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

Albert Diosi, Siniša Šegvić, Anthony Remazeilles, and François 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 significantly 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 fish-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 x-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 field of view between two reference image switches could have caused a problem due to covering up each tracked feature. In a later work [5], the authors in [4] added odometry to compensate for roll on a nonflat terrain.

The work described in [15] aimed at indoor navigation and can deal with occlusion but at the price of using 3-D information. A local 3-D reconstruction is done between two reference omnidirectional images. During navigation, tracked features that have been occluded get projected back into the current image. The recovered pose of the robot is used to guide the robot toward the target image.

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. Šegvić 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 figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 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 significant 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° field 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 field-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 filter. 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 filter 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 filter 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 refined 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 briefly describes the visual navigation framework that we implemented. The teaching of the robot (mapping) is first 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 offline 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 finding Harris points in the first image, initializing a Kanade–Lucas–Tomasi (KLT) feature tracker [36] and saving the first image as the first reference image. A version of the KLT[3] tracker was modified, 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 filled, 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. Birchfield can be found at http://www.ces.clemson.edu/~stb/klt/ [Accessed: February 22, 2011].

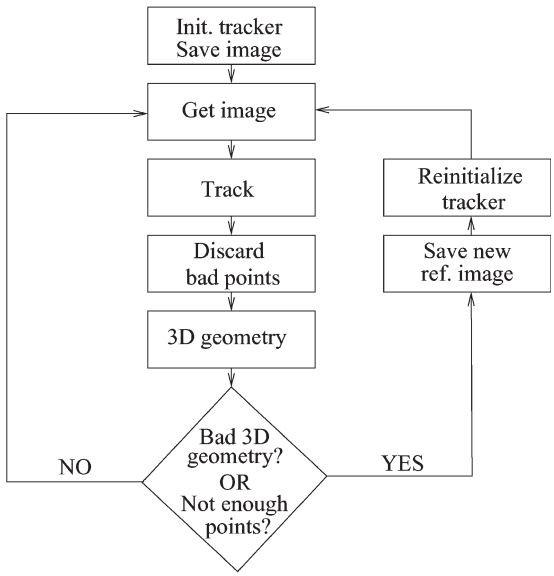

Fig. 1. Steps involved in building a representation of a path from a sequence of images, i.e., mapping.

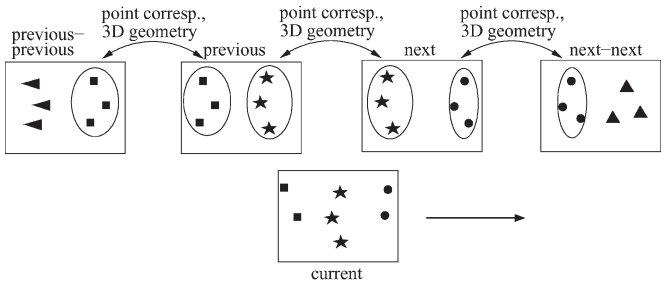

Fig. 2. Map consists of reference images, point correspondences, and 2-D and 3-D information. During navigation, the point features from the map are projected into the current image and are tracked.

3-D geometry between the previous reference and the current image. In the 3-D geometry estimation, the essential matrix is recovered using the calibrated five-point algorithm[4] [29] used in the maximum likelihood estimation sample consensus (MLESAC) [38] random sampling framework. The inlier points are then used in a final 3-D geometry calculation using the eight-point algorithm [38]. If the 3-D reconstruction error is low and there are enough tracked features, a new image is acquired. Otherwise, the previous image is saved as the new reference image. The relative pose of the previous image with respect to the previous reference image and the 2-D and 3-D coordinates of the point features shared with the previous reference image are also saved. Then, the tracker is reinitialized with new Harris points added to the old points, and the processing loop continues with a new image acquired by the camera.

To handle gaps in the image sequence and to close a loop between the first and the last images of the teaching sequence, wide-baseline matching is utilized, as described in Section III.

The resulting map (see Fig. 2) is used during autonomous navigation in the localization module to provide stable image points for image-based visual servoing.

[4]An implementation is available in the VW Library downloadable from http://www.doc.ic.ac.uk/~ajd/Scene/index.html [Accessed: February 22, 2011].

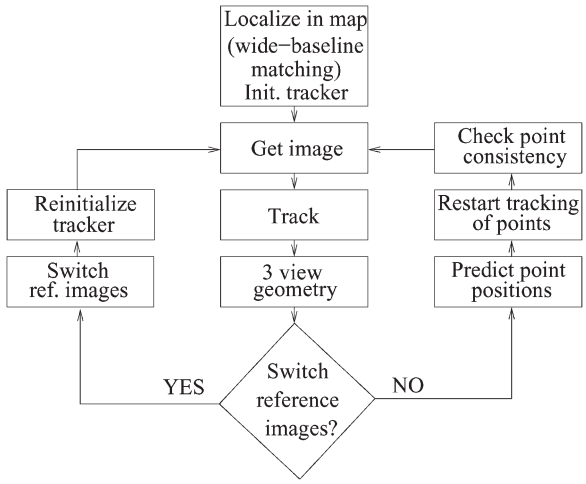

Fig. 3. Visual localization during navigation.

B. Localization

The localization process during navigation is depicted in Fig. 3. The navigation starts with initial localization, where the user selects a pair of reference images close to the robot's current location. Then, an image is acquired and matched to the selected reference images. Wide-baseline matching is done using a correlation-based approach [42] and by matching SIFT descriptors [21] applied to difference of Gaussians (DoG) [21], multiscale Harris [25], and maximally stable extremal regions (MSER) [23] keypoints. The estimation of the camera pose using the matched points enables us to project map points from the reference images into the current image. The projected points are then used to initialize a KLT tracker.

After the initial localization, a new image is acquired, and the point positions are updated by the tracker. Using the tracked points, a three-view geometry calculation (see Section III) is performed between the current image, the previous reference, and the next reference image (see Fig. 2). If the current image is found to precede the next reference image, then points from the map are projected into the current image using the estimated local pose. The projected points enable us to restart the tracking of points that are currently not tracked and to stop the tracking of points that are far from their projections. A new image is acquired next, and the whole cycle continues. However, if it is found that the current image comes after the next reference image, a topological transition is made, i.e., the second next reference image (see Fig. 2) becomes the next reference image. The tracker is then reinitialized with points from the map, and the process continues with acquiring a new image. Similar to forward transitions, a topological transition is performed backward if the current image precedes the previous reference image.

To achieve seamless switching between nodes, points from the second next, previous, and second previous reference images are also tracked in the current image (see Fig. 2).

Wide-baseline matching is only used outside the initial localization phase if most features are lost, e.g., due to a total obstruction of the camera's field of view. In such a case, the robot stops, and automatic reinitialization is carried out by matching with the nearest reference images.

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 Φ. 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 top 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* [39]. 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

across a multilevel resolution pyramid.[5] Then, to avoid drift accumulation, the current appearance is warped to achieve optimal resemblance with the stored template image or *reference*.[6] This alignment can be achieved by minimizing the norm of the *error image* obtained by subtracting the warped current feature from the reference [1]. Shi and Tomasi [36] have described the warp as a 2-D affine transform. An extended warp, which, in addition, compensates for affine photometric deformations of the gray-level value in the image, has been proposed in [17].

In the rest of this section, we first provide a formulation of the general point feature tracker [1], [17], [36] in Section III-B1, and then, in Section III-B2, we describe our variant of the concept, with which we obtained best results. The main changes of our final implementation with respect to the public KLT library are outlined in Section III-B3, whereas in Section III-B4, we summarize some computational considerations.

1) General Differential Tracker With Warp Correction: Let the feature in the current frame be given by $I(\mathbf{x})$, its appearance after a warp with parameters $\mathbf{p}$ by $I_W(\mathbf{x},\mathbf{p})$, and the corresponding reference by $I_R(\mathbf{x})$. Then, the differential tracking consists of finding $\hat{\mathbf{p}}$, which minimizes the norm of the error over the feature window. We have

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}} \left\| I_W(\mathbf{x},\mathbf{p}) - I_R(\mathbf{x}) \right\|. \tag{3}$$

The minimization is performed in a Gauss–Newton style, by employing a first-order Taylor expansion of the warped feature around the previous approximation of $\hat{\mathbf{p}}$. This formulation can be expressed in different ways [1], and here, we present a "forward-additive" formulation, with which the best accuracy has been obtained. In this formulation, the current feature warped with a sum of the previous parameter vector $\mathbf{p}$, and an unknown additive improvement $\Delta\mathbf{p}$ is approximated as

$$I_W(\mathbf{x},\mathbf{p}+\Delta\mathbf{p}) \approx I_W(\mathbf{x},\mathbf{p}) + \frac{\partial I_W}{\partial \mathbf{p}} \cdot \Delta\mathbf{p}. \tag{4}$$

The scalar residual norm that appears in (3) can now be represented as

$$\begin{aligned} R(\Delta\mathbf{p}) &= \sum_{\mathbf{x}} \left\| I_W(\mathbf{x},\mathbf{p}+\Delta\mathbf{p}) - I_R(\mathbf{x}) \right\| \\ &\approx \sum_{\mathbf{x}} \left\| I_W(\mathbf{x},\mathbf{p}) + \frac{\partial I_W}{\partial \mathbf{p}} \cdot \Delta\mathbf{p} - I_R(\mathbf{x}) \right\|. \end{aligned} \tag{5}$$

For clarity, we omit the arguments, denote the previous error image as e, and introduce $\mathbf{g}$ as the transposed warped feature gradient over the warp parameters. We have

$$R(\Delta\mathbf{p}) \approx \sum_{\mathbf{x}} \left\| e + \mathbf{g}^\top \Delta\mathbf{p} \right\|. \tag{6}$$

[5]We used two additional pyramid levels, which are iteratively obtained by subsampling each second pixel in a properly smoothed image.

[6]During mapping, the reference is obtained by simply storing the first appearance of the feature. During localization, the reference is taken from the corresponding key image.

The requirement (3) can be enforced by finding a $\Delta\hat{\mathbf{p}}$ for which the gradient of the residual vanishes. In case of the L2 norm, this approach is easy to perform, i.e.,

$$\frac{\partial R(\Delta\hat{\mathbf{p}})}{\partial \Delta\hat{\mathbf{p}}} \approx \sum_{\mathbf{x}} 2 \cdot (e + \mathbf{g}^\top \Delta\hat{\mathbf{p}}) \cdot \mathbf{g}^\top = \mathbf{0}^\top. \tag{7}$$

After transposing both ends of (7), we arrive at the final expression for an iteration in the context of a general warp (note that e is a scalar function) as follows:

$$\sum_{\mathbf{x}} (\mathbf{g}e + \mathbf{g}\mathbf{g}^\top \Delta\hat{\mathbf{p}}) = \mathbf{0}. \tag{8}$$

Thus, in each iteration, the additive improvement is calculated by solving a linear system of equations. The procedure stops when the norm of the improvement $\|\Delta\hat{\mathbf{p}}\|$ falls below a threshold, when the new feature position falls outside the image bounds, or when the determinant $|\mathbf{g}\mathbf{g}^\top|$ becomes very small.

