in order to identify the terms involved in its dynamics.

$$\begin{aligned} \dot{e}_S(t) &= \dot{\alpha}(t) - \alpha_D \dot{\varepsilon}(t) \\ &= -\omega(t) + \tfrac{v(t)}{d(t)} \sin(\alpha(t)) - \alpha_D \dot{\varepsilon}(t) \end{aligned} \tag{13}$$

In order to make $e_s(t)$ vanish, we propose to design a controller proportional to it which also compensate the other terms involved in (13). It leads to:

$$\omega(t) = \lambda_S e_S(t) + \frac{v(t)}{d(t)} \sin(\alpha(t)) - \alpha_D * \dot{\varepsilon}(t) \tag{14}$$

where λ_S is a positive scalar. The controller (14) makes possible for the robot to converge towards any specific spiral. In order to analyze the stability of the controller, we propose to define the following Lyaponuv function:

$$V_S(x_S(t)) = \frac{x_{S1}(t)^2 + x_{S2}(t)^2}{2} \tag{15}$$

with

$$x_S(t) = \begin{bmatrix} x_{S1}(t) \\ x_{S2}(t) \end{bmatrix} = \begin{bmatrix} \alpha(t) - \alpha_B - \alpha_D * \varepsilon(t) \\ d(t) - d^*(t) \end{bmatrix}$$

The equilibrium state $x_{SE} = [x_{S1}(t), x_{S2}(t)]^T = [0,0]^T$ corresponds to the robot following the spiral defined by α_B while being at the distance $d^*(t)$, *i.e.* when $\alpha(t) = \alpha_B$ and $d(t) = d^*(t)$. Moreover, $V_S(x_S(t)) > 0$ for $x_S(t) \neq x_{SE}$ and $V_S(x_S(t)) = 0$ when $x_S(t) = x_{SE}$. In order to study the evolution of $V_S(x_S(t))$, its derivative with respect to time is computed:

$$\begin{aligned} \dot{V}_S(x_S(t)) & \\ &= \dot{x}_{S1}(t)x_{S1}(t) + \dot{x}_{S2}(t)x_{S2}(t) \\ &= -\lambda_S e_S(t)^2 + [\dot{d}(t) - \dot{d}^*(t)][d(t) - d^*(t)] \\ &= -\lambda_S e_S(t)^2 \\ &\quad -v[\cos(\alpha(t)) - \cos(\alpha_B)][d(t) - d^*(t)] \end{aligned} \tag{16}$$

In equation (16), it can be seen that $\dot{V}_S(x_S(t)) = 0$ for $x_S(t) = x_{SE}$. However, it can not be proved that $\dot{V}_S(x_S(t)) < 0$ for $x_S(t) \neq x_{SE}$. Indeed even if $-\lambda_S e_S(t)^2 < 0$ for $x_B(t) \neq x_{SE}$, the second term $v[\cos(\alpha(t)) - \cos(\alpha_B)][d(t) - d^*(t)]$ is not always positive. For certain initial configurations the sign of $\cos(\alpha(t)) - \cos(\alpha_B)$ is different from the one of $d(t) - d^*(t)$. Moreover, the initial robot orientation may not allow it to reduce $||d(t) - d^*(t)||$ when it starts to move. This is due to the fact that only one control input, $\omega(t)$, is used to control the two degrees of freedom $\alpha(t)$ and $d(t)$ of a non-holonomic robot. Thus it might be first required to orientate the robot. During this orientation step the distance $||d(t) - d^*(t)||$ increases. Thus we can conclude that the controller is not globally asymptotically stable. However it is straightforward to show that:

$$\begin{aligned} &\text{If } d(t) > d^*(t): \\ &\quad \cos(\alpha(t)) - \cos(\alpha_B) > 0 \text{ if } \alpha(t) > \alpha_B \\ &\text{If } d(t) < d^*(t): \\ &\quad \cos(\alpha(t)) - \cos(\alpha_B) < 0 \text{ if } \alpha(t) < \alpha_B \end{aligned} \tag{17}$$

