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 3D geometries are reconstructed between the neighboring reference images in order to achieve fast feature prediction. This allows recovery from tracking failures. During navigation the robot is controlled using image-based visual servoing. The focus of the paper is on the results from a number of experiments conducted in different environments, lighting conditions and seasons. The experiments with a robot-car show that the framework is robust against moving objects and moderate illumination changes. It is also shown that the system is capable of on-line path learning.**

***Index Terms*—visual servoing, mapping, localization, visual memory, path following**

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 over 200km on a challenging desert route [37]. Systems based on visual odometry, stereo vision and inertial measurement unit based systems have achieved significantly high precision, for example just 9m error after a 9km 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 1000km 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 (go forward, turn left, turn right) as in [24]. There a mobile robot navigated through indoor corridors by applying template matching to current and reference images and by using the stored actions. However, storing the robot actions is not necessary for navigation. In [33] a robot navigates a 127m 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 3D reconstruction of the camera poses is performed of the frequently (approximately every 70cm) saved reference images. In the 3D reconstruction process applied to feature points of the reference images, global bundle adjustment is used which results in a long (1 hour) learning phase unsuitable for on-line 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 50cm/s velocity.

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

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

Similarly to [15], in the work described in [2] indoor navigation is performed using omnidirectional vision. However, the epipolar geometry is only calculated to validate SIFT descriptor [21] matches between current and reference image keypoints and to calculate the required heading direction. In a thorough experimental evaluation they demonstrated that their system was capable of planning and executing motions in pure appearance based topological maps while 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 754m 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

Manuscript received:

A. Diosi was with INRIA Rennes-Bretagne Atlantique and IRISA when performing this work, but now works in the industry. Email: albert.diosi@gmail.com

S. Šegvić was with INRIA Rennes-Bretagne Atlantique and IRISA when performing this work, but now is with the Dept. of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing, Unska 3, 10000 Zagreb, Croatia. Email: sinisa.segvic@fer.hr

A. Remazeilles was with INRIA Rennes when performing this work, but now works in the Health Unit of Fatronik-Tecnalia, San Sebastian. Email: aremazeilles@fatronik.com

F. Chaumette is with INRIA Rennes-Bretagne Atlantique and IRISA, Campus Beaulieu, 35042 Rennes cedex. Email: Francois.Chaumette@irisa.fr

The presented work received the financial support of the French national projects Predit MobiVIP and CityVIP.

to images covering 185 degree field of view. A step towards commercialisation is presented in [8] as a similar framework is applied to an indoor robot with a potentially inexpensive processing unit entailing an ARM9 microprocessor and an 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 3D 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 as well.

Convincing experimental results for outdoor visual path following using omnidirectional vision and odometry are presented in [40]. In the simple and effective approach, one dimensional 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 3D representation of the environment can also be done using monocular SLAM [11]. For example in [19] a robot mapped a 100m path outdoor 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 million 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 the field of 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]. The current paper on the other hand is oriented towards applying the same general idea for controlling real autonomous cars. An account of the employed vision system has been previously presented in [35]. In this paper the presented experiments describe the behaviour of the system in many different outdoor environments, and thus provide a qualitative and quantitative insight into the feasible range of navigation performance. Additionally, 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 is capable to 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 carried out mostly on roads using an autonomous electric vehicle capable of carrying two passengers.

The presented framework is similar to [15] in that only local 3D reconstruction is used and that 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 one, 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.

The paper is organized as follows: a description of the framework is given in Section II. More details of 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 the paper ends with conclusions.

II. VISUAL NAVIGATION

This section briefly describes the implemented visual navigation framework. The teaching of the robot (mapping) is described first, followed by the description of the navigation process consisting of localization and robot control.

A. Mapping

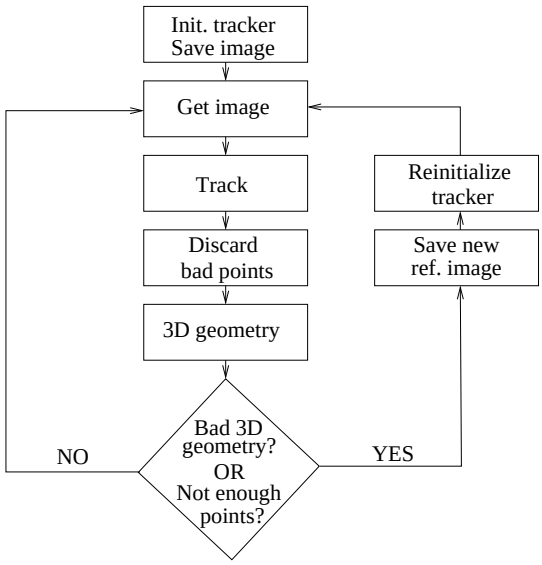

Fig. 1. The steps involved in building a representation of a path from a sequence of images, i.e. mapping.

Learning a path (i.e. mapping) starts with the manual driving of the robot on a reference path while processing (or storing for off-line mapping) the images from the robot's camera. From 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

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

[2]Videos showing 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]

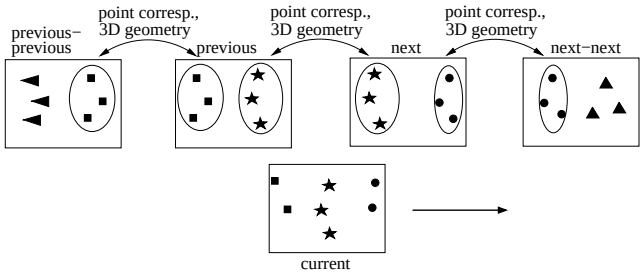

Fig. 2. The map consists of reference images, point correspondences, 2D and 3D information. During navigation, the point features from the map are projected into the current image and tracked.

as the first reference image. A version of the KLT[3] tracker was modified as proposed in [17] in order to improve performance in outdoor sequences acquired from a moving car. In the tracker, 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 3D geometry between the previous reference and the current image. In the 3D geometry estimation, the essential matrix is recovered using the calibrated 5 point algorithm[4] [29] used in the MLESAC [38] random sampling framework. The inlier points are then used in a final 3D geometry calculation using the 8-point algorithm [38]. If the 3D 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 2D and 3D 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 ones 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 image of the teaching sequence, wide-baseline matching is utilized as described in section III.

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

B. