Visual servoing for path reaching with nonholonomic robots

Andrea Cherubini, F. Chaumette, G. Oriolo

CAMBRIDGE UNIVERSITY PRESS ROBOTICA

Visual servoing for path reaching with nonholonomic robots

Journal:	*Robotica*
Manuscript ID:	ROB-REG-10-0115.R2
Manuscript Type:	Regular Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Cherubini, Andrea; INRIA Rennes - Bretagne Atlantique Chaumette, Francois; INRIA Rennes - Bretagne Atlantique Oriolo, Giuseppe; Dipartimento di Informatica e Sistemistica - Universita' di Roma "La Sapienza"
Keywords:	Control of Robotic Systems, Mobile Robots, Visual Servoing, Navigation, Automation
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
VisionBasedPathReaching-Robotica.tex figure.tar.gz VisionBasedPathReaching.mp4	

Visual servoing for path reaching with nonholonomic robots

Andrea Cherubini
INRIA Rennes - Bretagne Atlantique
Campus de Beaulieu
35042 Rennes, France
acherubi@irisa.fr

François Chaumette
INRIA Rennes - Bretagne Atlantique
Campus de Beaulieu
35042 Rennes, France
chaumette@irisa.fr

Giuseppe Oriolo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Ariosto 25, 00185 Roma, Italy
oriolo@dis.uniroma1.it

Abstract

We present two visual servoing controllers (*pose-based* and *image-based*) enabling mobile robots with a fixed pinhole camera to reach and follow a continuous path drawn on the ground. The first contribution is the theoretical and experimental comparison between pose-based and image-based techniques for a nonholonomic robot task. Moreover, our controllers are appropriate not only for path following, but also for path reaching, a problem that has been rarely tackled in the past. Thirdly, in contrast with most works, which require the path geometric model, only two path features are necessary in our image-based scheme, and three in the pose-based scheme. For both controllers, a convergence analysis is carried out, and the performance is validated by simulations, and outdoor experiments on a car-like robot.

1 Introduction

In recent research, automatic vehicle guidance is often done by utilizing vision sensors [1], which are very useful especially in urban environments, where numerous visual 'points of interest' exist. In a city, cameras can replace or integrate, GPS data [2], since satellite signals can be masked by tall buildings. Various participants of the DARPA Urban Challenge[1] exploit vision [3]. Appearance-based navigation, consisting of replaying a topological path defined by a set of images, has been accomplished in [4, 5]. Apart from navigation, other mobile robot tasks, that exploit camera data include localization [6, 7]. One of the

[1] www.darpa.mil/grandchallenge

prominent methods in vision-based navigation is visual servoing [8], which was originally developed for manipulators with a camera on their end-effector [9], but has also been applied on nonholonomic mobile robots [10]. In some cases, the method relies on the geometry of the environment and on other metrical information. In this case, *pose-based* visual servoing is used to reduce the error, which is estimated in pose space. Other visual navigation systems use no explicit representation of the environment. In this case, *image-based* visual servoing techniques can be used to reduce the error, which is measured directly in the image. The image-based approach eliminates the need for image interpretation, and errors due to camera modeling and calibration. Applying such techniques on wheeled robots, involves well known problems related to the nonholonomic constraint: the linearization of these systems is uncontrollable, and smooth feedback laws stabilizing these systems do not exist.

In this paper, we focus on the *path reaching task*: the controller must zero some suitable error function, indicating the robot pose with respect to a path. The path is a curve drawn on the ground, and the features used for control are at the intersection of the curve with the image borders. As we will show in the non-exhaustive survey below, the path following problem [11, 12] has been tackled in recent research. However, for path following, the initial error is assumed small (i.e., the robot is already on the path), whereas in the path reaching problem, the initial error can be arbitrarily large. The controller in [13] regulates the lateral displacement and orientation of the vehicle at a *lookahead* distance. Frezza and others [14] approximate the path by feasible cubic B-splines, and apply feedback linearization on the derivatives of the splines at the wheel axle. In [15] and [16], straight line following is implemented, respectively for a car-like, and hexapod robot. All these controllers are *pose-based*, and require a complete geometric representation of the path. In other works, *image-based* techniques have been used to avoid complete knowledge of the path geometry. In [17], a straight line follower for a mobile robot with para-catadioptric camera is presented. Coulaud and others [18] design a novel controller and discuss the stability of an equilibrium trajectory: for circular paths, asymptotic stability is guaranteed, whereas for continuously differentiable path curvature, the tracking error is bounded. In [19], the path is approximated by a fuzzy function, which drives the steering velocity. Two-step techniques enable robot pose regulation, using a ceiling camera, in [20].

We present two controllers (*pose-based* and *image-based*), enabling nonholonomic robots with a fixed pinhole camera to reach and follow a continuous path on the ground. The development of the two schemes has been described in [21], and [22]. The contribution of the present article with respect to those papers is the comparison of the two schemes. To our knowledge, a comparison between pose-based and image-based visual servoing for nonholonomic robot navigation has never been carried out. To achieve this, various theoretical details, and new experimental results and simulations, not present in [21], and [22], have been added here. From a theoretical viewpoint, a unique representation is given for all the closed-loop systems, leading to a unique formulation of the convergence analysis. Moreover, in contrast with [21], and [22], we numerically verify the closed-loop convergence of the two schemes, to get a deeper insight on their applicability and characteristics.

The contributions of our work are listed below.

1. An image-based and a pose-based path reaching approach, are compared.

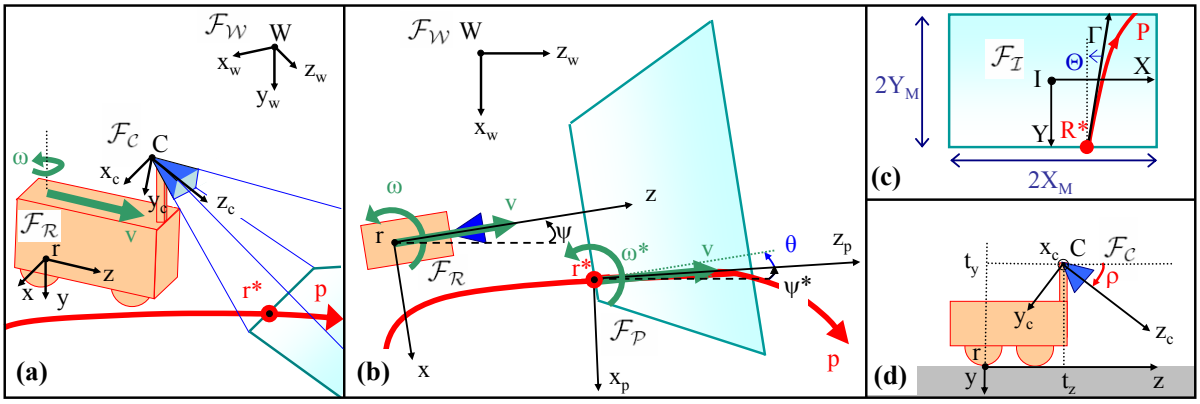

Figure 1: Relevant variables, and reference frames $\mathcal{F}_\mathcal{W}$, $\mathcal{F}_\mathcal{R}$, $\mathcal{F}_\mathcal{C}$, $\mathcal{F}_\mathcal{P}$ and $\mathcal{F}_\mathcal{I}$. The task for the robot (represented in orange), equipped with a fixed pinhole camera (blue) is to follow the red path p. The camera field of view and its projection on the ground are represented in cyan. (a) Perspective view: (b) Top view: robot configuration, desired configuration, applied (v, ω) and tracking (v^*, ω^*) control variables. (c) Image plane view. (d) Side view.

2. Under certain conditions, convergence is guaranteed even when the initial error is large. For this reason, we claim that our controllers are appropriate not only for path following, but also for path *reaching*, a problem that has not been tackled in the cited works, which impose constraints on the initial configuration.
3. As opposed to most approaches, which require a geometric model of the path, in our image-based scheme, only two features (the position of a path point, and the path tangent orientation at that point) are necessary.
4. The system is validated in an outdoor environment, with varying light, in contrast with most cited papers, where tests have been carried out only indoor, where controlled light facilitates image processing.
5. A convergence analysis is carried out.

The paper is organized as follows. In Sect. 2, the problem and variables are defined, along with the controllers, which are detailed in Sections 3 and 4. In Sect. 5, a convergence analysis is carried out. The experiments are presented in Sect. 6-8.

2 Problem statement

2.1 Definitions

We focus on the path reaching task for nonholonomic mobile robots equipped with a fixed pinhole camera. The ground is planar, and the *path* p to be followed is a curve which is drawn on the ground. For the pose-based control scheme, we assume that the curve is twice differentiable in $\mathbb{R}^2$. For the image-based control scheme, the curve can be differentiable only once. A *following direction* is associated to the path (see Fig. 1). We name r the point on the robot sagittal plane that should track the path. We define the frames (see Fig. 1): world frame $\mathcal{F}_\mathcal{W}\,(W, x_w, y_w, z_w)$, robot frame $\mathcal{F}_\mathcal{R}\,(r, x, y, z)$ and image frame $\mathcal{F}_\mathcal{I}(I, X, Y)$ (I

is the image plane center). The image width and height are respectively $2X_M$ and $2Y_M$. The robot configuration in $\mathcal{F}_\mathcal{W}$ is:

$$q = [x_w \; z_w \; \psi]^\top$$

where x_w and z_w represent the Cartesian position of r in $\mathcal{F}_\mathcal{W}$, and $\psi \in (-\pi, +\pi]$ is the positive counterclockwise orientation of z with respect to z_w. The camera optical axis has a constant tilt offset $0 < \rho < \frac{\pi}{2}$ with respect to the z axis, and the optical center C is positioned in the robot sagittal plane at:

$$\begin{cases} x = 0 \\ y = t_y \\ z = t_z \end{cases}$$

with $t_y < 0$, and $t_z \in I\!R$. We also define the camera frame $\mathcal{F}_\mathcal{C}(C, x_c, y_c, z_c)$, (see Fig. 1(d)). We denote the control variables by: $u = [v \; \omega]^\top$. These are the driving and steering velocities (positive counterclockwise) of the robot. Point r is chosen as the center of rotation. Then, the state equation of the robot in the world frame is:

$$\dot{q} = \begin{bmatrix} -\sin\psi \\ \cos\psi \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{1}$$

Although in this equation we have utilized a unicycle robot model, our approach can be extended to other vehicles (e.g., car-like).

2.2 Path reaching task

We hereby define the *path reaching* task, by recalling the characteristics of *path following*. The difference is that in path reaching the initial error can be arbitrarily large. Recalling [12], the goal of path following is to drive the robot configuration q to the desired configuration (shown in Fig. 1(b)): $q^* = [x_w^* \; z_w^* \; \psi^*]^\top$, such that:

- point $r^* = [x_w^* \; 0 \; z_w^*]^\top$ on the path p defines the desired robot position,
- $\psi^* \in (-\pi, +\pi]$ defines the desired robot orientation, i.e., the orientation of the path tangent at r^* in $\mathcal{F}_\mathcal{R}$. Note that ψ^* is always defined, since we have assumed that the path curve can be expressed by a differentiable function,
- $u^* = [v^* \; \omega^*]^\top$ is the tracking control at the desired state q^*.

In practice, the goal of driving q to q^* is equivalent to zeroing the error: $e = q - q^* \in I\!R^3$. Furthermore, in path following, this goal must be achieved under two conditions:

1. In opposition to the *trajectory tracking problem* [11], where the desired configuration is determined by a rigid law, (e.g., associated to time: $q^* = q^*(t)$), in path following we can arbitrarily choose the relationship that defines the evolution of q^*. Such relationship, called *path following constraint*, eliminates 1 of the 3 state components, so that the task consists of zeroing a new two-dimensional error $e = s - s^* \in I\!R^2$, by using appropriate control inputs v and ω. The state dynamics are:

$$\dot{s} = \mathbf{J}(s)u = \mathbf{J}_v(s)\, v + \mathbf{J}_\omega(s)\, \omega \tag{2}$$

where $\mathbf{J}_v$, and $\mathbf{J}_\omega$ are the columns of the Jacobian $\mathbf{J}(s)$ that relates u to $\dot{s}$.

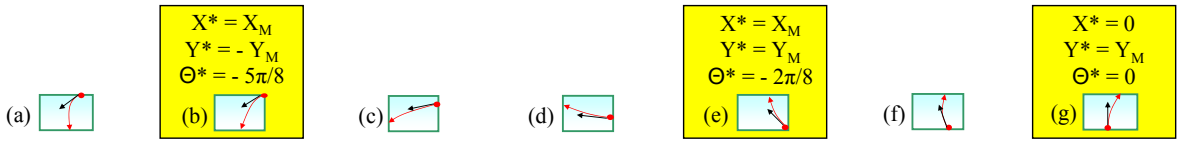

Figure 2: Outline of the control scheme, with 7 possible configurations of P in the image, including the 3 equilibrium configurations (yellow).

2. The robot should move at all times, while the control law ensures convergence to the path. This is the *motion exigency* condition defined in [12]:

$$|u| \neq 0 \qquad \forall s \in I\!R^2 \tag{3}$$

In all the cited works, as well as here, motion is restricted to the forward direction ($v > 0$) for security reasons (sensing obstacles in front is easier on most robots).

2.3 Control design

Since in our work the camera is the only sensor available, we want to ensure path visibility at all times. Hence, we shall use a path following constraint that keeps r^* in the camera field of view. The path following constraint that we chose will be detailed later in the paper. Similarly to [16], [18], and [19], since the only obstacle detecting sensor on our CyCab is a range scanner that points forward, we express the motion exigency as:

$$v = v^* = \text{const} > 0 \tag{4}$$

and we apply a nonlinear feedback on ω based on the features of a visible path point. Under the assumption that a portion of the path is initially visible, we utilize the features of the *first* (considering the path direction) visible path point r^*, of coordinates $r^* = [x\ 0\ z]^\top$ in $\mathcal{F}_\mathcal{R}$, which is projected to $R^* = [X\ Y]^\top$ on the image plane (see Fig. 1(c)). We denote by: P the projection of the path on the image plane, Γ the oriented (according to the path direction) tangent of P at R^* and $\Theta \in (-\pi, \pi]$ the angular offset from Γ to the $-Y$ axis (positive counterclockwise). Note that Γ and Θ are always defined, since we have assumed that the path curve is differentiable in $\mathcal{F}_\mathcal{W}$, and this property is preserved in $\mathcal{F}_\mathcal{I}$.

In both control schemes that we propose (*pose-based* and *image-based*), the task is defined by the path image features. As shown in Fig. 2(g), it consists of driving R^* to the bottom pixel row of the image plane with vertical tangent:

$$X^* = 0, \qquad Y^* = Y_M, \qquad \Theta^* = 0$$

Depending on the position of R^* in the image, we use either of two primitive controllers: a *row*, and a *column controller*. In both primitive controllers, the task is to drive the path features to a desired configuration, while R^* is constrained to a line in the image: a row of pixels (constant Y) in the first case, and a column (constant X) in the second case. These conditions determine the path following constraint introduced in Sect. 2.2. By using both controllers, the path can be reached from general initial configurations.

This is one of our main contributions. For example, the authors of [18], which use a similar approach, assume that in the initial configuration, the path already intersects the bottom

pixel row, and this implies a bound on the initial position error. Instead, we simply assume that the path is visible.