As it can be seen in equation (17), it is guaranteed that the controller is locally asymptotically stable once the $\alpha(t)$ overpasses α_B. Thus the obtained results do not completely fulfill our expectations, but are sufficient for our needs. A more accurate value of $\alpha(t)$ allowing a local stability could be found. It would depend on $v(t)$, λ_S and the initial conditions. Finally, as it will be shown in the simulation section, it is possible to sequence these two controllers to bring the robot in an initial state where the stability of the second control law is guaranteed.

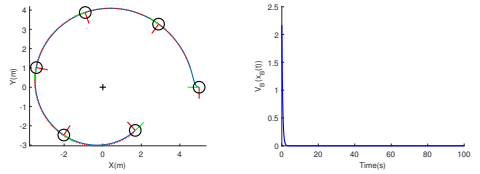
In this section two controllers have been designed. The first one allows to follow a spiral having an unknown distance $d^*(t)$. It can be used when controlling the robot behavior, *i.e.* increasing, decreasing or keeping constant $d(t)$ This controller is sufficient when the distance is not available, *eg.* when the data are provided by a single camera. The second controller allows to follow a spiral while controlling $d(t)$. It is then required to provide $d(t)$, using for example a stereo camera, a Lidar, etc.

3.4 Validation

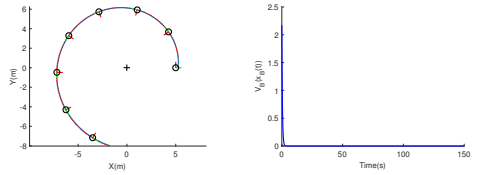
In this section we present simulations of a robot following a spiral using the two previously designed controllers. The results aim to illustrate the efficiency of our approach and also highlight its performances and limitations. Moreover we will propose ideas to investigate in order to overcome some issues and/or adapt the presented approach to another application.

For all the simulations the sampling time is $T_s = 0.1$ second. Moreover the coordinates of the spiral center are $[0,0]$ and the linear velocity is $v^* = v(t) = 0.2$ m/s. The first set of simulations, the results of which can be seen in figure 4, are performed using the first controller from equation (7) and $\lambda_B = 1$. The first test aims to follow an inward spiral by using $\alpha_B = \frac{15\pi}{32}$ when the robot initial state is $\chi(0) = [5,0,\pi]^T$ (see figure 4(a)). The second one follows an outward spiral with $\alpha_B = \frac{17\pi}{32}$ and an initial state $\chi(0) = [5,0,0]^T$ (see figure 4(c)). In both figures 4(a) and 4(c), the dotted red line is the path of reference whereas the solid blue one is the path performed by the robot while using the controller from equation (7). Moreover, some robot states, and especially the first one, are represented. The green and red lines represent $\vec{x}_r(t)$ and $\vec{y}_r(t)$, respectively. In both cases, the robot first orientates itself in order to make $\alpha(t)$ converge towards α_B, then it follows a spiral by keeping $\alpha(t) = \alpha_B$. This behavior is shown in figures 4(b) and 4(d) where the evolution of the Lyapunov function $V_B(x_B(t))$ is presented. Indeed $x_B(t)^2 = (\alpha(t) - \alpha_B)^2$ first decreases down to zero, and then stays at the constant null value. This result matches with the proposed analysis presented in section 3.2. However it should be noticed that the robot does not follow a predefined spiral, but the one met once $V_B x_B(t) = 0$. Thus we can conclude that the controller from equation (7) can be used in order to control the behavior of a robot with respect to a point, *i.e.* it increases, decreases or keeps constant $d(t)$, but it does not allow to follow a specific spiral, *i.e.* control the distance $d(t)$.

