Visual navigation with a time-independent varying reference

Andrea Cherubini and François Chaumette

***Abstract*— In this paper, we present a controller for visual navigation, which utilizes a time-independent varying reference in the feedback law. The navigation framework relies on a monocular camera, and the path is represented as a series of key images. The varying reference is determined using a vector field, derived from the previous and next key images. Results in a simulated environment, as well as on a real robot, show the advantages of the varying reference, with respect to a fixed one, in the image, as well as in the 3D state space.**

I. INTRODUCTION

Processing visual information is very useful for robot navigation in urban environments, where tall buildings can disturb satellite receiving and GPS localization, while offering numerous visual features. For this reason, many recent works have been carried out in the field of autonomous vehicle visual navigation. Most of these works [1 – 7] present a framework where a wheeled robot with an on-board camera autonomously tracks a reference path described by an ordered set of *key images*, stored in a database. A similarity score between the current view acquired by the camera and the database images, is used as input for the robot controller.

In [1], a framework of this type has been proposed, where a simple image-based visual servoing [8] controller was utilized: the difference in the image abscissa of the features centroid in the current and next key image was fed back into the controller [2]. The objective of this article is to improve the controller performance in the pose space without jeopardizing the good image space results obtained. This will be done by exploiting, on one hand, the results presented in [3] (where the performance of various controllers has been compared), and on the other hand, the use of a varying visual reference in the servoing scheme. This second aspect represents the main contribution of the present paper.

In fact, the objective of most image-based visual servoing systems is to achieve a desired configuration of the image features from an initial one; however, if the initial and final configurations are far, convergence can be difficult to ensure. As shown in [9], a possible solution is the use of a planning step joint with the servoing, to limit the tracking error along the planned path. In [10], an approach that uses an image-based visual servoing system to track a desired timed feature trajectory is presented. In contrast, a time-independent solution for tracking image trajectories, based on a movement flow to determine the desired configuration from the current one, is described in [11]. With these approaches, the desired configuration remains near the current one, and the local stability of the image-based visual servoing is always assured. This allows the desired image trajectory tracking, while guaranteeing the visibility of the target during operation. In this work, we propose the use of a similar approach to improve the performance of our visual navigation framework.

The paper is organized as follows. The variables and characteristics of our navigation framework are presented in Sections II and III. In Sect. IV, we explain how the varying visual reference is derived. Simulated and real experimental results are discussed respectively in Sections V and VI.

II. DEFINITIONS

In this work, we focus on a nonholonomic mobile robot, equipped with a fixed pinhole camera. The workspace where the robot moves is planar. With reference to Fig. 1, let us define the reference frames: world frame $\mathcal{F}_{\mathcal{W}}\,(W, x', z')$, and image frame $\mathcal{F}_{\mathcal{I}}(O, X, Y)$ (point O is the image plane center). The *robot configuration* is: $q = [x'\ z'\ \theta]^\top$, where $[x'\ z']^\top$ is the *cartesian position* of the robot center in $\mathcal{F}_{\mathcal{W}}$, and $\theta \in\ ]-\pi, +\pi]$ is the *robot heading* (positive counterclockwise) with respect to the world frame z' axis. We choose $u = [v\ \omega]^\top$ as the pair of control variables for our system: respectively the linear and angular velocities of the robot. In the case of a car-like robot, there is a constraint on the maximum applicable curvature, i.e., $\left|\frac{\omega}{v}\right|$ is bounded. The state equation of the robot is:

$$\dot{q} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u$$

We also define the camera frame $\mathcal{F}_{\mathcal{C}}(C, x, y, z)$, shown in Fig. 1 ($C$ is the optical center). The camera optical axis z is aligned with the robot forward axis, and the x axis is parallel to the motion plane. The distance between the y axis and the robot rotation axis is denoted by δ. Let us define:

$$u_c = [v_c\ \omega_c]^\top = [v_x\ v_y\ v_z\ \omega_x\ \omega_y\ \omega_z]^\top$$

the camera velocity expressed in $\mathcal{F}_{\mathcal{C}}$.

In the following, we consider that the camera parameters have been determined through a preliminary calibration

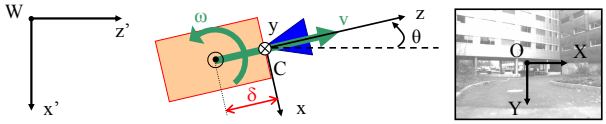

Fig. 1. Mobile robot (orange), equipped with fixed pinhole camera (blue), and applied control variables (v, ω)

A. Cherubini and F. Chaumette are with INRIA Rennes - Bretagne Atlantique, IRISA, Campus de Beaulieu 35042, Rennes, France.
{Andrea.Cherubini, Francois.Chaumette}@irisa.fr

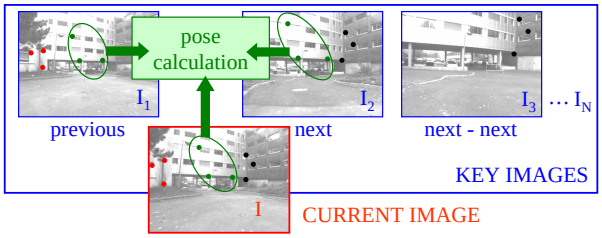

Fig. 2. Each neighboring pair of key images contains some common visual features. During navigation, such features are tracked in the current image to enable localization in the database and 3D pose calculation.

phase. The normalized perspective camera model is:

$$X = \frac{x}{z} \qquad Y = \frac{y}{z} \tag{1}$$

where (X, Y) and (x, y, z) are the coordinates of a visible point respectively in frames $\mathcal{F}_\mathcal{I}$ and $\mathcal{F}_\mathcal{C}$.

III. NAVIGATION FRAMEWORK

In this section, we outline our navigation framework, where the path is represented as a series of key images, such that each neighboring pair contains some common visual features (see fig. 2). We assume that the features are static. The *mapping* of the path will be described first, followed by the description of the autonomous navigation process, which consists of *localization* and *control*. Since the contribution of this paper is control, mapping and localization are merely outlined here; for further details on these aspects, the reader is referred to [1] and [2].

A. Mapping

Mapping starts with the manual driving of the robot on a taught *reference path*, while storing the images from the robot camera. We assume that during teaching, v is never null, i.e., there are no cusps on the path. From the images, a representation of the path is created. Harris points [12] are detected in the first image (which is saved as first key image), and a modified Kanade-Lucas-Tomasi (KLT) algorithm [13] is used to track the features in successive images. Tracked features are used to estimate the 3D geometry between the previous key image and the current image. If the 3D reconstruction error is high, the current image is saved as the next key image, and the tracker is reinitialized with the Harris points detected in this new key image. At the end of the mapping phase, the key images, the corresponding estimated 3D robot poses in $\mathcal{F}_\mathcal{W}$, as well as the 2D $\mathcal{F}_\mathcal{I}$ image coordinates and 3D $\mathcal{F}_\mathcal{C}$ camera coordinates of the feature points at key frames, are saved in the map database. We call I_i, $i = 1, \ldots, N$ the key images, and q_i the corresponding robot key poses.

B. Localization

The localization process during navigation is depicted in fig. 2. It is based on the comparison between the currently acquired image, noted I, and the map database. Using wide-baseline matching on SIFT descriptors [14], the first acquired image is localized in the topological map (i.e., the neighboring key images are identified), and the KLT tracker is initialized using the current features matched with the database key images. Throughout navigation, using the tracked points, a three-view pose calculation is performed between the previous, current and next key images, to allow the recovery from tracking failures and occlusions, and check whether the current image comes after the next key image. As soon as this occurs, a topological transition is made, i.e., the next key image becomes the previous key image and so on. The tracker is then reinitialized, and the process is iterated until the final key image is reached. At each localization iteration, for each pair of images (I, I_i), the n pairs of *matched points* is denoted $(P, P_i)_j$, $j = 1, \ldots, n$.

C. Control

The task of replaying the taught path is divided into N subtasks, each consisting of reaching the next database key image I_i. The contribution of this paper is the improvement of the controller from the one used in [1]. Here, we test the image jacobian controllers presented in [3], and, most importantly, we utilize a varying reference in the feedback law. The image jacobian is a well known tool in visual servoing [8], which is used to drive a vector of k visual features s to a desired value s^*, i.e., to minimize the error:

$$e = s\left(r\left(t\right)\right) - s^*\left(a\left(t\right)\right) \tag{2}$$

The error is implicitly time-dependent via the camera pose r and the mapping a that is used to compute the varying reference s^*, as will be explained in Sect. IV. The image jacobian has been previously applied in nonholonomic navigation (see e.g., [4] and [15]). Matrix $\mathbf{L}_\mathsf{S}$ relates the velocity of feature vector s to the robot velocity u:

$$\dot{s} = \mathbf{L}_\mathsf{S} u = \mathbf{L}_{\mathsf{S},v} v + \mathbf{L}_{\mathsf{S},\omega} \omega$$

