Experimental Evaluation of Autonomous Driving Based on Visual Memory and Image-Based Visual Servoing

Albert Diosi, Sini a egvic, Anthony Remazeilles, and Françoi Chaumette

***Abstract*—In this paper, the performance of a topological–metric visual-path-following framework is investigated in different environments. The framework relies on a monocular camera as the only sensing modality. The path is represented as a series of reference images such that each neighboring pair contains a number of common landmarks. Local 3-D geometries are reconstructed between the neighboring reference images to achieve fast feature prediction. This condition allows recovery from tracking failures. During navigation, the robot is controlled using image-based visual servoing. The focus of this paper is on the results from a number of experiments that were conducted in different environments, lighting conditions, and seasons. The experiments with a robot car show that the framework is robust to moving objects and moderate illumination changes. It is also shown that the system is capable of online path learning.**

***Index Terms*—Localization, mapping, path following, visual memory, visual servoing.**

I. Introduction

INTELLIGENT autonomous vehicles have performed amazing feats outdoors. They have driven thousands of kilometers on freeways [31], navigated on the surface of Mars [6], and driven more than 200 km on a challenging desert route [37]. Systems based on visual odometry, stereo vision, and inertial measurement units have achieved signi cantly high precision, e.g., an error of only 9 m after a 9-km travel [18]. Even monocular-vision-based map building is in the realm of mapping whole suburbs [27] or rapidly performing loop closure detection on images collected over a 1000-km path [10]. However, reliable autonomous navigation outdoors using one camera and no other sensor still remains an exciting challenge.

One of the approaches for autonomous navigation using monocular vision is visual path following. In visual path following, a path to follow can be represented by a series of reference images and corresponding robot actions (e.g., go forward, turn left, and turn right), as discussed in [24], where a mobile robot navigated through indoor corridors by applying template matching to current and reference images and using the stored actions. However, storing the robot actions is not necessary for navigation. In [33], a robot navigates a 127-m-long path outdoors while saving only a series of images from a camera with a sh-eye lens. To enable pose-based control of the robot in a global metric coordinate frame, a precise 3-D reconstruction of the camera poses is performed of the frequently (approximately every 70 cm) saved reference images. In the 3-D reconstruction process applied to feature points of the reference images, global bundle adjustment is used, which results in a long (1-h) learning phase unsuitable for online use. The length of the path measured by odometry is used to correct the scale of the map. After learning the path, the robot can very accurately reproduce it at a 50-cm/s velocity.

It turns out that reconstructing the robot s path or having 3-D information is not necessary. In [4], a robot navigated 140 m outdoors at a speed of 35 cm/s with only 2-D image information. During mapping, image features were tracked, and their image patches together with their image coordinates were approximately saved every 60 cm traveled. During navigation, the robot control was based on simple rules applied to the tracked feature coordinates shared between the next reference and the current image. The robot, however, relied on frequent reference image switches to recover from occlusions due to moving objects. A person who walks across the camera s eld

Manuscript received April 18, 2010; revised December 21, 2010; accepted February 3, 2011. Date of publication April 5, 2011; date of current version September 6, 2011. This work was supported in part through the French national projects Predit MobiVIP and CityVIP. The Associate Editor for this paper was L. Li.

A. Diosi was with the INRIA, Rennes-Bretagne Atlantique IRISA, Campus Beaulieu, 35042 Rennes Cedex, France (e-mail: albert.diosi@gmail.com).

S. egvic was with the INRIA, Rennes-Bretagne Atlantique IRISA, Campus Beaulieu, 35042 Rennes Cedex, France. He is now with the Department of Electronics, Microelectronics, Computer, and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia (e-mail: sinisa.segvic@fer.hr).

A. Remazeilles was with the INRIA, 35042 Rennes Cedex, France. He is now with the Health Unit, Fatronik-Tecnalia, 20009 Donostia-San Sebastian, Spain (e-mail: aremazeilles@fatronik.com).

F. Chaumette is with the INRIA, Rennes-Bretagne Atlantique IRISA, Campus Beaulieu, 35042 Rennes Cedex, France (e-mail: Francois.Chaumette@irisa.fr).

Color versions of one or more of the gures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identi er 10.1109/TITS.2011.2122334

Similar to [15], in the work described in [2], indoor navigation is performed using omnidirectional vision. However, the epipolar geometry is only calculated to validate that the scale-invariant feature transform (SIFT) descriptor [21] matches between the current and reference image keypoints and to calculate the required heading direction. In a thorough experimental evaluation, the authors demonstrated that their system can plan and execute motions in pure-appearance-based topological maps, whereas signi cant part of the robot s view was covered up by moving people.

Recently, Courbon *et al.* in [9] have successfully demonstrated outdoor visual path following on a 754-m-long outdoor track using a pose-based control strategy. Their topological map consisted of reference images. During navigation, the robot pose was estimated using homography recovery applied to images covering a 185 eld of view. A step toward commercialization is presented in [8], because a similar framework is applied to an indoor robot with a potentially inexpensive processing unit that entails an ARM9 microprocessor and a eld-programmable gate array (FPGA).

Not all robots in the visual-path-following literature use manually controlled map acquisition. In [13], the robot generated an indoor topological metric map by performing random motions. During mapping, the 3-D positions of point features of individual reference images were estimated using visual and odometry measurements fused together in a Kalman lter. The estimation of point feature positions continued during navigation.

Convincing experimental results for outdoor visual path following using omnidirectional vision and odometry are presented in [40]. In the simple effective approach, 1-D localization along the path is performed using a particle lter in conjunction with odometry and an effective patch-normalized implementation of correlation-based image matching. In the thorough experimental results, the accuracy and the effects of illumination were investigated.

Building an accurate and consistent 3-D representation of the environment can also be done using monocular simultaneous localization and mapping (SLAM) [11]. For example, in [19], a robot mapped a 100-m path outdoors using a monocular camera and odometry. There were only 350 features in the map, which may approach the limit that a simple Kalman lter SLAM implementation can handle in real time on current PCs. However, the simulation result in [14] of closing millions of landmark loops and building large hierarchical maps with monocular SLAM [7] predicts that monocular SLAM may be a viable choice for creating accurate maps with large numbers of landmarks.

In this paper, a visual-path-following framework is presented to the research community in intelligent transportation systems. General concepts such as representing paths as a series of images and extracting these series of images from an image database were presented in [32]. This paper, on the other hand, is oriented toward applying the same general idea for controlling real autonomous cars. An account of the employed vision system has previously been presented in [35]. In this paper, the presented experiments describe the behavior of the system in several different outdoor environments and thus provide qualitative and quantitative insights into the feasible range of navigation performance. In addition, this paper presents a more advanced implementation of the system, with a re ned control law and an improved implementation of the vision system. Consequently, the system presented in this paper exhibits faster, smoother, and safer motions and can perform online mapping. The contribution of this paper, based on [12],[1] is the application of the vision system to a robotic vehicle using an image-based visual servoing strategy and the experimental exploration of the implementation s limits.[2] Experiments were mostly carried out on roads using an autonomous electric vehicle that can carry two passengers.

The presented framework is similar to [15], because only local 3-D reconstruction is used, and occluded features get projected back into the image. However, the rest of the details are different. For example, in this paper, a standard camera is used instead of an omnidirectional camera, tracking is used for mapping instead of matching, experiments are done outdoors and not indoors, and the centroid of image features is used to control the robot.

This paper is organized as follows. A description of the framework is given in Section II. More details about the vision system for the interested reader are given in Section III, followed by a description of the experiments. After a discussion of the results, this paper ends with conclusions.

II. Visual Navigation

This section brie y describes the visual navigation framework that we implemented. The teaching of the robot (mapping) is rst described, followed by the description of the navigation process that consists of localization and robot control.

A. Mapping

Learning a path (i.e., mapping) starts with the manual driving of the robot on a reference path while processing (or storing for of ine mapping) the images from the robot s camera. Based on the images, an internal representation of the path is created, as summarized in Fig. 1. The mapping starts with nding Harris points in the rst image, initializing a Kanade Lucas Tomasi (KLT) feature tracker [36] and saving the rst image as the rst reference image. A version of the KLT tracker[3] was modi ed, as proposed in [17], to improve performance in outdoor sequences acquired from a moving car. In the tracker, the position, scale, and contrast parameters of features are tracked. In the next step, a new image is acquired, and the tracked features are updated. The tracking of features with a large appearance change compared to their reference image appearance is abandoned. The rest of the features are then used to estimate the

[1] Compared with [12], a gap in the experimental work has been lled, and more details of the vision system are given.

[2] Videos that show results presented in this paper can be accessed at http://www.irisa.fr/lagadic/video/CycabNavigation.mov and http://www.zemris.fer.hr~ssegvic/pubs/diosi_et_al_07iros_0581_VI_i.mp4 [Accessed: February 22, 2011].

[3] The source code of the KLT tracker maintained by S. Birch eld can be found at http://www.ces.clemson.edu/~stb/klt/ [Accessed: February 22, 2011].

C. Motion Control

In the motion control scheme, the robot is not required to accurately reach each reference image of the path or to accurately follow the learned path, because this condition may not be useful during navigation. In practice, the exact motion of the robot should be controlled by an obstacle avoidance module, which will constitute future work. Therefore, a simple control algorithm was implemented, where the difference in the x-coordinates (assuming that the forward-facing camera's horizontal axis is orthogonal to the axis of robot rotation) of the centroid of features in the current (x_c) and the next reference image (x_n) are fed back into the motion controller of the robot as the steering angle. We have

$$\Phi = -a(x_c - x_n) \tag{1}$$

where a is the gain. To smoothen rapid steering actions when switching reference images, a feedforward part is added to the steering angle. The calculation of the feedforward part is based on the centroids of the shared features in the current and second next (x_{nn}) reference images. Thus, the final equation is

$$\Phi = -a(x_c - x_n) - b(x_c - x_{nn}) \tag{2}$$

where b is the feedforward gain.

The translational velocity is set to a constant value, except during turns, where it is reduced to a smaller constant value to ease the tracking of rapidly moving features in the image. Such turns are automatically detected during navigation by thresholding the difference in the feature centroids in the current, next, and second next images.

The decision of when to stop when reaching the goal position is carried out similar to the reference-image-switching strategy in [4] by detecting when the variance of the difference between the current and last reference image feature coordinates starts to rise.

III. Vision Techniques and Algorithms

This section describes the main details about the vision techniques and algorithms employed. A key ability in vision path following is to correctly locate mapped features in images that were acquired during navigation. Therefore, we first introduce the following two basic approaches for establishing point correspondences between images: 1) wide-baseline matching in Section III-A and 2) tracking in Section III-B. Section III-C presents an empirical performance evaluation of the two correspondence approaches. The experiments indicate that correspondences recovered by tracking provide considerably more accurate 3-D reconstructions (aside from being several times faster to obtain). Consequently, we employ wide-baseline matching only for obtaining initial localization at the beginning of the navigation session (see the left of Fig. 3). Other localization steps and the whole mapping stage employ tracking, as detailed in Sections II-B and A, respectively.

The proposed localization subsystem often needs to (re)start the tracking of features that, for various reasons, were not tracked in the previous frame (see the right branch of the flowchart in Fig. 3). During the typical forward motion, tracked features gradually leave the field of view and need to be replaced by new features. In addition, tracked features may be lost in any moment due to local disturbances such as occlusion, motion blur, illumination effects, noise, or any combination thereof. A suitable geometric procedure has therefore been devised to predict the locations of mapped features that are currently not tracked. This procedure is described in Section III-D. Note that a part of this procedure (caching the recovered two-view geometries between the key images) is performed during the mapping stage (see Fig. 1), whereas the actual prediction is employed during navigation (see Fig. 3, predict point positions).

A. Keypoint Detection for Wide-Baseline Matching

The purpose of wide-baseline matching is to detect correspondences without any prior knowledge about the relative orientation of the two views. Our images are acquired from a moving car; therefore, we particularly require robustness to appearance distortion along the scale axis. The desired robustness can be achieved by matching *invariant feature descriptors* [26], which are independently extracted in both images. This approach is based on recent advances in robust and repeatable detection of characteristic image locations called *keypoints* [29]. Oftentimes, the detected keypoints are locally distinctive with respect to position, scale, and rotation, whereas some approaches even address affine invariance [25]. The descriptors obtained are exhaustively compared to the descriptors from the other image, typically with respect to the L2 distance. The correspondences are usually established as distinctive pairs, for which the best distance is less than 60% of the distance of the second best match [21].

We evaluated the following three keypoint detectors: 1) the maxima of the DoG [21]; 2) multiscale Harris corners [25]; and 3) MSER [23]. The three detectors extract different kinds of features (blobs, corners, and regions, respectively [39]) and complement each other with more or less success, depending on the scene. Our final procedure combines the correspondences obtained by individually matching the descriptors extracted by all three algorithms.

B. Point Feature Tracking

When approximate current feature locations are known (as is often the case when processing an image sequence), correspondences can be established by tracking. Two main point-feature-tracking approaches are given as follows: 1) iterative first-order differential approximation [17], [36] and 2) exhaustive matching of lightweight point features [22], [28], e.g., Harris corners. We believe that the former approach is better suited to appearance-based navigation, because it tends to be less susceptible to association errors and provides more accurate point tracks.

To tolerate significant interframe displacements, the features are first tracked between the previous and the current images

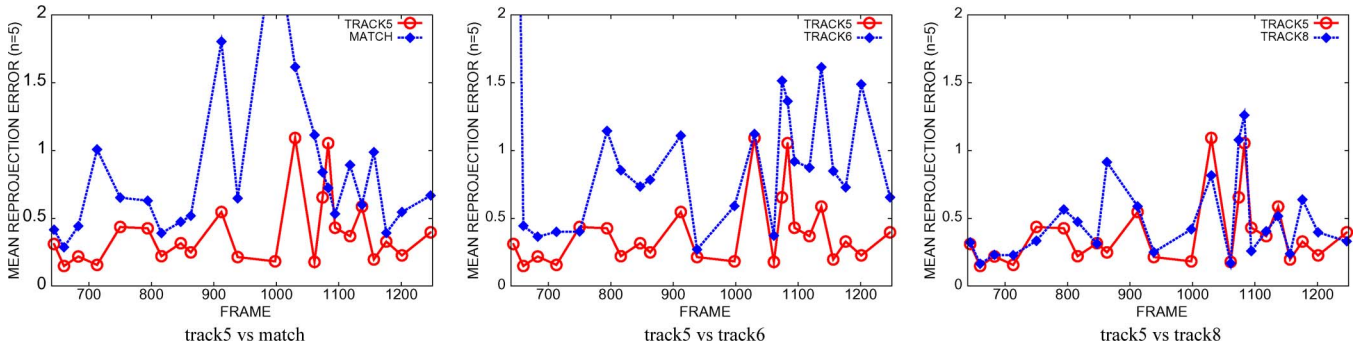

Fig. 4. Performance evaluation of four approaches for establishing correspondences between a given image and ve subsequent images in the sequenc The horizontal axis holds the sequential number of the frame, whereas the vertical axis shows the mean reprojection error.

the original code, but the most important among these modi cations are contrast compensation warp extensions as described in Section III-B2 and evaluated in Section III-C. In addition, we provided codes for (re)starting the tracking of features at predicted locations. Finally, we also improved warp correction results for features at large scales by employing the pyramid level that most closely matches the current feature size.

4) Computational Considerations: The performance of differential tracking comes at a price of considerable computational complexity. In fact, code pro ling[8] showed that the feature tracker is a major performance bottleneck of the navigation system presented in this paper. We are currently working on several opportunities to address this problem. Preliminary results indicate that the performance can be more than doubled by harnessing vector extensions of the x86 instruction set.