2) Differential Tracker With Isotropic Scaling and Contrast Compensation: To mitigate the danger that a physically unrelated image patch might be well transformed toward the reference, a tradeoff between modeling power and tracking security should carefully be chosen. For our application, a good balance is obtained by a 5-D warp that consists of a 2-D translational offset $(\mathbf{d})$, isotropic scaling (m), and the affine contrast compensation model (λ,δ) [17]. It is convenient to express the warp in terms of geometric and photometric components as $\mathbf{p} = (\mathbf{q},\mathbf{r})$, where $\mathbf{q} = (m,\mathbf{d})$, and $\mathbf{r} = (\lambda,\delta)$. The warped feature is then obtained as

$$I_W(\mathbf{x},\mathbf{p}) = \lambda \cdot I(m * \mathbf{x} + \mathbf{d}) + \delta = U\left(I(T(\mathbf{x},\mathbf{q})),\mathbf{r}\right). \tag{9}$$

To use the general formulation given in (8), an expression for $\partial I_W/\partial\mathbf{p} = [\partial U/\partial\mathbf{q} \quad \partial U/\partial\mathbf{r}]$ must be derived using the chain rule. The second term is simpler to obtain. We have

$$\frac{\partial U}{\partial \mathbf{r}}\left(I(T(\mathbf{x},\mathbf{q})),\mathbf{r}\right) = [I_T \quad 1] \tag{10}$$

where I_T is the current feature warped with T, $I_T = I(T(\mathbf{x},\mathbf{q}))$. The derivative of the first term is more involved as

$$\begin{aligned} &\frac{\partial U}{\partial \mathbf{q}}\left(I\left(T(\mathbf{x},\mathbf{q})\right),\mathbf{r}\right) \\ &\quad = \frac{\partial U}{\partial I}\left(I\left(T(\mathbf{x},\mathbf{q})\right),\mathbf{r}\right) \cdot \frac{\partial I}{\partial T}\left(T(\mathbf{x},\mathbf{q})\right) \cdot \frac{\partial T}{\partial \mathbf{q}}(\mathbf{x},\mathbf{q}) \\ &\quad = \lambda \cdot I_T^x \cdot \begin{bmatrix} x_1 & 1 & 0 \\ x_2 & 0 & 1 \end{bmatrix} = \lambda \begin{bmatrix} I_T^x\mathbf{x} & I_T^{x1} & I_T^{x2} \end{bmatrix} \end{aligned} \tag{11}$$

where I_T^x is the gradient in the feature warped by T, $I_T^x = \frac{\partial I}{\partial T}(T(\mathbf{x},\mathbf{q}))$. The combined result, (10) and (11), can be inserted into (8), with $\mathbf{g}$ given by

$$\mathbf{g}^\top = \begin{bmatrix} \lambda I_T^x\mathbf{x} & \lambda I_T^{x1} & \lambda I_T^{x2} & I_T & 1 \end{bmatrix}. \tag{12}$$

3) Implementation: Our implementation of the KLT tracker derives from the public library maintained by S. Birchfield at Clemson University.[7] We performed several modifications to

[7]URL: http://www.ces.clemson.edu/~stb/klt/ [Accessed: February 22, 2011].

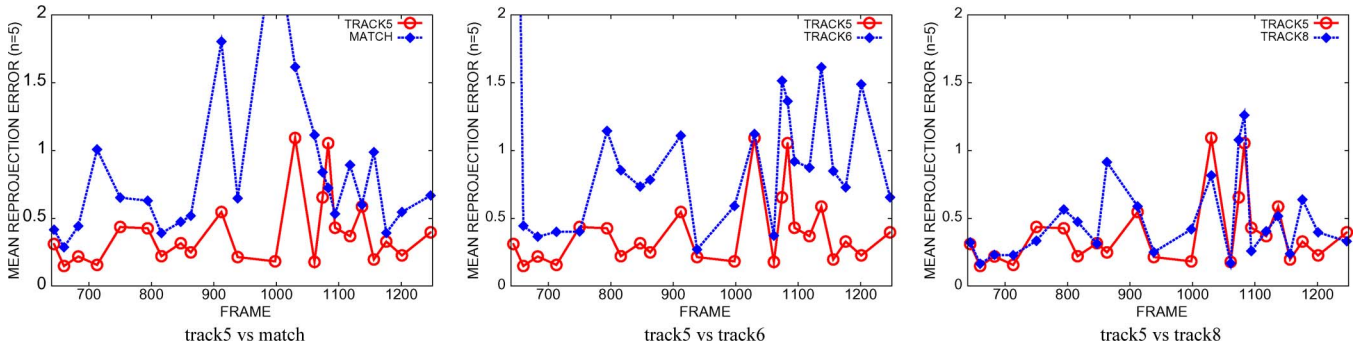

Fig. 4. Performance evaluation of four approaches for establishing correspondences between a given image and five subsequent images in the sequence. The horizontal axis holds the sequential number of the frame, whereas the vertical axis shows the mean reprojection error.

the original code, but the most important among these modifications 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 profiling[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 difficult 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);
- affine warp with contrast compensation (track8);
- affine warp without contrast compensation (track6);
- wide-baseline matching by employing Lowe's keypoints and SIFT descriptors (match).

[8]We employed the GNU profiler gprof.

The experiment is designed as follows. For each of the 23 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 five 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 five reprojection errors for geometries from $(i, i+1)$ to $(i, i+5)$. The results are summarized in Fig. 4 as means of the five reprojection errors.

The results illustrate that the correspondences obtained by tracking with contrast compensation yield the overall better and significantly more stable two-view geometries than the correspondences obtained by matching (track5 versus match). The figure also shows that contrast compensation provides a significant performance gain when tracking outdoors (track5 versus track6). Finally, the figure suggests that track5 is somewhat better than track8 (track5 versus track8). Our result that tracking performs better than matching is consistent with the findings 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.

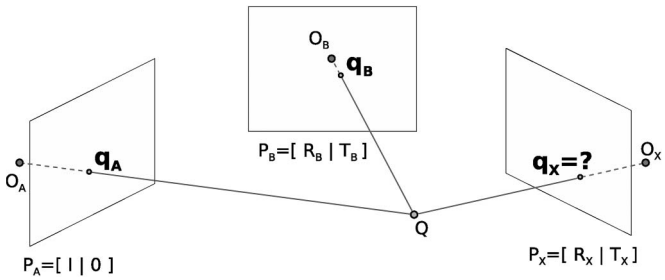

Fig. 5. Point transfer problem. Given two known projections of the same point $\mathbf{Q}$ onto key images A and B, find its projection in a current view X. The decomposed solution of that problem is given as follows: 1) Image correspondences are used to recover the two-view geometry (A, B); 2) the two 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.

