Comparing appearance-based controllers for nonholonomic navigation from a visual memory

Andrea Cherubini, Manuel Colafrancesco, Giuseppe Oriolo, Luigi Freda and Francois Chaumette

Abstract **In recent research, autonomous vehicle navigation has been often done by processing visual information. This approach is useful in urban environments, where tall buildings can disturb satellite receiving and GPS localization, while offering numerous and useful visual features. Our vehicle uses a monocular camera, and the path is represented as a series of reference images. Since the robot is equipped with only one camera, it is dif cult to guarantee vehicle pose accuracy during navigation. The main contribution of this article is the evaluation and comparison (both in the image and in the 3D pose state space) of six appearance-based controllers (one pose-based controller, and ve image-based) for replaying the reference path. Experimental results, in a simulated environment, as well as on a real robot, are presented. The experiments show that the two image jacobian controllers, that exploit the epipolar geometry to estimate feature depth, outperform the four other controllers, both in the pose and in the image space. We also show that image jacobian controllers, that use uniform feature depths, prove to be effective alternatives, whenever sensor calibration or depth estimation are inaccurate.**

I. INTRODUCTION

In recent research, mobile robot navigation has been often done by processing visual information [1]. This approach can be useful for navigation in urban environments, where tall buildings can disturb satellite receiving and GPS localization, while offering numerous and useful visual features. The most widespread approaches to visual navigation are the *model-based*, and the *appearance-based* approaches, which we shall brie y recall. Model-based approaches rely on the knowledge of a 3D model of the navigation space. The model utilizes perceived features (e.g., lines, planes, or points), and a learning step can be used for estimating it. Conversely, the appearance-based approach does not require a 3D model of the environment, and works directly in the sensor space. The environment is described by a topological graph, where each node corresponds to the description of a position, and a link between two nodes de nes the possibility for the robot to move autonomously between the two positions.

In this work, we focus on appearance-based navigation, with a single vision sensor. The environment descriptors correspond to images stored in an *image database*. A similarity score between the view acquired by the camera and the database images, is used as input for the controller that leads the robot to its nal destination (which corresponds to a *goal image* in the database). Various strategies can be used to control the robot during navigation. An effective method is visual servoing [2], which was originally developed for manipulator arms, but has also been used for controlling nonholonomic robots (see, for instance, [3]).

The main contribution of this paper is the comparison between six controllers for nonholonomic appearance-based navigation using monocular vision. In particular we investigate the performance of this controllers both in the image, and in the 3D pose state spaces. The paper is organized as follows. In Sect. II, a survey of related works is carried out. In Sect. III, the problem of appearance-based nonholonomic navigation from a visual memory is de ned. Although the scope of this paper is the discussion of the control strategies, in Sect. IV, we outline the image processing and the 3D reconstruction algorithms used in our navigation framework. In Sect. V, we present and illustrate the six controllers. The simulated and experimental results are presented in Sect. VI.

II. RELATED WORK

Recent works in the eld of appearance-based autonomous vehicle navigation are surveyed hereby. Most of these works [3 13] present a framework with these characteristics:

- a wheeled robot with an on-board camera is considered;
- during a preliminary phase, the *teaching phase*, the robot motion is controlled by a human operator, and a set of images is acquired and stored in a database;
- an *image path* to track is then described by an ordered set of *reference images*, extracted from the database;
- during the *replaying phase*, the robot (starting 'near' the teaching phase initial position) is required to repeat the same path;
- the replaying phase relies on a matching procedure (usually based on correlation) that compares the currently observed image with the reference images;
- although the control strategy enabling the robot to track the learned path varies from one work to the other, it relies, in all cases, on the comparison between the current and reference images.

The methods presented hereby can be subdivided in two main areas. In some works, a three dimensional reconstruction of the workspace is used. The other navigation frameworks, instead, rely uniquely on image information.

We rstly survey the works where 3D reconstruction is utilized. In 1996, Ohno and others [4] propose to use the image database to reconstruct the robot pose in the workspace (i.e., position and orientation) which is utilized for control. In [5], a three dimensional representation of the

A. Cherubini, M. Colafrancesco, G. Oriolo and L. Freda are with the Dipartimento di Informatica e Sistemistica, Universita di Roma La Sapienza , Via Ariosto 25, 00185 Roma, Italy {cherubini, oriolo, freda}@dis.uniroma1.it

F. Chaumette is with INRIA-IRISA, Campus de Beaulieu 35042, Rennes, France Francois.Chaumette@irisa.fr

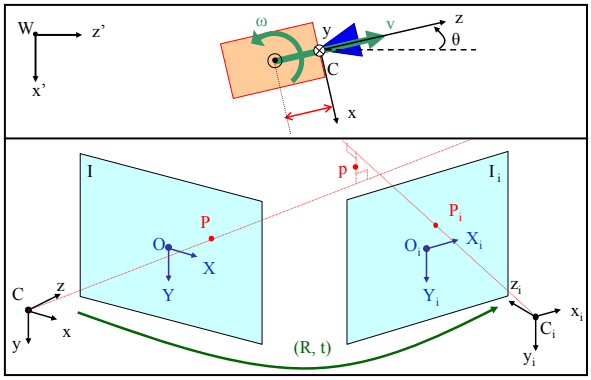

Fig. 1. Relevant variables utilized in this work. Top: mobile robot (orange), equipped with a fixed pinhole camera (blue), and applied control variables (v, ω). Bottom: two different views (distinct camera placements) of the same 3-D point p, i.e., in the current (left) and reference (right) images.

taught path is built from the image sequence, and a classic path following controller is used for navigation. Similarly, in [6], pairs of neighboring reference images are associated to a straight line in the 3D workspace, that the robot must track. The epipolar geometry and a planar floor constraint are used to compute the robot heading used for control in [7]. Similarly, in [8], 3D reconstruction is used to improve an omnidirectional vision-based navigation framework.

In general, 3D reconstruction is unnecessary, since moving from one reference image to the next, can also be done by relying uniquely on visual information, as shown in many papers. For instance, in [3], the vehicle velocity commands and the camera pan angle are determined using an image-based visual servoing scheme. In [9], a particular motion (e.g., 'go forward', 'turn left') is associated to each image, in order to move from the current to the next image in the database. In [10], a proportional control on the position of the feature centroid in current and reference images drives the robot steering angle, while the translational velocity is set to a constant value. The controller presented in [11] exploits angular information regarding the features matched in panoramic images. Energy normalized cross correlation is used to control the robot heading in [12]. In [13], a specific image jacobian, relating the change of some image features with the changes in motion in the plane, is used for control.

In summary, a large variety of control schemes has been applied for achieving nonholonomic navigation from a visual memory. However, a comparison between the various approaches has never been carried out. Moreover, in most of the cited articles, the focus has been the qualitative evaluation of the proposed navigation framework in real, complex, environments, without a quantitative assessment of the controller performance. In this paper, we shall compare the performance of six approaches to nonholonomic navigation from a visual memory. The controllers will be assessed using various metrics, both in simulations, and in real experiments. In particular, we will compare the controller accuracy both in the image and in the pose state space, since both are fundamental for precise unmanned navigation.

III. PROBLEM DEFINITION

A. System characteristics

In this work, we focus on a nonholonomic mobile robot of unicycle type, equipped with a fixed pinhole camera. The workspace where the robot moves is planar: $\mathcal{W} = \mathbb{R}^2$. 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: $q = [x'\ z'\ \theta]^\mathsf{T}$, where $[x'\ z']^\mathsf{T}$ 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 $u = [v\omega]^\mathsf{T}$ as the pair of control variables for our system; these represent respectively the linear and angular velocities (positive counterclockwise) of the robot. 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 distance between the y axis and the robot rotation axis is denoted by δ. A pinhole camera model is considered; radial distortion is neglected. Hence, the camera intrinsic parameters are the principal point coordinates and the focal lengths in horizontal and vertical pixel size: f_X , and f_Y . In the following, we consider that the camera parameters have been determined through a preliminary calibration phase, although we shall partially relax this assumption later in the paper. Image processing is based on the grey-level intensity of the image, called $I(P)$ for pixel $P = (X, Y)$.

As outlined in Sect. II, our navigation framework relies on a teaching and on a replaying phases. These phases will be described in the rest of this section.

B. Teaching phase

During the teaching phase, an operator guides the robot stepwise along a continuous path. Each of the N *teaching steps* starts at time t_{i-1} and ends at $t_i > t_{i-1}$ ($i = 1, \ldots, N$). At each step i, the control input u is assigned arbitrarily by the operator. In this work, we assume that throughout teaching, the robot moves forward, i.e., $v > 0$. At the end of each teaching step, the robot acquires a reference image, that we call I_i, and stores it in a database. Visual features are detected in each I_i. We call $\mathcal{F}_{\mathcal{C}i}(C_i, x_i, y_i, z_i)$ and $\mathcal{F}_{\mathcal{I}i}(O_i, X_i, Y_i)$ (see Fig. 1) the N camera and corresponding N image frames associated to the reference configurations q_i reached at the end of each teaching step.

C. Replaying phase

At the beginning of the replaying phase, the robot is placed at the starting position of the teaching phase. During the replaying phase, the robot must autonomously track the path executed during the teaching phase. The task of replaying the taught path is divided into N subtasks, each consisting of zeroing the visual error between the currently acquired image (called I) and the next reference image ($I_1, I_2, \ldots, I_N$) in

the database. In practice, as soon as the visual error between I and goal image I_{i} is 'small enough', the subtask becomes that of reaching image I_{i+1}. Both the visual error and the switching condition will be detailed in Sect. V. Throughout replaying, the linear velocity is fixed to a constant value $\bar{v} > 0$, while the angular velocity ω is derived with a feedback law dependent on the visual features. In all six feedback controllers that we have tested, at each iteration of subtask i, ω is based on the feature points matched between the current image I and the reference image I_{i}.

IV. VISION ISSUES

A. Image processing

During both teaching and replaying, the images acquired by the robot camera must be processed in order to detect feature points. Besides, during the replaying phase, correspondences between feature points in images I and I_{i} are required to generate the set of matched points which is used to control the robot. In both teaching and replaying phases, we detect feature points with the well known Harris corner detector [14]. Every iteration of the replaying phase relies on image matching between Harris corners in the current image I and in the nearest next reference image in the database I_{i}. For each feature point P in image I, we use a correlation technique to select the most similar corresponding point P_{i} in image I_{i}. For each pair of images (I, I_{i}), the algorithm returns the n pairs of *matched points* $(P, P_{i})_{j}$, $j = 1, \ldots, n$.

B. Deriving 3D information

In one the control schemes used in this work (i.e., the *robot heading controller*), it is necessary to estimate the camera pose variation (rotation $\mathbf{R}$ and translation $\mathbf{t}$, see Fig. 1) between the current view I and the next reference view I_{i} during replay. Moreover, in two of the five image jacobian controllers used, the z coordinates in $\mathcal{F}_{\mathcal{C}}$ (i.e., the *depths*) of the retroperspective projection p of feature points must be estimated. The depths can also be derived from the camera pose variation. The problem of estimating the camera pose variation ($\mathbf{R}$, $\mathbf{t}$) is a typical *structure from motion* problem.