Let us focus on the initial configuration where R^* is on the top row of the image (Fig. 2(a)). Initially, the row controller must be used to drive R^* to a lateral column. The column selected depends on the initial value of Θ: for $\Theta \geq 0$ (respectively, $\Theta < 0$) the left (right) column is chosen. For the case in Fig. 2, the right column is selected. The equilibrium configuration for the top row controller (TRC), is the top right corner, with desired tangent orientation $\Theta^* = -5\pi/8$ rad (Fig. 2(b)). Then, the right column controller (RCC) will be used to drive R^* (Fig. 2(c) and 2(d)) to the bottom right corner: $\Theta^* = -2\pi/8$ rad (Fig. 2(e)). Finally, the bottom row controller (BRC) will drive R^* along the bottom row (Fig. 2(f)) to the center, with vertical tangent $\Theta^* = 0$ (Fig. 2(g)). If the left column is initially selected, the equilibrium configurations are symmetric to the ones in the figure, and they are considered reached by thresholding the point position error, i.e., the coordinates of R^*.

This composition of row and column controllers will be used in both pose-based and image-based control schemes, although the state variables will be different (i.e., defined in the pose, or in the image space). In the first case, the 3D path error dynamics, will depend on the curvature at the desired point, denoted c^*, and related to the tracking control inputs by:

$$c^* = \frac{\omega^*}{v^*} \tag{5}$$

Instead, in the image-based scheme X, Y, and Θ will be used, without taking into account the curvature. In both cases, by imposing the motion exigency (4), system (2) becomes:

$$\dot{s} = \mathbf{J}_v(s)v^* + \mathbf{J}_\omega(s)\omega \tag{6}$$

and we apply the following feedback control:

$$\omega = -\mathbf{J}_\omega{}^+ (\lambda e + \mathbf{J}_v v^*) \tag{7}$$

with $\lambda > 0$, and $\mathbf{J}_\omega{}^+$ the Moore-Penrose matrix pseudoinverse of $\mathbf{J}_\omega$. Control (7) allows $\|\omega\|$ to be minimal, under condition (3).

In practice, condition (3) is guaranteed by (4), and the term $\mathbf{J}_v v^*$ compensates the feature displacements due to the known driving velocity. In the next sections, we will instantiate this formulation for the two control schemes.

3 Pose-based path follower

3.1 Deriving the path 3D features

For the pose-based approach, the path 3D features in $\mathcal{F}_\mathcal{R}$ must be derived from the image features, by considering a pinhole camera model. The four camera parameters used for projecting are: the focal length in pixels f, and ρ, t_y and t_z (see Fig. 1(d)). For simplicity, let us consider a *normalized perspective camera model*:

$$X = \frac{x}{z} \qquad\qquad Y = \frac{y}{z}$$

The mapping between the $\mathcal{F}_\mathcal{I}$ and $\mathcal{F}_\mathcal{C}$ coordinates of a ground point gives:

$$x_c = \frac{X t_y}{\sin\rho + Y\cos\rho}$$
$$y_c = \frac{Y t_y}{\sin\rho + Y\cos\rho}$$
$$z_c = \frac{t_y}{\sin\rho + Y\cos\rho}$$

Note that these equations do not present singularities, since by construction the image projection of any ground point has $Y > -\tan\rho$. Then, the robot frame coordinates of the ground point can then be easily derived, by using the homogeneous transformation from $\mathcal{F}_\mathcal{C}$ to $\mathcal{F}_\mathcal{R}$. For the orientation of the tangent at r^*, we obtain:

$$\theta = \text{ATAN2}\left(\sin\Theta\left(\sin\rho + Y\cos\rho\right) - X\cos\Theta\cos\rho,\ \cos\Theta\right)$$

To derive c^*, the path points 'near' R^* are first projected to $\mathcal{F}_\mathcal{R}$. Then, the equation of the path osculating circle in r^* (thus, the value of c^*) is derived by least square interpolation.

3.2 Row controller

The goal of the pose-based row controller is to drive (x, z, θ) to a desired state (x^*, z^*, θ^*) while constraining R^* to a row in the image: $Y = \text{const} = Y^*$. This is equivalent to zeroing the error $e = [x - x^* \ \ \theta - \theta^*]^\top$, while constraining r^* to the projection of the row on the ground (see Fig. 1(b)), which is equivalent to applying the path following constraint:

$$z = \text{const} = z^* = \frac{t_y}{\sin\rho + Y^*\cos\rho}$$

This equation can be projected in the *path frame* $\mathcal{F}_\mathcal{P}$ (see Fig. 1(b)), where the robot coordinates are: $r = [x_p\ 0\ z_p]^\top$. Frame $\mathcal{F}_\mathcal{P}$ lies on the path, with origin at r^*, y_p parallel to y and z_p coincident with the path tangent at r^* in the following direction. For the nonholonomic model (1), the errors dynamics in $\mathcal{F}_\mathcal{P}$ can be derived, as we have shown in [21], to obtain:

$$\begin{cases} \dot{x}_p = \omega^*\ z_p - v\ \sin\theta \\ \dot{z}_p = -v^* - \omega^*\ x_p + v\ \cos\theta \\ \dot{\theta} = \omega - \omega^* \end{cases} \tag{8}$$

Then, plugging (8) into $\dot{z} = \frac{d}{dt}\left(x_p\sin\theta - z_p\cos\theta\right) = 0$, leads to:

$$v - v^*\cos\theta + \omega x = 0 \tag{9}$$

This expression of the path following constraint relates the desired (v^*) and applied (v) driving robot velocities. Similarly, replacing (8) and (9) in the expression of $\dot{x}$, yields:

$$\dot{x} = (\tan\theta)\, v + (z^* + x\tan\theta)\,\omega$$

On the other hand, replacing (5) and (9) in the third equation of (8) leads to:

$$\dot{\theta} = \left(-\frac{c^*}{\cos\theta}\right) v + \left(1 - \frac{c^* x}{\cos\theta}\right)\omega$$

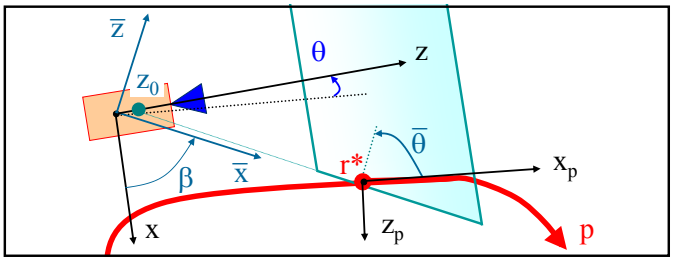

Figure 3: Relevant variables utilized for the pose-based column controller: frames $\mathcal{F}_\mathcal{R}$, $\mathcal{F}_\mathcal{P}$, and $\bar{\mathcal{F}}_\mathcal{R}$, robot current and desired configuration, z_0 and β.

Hence, the system state equations are:

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} = \mathbf{J}_v v + \mathbf{J}_\omega \omega \qquad \text{with:} \quad \mathbf{J}_v = \begin{bmatrix} \tan\theta \\ -\frac{c^*}{\cos\theta} \end{bmatrix}, \ \mathbf{J}_\omega = \begin{bmatrix} z^* + x\tan\theta \\ 1 - \frac{c^* x}{\cos\theta} \end{bmatrix} \tag{10}$$

under the constraint that $|\theta| \neq \frac{\pi}{2}$, which can be avoided by temporarily using the pose-based column controller while Γ is parallel to X.