C. Performance Evaluation of the Correspondence Approaches

Evaluating the correspondence performance for real scenes is tricky, because ground-truth correspondences cannot typically be recovered in experiments with real 3-D scenes. Consequently, it is very dif cult to assess the alignment accuracy of the correspondences. However, a correct correspondence alignment is very important in feature-based navigation, because the existing correspondences are employed to predict locations of previously unseen features. Bad predictions can exceptionally be troublesome, because they may give rise to association errors and subsequent degradation of the geometrical quality within a positive feedback spiral. Here, we estimate the correspondence alignment accuracy by looking at the reprojection error of the recovered two-view geometries. The smaller the reprojection error of the resulting two-view geometry, the better the correspondence approach becomes. The four evaluated correspondence approaches are listed as follows:

- isotropic scaling with contrast compensation (track5);
- af ne warp with contrast compensation (track8);
- af ne warp without contrast compensation (track6);
- wide-baseline matching by employing Lowe s keypoints and SIFT descriptors (match).

[8]We employed the GNU pro ler gprof.

The experiment is designed as follows. For each of the 20 key images of the sequence referenced in Section IV-G, we look at correspondences between the key image (index i in the sequence), and the ve subsequent images at indices $i+1$, $i+2$, $i+3$, $i+4$, and $i+5$. For matching, we simply match the pairs $(i, i+1)$, $(i, i+2)$, and so on. For tracking, we initialize the tracker at index i, and then, track5 frames forward. In each case, we record ve reprojection errors for geometries from $(i, i+1)$ to $(i, i+5)$. The results are summarized in Fig. 4 as means of the ve reprojection errors.

The results illustrate that the correspondences obtained by tracking with contrast compensation yield the overall better and signi cantly more stable two-view geometries than the correspondences obtained by matching (track5 versus match). The gure also shows that contrast compensation provides a signi cant performance gain when tracking outdoors (track5 versus track6). Finally, the gure suggests that track5 is somewhat better than track8 (track5 versus track8). Our result that tracking performs better than matching is consistent with the ndings in [34], where a similar comparison was performed.

D. Decomposed Point Transfer in the Calibrated Context

The main shortcoming of tracking is that it requires an auxiliary technique for establishing initial correspondences and recovering from tracking failures. We address this problem by providing a module for predicting the locations of features that are currently not tracked. After an approximate feature location has been provided by the prediction module, the correct location can be recovered by differential tracking with warp correction with respect to the reference appearance acquired during the mapping stage. Feature prediction is therefore a critical task that enables the system to deal with large motions and local disturbances by providing means for a dynamic update of tracked features.

The adopted feature prediction approach exploits geometric constraints provided by currently tracked features and their mapped correspondences within the frame of a technique known as *point transfer* [16]. Point transfer locates an unknown 2-D point in the current image by employing the following elements: 1) the known projections of the same 3-D point in two other images and 2) some additional correspondences across the three images. This problem is illustrated in Fig. 5.

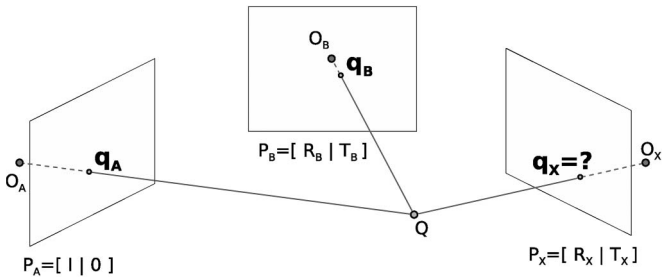

Fig. 5. Point transfer problem. Given two known projections of the same point $\mathbf{Q}$ onto key images A and B, nd its projection in a current view X. The decomposed solution of that problem is given as follows: 1) Image correspondences are used to recover the two-view geometry (A, B); 2) the two known projections $\mathbf{q_A}$ and $\mathbf{q_B}$ are used to triangulate the 3-D point $\mathbf{Q}$; 3) the two-view geometry (A, X) is recovered and put into the frame of the geometry (A, B); and 4) the desired point $\mathbf{q_X}$ is obtained by projecting $\mathbf{Q}$ onto image X.