The second set of simulations uses the spiral following controller presented in equation (14). For the first test (see figure 5), $\lambda_S = 1$, $\alpha_B = \frac{15\pi}{32}$ and $d^*(0) = 5$ m. Moreover the robot initial state is $\chi(0) = [8,0,-\frac{\pi}{4}]^T$. In this simulation the robot has to follow the specific spiral defined by $\alpha_B = \frac{15\pi}{32}$ and passing by the point



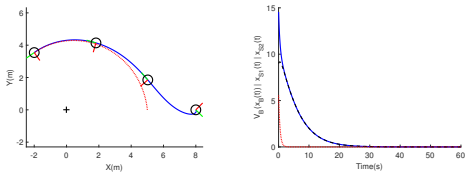
(a) Robot trajectory (b) $V_B(x_B(t)) \mid \alpha_B = \frac{15\pi}{32}$ $\mid \alpha_B = \frac{15\pi}{32}$



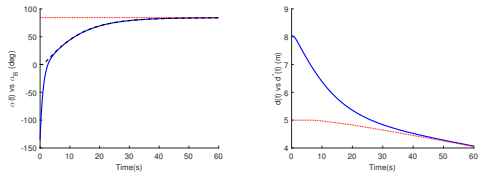
(c) Robot trajectory (d) $V_B(x_B(t)) \mid \alpha_B = \frac{17\pi}{32}$ $\mid \alpha_B = \frac{17\pi}{32}$

Figure 4: Spiral following using the first controller - Black cross: center of the spiral - Solid line: robot trajectory - Doted line: spiral of reference.

whose coordinates are $[5,0]$. In figure 5(a), it can be seen that the robot successfully manages to follow the predefined spiral by simultaneously orienting itself toward the spiral (see figure 5(c)) and making the distance $d(t)$ converge towards $d^*(t)$ (see figure 5(d)). Figure 5(b) presents the evolution of $V_S(x_S(t))$, $x_{S1}(t)$ and $x_{S2}(t)$. It can be seen that $x_{S1}(t) = e_S(t)$ is permanently decreasing and converges towards zero. Then it should be noticed that $x_{S2}(t) = d(t) - d^*(t)$ slightly increases before decreasing down to zero. Moreover it vanishes after $x_{S1}(t)$. This behavior is due to the fact that the non-holonomic robot is initially not well oriented. Thus, while orienting itself, $x_{S2}(t)$ increases. Once the robot is correctly oriented towards the spiral, vanishing $e_S(t)$ allows the robot to converge towards the specific spiral and then to follow it. It can be also seen that $V_S(x_S(t))$ is constantly decreasing and is then equal to zero when the robot is on the spiral. In this test the increase of $x_{S2}(t)$ due to the orientation step is not large enough with respect to the reduction of $x_{S1}(t)$ to not allow $V_S(x_S(t))$ to be constantly decreasing. Those results match the stability analysis proposed in section 3.3. First, $V_S(x_S(t)) = \frac{x_{S1}(t)^2}{2}$ cannot be used as a Lyapunov function to guarantee that the robot follow a spiral. Indeed, $x_{S1}(t) = 0$ before the robot is on the spiral. Then, only an appropriate orientation of the robot guarantees the decrease of $x_{S2}(t)$ and then $V_S(x_S(t))$. However, despite of not being initially well oriented, the robot manages to perform its task successfully. Finally it should be noticed that in our case, the robot is always moving forward. A solution to obtain an monotonically decreasing $x_{S2}(t)$ could be to allow the robot to move backward in order to perform a maneuver. This solution has not been selected in this work because the designed controllers have to be used to perform obstacles avoidance or u-turn while navigating. Moving backward would be an inappropriate motion with respect to such a navigation task.