By imposing the motion exigency (4), the system state equations (10) become:

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} = \mathbf{J}_v v^* + \mathbf{J}_\omega \omega$$

This system can be controlled using the feedback law:

$$\omega = -\mathbf{J}_\omega{}^+ \left(\lambda e + \mathbf{J}_v v^*\right)$$

3.3 Column controller

The goal of the pose-based column controller is to drive (x, z, θ) to a desired state (x^*, z^*, θ^*) while constraining R^* to a column in the image: $X = \text{const} = X^*$. This is equivalent to constraining r^* to the projection of the column on the ground (see Fig. 3), i.e. to the line:

$$z = z_0 + x\tan\beta$$

where z_0 and $\beta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ (shown in Fig. 3) are:

$$\begin{cases} z_0 = t_z - t_y \tan\rho \\ \beta = \text{ATAN2}\left(1, X\cos\rho\right) \end{cases}$$

Let us redefine the variables in a new frame $\bar{\mathcal{F}}_\mathcal{R}(r, \bar{x}, \bar{y}, \bar{z})$, obtained by rotating $\mathcal{F}_\mathcal{R}$ by β around $-y$ (Fig. 3). In $\bar{\mathcal{F}}_\mathcal{R}$, denoting by $\bar{\theta} = \theta + \beta$ the orientation error between z_p and $\bar{z}$, the task will consist of zeroing: $e = \left[\bar{x} - \bar{x}^* \ \ \bar{\theta} - \bar{\theta}^*\right]^\top$, under the path following constraint:

$$\bar{z} = \text{const} = \bar{z}^*$$

with $\bar{z}^* = z_0 \cos\beta$. Hence, as before, but in $\bar{\mathcal{F}}_\mathcal{R}$, using (8), simple calculations yield:

$$v^* \cos\bar{\theta} - v\cos\beta - \omega\bar{x} = 0 \tag{11}$$

and using (8) and (11) gives:

$$\dot{\bar{x}} = \left(\tan\bar{\theta}\cos\beta - \sin\beta\right) v + \left(\bar{z}^* + \bar{x}\tan\bar{\theta}\right)\omega$$

On the other hand, replacing (5) and (11) in the third equation of (8) leads to:

$$\dot{\bar{\theta}} = \left(\frac{-c^* \cos\beta}{\cos\bar{\theta}}\right) v + \left(1 - \frac{c^* \bar{x}}{\cos\bar{\theta}}\right) \omega$$

Hence, the system state equations are:

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{\theta}} \end{bmatrix} = \mathbf{J}_v v + \mathbf{J}_\omega \omega \qquad \text{with:} \quad \mathbf{J}_v = \begin{bmatrix} \tan\bar{\theta}\cos\beta - \sin\beta \\ -\frac{c^* \cos\beta}{\cos\bar{\theta}} \end{bmatrix}, \ \mathbf{J}_\omega = \begin{bmatrix} \bar{z}^* + \bar{x}\tan\bar{\theta} \\ 1 - \frac{c^* \bar{x}}{\cos\bar{\theta}} \end{bmatrix} \quad (12)$$

under the constraint that $\left|\bar{\theta}\right| \neq \frac{\pi}{2}$, which can be avoided by temporarily using the pose-based row controller while Γ is parallel to Y.

By imposing the motion exigency (4), the system state equations (12) become:

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{\theta}} \end{bmatrix} = \mathbf{J}_v v^* + \mathbf{J}_\omega \omega$$

This system can be controlled using the feedback law:

$$\omega = -\mathbf{J}_\omega{}^+ \left(\lambda e + \mathbf{J}_v v^*\right)$$

4 Image-based path follower

Similarly to the pose-based path follower, the image-based path follower utilizes a row and a column primitive controllers. However, in this case, the controllers are based on the reference path point image features instead of its 3D features. Let us denote by: $u_c = \begin{bmatrix} v_{c,x} & v_{c,y} & v_{c,z} & \omega_{c,x} & \omega_{c,y} & \omega_{c,z} \end{bmatrix}^\top$ the robot velocity expressed in $\mathcal{F}_\mathcal{C}$. The interaction matrix $\mathbf{L}_s$, which relates the dynamics of visual features $s = (X, Y, \Theta)$ to u_c, has been derived, for the normalized perspective camera model, in [9]:

$$\mathbf{L}_s = \begin{bmatrix} -\frac{1}{z_c} & 0 & \frac{X}{z_c} & XY & -1 - X^2 & Y \\ 0 & -\frac{1}{z_c} & \frac{Y}{z_c} & 1 + Y^2 & -XY & -X \\ \frac{C\rho C^2\Theta}{t_y} & \frac{C\rho C\Theta S\Theta}{t_y} & -\frac{C\rho C\Theta (YS\Theta + XC\Theta)}{t_y} & -(YS\Theta + XC\Theta)\, C\Theta & -(YS\Theta + XC\Theta)\, S\Theta & -1 \end{bmatrix} \quad (13)$$

with $C\Theta$, $S\Theta$, $C\rho$ and $S\rho$ respectively denoting: $\cos\Theta$, $\sin\Theta$, $\cos\rho$, and $\sin\rho$. This expression is an approximation, since the observed 3D point corresponding to R^*, is varying. At low velocities, this assumption is valid, and the visual feature dynamics can be expressed by:

$$\dot{s} = \mathbf{L}_s u_c + \frac{\partial s}{\partial t} \quad (14)$$

The term $\frac{\partial s}{\partial t}$ corresponds to the feature motion. However, assuming low robot velocities, we can neglect this term in the control laws.

The robot velocity in $\mathcal{F}_\mathcal{C}$ can be expressed in function of $u = \begin{bmatrix} v & \omega \end{bmatrix}^\top$ as:

$$u_c = {}^C\mathbf{T}_R u \quad (15)$$

where ${}^{C}\mathbf{T}_{R}$ is the homogeneous transformation from $\mathcal{F}_{\mathcal{R}}$ to $\mathcal{F}_{\mathcal{C}}$:

$${}^{C}\mathbf{T}_{R} = \begin{bmatrix} 0 & -t_z \\ -\sin\rho & 0 \\ \cos\rho & 0 \\ 0 & 0 \\ 0 & -\cos\rho \\ 0 & -\sin\rho \end{bmatrix}$$

In the following, we will denote by $\mathbf{T}_v$ and $\mathbf{T}_\omega$ respectively the first and second columns of ${}^{C}\mathbf{T}_{R}$, and by $\mathbf{L}_X$, $\mathbf{L}_Y$ and $\mathbf{L}_\Theta$ (top to bottom) the rows of $\mathbf{L}_s$. Replacing (15) in (14) yields:

$$\dot{s} = \mathbf{L}_s {}^{C}\mathbf{T}_{R} u + \frac{\partial s}{\partial t} \tag{16}$$

This equation will be used to design the two image-based primitive controllers below.

4.1 Row controller

The task of the row controller is to drive (X, Θ) to a desired state (X^*, Θ^*), while constraining R^* to a row in the image. This is equivalent to zeroing: $e = [X - X^* \;\; \Theta - \Theta^*]^\top$, under the path following constraint:

$$Y = \text{const} = Y^*$$

Since $\dot{Y} = 0$, the system state equations are:

$$\begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix} = \mathbf{J}_v v + \mathbf{J}_\omega \omega + \begin{bmatrix} \frac{\partial X}{\partial t} \\ \frac{\partial \Theta}{\partial t} \end{bmatrix} \tag{17}$$

where the expressions of $\mathbf{J}_v$ and $\mathbf{J}_\omega$, can be derived from (16):

$$\mathbf{J}_v = \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_\Theta \end{bmatrix} \mathbf{T}_v \qquad\qquad \mathbf{J}_\omega = \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_\Theta \end{bmatrix} \mathbf{T}_\omega$$

By imposing the motion exigency (4), the system state equations (17) become:

$$\begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix} = \mathbf{J}_v v^* + \mathbf{J}_\omega \omega + \begin{bmatrix} \frac{\partial X}{\partial t} \\ \frac{\partial \Theta}{\partial t} \end{bmatrix}$$

This system can be controlled using the feedback law:

$$\omega = -\mathbf{J}_\omega{}^{+} \left(\lambda e + \mathbf{J}_v v^*\right)$$

Note that, as aforementioned, this control law does not compensate the terms $\frac{\partial X}{\partial t}$ and $\frac{\partial \Theta}{\partial t}$.