The time variation of error (2) is given by:

$$\dot{e} = \dot{s} - \dot{s}^* = \mathbf{L}_{\mathsf{S},v} v + \mathbf{L}_{\mathsf{S},\omega} \omega - \frac{\partial s^*}{\partial a} \frac{\partial a}{\partial t} \tag{3}$$

To ensure an exponential decoupled decrease of e (that is, $\dot{e} = -\lambda e$), we set v to a constant, non-null, value[1] and we select as control law on ω:

$$\omega = -\mathbf{L}_{\mathsf{S},\omega}{}^{+} \left(\lambda e + \mathbf{L}_{\mathsf{S},v} v - \frac{\partial s^*}{\partial a} \frac{\partial a}{\partial t} \right) \tag{4}$$

where λ is a given positive gain, $\mathbf{L}_{\mathsf{S},\omega}{}^{+} \in \mathbb{R}^{1\times k}$ is the Moore-Penrose matrix pseudoinverse of $\mathbf{L}_{\mathsf{S},\omega}$, i.e., $\mathbf{L}_{\mathsf{S},\omega}{}^{+} = \left(\mathbf{L}_{\mathsf{S},\omega}{}^{\top} \mathbf{L}_{\mathsf{S},\omega}\right)^{-1} \mathbf{L}_{\mathsf{S},\omega}{}^{\top}$, and $\frac{\partial s^*}{\partial a} \frac{\partial a}{\partial t}$ must be introduced in the control law to compensate for the desired feature motion. In the experiments, we will however neglect $\frac{\partial s^*}{\partial a} \frac{\partial a}{\partial t}$ in (4).

In all five feedback controllers, s is related to the feature points matched between I and I_i. In the *image jacobian points controller* (IJP), the visual features s used for reaching image I_i are the current image coordinates of the n matched points ($k = 2n$). In the *image jacobian centroid controller* (IJC), the s are the current image coordinates of the matched points' centroid ($k = 2$). The same features are

[1]This is plausible, since the path does not contain any cusps. In Sect. VI, we will relax the assumption of constant v.

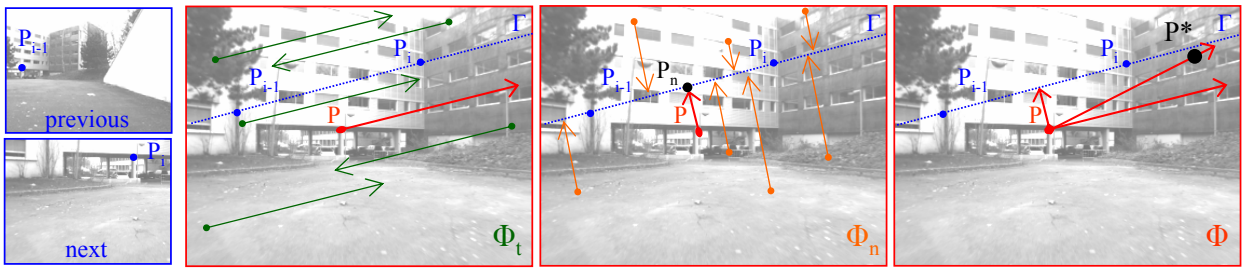

Fig. 3. Derivation of the varying reference in the current image, using the previous and next key point positions, P_{i-1} and P_i.

used respectively by the *image jacobian points controller with uniform depths* (IJPU) and the *image jacobian centroid controller with uniform depths* (IJCU); however, for these controllers, which should be used when depth estimation is unreliable, all point depths are assumed identical and set to a fixed value tuned by the user. In the *approximated image jacobian centroid abscissa controller* (AIJCA), which has been used in [1], the visual feature s is the abscissa of the centroid of the n matched points ($k = 1$), and no metrical knowledge of the 3-D scene is used. The expressions of $\mathbf{L}_{\mathrm{S},v}$ and $\mathbf{L}_{\mathrm{S},\omega}$ for the 5 controllers are given in [3]. In that same work, we have also shown that in all 5 cases, (4) is valid, since ${\mathbf{L}_{\mathrm{S},\omega}}^{\top}\mathbf{L}_{\mathrm{S},\omega}$ is never singular.