(a) Solid line: Robot trajectory - Dotted line: reference spiral (b) Solid line: V_B - Dotted line: x_{S1} - Dashed line: x_{S2}

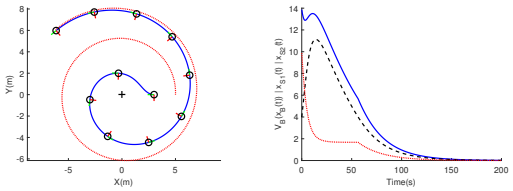


(c) Solid line: $\alpha(t)$ - Dotted line: α_B - Dashed line: $\alpha_S(t)$ (d) Solid line: $d(t)$ - Dotted line: $d^*(t)$

Figure 5: Spiral following using the second controller with $\alpha_B = \frac{15\pi}{32}$.

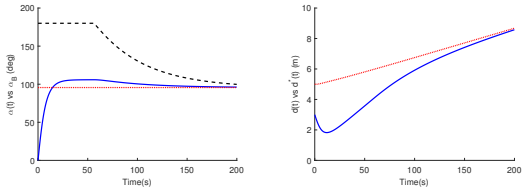
For the second simulation using the controller given by equation (14) (see figure 6), $\lambda_S = 0.2$, $\alpha_B = \frac{17\pi}{32}$ and $d^*(0) = 5$ m. Moreover, the robot initial state is $\chi(0) = [3,0,\pi]$. The robot has to follow the specific spiral defined by $\alpha_B = \frac{17\pi}{32}$ and passing by the point whose coordinates are $[5,0]$. As previously the robot successfully follows the predefined spiral even if the time to converge towards it is much greater than before (see figure 6(a)). Similarly to the previous simulation the initial robot state requires an orientation step leading to an increase of $x_{S2}(t)$ (see figures 6(d) and 6(b)). Moreover, in order to make $x_{S2}(t)$ vanish, $\alpha(t)$ has to overshoot α_B (see figure 6(c)). This is due to the term $\alpha_D * \varepsilon(t)$ in equation (10). However it does not lead to any increase of $x_{S1}(t)$ over the spiral following (see figure 6(b)). In this particular simulation, having $\lambda_B = 0.2$ introduces an increase of $V_S(x_S(t))$ at the beginning of the simulation. Indeed, the variations of $x_{S2}(t)$ due to the orientation step are no more compensated by the strong decrease of $x_{S1}(t)$. One more time the reduction of $V_S(x_S(t))$ is only guaranteed once the robot is correctly oriented with respect to the spiral. Despite this issue in term of stability analysis, the robot manages to converge towards the spiral and then to follow it.

In a last simulation (see figure 7), we propose to use successively both controllers. The spiral to follow, the robot initial state and the parameter values are identi-



(a) Solid line: Robot trajectory - Dotted line: reference spiral

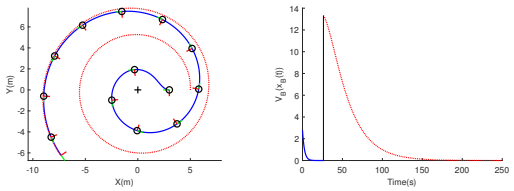
(b) Solid line: V_B - Dotted line: x_{S1} - Dashed line: x_{S2}



(c) Solid line: $\alpha(t)$ - Dotted line: α_B - Dashed line: line: $\alpha_S(t)$

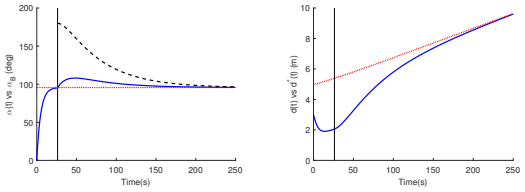
(d) Solid line: $d(t)$ - Dotted line: $d^*(t)$

Figure 6: Spiral following using the second controller with $\alpha_B = \frac{17\pi}{32}$.