In some works (see, for instance, [5]), the camera pose is estimated by using bundle adjustment methods, which result in long computation processing, unsuitable for on-line use. Here, we have decided to perform on-line 3D reconstruction, by using only the pair of images (I, I_{i}), instead of I with the whole database. This choice inevitably implies lower computational time to the detriment of the 3D reconstruction accuracy. The technique that we used for camera pose estimation is epipolar geometry (see [15], for further details). Using an estimate of the distance from q to q_{i} for $\|\mathbf{t}\|$, four alternative solutions $(\mathbf{R}, \mathbf{t})$ can be derived. For each of the four possible pose variations, we use the technique described in [16] to derive the feature point 3D position p, as the midpoint on the perpendicular to the projecting rays in the two camera frames (see Fig. 1). Finally, we select the pose variation $(\mathbf{R}, \mathbf{t})$ with the greatest number of positive depths in both camera frames $\mathcal{F}_{\mathcal{C}}$ and $\mathcal{F}_{\mathcal{C}i}$, since feature points must lie in front of both image planes.

V. CONTROL SCHEMES

In this section, we describe the characteristics of the six controllers on ω that we have tested in the replaying phase (v is fixed to constant value $\bar{v}$, see Sect. III). In all cases, we consider that subtask i (i.e., reaching image I_{i}) is achieved, and we consequently switch to reaching image I_{i+1}, as soon as the average feature error:

$$\epsilon_{i} = \frac{\sum_{j=1}^{n} \| P_{j} - P_{i;j} \|}{n}$$

is below a threshold τ, and starts to rise.

The first feedback law that we will describe, is *pose-based*: the feedback law is expressed in the robot workspace, by using the 3D data derived from image matching as described in Sect. IV-B. The 5 other feedback laws, instead, are *image-based*: both the control task, and the control law are expressed in the image space, by using the well known image jacobian paradigm. In practice, an error signal measured directly in the image is mapped to actuator commands. Two of the 5 image jacobian controllers require camera pose estimation to derive the depth of feature points. For the 3 others, some approximations on the feature depths are used, as will be shown below.

We hereby recall the image jacobian paradigm which is used by the five image-based controllers. The image jacobian is a well known tool in image-based visual servo control [2], which is used to drive a vector of k visual features s to a desired value s^{*}. It has been previously applied for solving the problem of nonholonomic appearance-based navigation from a visual memory (see, e.g., [3] and [13]). Let us define:

$$u_{c} = [v_{c;x}\ v_{c;y}\ v_{c;z}\ \omega_{c;x}\ \omega_{c;y}\ \omega_{c;z}]^{T}$$

the camera velocity expressed in $\mathcal{F}_{\mathcal{C}}$. The matrix $\mathbf{L}_{S}$ relates the velocity of feature s to u_{c}:

$$\dot{s} = \mathbf{L}_{S} u_{c} \tag{1}$$

For the robot model that we are considering, the camera velocity u_{c} can be expressed in function of $u = [v\ \omega]^{T}$ by using the homogeneous transformation:

$$u_{c} = {}^{C}\mathbf{T}_{R} u \tag{2}$$

with:

$${}^{C}\mathbf{T}_{R} = \begin{bmatrix} 0 & -\delta \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}$$

In the following, we will call T_{v} and T_{ω} the first and second columns of ${}^{C}\mathbf{T}_{R}$. Injecting (2) in (1), we obtain:

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

Figure 4: Tracking in complex environment within visual servoing: Images are acquired and processed at video rate (25Hz). Blue: desired position defined by the user Green: position measured after pose calculation. (a) first image initialized by hand, (b) partial occlusion with hand, (c) lighting variation, (d) final image with various occlusions

and partial occlusions by an hand and various work-tools, as well as modification of the lighting conditions were imposed during the realization of the positioning task. On the third experiments (see Figure 6), after a complex positioning task (note that some object faces appeared while other disappeared), the object is handled by hand and moved around. Since the visual servoing task has not been stopped, robot is still moving in order to maintain the rigid link between the camera and the object.

For the second experiment, plots are also shown which helps to analyse the pose estimation, the robot velocity and the error vector. We can see that the robot velocity reaches 23 cm/s in translation and 85 dg/s in rotation. In other words, less than 35 frames were acquired during the entire positioning task up until convergence despite the large displacement to achieve (see Figure 5e). Therefore the task was accomplished in less than 1 second. Let us note that in all these experiments, neither a Kalman filter (or other prediction process) nor the camera displacement were used to help the tracking.