Since all controllers use image point coordinates as visual features, to implement (4), we must choose the appropriate desired reference point P^* associated to a point P.

IV. DERIVING THE VARYING REFERENCE

In order to properly choose P^*, we study the trajectory P of the image projection of a 3-D point as the robot moves with velocity commands u between key poses q_{i-1} and q_i. Since in practice, time tracking is not a specification for our system, and since it requires reliable time measurements, we have designed a time-independent varying reference, as in [11]. If poses q_{i-1} and q_i are accurately estimated, and if we assume that, during teaching, u was constant on the path portion, it is possible to derive the camera velocity u_c, which will also be constant on the path portion, and related to u by a homogeneous transformation. Then, if the point depths z_{i-1} and z_i at the key poses are also known, the 3D trajectory of a point in $\mathcal{F}_\mathcal{C}$ can be derived by using the *exponential map* [16], from which we can deduce the 2D image point trajectories.

If we approximate the robot path portion between q_{i-1} and q_i with an arc of circle: $v_x = -\delta\omega$, $v_z = v$, and $\omega_y = -\omega$. Then, the exponential map yields:

$$\begin{cases} x_i = \Delta_x C + \Delta_z S - \frac{v}{\omega} \\ y_i = y_{i-1} \\ z_i = -\Delta_x S + \Delta_z C - \delta \end{cases} \tag{5}$$

where we have used the notations:

$$S = \sin(\theta_i - \theta_{i-1}) \qquad C = \cos(\theta_i - \theta_{i-1})$$
$$\Delta_x = \frac{v}{\omega} + x_{i-1} \qquad \Delta_z = \delta + z_{i-1}$$

Plugging (5) into (1) gives:

$$\begin{cases} X_i = \frac{\Delta_x C + \Delta_z S - \frac{v}{\omega}}{-\Delta_x S + \Delta_z C - \delta} \\ Y_i = \frac{z_{i-1} Y_{i-1}}{-\Delta_x S + \Delta_z C - \delta} \end{cases} \tag{6}$$

These equations can be used to derive the parametric equations of the point trajectory as the robot moves between the two key poses, and hence to design the desired varying reference for (4). However, for practical implementation, a very reliable 3D reconstruction is required, due to the importance of the point depths and of the robot pose in (6).

Instead, since 3D reconstruction in our navigation framework is not accurate enough for such implementation, we have decided to design the varying reference by considering the line connecting P_{i-1} and P_i. Note that in the case of null curvature (i.e., for pure translations of the robot), the visible point trajectories are in fact straight lines.

A vector field $\Phi(P)$ is used to determine the desired reference P^* associated to current point P; we set:

$$P^* = P + \|P_{i-1}P_i\| \frac{\Phi(P)}{\|\Phi(P)\|}$$

Note, from the above equation, that the norm of the error $P - P^*$ is maintained constant throughout the path portion. Its value coincides with the distance between the previous and next key point position. In the simulations, and experiments, we will show how this choice improves the system performance.

The technique used to generate vector field $\Phi(P)$ is similar to that used in [11], and is illustrated in fig. 3. For each path portion linking two consecutive key poses, we consider the straight line Γ connecting P_{i-1} and P_i. The vector field at each point of Γ must be tangent to it, and point towards P_i. At points outside the line it should decrease the tracking error. In practice, the vector field indicates the direction in which the desired reference must be located, in order to track the trajectory and drive P to P_i. Based on these considerations, the vector field $\Phi(P)$ associated to point P is defined as the sum of a tangential and a normal effect:

$$\Phi(P) = \Phi_t(P) + \Phi_n(P)$$

where Φ_t and Φ_n are obtained as the negative gradients of potential fields U_t and U_n that will be defined below.

The tangential potential field U_t is the scalar product:

$$U_t = |PP_i \cdot P_{i-1}P_i|$$

Its norm is the product of $\|P_{i-1}P_i\|$ with the distance between P_i and the normal projection of P (denoted P_n) on Γ. Thus, its contribution to the vector field

$$\Phi_t = -\bigtriangledown U_t = \pm P_{i-1}P_i$$

has constant norm (except at P_i, where it is null), is parallel to Γ, and points towards P_i. Note that there is a discontinuity in the norm of Φ_t when P crosses the normal to Γ through P_i. This is not a problem, since, as explained in III-B, the controller will switch to the following key image I_{i+1} when I_i is reached (and eventually stop, if $i = N$).

The normal potential field U_n is:

$$U_n = \frac{d^2(P, \Gamma)}{2}$$

where $d(P, \Gamma)$ indicates the signed distance from P to Γ. In practice, it attains its minimum value (zero) for line points, and increases quadratically with the distance from the line. Thus, its contribution to the vector field:

$$\Phi_n = - \bigtriangledown U_n = PP_n$$

is necessary to 'attract' reference P^* towards Γ.

Both vector fields Φ_t and Φ_n, are shown in fig. 3.

V. SIMULATIONS

For preliminary simulations, we made use of Webots[2], where a mobile robot has been designed. In successive simulations, we let the robot navigate on a taught path using feedback law (4) with the 5 controllers outlined in Sect. III-C (i.e., with the corresponding instances of $\mathbf{L}_{S,v}$ and $\mathbf{L}_{S,\omega}$), and with the 3 different visual references s^* that will be described below.

The objective of the first simulations is to show the effect, on navigation, of the choice of s^* in (4). This is done by running 3 series of simulations, each one on all 5 controllers:

- in the *key fixed reference* simulations, we use the references detected on the $N = 17$ key images, as in [1];
- in the *all fixed reference* simulations, during mapping, we store the data acquired at *every* manual driving iteration, instead of storing only *key* data; hence, in (4), we use the fixed references detected on all images, and a much larger map database is obviously required in this case ($N = 510$):
- in the *varying reference* simulations, only data at key poses is considered, as in the first case; the reference to be tracked is computed as explained in Sect. IV.

Using the Webots GPS sensor, we can derive the 3D paths tracked by the simulated robot in the 15 experiments and compare them to the taught path. With all 5 controllers and all 3 types of references, the robot is able to successfully complete path tracking. Along with the 3D position error, we consider, as performance metric, the image error with respect to the next key image:

$$\epsilon_i = \frac{1}{n} \sum_{j=1}^{n} \| P_j - P_{i,j} \|$$

In Figure 4, we have plotted, for all 15 simulations, the average image error ϵ_i when the key images are reached, and the average position error with respect to the taught path. As is shown in the figure, all controllers are much

[2]www.cyberbotics.com

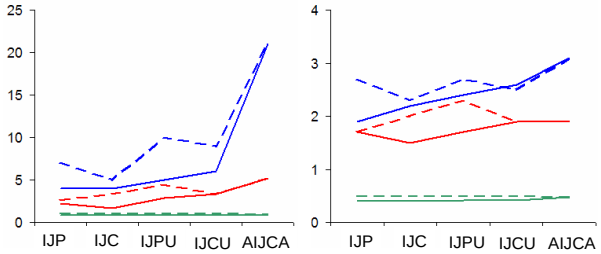

Fig. 4. Average position error (left, in cm) and average image error (right, in pixels) for the 5 controllers in the simulations with correct (full curves) and coarse (dashed curves) camera calibration, using *key fixed reference* (blue), *all fixed reference* (green) and *varying reference* (red).

less accurate with the key references (blue) than with all references (green). This is reasonable, since using all references is equivalent to increasing the signal frequency on the feedback loop of the control scheme. The figure shows that both errors, are more than four times higher with key fixed than with all fixed references using either of the 5 controllers. On the other hand, when varying references are used, the controller accuracies are good (less than 5 cm in 3D, and 2 pixels in the image) while the database size is approximately thirty times smaller than if all images are used. This represents an excellent trade-off, since both precision and feasibility should be taken into account when designing a real navigation system, and if all references are used, a database of approximately 1 gigabyte would be required for each path km. Along with the database size, the mapping computation time is sensibly increased (almost ten times) when all images are used as references. This confirms the utility of the varying reference. When fixed key references are used, slightly better results are obtained if the depth is estimated using 3D reconstruction (IJP and IJC), than if it is fixed (IJPU and IJCU) or not considered at all (AIJCA). This aspect, which had already been outlined in [3], is less evident in the two other cases: the 3D estimation is less influent in the case of a frequent reference signal (measured when all images are considered, and estimated when the varying reference is considered).