(a) Solid line: Robot trajectory - Dotted line: reference spiral

(b) Solid line: V_B - Dotted line: V_S



(c) Solid line: $\alpha(t)$ - Dotted line: α_B - Dashed line: line: $\alpha_S(t)$

(d) Solid line: $d(t)$ - Dotted line: $d^*(t)$

Figure 7: Spiral following using successively both controllers. The switch occurs at $t = 26.2$ s.

cal to the previous simulation. Initially the robot is controlled using the first controller and thus it follows a non-specific spiral having $\alpha_B = \frac{17\pi}{32}$. Once $\alpha(t) = \alpha_B$, the robot then switches to the second controller to follow the spiral having $\alpha(t) = \alpha_B$ and passing by the point with coordinates $[5,0]$. This approach allows to obtain the robot to reach an initial state that guarantees the convergence of the second controller. Indeed, in figure 7(b), it can be seen that both $V_B(x_B(t))$ and $V_S(x_S(t))$ are decreasing. Other than the Lyapunov functions, the path and the evolution of $\alpha(t)$ and $d(t)$ are almost the same as in the previous simulation (figures 7(a) 7(c) and 7(d)). This approach allows to guarantee the convergence of the robot towards a specific spiral despite its initial state, but it does not drastically change its performances.

4 NAVIGATION STRATEGY

In this section we propose to present how the sensor-based spiral controller can be used into a navigation architecture to allow a robot to drive through an orchard. First, a possible navigation is described: robot, sensors, map, controllers and supervision. Then, we present results obtained in simulation to highlight the efficiency of the sensor-based approach.

4.1 Robot and Sensory Data

To navigate through the orchard, we consider the Toro workman MDE vehicle, which is a car-like robot equipped with three laser rangefinders (see figures 8 and 9). The state vector is $\chi_c(t) = [x(t), y(t), \theta(t), \gamma(t)]^T$, with $\gamma(t)$ the steering angle, and the input vector is $u_c(t) = [v(t), \gamma(t)]^T$. It should be noticed that the second term of the control input vector, which used to be an angular velocity for the differential model, is now an angular position. It is then required to convert the angular velocity $\omega(t)$ to an angular position $\gamma(t)$ allowing the robot to rotate by the same amount. To do so, we use the following equation:

$$\gamma(t) = \tan^{-1}\left(\frac{L\,\omega(t)}{v(t)}\right) \tag{18}$$

where L is the distance between the front axle and the rear axle.

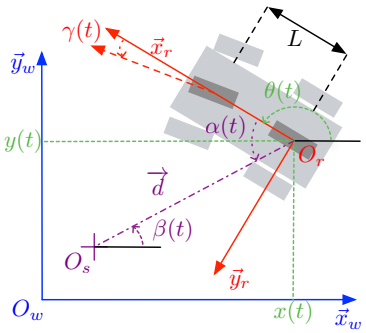


Figure 8: Car-like model.

The first laser rangefinder mounted in front of the robot and facing towards, aims at detecting the trees in order to drive trough the rows. The acquired data are processed in order to identify the trees in the field of view of the laser rangefinder (the circled trees in figure 9). Next the trees are separate into two clusters: left row (red circles) and right row (green circles). Next, the best fitting lines Δ_L and Δ_R are calculated for the left and right cluster respectively. Then, Δ_M, the line median to Δ_L and Δ_R is computed. Finally, the orientation error ε_θ and the lateral position error ε_Y are calculated in the front rangefinder frame $F_f = (O_f, \vec{x}_f, \vec{y}_f, \vec{z}_f)$[2] (see figure 9). These last two values are used to drive through the row.

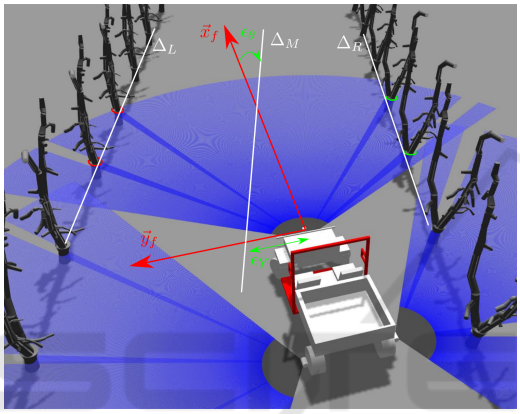


Figure 9: Data acquisition during a row following.