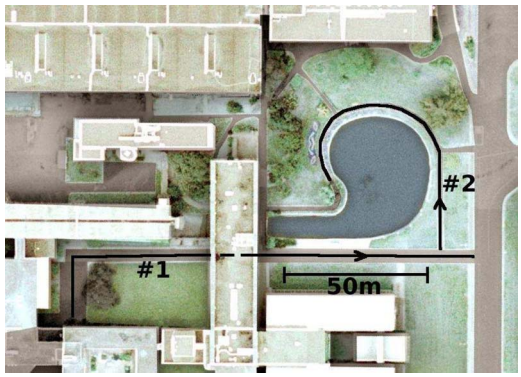

Fig. 6. Paths for experiments 1 and 2.

To perform the point transfer, we need to recover the three-view geometry between the current image and two key images from the map.

There are several ways of computing the three-view geometry, with different assumptions and performance requirements. The golden standard method described in [16] involves bundle adjustment with respect to the reprojection error in all views, which may be costly for a real-time implementation. A more suitable solution would observe that several three-view geometries need to be recovered for the same key-image pair during navigation and therefore strive to reuse precomputed two-view geometries for such pairs. Such a decomposed solution has been proposed in [20]. A similar approach has been employed in this paper but within the calibrated context, i.e., by assuming that all observed points have been expressed in normalized coordinates[9] that correspond to the case of unit focal distance [22]. Some implementation details of our solution will be described as follows.

Each of the two geometries (A, B) and (A, X) (see Fig. 5) is independently recovered. The two essential matrices are estimated by the random sampling scheme MLESAC [38], using the recent five-point algorithm [29] as the generator of motion hypotheses. The implementation employed has been provided within the library VW34[10] maintained at the Imperial College, London, U.K.. The decomposition of the essential matrix into motion components is performed next, followed by the triangulation of 3-D points [16].

Consequently, the geometries (A, B) and (A, X) (see Fig. 5) are expressed in the common frame. In the calibrated context, the adjustment involves the estimation of only one parameter (scale), whereas in the projective context, the ambiguity has 4 degrees of freedom (DOFs) [20]. The scale factor between two metric frames is estimated by requiring that pairs of corresponding points visible in both frames have the same depth. In practice, different points vote for different scale factors due to noise, but a robust result is, in the end, obtained as the median of all individual factors.

Three-dimensional coordinates of the desired point $\mathbf{Q}$ are obtained by triangulating its projections onto the two key images A and B (see Fig. 5). This approach can be performed of line during mapping. The desired prediction $\mathbf{q_X}$ of the triangulated point $\mathbf{Q}$ to the current image X is finally obtained by simple projection.

The aforementioned prediction procedure is very sensitive to the accuracy of the estimated two-view geometries. Thus, it makes sense to disregard the predictions when the estimates appear to be inaccurate with respect to the reprojection error [16]. The reprojection error may be determined either in a straightforward manner or, as calculated in this paper, by taking into account the probability that a bad geometry may produce a low reprojection error by chance (as proposed in [35]).

IV. Experimental Results

The goal of our experiments is to explore the possibilities and limits of the current implementation of the framework by navigating in different scenarios, environments with different proportions of vegetation to human made structures, and different illumination conditions. We also explore the limit in speed and in lateral deviation from the path. A practical application of online mapping and autonomous parking is also given. The results are quantitatively evaluated.

In all but the last experiment, a CyCab a French-made four-wheel-drive four-wheel-steered intelligent vehicle designed to carry two passengers was used. On our CyCab, all computations, except for the low-level control, were carried out on a laptop with a 2-GHz Pentium M processor. A 70° field of view, forward looking, B&W Allied Vision Marlin (F-131B) camera was mounted on the robot at a 65-cm height. Except for experiment 3, the camera was used in the autoshutter mode, with the rest of the settings constant. During all experiments (except the last experiment), no software parameters were changed, except for the forward and turning speed. Mapping has been performed of line, except in experiment 6. The image resolution in the experiments was 320×240. Tracked feature patch sizes were 15 × 15 pixels.

[9]We employ the usual model for transforming pixels into normalized coordinates that comprise a 5-DOF linear transformation and the fourth-order radial distortion model [41]. We recovered calibration parameters for our cameras by employing our own implementation of the procedure with a planar calibration target as described in [41].

[10]URL http://www.doc.ic.ac.uk/~ajd/Scene/Release/vw34.tar.gz [Accessed: February 22, 2011].