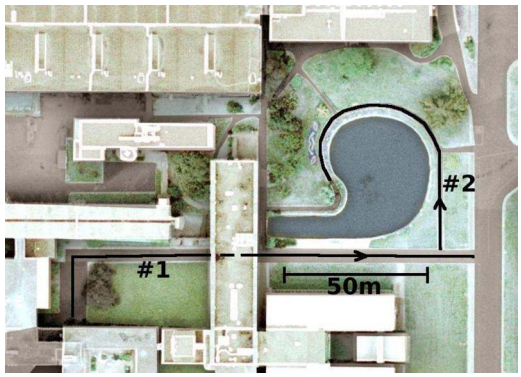

Fig. 6. Paths for experiments 1 and 2.

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 offline 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 offline, 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

	Learning						Navigation					
exp.	raw images	ref. images	proc. time [s]	fps	path [m]	meters per ref. image	images	time [s]	fps	v forw. [cm/s]	v turning [cm/s]	human interv.
1	958	77	257	3.7	158	2	934	226	4.1	90	50	0
2	862	51	208	4.1	96	1.9	532	262	2	50	30	1
3	2454	97	592	4.1	304	3.1	2272	516	4.4	80	30	0
4	1425	48	237	6	119	2.5	1812	385	4.7	50	40	0
5	785	32	167	4.7	100	3.1	280	78	3.6	180	40	0
6	371	22	102	3.6	50	2.4	406	94	4.3	80	40	0

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 difficult 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 influenced 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. 8. Every second frame of a sequence from experiment 1 demonstrates robust feature (light colored crosses) tracking resumption after occlusion by a passing car.

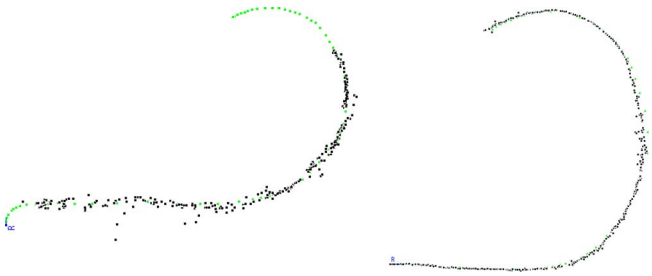

Fig. 9. Navigation results in experiment 2 (left) using a map created four months earlier. As shown in the proportion of black and lighter colored dots, CyCab completed about 80% of the path. A successfully repeated experiment (right) with a new map suggests the previous experiment failed near the end because of large changes in the appearance of the environment.

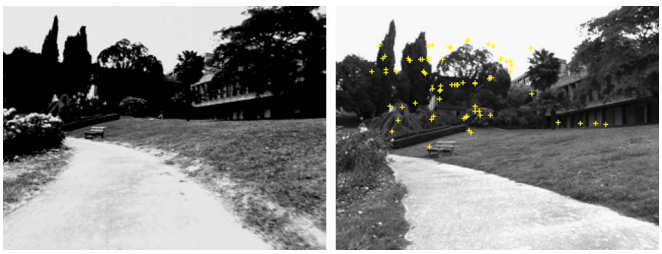

Fig. 10. Large difference in illumination and vegetation between a four-month-old reference image (left) and a current image used during navigation in experiment 2.

Fig. 11. Difference between the reference image (left) and current image (right) in experiment 2, which the vision system could no longer handle. Notice the missing flowers in the flowerbed.

points in Fig. 14, e.g., at the right bottom part of the path. In this case, the 3-D pose error resulted in an early switching of a few reference images during turning and, subsequently, following the learned path with a 1-m lateral error for a short section of the path. Other than this case, CyCab performed excellently, even when the sun was shining into its camera, as shown in Fig. 15. With seamless motion over the first and final reference frames, CyCab demonstrated that the framework does not require global consistency in the 3-D reconstruction.

Fig. 12. CyCab autonomously drives on the narrow path in experiment 2.

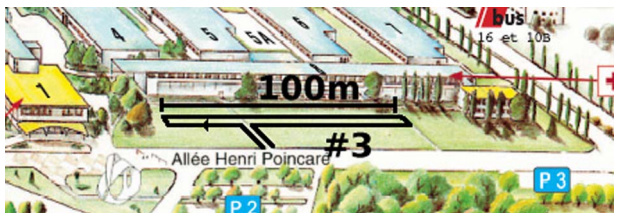

Fig. 13. Path for experiment 3.

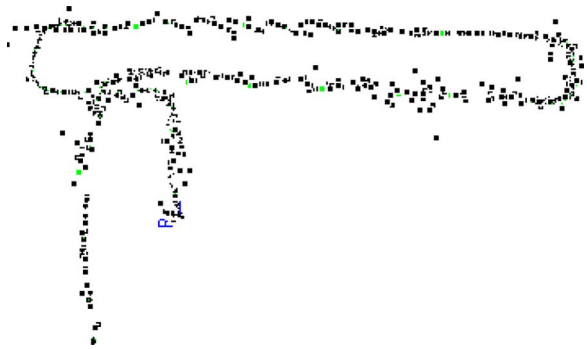

Fig. 14. Larger noise in the reconstructed robot poses, where all features are far away in experiment 3.

Fig. 15. Sun shines into the camera in the reference image (left) but not in the current image (right) during navigation in experiment 3.

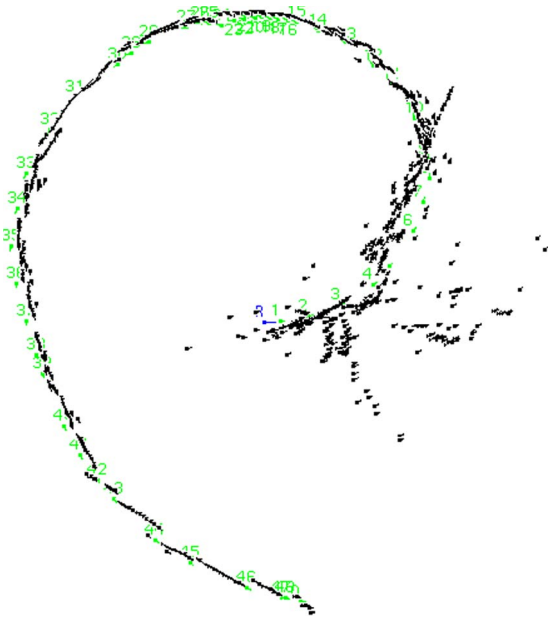

Fig. 16. Navigation results in the loop closing experiment (experiment 4).

Fig. 17. Sun shines into the camera in the reference image (left) of 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.

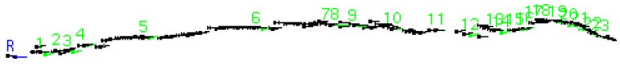

Fig. 20. Navigation results in experiment 6.

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.

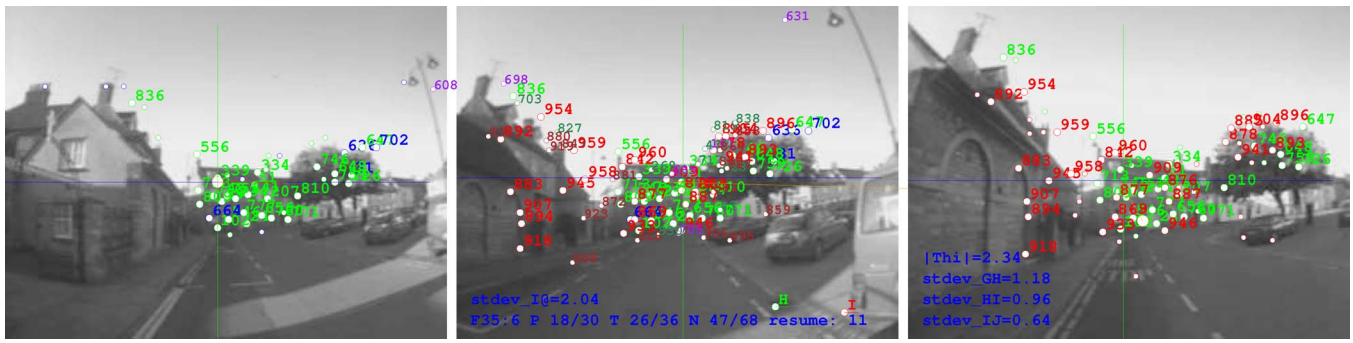

Fig. 21. View difference example in experiment 7. (Left and right) Previous and next image from the map captured on the left side of the road, respectively. (Middle) Current image captured on the right side of the road. Notice the large separation between the reference images and the large lateral displacement of the current image from the reference images.

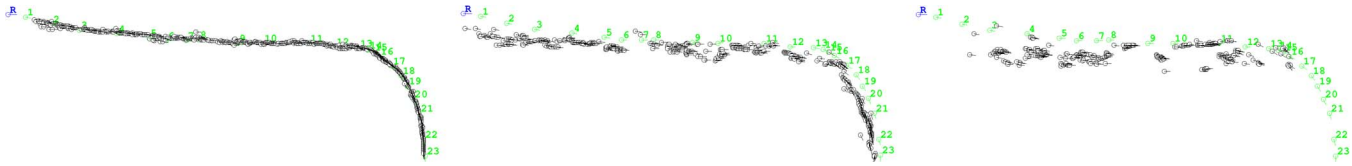

Fig. 22. Offline localization result in experiment 7 while driving on the reference path (left), in the middle (center), and on the right (right) of the path. Notice the increase of noise in the reconstructed robot poses with increasing lateral deviation. The point of getting lost in the rightmost track coincided with performing a right angle turn while being close to the tracked points.

G. Experiment 7: Robustness to Lateral Deviation

A navigation system based on vision should also handle situations where the autonomous vehicle is required to deviate from the reference path to avoid an obstacle. Because obstacle detection and avoidance is out of the scope of this paper, in this experiment only the maximum possible lateral deviation from the reference path is investigated. Unlike in the previous experiments, a firewire color webcam, the Unibrain Fire-I, was mounted on the top of a 1995 right-hand-drive Renault Clio. During the experiment, images were logged at 30 Hz, whereas the vehicle was traveling at approximately 5 m/s.

The experiment was conducted on a single-direction double-lane L-shaped road of a small town at 7:30 on a sunny Saturday morning in June. The time of the experiment was chosen to minimize the effect of moving objects, because the goal was to test the sensitivity of localization to lateral deviation. The place of the experiment was chosen to emulate an unfavorable scenario where the houses are close to the road (see Fig. 21). Such situations where the lateral deviation is large compared to the distance from the scene are challenging, because the tracked points undergo a large amount of appearance and position change. The distance between the camera and the nearest house on the left was often only 2 m during the mapping of the approximately 100-m-long path. The right side of the road was occupied by parked cars. During mapping, one car drove past. Data were gathered for localization in the subsequent runs, at estimated lateral deviations of 0 m (left side), 2.5 m (middle), and 5 m (right side) from the reference path.

Offline localization during the 0-m deviation and the 2.5-m deviation was successful; however, the initial localization with wide-baseline matching at the 5-m lateral deviation failed for the first few reference images. After a later successful initial localization, the framework kept the camera localized until the pose tracking failed right after the turn. We can observe

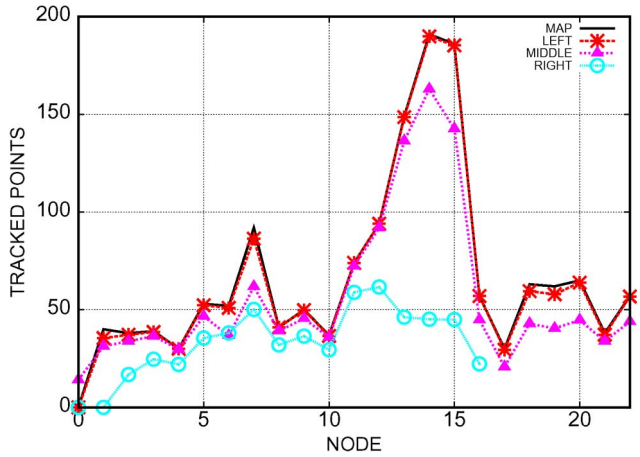

Fig. 23. Number of points in the map and the average number of tracked points for each node in experiment 7. With increasing lateral deviation, the average number of tracked points decreases.

the increase of jitter with increasing lateral deviation in the localization results (see Fig. 22). A decline in the number of tracked points with increasing lateral deviation is shown in Fig. 23. It is also visible in the figure that the number of points in the map increased as the car approached the turn and decreased as it came out of the turn.

The results indicate that, with increasing lateral deviation, the localization accuracy and the number of tracked points decrease. The limit of the vision system for lateral deviation in the environment tested lies between 2.5 and 5 m.

V. Discussion and Lessons Learned

A. Scalability and Performance

Statistics from experiments 1–6 are presented in Table I.

By performing simple image-based visual servoing instead of position-based control of the robot, we can have several advantages. Because there is no need for an accurate robot pose during navigation, we can allow a larger 3-D reconstruction error during mapping. Because of this case, there is no need to perform a computationally costly global bundle adjustment, and mapping can be done online. During the experiments, it was noticed that, after the baseline between reference images has increased beyond a certain distance, the 3-D reconstruction error also increased. Therefore, if a larger 3-D reconstruction error is allowed, we can have larger distances between reference images, and the memory requirement for storing the map is reduced. This condition is shown, for example, in experiment 3, where the average distance between reference images was 3.1 m. Sparse reference images improve not only scalability but performance as well, because the overhead associated with the loading of reference images and their switching is reduced.

The framework enables the learning and navigation of long paths, because the total memory and computational requirements for creating a map linearly grow with the length of the path. The computational cost during each navigation step is approximately constant. As for the memory requirements, when calculating with 3.1 m between reference images, a 1-km-long path can be represented using 25 MB of storage if we use 320×240 uncompressed images and neglect the stored feature point coordinates. Because we can store the reference images on a hard drive, a 1-TB drive may store an approximately 40 000-km worth of path.

There is a relationship between camera field of view and distance between reference images. In experiments not described in this paper due to the lack of space, we noticed that, when using only the center half of the images or when using a Logitech Quickcam Pro 4000 camera, the average distance between reference images increased up to 12 m. The detailed study of the effects of field of view constitutes future work.

B. Vision Techniques

The contrast compensation implemented in the tracker can handle large affine changes of illumination between the reference and current images, which was crucial, for example, during experiment 2 (see Fig. 10). Although the tracker was fairly resilient to illumination changes, the same is not true of the wide-baseline matching. Problems occurred from time to time when buildings that hold the majority of the features reflected the sunlight directly into the camera. The matching of overexposed features with well-exposed features using SIFT descriptors often failed, even when the tracker can track them. Because initial localization or relocalization is done on a stationary robot, the use of exposure bracketing (see [30] for stereo vision) and the utilization of points resulting from all images in the matching process may alleviate this problem.

The use of 3-D information enables us to restart the tracking of features just becoming visible after occlusion, as shown in Fig. 8. This property is important in dynamic environments. In addition, having 3-D information enables the system to check the consistency of the tracked features. Tracked points that "jump" from the background onto a moving object in the foreground are discarded. Although having 3-D information may not be necessary for path following, as stated in the Introduction, it may extend the area of applicability of an outdoor path-following system.

Because only features that were reliably tracked are kept between two possibly distant reference images, the feature selection for 3-D geometry estimation did not pose a significant problem. We may intuitively think that maps built with an extended Kalman filter (EKF)-based monocular SLAM implementation are more accurate due to a larger amount of information integrated into the maps. However, the superiority of several EKF-based monocular SLAM implementations is not very clear, because unstable features or features located on slowly moving objects (e.g., clouds) may be tracked and incorporated into the map before being discarded. This incorporation of bad features may gradually compromise the integrity of the map. In contrast, errors in the 3-D reconstructions always stay local in our framework and do not affect other nodes of the map. A similar effect can also be achieved with local SLAM maps. Local SLAM maps may also alleviate the effects of linearization errors in EKF implementations.

The use of normalized image coordinates in the vision system together with tracked image patch scale estimation does not preclude performing mapping with one camera and navigating with a different but reasonably similar camera. Such capacity enables mapping by one vehicle and sharing the map by several vehicles.

C. Limitations

As shown in experiment 7, the framework has handled lateral deviations in excess of 2.5 m, even when used with a noisy camera with a high radial distortion. This condition indicates that the framework may enable obstacle avoidance, as long the scene is not totally covered up by the obstacle.

In the current implementation, the framework relies on a 3-D pose to switch reference images. In cases where the 3-D pose is less accurately recovered, it can happen that a reference image switch is not performed or is performed in the wrong direction. Such behavior occasionally happens when most of the observed points are located on a plane or on a tree. A wrong reference image switch more likely caused problems in turns where not turning in the right direction quickly reduced the number of visible feature points. With fewer points, the reconstructed geometry is often less accurate, which further worsens reference image switching and reduces the accuracy of points projected from the map into the image. When there is no replacement for lost feature points, the number of feature points declines. To address the issue of reference image switches, we plan to investigate a reference-image-switching strategy based on the more stable image information. Pose estimation based on homography for planar scenes is also an option.

One further limitation is that of illumination. Extreme illumination changes, e.g., the sun shining into the camera during mapping but not during navigation or the lack of light, may impair the performance of the framework, particularly the performance of the matcher.