4.2 Column controller

The task of the column controller is to drive (Y, Θ) to a desired state (Y^*, Θ^*) while constraining R^* to a column in the image. This is equivalent to zeroing: $e = [Y - Y^* \;\; \Theta - \Theta^*]^\top$, under the path following constraint:

$$X = \text{const} = X^*$$

control scheme	pose-based		image-based	
primitive controller	row	column	row	column
s_1	x	$\bar{x}$	X	Y
s_2	θ	$\bar{\theta}$	Θ	Θ
$\mathbf{J}_{v1}$	$\tan\theta$	$\tan\bar{\theta}\cos\beta - \sin\beta$	$\mathbf{L}_X\mathbf{T}_v$	$\mathbf{L}_Y\mathbf{T}_v$
$\mathbf{J}_{v2}$	$-c^*/\cos\theta$	$-c^*\cos\beta/\cos\bar{\theta}$	$\mathbf{L}_\Theta\mathbf{T}_v$	$\mathbf{L}_\Theta\mathbf{T}_v$
$\mathbf{J}_{\omega 1}$	$z^* + x\tan\theta$	$\bar{z}^* + \bar{x}\tan\bar{\theta}$	$\mathbf{L}_X\mathbf{T}_\omega$	$\mathbf{L}_Y\mathbf{T}_\omega$
$\mathbf{J}_{\omega 2}$	$1 - c^*x/\cos\theta$	$1 - c^*\bar{x}/\cos\bar{\theta}$	$\mathbf{L}_\Theta\mathbf{T}_\omega$	$\mathbf{L}_\Theta\mathbf{T}_\omega$
e_1	$x - x^*$	$\bar{x} - \bar{x}^*$	$X - X^*$	$Y - Y^*$
e_2	$\theta - \theta^*$	$\bar{\theta} - \bar{\theta}^*$	$\Theta - \Theta^*$	$\Theta - \Theta^*$

Table 1: Components of: s, $\mathbf{J}_v$, $\mathbf{J}_\omega$, and e for the four controllers

Since $\dot{X} = 0$, the system state equations are:

$$\begin{bmatrix} \dot{Y} \\ \dot{\Theta} \end{bmatrix} = \mathbf{J}_v v + \mathbf{J}_\omega \omega + \begin{bmatrix} \frac{\partial Y}{\partial t} \\ \frac{\partial \Theta}{\partial t} \end{bmatrix} \tag{18}$$

where the expressions of $\mathbf{J}_v$ and $\mathbf{J}_\omega$, can be derived from (16):

$$\mathbf{J}_v = \begin{bmatrix} \mathbf{L}_Y \\ \mathbf{L}_\Theta \end{bmatrix} \mathbf{T}_v \qquad\qquad \mathbf{J}_\omega = \begin{bmatrix} \mathbf{L}_Y \\ \mathbf{L}_\Theta \end{bmatrix} \mathbf{T}_\omega$$

By imposing the motion exigency (4), the system state equations (18) become:

$$\begin{bmatrix} \dot{Y} \\ \dot{\Theta} \end{bmatrix} = \mathbf{J}_v v^* + \mathbf{J}_\omega \omega + \begin{bmatrix} \frac{\partial Y}{\partial t} \\ \frac{\partial \Theta}{\partial t} \end{bmatrix}$$

This system can be controlled using the feedback law:

$$\omega = -\mathbf{J}_\omega{}^+ (\lambda e + \mathbf{J}_v v^*)$$

As aforementioned, this control law does not compensate the terms $\frac{\partial Y}{\partial t}$ and $\frac{\partial \Theta}{\partial t}$.

5 Convergence of the primitive controllers

The four dynamic systems that we have studied can all be expressed by (6), and the corresponding primitive controllers, by (7), with the components of the two-dimensional vectors s, $\mathbf{J}_v$, $\mathbf{J}_\omega$, and e recalled in Table 1. This general formulation will be exploited, in the following, to analyze the convergence of all four closed loop systems. A sufficient condition for convergence of control law (7) is defined in the following theorem.

Theorem 1 *Under assumption:*

$$e \notin \ker \boldsymbol{J}_\omega{}^+ \tag{19}$$

a sufficient condition for the convergence of the closed loop system at the desired state s^* is:

$$\lambda > \frac{e^\top \left(\boldsymbol{J}_v - \boldsymbol{J}_\omega \boldsymbol{J}_\omega{}^+ \boldsymbol{J}_v \right)}{e^\top \boldsymbol{J}_\omega \boldsymbol{J}_\omega{}^+ e} \; v^* \tag{20}$$

Proof: Let us consider the Lyapunov function $V(e) = \frac{e^\top e}{2}$, which is positive definite $\forall e \neq 0$. The time derivative of V along the closed-loop system, using (6) and (7) is:

$$\dot{V} = e^\top \dot{s} = e^\top \left(\mathbf{J}_v v^* - \mathbf{J}_\omega \mathbf{J}_\omega{}^+ (\lambda e + \mathbf{J}_v v^*) \right)$$

Since $v^* > 0$, $\dot{V}$ is negative definite if and only if:

$$e^\top \left(\mathbf{J}_v - \mathbf{J}_\omega \mathbf{J}_\omega{}^+ \mathbf{J}_v \right) - \frac{\lambda}{v^*} e^\top \mathbf{J}_\omega \mathbf{J}_\omega{}^+ e < 0 \tag{21}$$

Since we have assumed that $\mathbf{J}_\omega{}^+ e$ is non null, we also have $\mathbf{J}_\omega{}^\top e \neq 0$. As a consequence, $e^\top \mathbf{J}_\omega \mathbf{J}_\omega{}^+ e = \left(\mathbf{J}_\omega{}^\top e \right)^2 / \mathbf{J}_\omega{}^\top \mathbf{J}_\omega > 0$, and condition (21) can be rewritten as:

$$\frac{e^\top \left(\mathbf{J}_v - \mathbf{J}_\omega \mathbf{J}_\omega{}^+ \mathbf{J}_v \right)}{e^\top \mathbf{J}_\omega \mathbf{J}_\omega{}^+ e} < \frac{\lambda}{v^*} \tag{22}$$

which leads to condition (20), since $\mathbf{J}_\omega \mathbf{J}_\omega{}^+ \neq \mathbf{I}$.

■

We have decided not to present the complete expressions of (19) and (20) for the four controllers, since these were very lengthy. However, they can be derived, by using the vector component values in Table 1. In all four cases, (19) is an inequality constraint which is linear in e and depends on both the features and the robot parameters, hence very difficult to verify analytically. Instead, we will verify it numerically in Sect. 6. Equation (20) gives a sufficient condition for convergence of the closed loop system at $e = 0$, valid for all four primitive controllers. Note that this equation cannot be verified off line: since it is related to the state, the second part of (20) evolves during the experiment. However, Theorem 1 can be used to guarantee the convergence of the closed loop system at run time, by updating the gain λ at every iteration, to meet (20). Then, since, under assumption (19), all primitive controllers converge to their equilibrium point, the complete control scheme, which is defined as a cascade of the primitive controllers, will also converge. An exception occurs if the pose-based scheme singularities ($\theta = \frac{\pi}{2}$ and $\bar{\theta} = \frac{\pi}{2}$) are reached. As we mentioned in Sect. 3, these are avoided by switching temporarily to the other primitive controller. In this paper, however, we do not analyze the convergence of the complete system in the particular case when the switching is done. The switching was never required in the numerous experiments that we carried out.

6 Experimental Setup

We hereby report the experiments obtained with the two control schemes. Videos are available at: www.irisa.fr/lagadic/demo/demo-cycab-path-following/cycab-path-following.

Experiments took place outdoor using a CyCab, in varying light conditions. Our CyCab is a 4 wheel steered intelligent vehicle equipped with a 70° field of view, forward looking, B&W

Figure 4: Initial conditions used in the outdoor experiments. The point R^* and tangent Γ derived by image processing are indicated respectively by the black circle and arrow.