Two laser rangefinders are mounted at the back of the robot and face the right and left sides respectively (see figure 9). They are used to detect the trees when performing a u-turn. The data acquired in the rangefinder frame are first projected in the robot frame. Next, they are processed to detect the closest tree. Then, the angle $\alpha(t)$ and the distance $d(t)$ are computed in the robot frame (see figure 10). These values are used to control the robot while performing a u-turn.

4.2 Environment Modeling

In this simulation, we consider an orchard composed of 4 rows made of 8 trees (see figure 11). It forms tree rows for the robot to navigate through.The rows width is 8 meters whereas the trees are spaced by 3 meters. To model the orchard, we have chosen to use the topological map presented in figure 12 which consists in an oriented graph. Here, the nodes (U_1, D_1), (U_2, D_2) and (U_3, D_3) represent the first, second and third row respectively. U_x is used when the robot drives upward

[2]These parameters have to expressed in the robot frame. However, their values are the same in F_r and F_f.

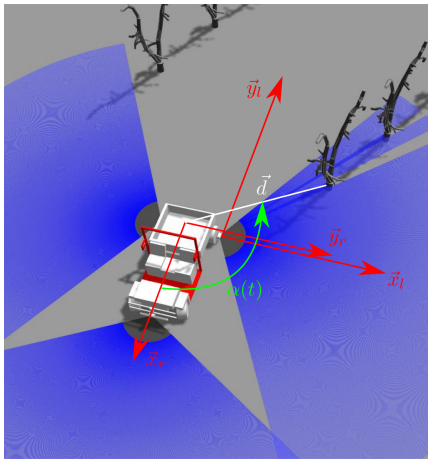


Figure 10: Data acquisition during a u-turn.

in row x, whereas D_x is used when it drives downward. The nodes R_{yz} and L_{yz} represent the right and left turns from row y towards row z. At the beginning of the navigation, the user provides initial and final nodes. Thanks to these informations, a path is computed. For example from U_3 to U_1 the path is $P = \{U_3, L_{32}, D_2, R_{21}, U_1\}$.

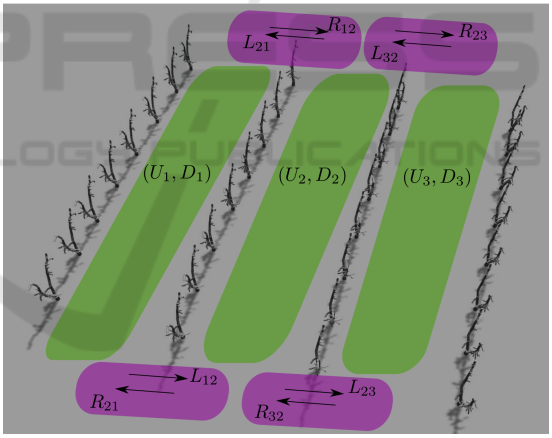


Figure 11: Simulated orchard.

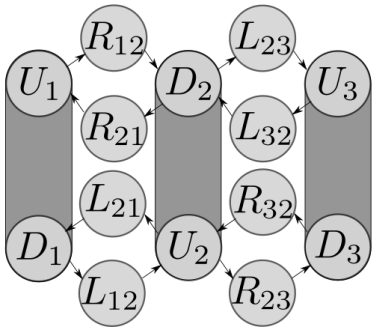


Figure 12: Topological map.