The navigation at night remains an open question. Sensitive cameras and artificial illumination may help in some cases. Encouraging results in localization have been described in [3], where image-based localization was demonstrated at night using headlights in sequences taken also at night. The localization in sequences taken during the day was not very successful.

Although the mapping and localization part of the system is 3-D, the control algorithm is 2-D. This case does not imply that the framework can only handle flat terrains. Several the tests were performed on moderately sloping terrains. The system also handled twists in the slopes.

Navigation frameworks for uncontrolled environments, as described in this paper, should detect and avoid obstacles. Because this case is not yet implemented in the framework, it constitutes part of the future work.

The choice of the speed of the robot should depend on the following factors: 1) exposure time as it influences motion blur; 2) frame rate and distance to features (as they influence how much features move between frames); and 3) safety considerations.

When considering navigation based on maps created a long time ago, we can expect vegetation to significantly change. This condition restricts the long-term application of such a system to places with a slower rate of change, e.g., to urban areas where buildings are visible.[11] It seems reasonable to assume that the appearance of buildings slowly changes. However, old buildings are rebuilt, and new buildings are erected all the time. A wide-field-of-view camera or a panoramic camera may help capture parts of the scene that have not changed. To even more increase the robustness of the system, a mechanism should be added to the framework, through which new map points can be added to the map during navigation. Even then, snow may sufficiently change the facades of buildings to stop the system from working, which may restrict the use of the framework to climates without snow.

Experts may easily assess environments for vision-system-related risks of failures during navigation. However, commercial systems would benefit of such output as part of the mapping process.

D. Applications

Frameworks such as our approach may be used one day on arbitrary systems which have to move on a previously completed track. Such systems include people carriers, street-cleaning robots, and robots that transport goods between buildings of a factory. The framework is not limited to systems with wheeled or tracked locomotion. Because the only sensing modality is a single camera (no odometry), coupled with full 3-D geometry estimation, we could likely use the framework on hovercrafts, blimps, helicopters, and airplanes. However, for aircraft, the affine tracking of the tracker should be enabled (for the experiments in this paper, this property was disabled to obtain more accurate results) to handle rotated image patches, and the control algorithm should be changed to handle 3-D motion. We could also envisage the use of such a system on autonomous boats in places such as canals in some cities where several stationary features are visible.

Because it is reasonable to expect that the framework can handle teaching while moving forward and executing the path while moving backward, it could be used on transportation devices that drive themselves back to their base after use.

In safety-critical applications, the addition of the inertial measurement unit (IMU), Global Positioning System (GPS), odometry, or a motion model (predicting the motion of the vehicle) may be considered to ensure that eventual vision system failures are appropriately handled.

[11] In a wider context, we could also consider space objects with slowly changing landscapes such as the moon.

VI. Conclusion

An experimental evaluation of a framework for visual path following in outdoor urban environments using only monocular vision has been presented in this paper. In the framework, no sensor other than a camera was used. The path to follow was represented as a series of images with overlapping landmarks. It was shown that the use of local 3-D information, contrast compensation, and image-based visual servoing can lead to a system that can navigate in diverse outdoor environments with reasonable changes in lighting conditions and moving objects. Online learning was also demonstrated.

Because the framework does not rely on odometry, the range of applications may also include boats that navigate on urban canals or aircraft.

Acknowledgment

The authors would like to thank F. Spindler, A. Cherubini, H. Tran, and F. Servant for their help during the experiments, M. Pupilli for lending the equipment, enabling the conduction of experiment 7, and for his comments on this paper, and the reviewers for their comments.

References

[1] S. Baker and I. Matthews, "Lucas–Kanade 20 years on: A unifying framework," *Int. J. Comput. Vis.*, vol. 56, no. 3, pp. 221–255, Mar. 2004.
[2] O. Booij, Z. Terwijn, Z. Zivkovic, and B. Krosse, "Navigation using an appearance-based topological map," in *Proc. ICRA*, 2007, pp. 3927–3932.
[3] D. Bradley, R. Patel, N. Vandapel, and S. Thayer, "Real-time image-based topological localization in large outdoor environments," in *Proc. IROS*, 2005, pp. 3670–3677.
[4] Z. Chen and S. T. Birchfield, "Qualitative vision-based mobile robot navigation," in *Proc. ICRA*, Orlando, FL, 2006, pp. 2686–2692.
[5] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *IEEE Trans. Robot.*, vol. 25, no. 3, pp. 749–754, Jun. 2009.
[6] Y. Cheng, M. W. Maimone, and L. Matthies, "Visual odometry on the Mars exploration rovers—A tool to ensure accurate driving and science imaging," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 54–62, Jun. 2006.
[7] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tardos, "Mapping large loops with a single hand-held camera," in *Proc. RSS*, 2007, pp. 1–8.
[8] J. Courbon, Y. Mezouar, and P. Martinet, "Indoor navigation of a nonholonomic mobile robot using a visual memory," *Auton. Robot*, vol. 25, no. 3, pp. 253–266, Oct. 2008.
[9] J. Courbon, Y. Mezouar, and P. Martinet, "Autonomous navigation of vehicles from a visual memory using a generic camera model," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 392–402, Sep. 2009.