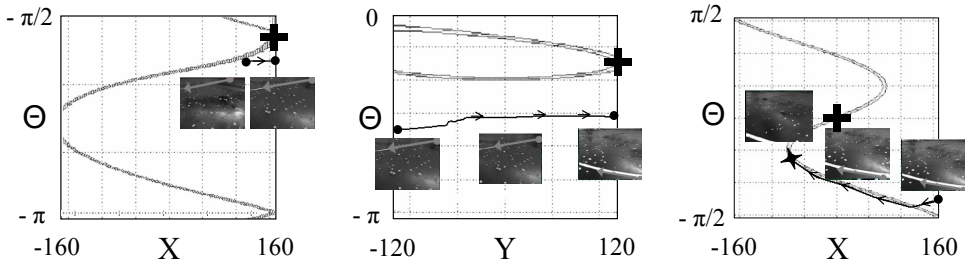

Figure 5: For the image-based scheme, the state loci (X, Y in pixels, Θ in rad) where $e \in \ker \mathbf{J}_\omega{}^+$ are shown in gray for: top row controller (left), right column controller (center), and bottom row controller (right). The desired states are indicated with the black crosses. The loci (black curves) and images during the third image-based experiment are also indicated.

Marlin F-131B camera with image resolution 320×240 pixels. It is used in car-like mode (only the front wheels are used for steering). This induces a bound on the curvature, which, for our CyCab, is: $|c| < 0.35\,\mathrm{m}^{-1}$. The path features are tracked with the ViSP software [23], which must be initialized at the beginning of the experiment by clicking on five path points indicating the desired path following direction. This is the only human intervention: at run time, the tracker detects the position of R^*, and selects the corresponding primitive controller. The tracker proved always effective, and the path was never lost. In the real experiments, we set $v^* = 0.2$ ms $^{-1}$. This velocity was limited for security reasons, and because of the low camera frequency (10 Hz). The system was coarsely calibrated, to obtain: $f = 240$ pixels, $\rho = 0.545$ rad, $t_y = -0.55$ m, and $t_z = 1.63$ m. Ideally, the vehicle trajectory should be assessed using a sensor independent from the camera. This was possible in the simulations, by using an ideal GPS, but since such sensor is not available on our CyCab, the camera was used to qualitatively assess the performance during the real experiments.

For each of the two control schemes, experiments with 3 different initial conditions have been carried out. The 3 experiments are enumerated below.

1. CyCab is initially positioned on the path. Hence, R^* is on the bottom image row (Fig. 4, left). The top row controller (TRC) is used to drive R^* to $X^* = \Theta^* = 0$.
2. CyCab is initially near the path, with R^* on the right column (Fig. 4, center). Initially, the right column controller (RCC) drives R^* to the bottom right corner. Then, the BRC drives R^* along the bottom row to $X^* = \Theta^* = 0$.
3. CyCab is initially far from the path, with R^* on the top row of the image (Fig. 4, right). Initially, the top row controller (TRC) drives R^* to the right pixel column. Then, the RCC drives R^* to the bottom right corner. Finally, the BRC drives R^* along the bottom row to $X^* = \Theta^* = 0$.

Moreover, we have numerically verified condition (19), as the state evolves. For the pose-

based approach, with the CyCab parameters and state ranges, condition $e \notin \ker \mathbf{J}_\omega{}^+$ is always met, except, of course, at $e = 0$. This is true both for straight ($c^* = 0$), and curved ($c^* \neq 0$) path portions. This implies that for any initial condition, and appropriate gain tuning to guarantee (20), the pose-based primitive controllers will converge to their equilibrium point, leading to convergence of the cascaded scheme. Instead, for the image-based controller, (19) is not always met. In Fig. 5, we have represented in gray the state loci where (19) is not verified in the image-based scheme, and with a black cross the desired states for each primitive controller. In practice, as long as the state variables do not enter the gray loci, and the gain guarantees (20), the image-based primitive controllers will converge to their equilibrium point, leading to convergence of the cascaded scheme. If the state variables enter the gray loci, convergence cannot be guaranteed anymore (see Theorem 1). However, for both schemes, the states that do not meet (19) can be numerically determined off line, according to the robot parameters and state ranges, and this information can be used to select the appropriate scheme to avoid that the states enter the loci. In the following sections, the loci in Fig. 5 will be used to verify (19) during the experiments.

7 Simulations

For simulations, we have adopted Webots[2], an environment for modeling and controlling mobile robots, where we have designed a robot with the same characteristics as CyCab. A circular path of radius 12.5 m (i.e., $c^* = \text{const} = 0.08$ m^{-1}) has been drawn, and the two control schemes (pose-based and image-based) have been simulated starting with the path intersecting the right column (see Fig. 4, center). The RRC and BRC are used. All gains are tuned off line to avoid abrupt changes in the steering velocity at the changing point.

With both control schemes, the robot is able to reach and follow the path, and condition (19) is always met. For the image-based case, the robot positions and processed images during the simulation are shown in Fig. 6. The relevant variables (state errors and applied curvature $c = \omega/v^*$) for the two schemes, are plotted in Fig. 7. For the image-based error, instead of $e_1 = X - X^*$ and $e_1 = Y - Y^*$, we have plotted the scaled values $e_{1,n} = \frac{X-X^*}{2X_M}$ for the row controller, and $e_{1,n} = \frac{Y-Y^*}{2Y_M}$ for the column controller. Note that the pose-based controller initially saturates the curvature to its maximum 0.35 m^{-1}, to enable path reaching. Since in Webots we can add a GPS to the robot, the controllers have been assessed by measuring the distance from the path. For the pose-based controller, the average distance is 4.1 cm, whereas for the image-based controller, it is 4.3 cm. Both results are excellent, since they are below 0.5% of the path radius. In both cases, at the end of the first phase (after the RRC has been applied) the orientation error e_2 has not reached 0, because the BRC is activated only by e_1. Nevertheless, when the BRC is applied, the tracking errors converge, and the

[2] www.cyberbotics.com

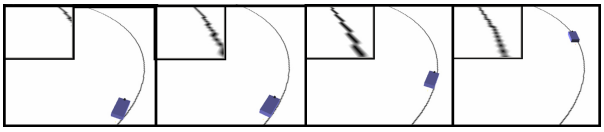

Figure 6: Image-based simulation with robot positions and corresponding images.

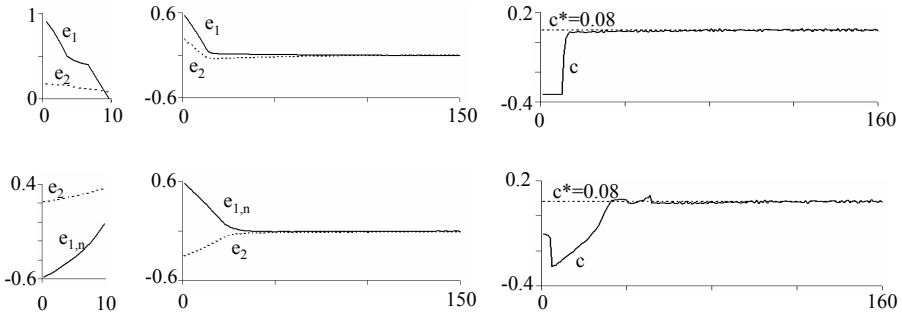

Figure 7: Evolution of relevant variables during the pose-based (top) and image-base (bottom) simulations. Pose-based errors e_1 (solid, in m) and e_2 (dotted, in rad), and image-based errors $e_{1,n}$ (solid, dimensionless) and e_2 (dotted, in rad) for column (left) and row (center) controllers. Applied (solid) and desired (dotted) curvatures in m^{-1} (right).

mean of the curvature at steady state is as expected: $c^* = 0.08\ \mathrm{m}^{-1}$.