4.3 Control Strategy

Over the navigation, the robot is controlled using two controllers. The first one allows the robot to drive through the rows while staying in the middle of it. It is defined as:

$$\omega = \lambda_\theta \varepsilon_\theta + \lambda_Y \varepsilon_Y \quad (19)$$

where λ_θ and λ_Y are two positive scalar gains. Next, the controller to perform a u-turn, is one of the two ones previously presented, *i.e.* either controller (7) or (14). Finally, it has to be determined when the robot has to switch from one controller to the other based on the data provided by the on-board sensors. First, it is required to switch from the row following controller to the u-turn one when i) the front laser range finder does not perceive a tree anymore, ii) $\alpha_{l,r} < |\pi/2 + \varepsilon_s|$, where $\alpha_{l,r}$ are the $\alpha(t)$ values provided by the left and right camera and ε_s is a very small angle. In other words, the robot has to be at the end of the row, with the two side laser rangefinders at the level of the last two trees. Finally, the robot switches from the u-turn mode to the row following one, when the side sensor which is not used to perform the u-turn perceives two trees. It should be noted that the linear velocity is defined by the user when both following the rows and performing a u-turn.

4.4 Validation

The presented navigation strategy has been implemented in ROS and coupled with Gazebo in order to provide a simulation environment. The following gains have been selected: $\lambda_\theta = 1$, $\lambda_Y = 1$, $\lambda_B = 5$ and $\lambda_S = 5$. Moreover the linear speed is $v(t) = 2$ m/s when the robot is in the rows, whereas $v(t) = 1$ m/s when it performs a u-turn. Finally $\alpha_B = \pm\pi/2$, forcing the robot to follow a circle when switching rows. During the first simulation, the robot starts at D_1 and follows the path $P = \{D_1, L_{12}, U_2, R_{23}, D_3\}$. In order to switch rows, controller (7) is used. As it can be seen in figure 13, the robot successfully switch from one controller to the other in order to navigate through the orchard and reach its destination. During L_{12} and R_{23}, the distance to the trees is not controlled. However, because it is initially located at the center of the row, the robot manages to switch rows.

In a second simulation the controller (14) is used to switch rows. The robot starts in U_1 and has to follow the path $P = \{U_1, R_{12}, D_2, L_{23}, U_3\}$. Once again, it manages to use both controllers to drive through the orchard and reach the goal. In R_{12} and L_{23}, the distance to the trees is controlled and fixed to 4 m. It allows to guarantee that the robot will enter the n the middle of the following row.

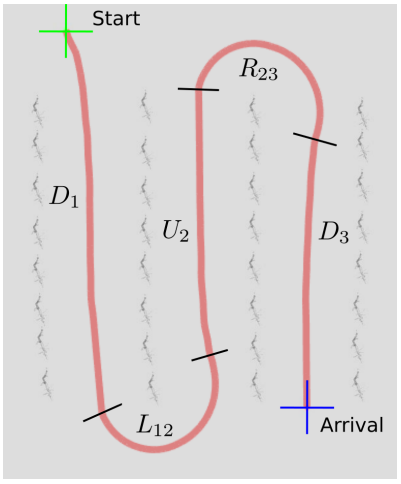


Figure 13: Orchard navigation #1.

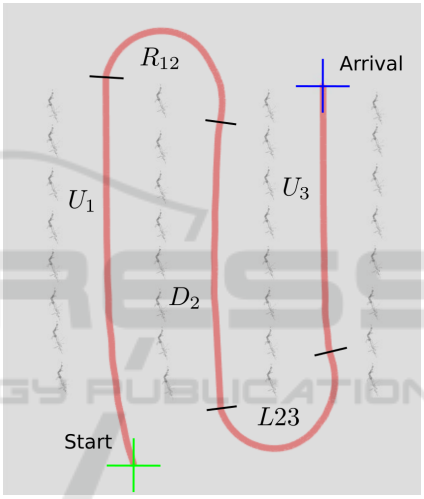


Figure 14: Orchard navigation #2.