To further investigate the controller performances, we have plotted, in Fig. 5 (left), the evolution of some relevant variables (the robot position in $\mathcal{F}_\mathcal{W}$, as well as ω and ϵ_i) during a circular path portion, when the best controllers of *key fixed reference* (i.e., IJP), *all fixed reference* (IJC) and *varying reference* (IJC) are used. Although we focus on this portion, the trends are similar throughout the path. During teaching, the angular velocity applied to the robot on this path portion was -0.05 rad s^{-1}. The robot paths (top) show that when the signal is frequent (red and green curves) the autonomous behavior is similar to the taught one (black). The reason for this can be found in the ω curves (center): when the signal is frequent, $\omega \simeq -0.05$ rad s^{-1}. Correspondingly, ϵ_i (bottom) decreases approximately linearly when the varying reference is used, and remains low when all images are used. This leads to the nice performances described above. On the other hand, when fixed key references are utilized (blue curve), both the control input and error have an exponential

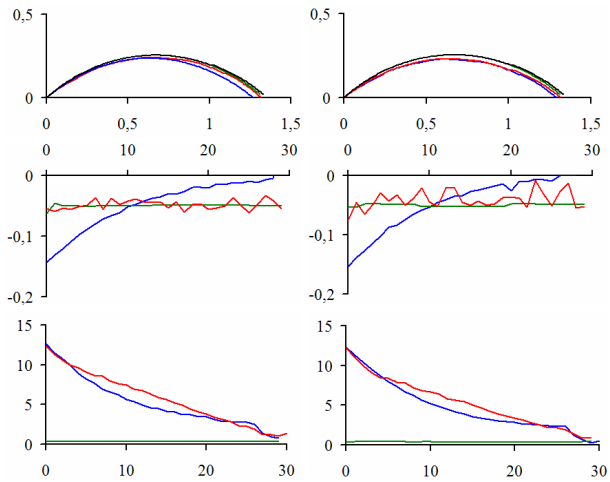

Fig. 5. Evolution of relevant variables at successive iterations while the simulated robot moves on a circular portion of the path using the best controllers of *key fixed reference* (blue), *all fixed reference* (green) and *varying reference* (red) with correct (left) and coarse (right) camera calibration. Top: robot position (x', z') in $\mathcal{F}_{\mathcal{W}}$ (meters); the taught path is indicated in black. Center: ω (rad/s). Bottom: ϵ_i (pixels).

decreasing trend, which leads to abrupt changes when the key images are switched, and to less accurate path tracking.

To verify the controllers' robustness, the 15 simulations have been repeated with a random calibration error of either $+10\%$ or -10% on focal length parameters f_X and f_Y, and distance δ. The relevant metrics, and their evolution, are shown respectively in Fig. 4 (dashed curves), and Fig. 5 (right). The robot is again able to successfully follow the path in all cases, although path tracking is less precise than in the calibrated camera case (see Fig. 4). The best results are again obtained with: *all fixed reference*, *varying reference*, and *key fixed reference* (in this order).

VI. EXPERIMENTS

After the simulations, we tested the framework on a CyCab vehicle with a coarsely calibrated 70° field of view, forward looking, B&W camera. CyCab is a 4 wheel drive, 4 wheel steered vehicle, which we use in car-like mode (i.e., only the front wheels are used for steering). The objective of the experiments is to show the improvement in the navigation performance, with respect to [1]. The changes with respect to the existing navigation framework concern control.

A minor difference with respect to the existing controller concerns the linear velocity v, which must be reduced, in the presence of sharp turns, to ease the tracking of quickly moving features in the image. In [1], v was switched between two constant values $\bar{V}$ and $\bar{v} < \bar{V}$ by thresholding the differences between the centroid abscissa in the current, next and next-next key images. This caused abrupt accelerations, unpleasant for passengers. Here, instead, we use a hyperbolic tangent function to map the centroid abscissa variation to v:

$$v = \bar{V} + \bar{v} + \frac{\bar{V} - \bar{v}}{2} \tanh\left(\pi - \frac{|2X_i + X_{i+1} - 3X|}{\gamma}\right) \quad (7)$$

where γ is a positive parameter that determines the function inflection point. This design choice guarantees the $\bar{v}$ and $\bar{V}$ bounds, while smoothening the transitions.

TABLE I

COMPARING THREE CONTROLLERS ON THE REAL ROBOT

controller	AIJCA fixed	AIJCA varying	IJCU varying
average ϵ_i (pixels)	28	21	17
average v (m s^{-1})	0.67	0.74	0.80

The main difference with respect to the framework presented in [1] concerns the angular velocity, which is still calculated using (4), but with three control strategies. In Sect. V, we have shown that the use of a varying reference improves navigation. Using all images, while guaranteeing higher accuracy, induces slower mapping and greater memory requirements. Thus, we have decided to compare the controller used in [1] (AIJCA with key fixed references) with the AIJCA controller with varying reference, and with the IJCU controller with varying reference. The comparison between the first two controllers aims at emphasizing the advantages of the varying reference, when the same control law is used. For the third experiment, we have chosen IJCU for three reasons. Firstly, the simulations (see Fig. 4) have shown that the image jacobian controllers slightly outperform AIJCA, especially in the pose space. Moreover, to avoid the computation of the vector field at all points, which is computationally costly, centroid controllers IJC and IJCU are preferable. Finally, since we have shown in [3] that the use of uniform feature depths is a valid alternative when depth estimation is inaccurate, as is the case here, we have chosen IJCU.

A taught path of approximately 40 m, composed of $N = 9$ key images, has been replayed using the three controllers. We set $\bar{V} = 0.9$ m s^{-1}, $\bar{v} = 0.5$ m s^{-1} and $\gamma = 30$. The gains are tuned respectively to $\lambda = 0.25$ and $\lambda = 0.15$, for the fixed and varying reference experiments[3]. Values of the main metrics are reported in Table I, and the evolution of v, ω, and ϵ_i during the experiments are shown in Fig. 6. The replayed paths, estimated from snapshots of the 3 experiments are shown, along with the taught path (white) in Fig. 7.

The experiments confirm the controllers' characteristics seen in Webots. Indeed, both in the image (see Table I and Fig. 6) and in the 3D state space (Fig. 7), the two controllers using the varying reference outperform the one that uses the fixed reference. In fact, maintaining constant $\|e\|$ in (4), keeps the value of ω smooth throughout navigation. This is an important advantage, since during manual drive, the angular velocities are generally smooth, and this is hardly reproducible with the fixed reference, leading to imprecise 3D tracking. In particular, note from Fig. 6, that the variation in ω at key image shifts is smaller when the varying reference is used (for both controllers, approximately 0.03 rad s^{-1}, averaged over the eight image shifts), than when the fixed reference is used (0.045 rad s^{-1}). This is an advantage both for passenger comfort, and for the localization algorithm, since feature tracking is facilitated. In fact, better feature tracking enables faster navigation: as shown in Table I, the average v can be 10% higher when the varying reference is used with the same controller, i.e., AIJCA.

[3]Since on each path portion, $\|e\|$ is constantly equal to its initial value in the varying case, and decreases from that value in the fixed case, the gain with fixed reference must be larger than with varying reference.

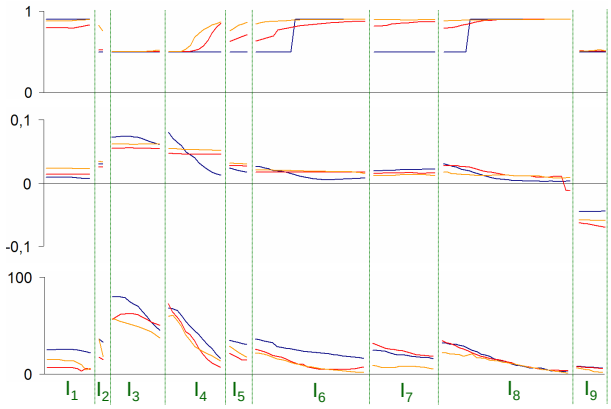

Fig. 6. Evolution of v (top, in m s^{-1}), ω (center, in rad s^{-1}) and ϵ_i (bottom, in pixels) at successive iterations while Cycab moves on the path using: AIJCA with key fixed references (blue), AIJCA with varying reference (red), and IJCU with varying reference (orange).

As shown in the top plot of Fig. 6, using law (7) keeps the linear velocity smooth, and therefore improves the passenger comfort. Finally, as prefigured in the simulations and in [3], among the two varying reference controllers, IJCU is better than AIJCA, both in the image, and in the pose space. The corresponding experiment (IJCU with varying reference) is shown in the video clip attached to this paper.