8 Experiments

After the simulations, the two control schemes have been tested outdoor on the real CyCab. The path is composed of two straight lines of length 6 m joined by a 60° arc of circle of radius 10 m (i.e., $c^* = \pm 0.1\ \mathrm{m}^{-1}$, with the sign of c^* depending on the path direction to be followed by the robot). The primitive controller gains are the same as in the simulations. To verify the robustness of the controllers, the experiments have been repeated with a random calibration error of either +10% or −10% on each of the four camera parameters.

8.1 Pose-based experiments

In the first pose-based experiment (see Fig. 8), R^* is on the bottom pixel row of the image plane. The row controller is used. The evolution of the relevant variables during the experiment is shown in Fig. 9. The robot successfully follows the path, and the tracking errors (solid and dotted black curves) are low throughout the experiment. At the end of the experiment, both errors are below 0.1. Both errors increase when the robot reaches the discontinuity in the path curvature (frame 270). Correspondingly, ω and, therefore, $c = \omega/v^*$, increases to compensate the error and enable CyCab to follow the curve.

In the second experiment, CyCab is initially near the path, but with R^* on the right column. The relevant variables (state errors, as well as path and applied curvature) are plotted in Fig. 10. The robot successfully reaches and follows the path. Again, when the robot reaches the path curve (frame 285), the error increases. However, ω compensates the error, and

Figure 8: First pose-based experiment (CyCab is initially positioned on the path with small error), with robot positions and corresponding processed images at various iterations.

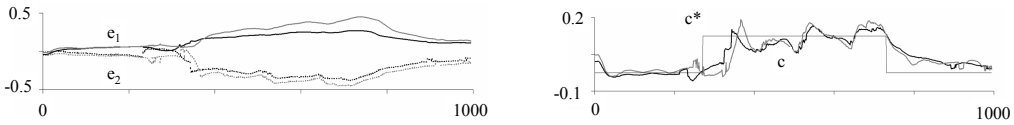

Figure 9: Evolution of relevant variables during the first pose-based experiment, with correct (black) and coarse (gray) camera calibration. Left: errors e_1 (solid, in m) and e_2 (dotted, in rad). Right: applied (thick) and desired (thin) curvatures in m^{-1}.

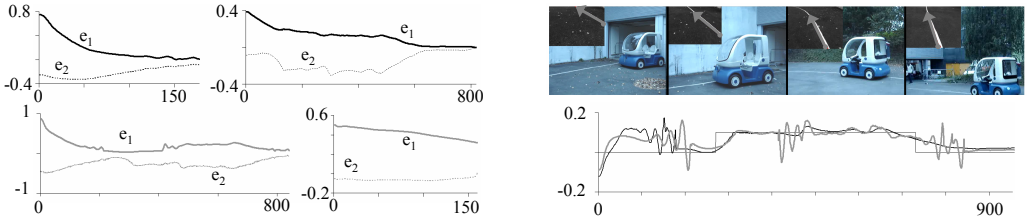

Figure 10: Variables and snapshots of the second pose-based experiment, with correct (black) and coarse (gray) calibration. Errors e_1 (solid, in m) and e_2 (dotted, in rad), with RCC (left) and BRC (center), and applied (thick) and desired (thin) curvatures (right, m^{-1}).

enables the robot to follow the curve, and to zero both state errors.

In the third experiment (Fig. 11), CyCab is initially far from the path, with R^* on the top row. Once again, the robot is able to successfully follow the path. The curvature is initially saturated to 0.35 m^{-1} to enable the robot to reach the path. At the end of the experiment, both errors are below 0.1.

The three pose-based experiments have been repeated with camera calibration error. The variables in the coarse calibration experiments are also shown in Fig. 9, 10, and 11 (gray curves), for comparison with the calibrated case (black). Although CyCab follows the path in all three cases, the convergence is slower than in the calibrated experiments. In particular, in the second experiment, the performance is slightly worsened (see Fig. 10, center and bottom): the RCC convergence is slower than in the calibrated case, the final error is higher (0.35 m instead of 0.05 m), and the applied curvature oscillates more.

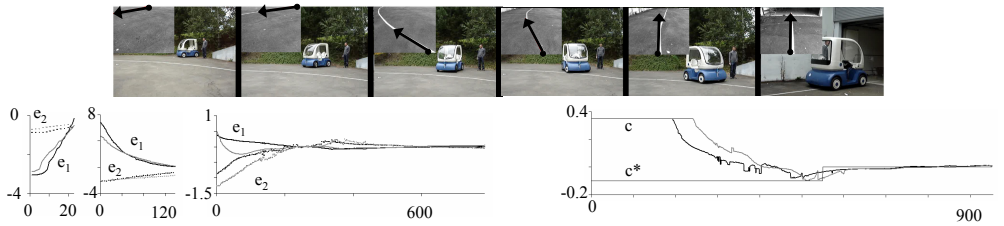

Figure 11: Snapshots and variables for the third pose-based experiment, with correct (black) and coarse (gray) calibration. Left: errors e_1 (solid, in m) and e_2 (dotted, in rad), with TRC, RCC, and BRC (left to right). Right: applied (thick) and desired (thin) curvatures (m^{-1}).

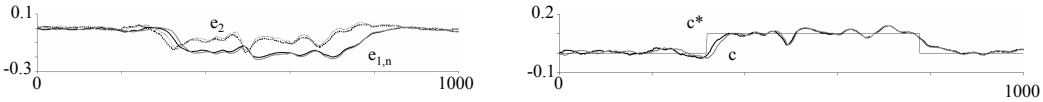

Figure 12: Evolution of relevant variables during the first image-based experiment, with correct (black) and coarse (gray) calibration. Left: errors $e_{1,n}$ (solid, dimensionless), and e_2 (dotted, in rad). Right: applied (thick) and desired (thin) curvatures in m^{-1}.

8.2 Image-based experiments

In the first image-based experiment, R^* is on the bottom row. The relevant variables are shown in Fig. 12. CyCab follows the path, and the tracking errors are low throughout the experiment (at the end, both are below 0.03). As in the pose-based experiment, both errors increase when the robot reaches the discontinuity in the path curvature, and correspondingly, c increases in order to compensate for the error. Using Fig. 5, we verify that throughout the experiment, the state variables verify condition (19).

In the second experiment, CyCab is initially near the path, but with R^* on the right column. The variables are plotted in Fig. 13. CyCab successfully reaches and follows the path, and at the end, both state errors are zeroed. The initial trend of c is completely different from the pose-based experiment (Fig. 10, bottom). Condition (19) is always verified.

In the third experiment, CyCab is initially far from the path, with R^* on the top pixel row. The experiment fails while using the BRC, as the path exits the field of view. Tests with other values of λ are also unsuccessful. The reason is the failure of (19) during control with the BRC, at the iterations highlighted in Fig. 14. To clarify this, we have plotted in black, in Fig. 5, the state evolution during this experiment, and the corresponding images. As the curves show, with TRC and RCC, the state is consistent with (19). Instead, during control with the BRC, the state error enters the kernel of ${\mathbf{J}_\omega}^+$, and (6) cannot be controlled using (7).

As we mentioned, a flaw of the image-based scheme is that it does not consider the curvature c^*, nor the feature motion $\frac{\partial s}{\partial t}$ (see (14)). This is relevant here, and causes the failure. In fact (see the third snapshot on the left of Fig. 14), the BRC is activated with a large error on e_2, in a critically curve path portion. In contrast with the pose-based scheme, the error cannot be regulated since the curvature is not in the feedback law. However, the initial state of the BRC controller cannot be changed, since it is determined by the fact, that the gain λ used with TRC and RRC must be tuned to saturate the c to its maximum 0.35 m^{-1}.

The two successful image-based experiments have been repeated by considering camera calibration error. The results are also shown in Fig. 12 and 13 (gray curves), for comparison

Figure 13: Variables for the second image-based experiment, with correct (black) and coarse (gray) calibration. Errors $e_{1,n}$ (solid, dimensionless), and e_2 (dotted, in rad), with RCC (left) and BRC (center). Applied (thick) and desired (thin) curvatures (right, in m^{-1}.)