5 CONCLUSION

We have considered the problem of designing sensor-based controllers allowing to navigate in orchards. More particularly, we have focused on the realization of u-turns which are traditionnally realized using deadreckoning only. To improve the quality of the maneuver, we have developed two sensor-based controllers. They allow to follow particular spirals which are defined from laser data and adapted to realize the desired maneuvre. It is important to note that the performances are different for both controllers. The first one allows to follow a spiral, but the distance to its center is not controlled. The second controller is more advanced and allows to follow a specific given spiral, *i.e.* angle $\alpha(t)$ and distance $d(t)$ are controlled. In this approach data from the sensor space are sufficient to

control the robot path. No localization are required with the proposed method. In addition a stability analysis of those controllers and simulations highlighting the efficiency of our approach have been provided. In a second step, we have embedded them in a complete navigation strategy allowing to navigate through an orchard. We have explained how the different data necessary for the control are derived, how the environment is modeled and how the robot is controlled using the appropriate controllers. Finally a complete simulation using ROS and Gazebo shows the efficiency of the chosen approach in our agricultural context. The next step of our work is to implement the controllers on the Toro workman MDE robot. Finally, we also plan to develop this sensor-based approach and take advantage of its reactivity in order to navigate in a dynamic environment.

REFERENCES

Bak, T. and Jakobsen, H. (2004). Agricultural robotic platform with four wheel steering for weed detection. *Biosystems Engineering*, 87(2):125–136.

Barawid, O. C., Mizushima, A., Ishii, K., and Noguchi, N. (2007). Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering*, 96(2):139–149.

Bayar, G., Bergerman, M., Koku, A. B., and ilhan Konukseven, E. (2015). Localization and control of an autonomous orchard vehicle. *Computers and Electronics in Agriculture*, 115:118–128.

Boyadzhiev, K. N. (1999). Spirals and conchospirals in the flight of insects. *The College Mathematics Journal*, 30(1):21–31.

Durand-Petiteville, A., Cadenat, V., and Ouadah, N. (2015). A complete sensor-based system to navigate through a cluttered environment. In *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*, volume 2, pages 166–173. IEEE.

Eaton, R., Katupitiya, J., Siew, K. W., and Howarth, B. (2009). Autonomous farming: Modelling and control of agricultural machinery in a unified framework. *International journal of intelligent systems technologies and applications*, 8(1-4):444–457.

Fang, H., Fan, R., Thuilot, B., and Martinet, P. (2006). Trajectory tracking control of farm vehicles in presence of sliding. *Robotics and Autonomous Systems*, 54(10):828–839.

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., Polasky, S., Rockström, J., Sheehan, J., Siebert, S., Tilman, D., and Zaks, D. P. M. (2011). Solutions for a cultivated planet. *Nature*, 478:337–342.

Futterlieb, M., Cadenat, V., and Sentenac, T. (2014). A Navigational Framework Combining Visual Servoing and Spiral Obstacle Avoidance Techniques. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)* , pages 57–64, Vienna, Austria. INSTICC.

Hansen, S., Bayramoglu, E., Andersen, J. C., Ravn, O., Andersen, N., and Poulsen, N. K. (2011). Orchard navigation using derivative free kalman filtering. In *American Control Conference (ACC), 2011*, pages 4679–4684. IEEE.

Johnson, D. A., Naffin, D. J., Puhalla, J. S., Sanchez, J., and Wellington, C. K. (2009). Development and implementation of a team of robotic tractors for autonomous peat moss harvesting. *Journal of Field Robotics*, 26(6-7):549–571.

Mcfadyen, A., Durand-Petiteville, A., and Mejias, L. (2014). Decision strategies for automated visual collision avoidance. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 715–725.

Reid, J. (2011). The impact of mechanization on agriculture. *Nat. Acad. Eng. Bridge, Issue Agric. Inf. Technol.*, 41(3):22–29.

Siciliano, B. and Khatib, O. (2016). *Springer handbook of robotics*. Springer.

Zhang, J., Maeta, S., Bergerman, M., and Singh, S. (2014). Mapping orchards for autonomous navigation. In *Proc. American Society of Agricultural and Biological Engineers Annu. Int. Meeting*.