VII. CONCLUSIONS AND FUTURE WORK

We have proposed the use of a varying reference in visual navigation frameworks for replaying paths represented as a series of key images. The reference has been designed using a vector field, which is independent from time, and from camera pose estimation. This choice is appropriate for application on real systems, where timed trajectory tracking can be difficult, and pose reconstruction is often inaccurate. Simulations and real results show that better image and pose accuracy, as well as higher navigation speeds, can be obtained when the reference is varied. The experiments also confirm the characteristics of the 5 controllers introduced and assessed in [3]: although the 2 controllers, which combine both image data and feature depth (IJP and IJC) outperform the others, valid alternatives, when 3D reconstruction is inaccurate, are the uniform depth controllers IJPU and IJCU.

In our opinion, the varying reference approach, will be useful for the future work we plan to do on obstacle avoidance. In fact, taking into account obstacles present in the workspace, while tracking a path represented by key images, requires high accuracy both in the image, and in the pose space. This can be obtained by using a varying reference in the feedback law.

VIII. ACKNOWLEDGMENTS

The authors thank Fabien Spindler, Albert Diosi and Nicolas Melchior for their help in the experiments. This work was funded in part by the ANR CityVIP project.

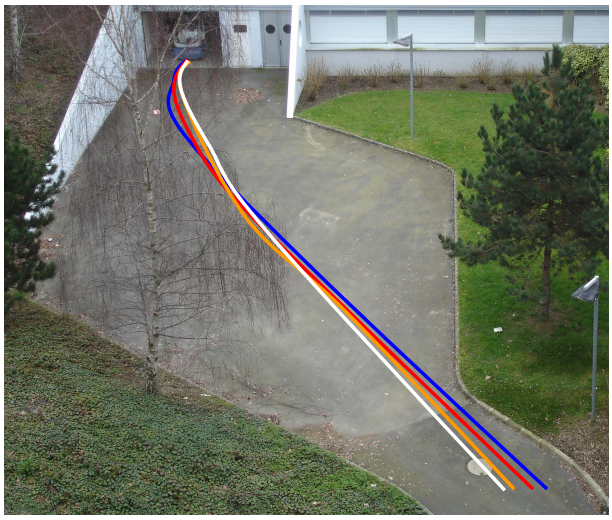

Fig. 7. Replaying a taught path (white) using controllers: AIJCA with key fixed references (blue), AIJCA with varying reference (red), and IJCU with varying reference (orange).

REFERENCES

[1] S. Šegvić, A. Remazeilles, A. Diosi and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation", *Computer Vision and Image Understanding*, vol 113, pp. 172–187, 2008.

[2] A. Diosi, A. Remazeilles, S. Šegvić and F. Chaumette, "Outdoor Visual Path Following Experiments", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.

[3] A. Cherubini, M. Colafrancesco, G. Oriolo, L. Freda and F. Chaumette, "Comparing appearance-based controllers for nonholonomic navigation from a visual memory", *ICRA 2009 Workshop on safe navigation in open and dynamic environments: application to autonomous vehicles*.

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

[5] S. S. Jones, C. Andersen and J. L. Crowley, "Appearance based processes for visual navigation", *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1997.

[6] 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.

[7] 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.

[8] F. Chaumette and S. Hutchinson, "Visual servo control, part I: basic approaches", *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, 2006.

[9] P. Zanne, G. Morel and F. Plestan, "Sensor-based robot control in the presence of uncertainties: bounding the task function tracking errors", *IEEE Int. Conf. on Robotics and Automation*, 2002.

[10] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control", *IEEE Trans. on Robotics and Automation*, vol. 18, no. 4, pp. 534 – 549, 2002.

[11] J. Pomares and F. Torres, "Time independent tracking using 2-D movement flow based visual servoing", *IEEE Int. Conf. on Robotics and Automation*, 2005.

[12] C. Harris and M. Stephens, "A combined corner and edge detector", *4th Alvey Vision Conference*, pp. 147–151, 1988.

[13] J. Shi and C. Tomasi, "Good features to track", *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[14] D. G. Lowe. "Distinctive image features from scale-invariant keypoints", *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] D. Burschka and G. Hager, "Vision-based control of mobile robots", *IEEE Int. Conf. on Robotics and Automation*, 2001.

[16] Y. Ma, S. Soatto, J. Košecká and S. S. Sastry, "An Invitation to 3-D Vision: from Images to Geometric Models". New York: Springer, 2003.