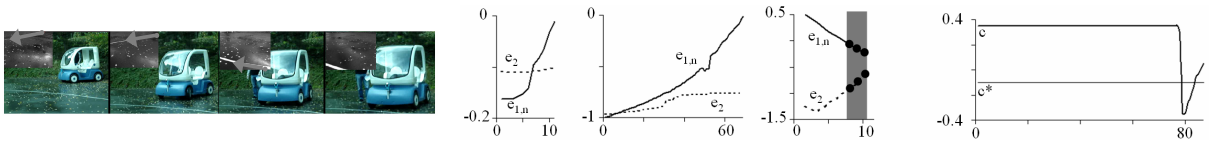

Figure 14: Snapshots and variables during the third image-based experiment (correct camera calibration). Left to right: errors $e_{1,n}$ (solid, dimensionless), and e_2 (dotted, in rad), with TRC (left), RCC (center), BRC (right), and applied (thick) and desired (thin) curvatures (m^{-1}). The iterations where: $e \in \ker \mathbf{J}_\omega{}^+$ are highlighted with a rectangle.

with the calibrated experiments (black). The robot successfully follows the path in both experiments, and again, (19) is always verified. The convergence is slightly slower than in the calibrated experiments. However, in particular for the second experiment, the image-based approach outperforms the pose-based approach when the camera is coarsely calibrated.

8.3 Comparison between the two control schemes

To compare the two control schemes, in Table 2, we show the error norms averaged over each of the real experiments. When the initial error is small (experiments 1 and 2), the image-based approach is better (smaller $|e|$, as seen in the table, and smoother ω, as seen in the curves), and more robust to calibration errors. Instead, experiment 3 fails with the image-based approach, and succeeds with the pose-based approach. As explained above, this is due to the failure of (19) for large error. In a series of simulations with path intersecting the top row with various values of the initial error, we have verified that when the initial $|e|$ is larger than 0.22, the robot is not able to follow the path with the image-based control scheme. In all cases, the path is lost during BRC control, like in experiment 3. The pose-based simulations, instead, are all successful. In summary, image-based control is less effective when the initial error is large, since it does not rely on the curvature, which, acting as a second derivative, fosters the prediction of the error dynamics. On the other hand, since it uses the curvature, which is typically more biased than path position and tangent orientation measurements, the pose-based controller is less smooth and less robust to calibration errors.

9 Conclusions

In this paper, we presented two visual servoing control schemes (pose-based and image-based) enabling nonholonomic mobile robots to reach and follow a path on the ground. The controllers require only a small set of path features: the position of a path point,

control scheme	pose-based		image-based	
calibration	correct	coarse	correct	coarse
experiment 1	0.22	0.30	0.09	0.09
experiment 2	0.21	0.32	0.16	0.17
experiment 3	0.38	0.42	failed	skipped

Table 2: Error norm $|e|$ averaged over the experiments.

and the path tangent orientation at that point. For the pose-based controller, the path curvature at the point is also necessary. Although the two schemes had been introduced in [21], and [22], this article presents a unique representation for all the closed-loop systems used, which eases comparison and leads to a unique formulation of the convergence analysis. New experimental results, not present in [21], and [22], have also been added here, along with the numerical analysis of the closed-loop convergence, which gives a deeper insight on the controllers characteristics. The main results of our comparison are that the pose-based controller can be used in general initial conditions, and is appropriate for path *reaching*, whereas the image-based controller, is more precise and robust, and should be preferred for path *following* (i.e., when the error is small). In future work, we aim at integrating the two schemes, to design a general framework which is able to exploit the advantages of both. We also plan to vary the driving velocity v, in order to tackle more sophisticated scenarios.

References

[1] F. Bonin-Font, A. Ortiz and G. Oliver, "Visual Navigation for Mobile Robots: A Survey", *Journal of Intelligent and Robotic Systems*, 2008.

[2] M. Agrawal and K. Konolige, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS", *Int. Conf. on Pattern Recognition*, 2006.

[3] J. Leonard et al., "A perception-driven autonomous vehicle", *Journal of Field Robotics*, vol. 25, no. 10, pp. 727 - 774, 2008.

[4] A. A. Argyros, K. E. Bekris, S. C. Orphanoudakis and L. E. Kavraki, "Robot homing by exploiting panoramic vision", *Autonomous Robots*, vol. 19, no. 1, pp. 7 – 25, 2005.

[5] E. Royer, M. Lhuillier, M. Dhome and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation", *Int. Journal of Computer Vision*, vol. 74, no. 3, pp. 237 – 260, 2007.

[6] D. Scaramuzza and R. Siegwart, "Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles", *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1015 – 1026, 2008.

[7] J. J. Guerrero, A. C. Murillo and C. Sagues, "Localization and Matching using the Planar Trifocal Tensor with Bearing-only Data", *IEEE Trans. on Robotics*, vol. 24, no. 2, pp. 494 – 501, 2008.

[8] F. Chaumette and S. Hutchinson, "Visual servo control tutorial, part I and II", *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, and vol. 14, no. 1, 2007.

[9] B. Espiau, F. Chaumette and P. Rives, "A new approach to visual servoing in robotics", *IEEE Trans. on Robotics and Automation*, vol. 8, no. 3, pp. 313 – 326, 1992.

[10] Y. Masutani, M. Mikawa, N. Maru and F. Miyazaki, "Visual servoing for non-holonomic mobile robots", *IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems*, 1994.

[11] C. Canudas de Wit, B. Siciliano and G. Bastin Eds., *Theory of Robot Control*, Communication and Control Engineering, Springer-Verlag, London, 1996.

[12] F. Diaz del Rio, G. Jimenez, J. L. Sevillano, S. Vicente and A. Civit Balcells, "A generalization of path following for mobile robots", *IEEE Int. Conf. on Robotics and Automation*, 1999.

[13] M. F. Hsieh and U. Ozguner, "A path following control algorithm for urban driving", *IEEE Int. Conf. on Vehicular Electronics and Safety*, 2008.

[14] R. Frezza, G. Picci, and S. Soatto, "A Lagrangian formulation of nonholonomic path following", *The Confluence of Vision and Control*, Springer Berlin / Heidelberg, 1998.

[15] A. K. Das, R. Fierro, V. Kumar, B. Southall, B. Spletzer, and J. Taylor, "Real-time vision-based control of a nonholonomic mobile robot", *IEEE Int. Conf. on Robotics and Automation*, 2001.

[16] S. Skaff, G. Kantor, D. Maiwand and A. Rizzi, "Inertial navigation and visual line following for a dynamical hexapod robot", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1808 – 1813, 2003.

[17] H. H. Abdelkader, Y. Mezouar, N. Andreff and P. Martinet, "Omnidirectional Visual Servoing From Polar Lines", *IEEE Int. Conf. on Robotics and Automation*, 2006.

[18] J. B. Coulaud, G. Campion, G. Bastin and M. De Wan, "Stability analysis of a vision-based control design for an autonomous mobile robot", *IEEE Trans. on Robotics*, 2006.

[19] A. Rezoug and M. S. Djouadi, "Visual based lane following for non-holonomic mobile robot", *IEEE EUROCON*, 2009.

[20] C. Wang, W. Niu, Q. Li, and J. Jia, "Visual servoing based regulation of nonholonomic mobile robots with uncalibrated monocular camera", *IEEE Int. Conf. on Control and Automation*, 2007.

[21] A. Cherubini, F. Chaumette and G. Oriolo, "A position-based visual servoing scheme for following paths with nonholonomic mobile robots", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.

[22] A. Cherubini, F. Chaumette and G. Oriolo, "An image-based visual servoing scheme for following paths with nonholonomic mobile robots", *Int. Conf. on Control, Automation, Robotics and Vision*, 2008.

[23] E. Marchand, F. Spindler and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills", *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40 – 52, 2005.