[10] M. Cummins and P. Newman, "Highly scalable appearance-only SLAM-FAB-MAP 2.0," in *Proc. RSS*, 2009, pp. 1–8.
[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single-camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 1052–1067, Jun. 2007.
[12] A. Diosi, A. Remazeilles, S. Segvic, and F. Chaumette, "Outdoor visual path following experiments," in *Proc. IROS*, 2007, pp. 4265–4270.
[13] D. Fontanelli, A. Danesi, F. A. W. Belo, P. Slaris, and A. Bicchi, "Visual servoing in the large," *Int. J. Robot. Res.*, vol. 28, no. 6, pp. 802–814, Jun. 2009.
[14] U. Frese and L. Schroder, "Closing a million-landmarks loop," in *Proc. IROS*, Beijing, China, 2006, pp. 5032–5039.
[15] T. Goedeme, T. Tuytelaars, G. Vanacker, M. Nuttin, and L. Van Gool, "Feature-based omnidirectional sparse visual path following," in *Proc. IROS*, Edmonton, AB, Canada, Aug. 2005, pp. 1806–1811.
[16] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[17] H. Jin, P. Favaro, and S. Soatto, "Real-time feature tracking and outlier rejection with changes in illumination," in *Proc. ICCV*, 2001, vol. 1, pp. 684–689.
[18] K. Konolige, M. Agrawal, and S. Sola, "Large-scale visual odometry for rough terrain," in *Proc. Int. Symp. Res. Robot.*, 2007, pp. 1–12.
[19] T. Lemaire, C. Berger, I. Jung, and S. Lacroix, "Vision-based SLAM: Stereo and monocular approaches," in *Proc. IJCV/IJRR Special Joint Issue*, 2007, pp. 343–364.
[20] M. I. A. Lourakis and A. A. Argyros, "Fast trifocal tensor estimation using virtual parallax," in *Proc. ICIP*, Genoa, Italy, Jun. 2005, pp. 169–172.
[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[22] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. New York: Springer-Verlag, 2004.
[23] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," in *Proc. Brit. Mach. Vis. Conf.*, 2002, vol. 1, pp. 384–393.
[24] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *Proc. ICRA*, Minneapolis, MN, Apr. 1996, pp. 83–88.
[25] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.
[26] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.
[27] M. Milford and G. Wyeth, "Mapping a suburb with a single camera using a biologically inspired SLAM system," *IEEE Trans. Robot.—Special Issue Visual SLAM*, vol. 24, no. 5, pp. 1038–1053, Oct. 2008.
[28] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Washington, DC, 2004, pp. 652–659.
[29] D. Nister, "An efficient solution to the five-point relative-pose problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 756–770, Jun. 2004.
[30] N. Nourani-Vatani, J. Roberts, and M. Srinivasan, "Practical visual odometry for carlike vehicles," in *Proc. ICRA*, 2009, pp. 3551–3557.
[31] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, no. 2, pp. 19–27, Apr. 1996.
[32] A. Remazeilles and F. Chaumette, "Image-based robot navigation from an image memory," *Robot. Auton. Syst.*, vol. 55, no. 4, pp. 345–356, Apr. 2007.
[33] E. Royer, J. Bom, M. Dhome, B. Thuillot, M. Lhuillier, and F. Marmoiton, "Outdoor autonomous navigation using monocular vision," in *Proc. IROS*, Edmonton, AB, Canada, Aug. 2005, pp. 1253–1258.
[34] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Proc. ICRA*, 2009, pp. 4293–4299.
[35] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette, "A mapping and localization framework for scalable appearance-based navigation," *Comput. Vis. Image Understand.*, vol. 113, no. 2, pp. 172–187, Feb. 2009.
[36] J. Shi and C. Tomasi, "Good features to track," in *Proc. CVPR*, 1994, pp. 593–600.
[37] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley, the robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006.
[38] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Comput. Vis. Image Understand.*, vol. 78, no. 1, pp. 138–156, Apr. 2000.
[39] T. Tuytelaars and K. Mikolajczyk, *Local Invariant Feature Detectors: A Survey*. Hanover, MA: Now, 2008.
[40] A. Zhang and K. Kleeman, "Robust appearance-based visual route following in large scale outdoor experiments," in *Proc. ACRA*, 2007.
[41] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.
[42] Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artif. Intell.*, vol. 78, no. 1/2, pp. 87–119, Oct. 1995.

Albert Diosi received the B.S and M.S. degrees in control systems engineering from the Slovak University of Technology, Bratislava, Slovakia, in 1999 and 2001, respectively, and the Ph.D. degree in electrical engineering from Monash University, Melbourne, VIC, Australia, in 2006.

In 2006 and 2007, for 10 months, he held a Postdoctoral Research post with the INRIA, Rennes-Bretagne Atlantique–IRISA, Campus Beaulieu, Rennes, France. He is currently a Robotic Systems Engineer.

Siniša Šegvić received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from the University of Zagreb, Zagreb, Croatia.

In 2005 and 2006, he spent one year as a Postdoctoral Researcher with the INRIA, Rennes-Bretagne Atlantique–IRISA, Campus Beaulieu, Rennes, France. In 2007, he completed another one-year postdoctoral position with Graz University of Technology, Graz, Austria. He is currently an Assistant Professor with the Department of Electronics, Microelectronics, Computer, and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia. His research and professional interests include various applications of computer vision, particularly in robot navigation and the analysis of video acquired from a moving vehicle.

Anthony Remazeilles received the B.Eng. degree in computer science from the INRIA, Rennes, France, in 2001, the M.S. degree in artificial intelligence and computer vision from the University of Rennes I, and the Ph.D. degree from the Institut National des Sciences Appliquées de Rennes (INSA–Rennes) in 2004.

From 2001 to 2006, he was with INRIA Rennes, under the supervision of F. Chaumette (Lagadic Group), and P. Gros (TeXMeX Group). He was then a Teaching Assistant with the Computer Science Department, INSA Rennes. In 2006 and 2007, he held a Postdoctoral post with LIST/CEA, Fontenay aux Roses, France, on robotic assistance for injured people. Since 2008, he has been with the Health Unit, Fatronik-Tecnalia, Donostia-San Sebastian, Spain.

François Chaumette received the B.S. degree from the École Nationale Supérieure de Mécanique, Nantes, France, in 1987 and the Ph.D. degree in computer science from the University of Rennes, Rennes, France, in 1990.

Since 1990, he has been with the INRIA, Rennes, where he is currently the Directeur de Recherches and the Head of the Lagadic Group (http://www.irisa.fr/lagadic). His research interests include robotics and computer vision, particularly visual servoing and active perception.