TABLE I
SUMMARY OF THE VISUAL-PATH-FOLLOWING EXPERIMENTS

Fig. 7. Navigation results in experiment 1 shown as reconstructed robot poses (black) overlaid on 77 reconstructed reference image poses (lighter colored dots and barely visible sequence numbers). (Bottom left) First reference image pose.

A. Experiment 1: Basic Experiment

Experiment 1 (see Fig. 6) was conducted on an overcast day with a short time between mapping and navigation. Most views on the 158-m-long path contained buildings which provided stable image features. The main potential challenges in this experiment were given as follows: 1) motion blur in the teaching sequence caused by fast driving for the used exposure times; 2) driving under a building, which caused a quick illumination change; and 3) people (more than ten) and cars covering up features during navigation.

In the teaching phase, 958 logged images were reduced into 77 reference images in 257 s (3.7 fps). While the robot was moving at 50 cm/s in turns and at 90 cm/s otherwise during navigation, 934 images were processed at 4.1 fps on the average. Statistics with regard to mapping and navigation are shown in Table I. Reconstructed robot and reference image poses shown in Fig. 7 were used only for assessing the performance of the system.

The quick illumination change when driving under the building was easily handled due to the implemented illumination compensation in the tracker [17]. Motion blur in the teaching sequence did not impair the performance of the system. The moving objects and persons did not affect the navigation, because the tracking of features that reappear after occlusion was immediately restarted due to the feature reprojection scheme. Fig. 8 contains images processed at the end of the navigation. They describe an interesting situation where a moving car progressively occludes most features. It is shown that the tracking of reappearing features is restarted, because there were enough good features that were tracked for the camera pose estimation used in point reprojection.

B. Experiment 2: Robustness to Environment Changes

Experiment 2 was conducted on a narrow path along a small lake (see Figs. 6 and 12). Mapping was carried out in June, under the strong summer sun (see Fig. 9). Navigation took place in October, when vegetation and illumination conditions were very different (see Fig. 10). Despite the large change in the environment, CyCab managed to navigate about 80% of the path, with only one human intervention. At one place, CyCab started to brush the rose plants on the left side of the path (the inside of the bend) in Fig. 10; therefore, we stopped the vehicle. Such a corner-cutting behavior naturally comes with wide separation between reference images and the chosen control strategy. Without stopping the vision system, CyCab was moved 50 cm to the right, and its automatic motion was resumed. CyCab s vision system gave up close to the end of the track when the change in the environment was very large (see Figs. 11 and 12). Although CyCab did not complete the whole path (see the left side of Fig. 9, where it failed), this experiment still represents a success because of the dif cult conditions that CyCab handled.

Shortly after CyCab got lost, we have repeated the experiment using a new map. As it shown in Fig. 9, right, CyCab completed the path without any problems and with smaller localization noise. Note that, because image-based visual servoing was used, localization noise had only an indirect effect on the motion of the robot, because it in uenced only feature point reprojection and reference image switching.

This experiment indicates that seasonal vegetation changes may negatively affect the performance of the framework in environments where most features are provided by the vegetation. This experiment also suggests that, in the short term, under favorable conditions, vegetation may provide a large number of well-textured features, which can result in high-quality 3-D geometry estimation. However, unfavorable conditions such as wind or rain may easily degrade the quality of the created map. The frame rates during navigation are lower in this experiment (see Table I) due to temporary implementation and processing platform limitations.

C. Experiment 3: Deterioration Due to Distant Features

In experiment 3, CyCab completed an approximately 304-m track, where in some places (see the right side of Fig. 13), the closest features were more than 100 m away. Because the width of the footpath matched the width of CyCab, it was easy to observe the lateral error during navigation. The mapping and navigation part of the experiment was conducted in succession, under very bright lighting conditions. Instead of the usual autoshutter mode, the camera was used in its high-dynamic-range mode. The start and end positions were identical.

As we can expect, the error in the estimated pose during navigation was the largest at places where there were no close features. Such large pose errors are represented by cluttered

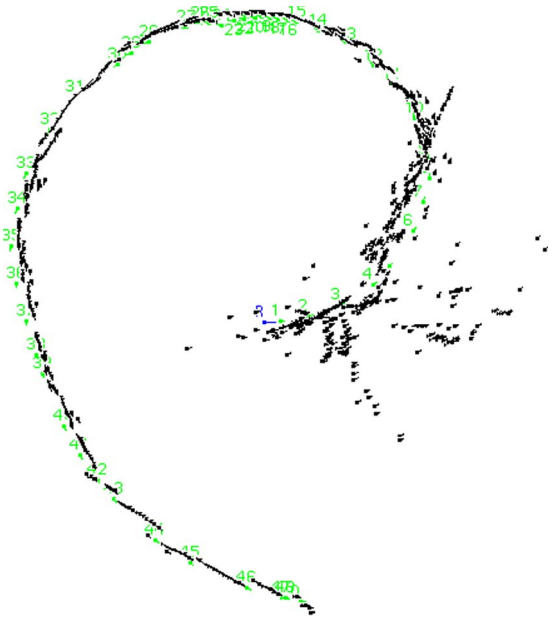

Fig. 16. Navigation results in the loop closing experiment (experiment 4).

Fig. 17. Sun shines into the camera in the reference image (left) experiment 4 but not in the current image (right) during navigation.

D. Experiment 4: Driving in a Loop

The aim of this experiment was to investigate navigation in a loop. The teaching was performed by driving CyCab in a full loop in a circular parking lot of an approximately 119-m circumference. The beginning and end of the loop were closed by matching the first and last images of the teaching sequence. If neighboring nodes were connected with line segments, then the first and the last light colored dots in Fig. 16 were connected.

CyCab managed to complete 1.25 loops, although the experiment was conducted at the end of day, where people were driving their cars away from the car park, and the sun was shining into the camera (see Fig. 17). The change in the scene worsened at the beginning of the second loop, where one of these cars provided the only close features. The lack of good features in conjunction with a lateral error resulted in very poor pose estimates, as shown in Fig. 16. Therefore, this experiment demonstrated that the lack of global consistency in pose does not preclude navigation, as long as local consistency is ensured (in Fig. 16, the path does not join up into a circle). This case is due to the ability of the image-based visual servoing scheme to handle situations where pose-based schemes may struggle when fed with poor pose estimates.

Eventually, CyCab was manually stopped when it no longer followed the curvature of the road (see the short straight section of black dots in Fig. 16, where it happened); however, the experiment was a success, because it demonstrated that CyCab can connect the beginning and end of a loop and drive through the joint.

Fig. 18. First images during navigation in experiments 5 (left) and 6 (right). In experiment 5, the robot drove until the end of the road while maintaining a 1.8-m/s speed. In experiment 6, after online path learning, the robot parked itself into the garage close to the center of the image.

Fig. 19. Navigation results in experiment 5.

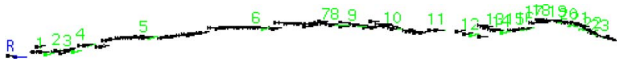

Fig. 20. Navigation results in experiment 6.

E. Experiment 5: Robustness to Speed

This experiment investigates how fast CyCab can navigate on a straight path. On the track shown in Figs. 18 and 19, CyCab completed a 100-m straight path at a 1.8-m/s (6.5-km/h) speed. Raising the speed even higher caused oscillations in the robot's motion to appear. The oscillations were presumably caused by the delay between image measurements and control action and by the frame rate.

F. Experiment 6: Application to Automatic Parking With Online Mapping

In this experiment, online mapping (i.e., processing the images as they are grabbed) and a practical application are demonstrated. In the current state of the navigation system, i.e., without obstacle detection and avoidance, the practical applications are limited. However, even now, the framework can be used for automatic parking on private properties, which are under the control of the user.

During the experiment, a map was created online while driving CyCab from the entrance of the Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA) to the CyCab garage approximately 50 m away (see Figs. 18 and 20) at about 50 cm/s. Then, CyCab was manually driven to the entrance of IRISA, where the driver got out, and CyCab drove itself into the garage. During mapping, clouds covered the sun, whereas during navigation, the sky was clear. CyCab even handled the transition from strong sunshine to the darkness of the garage.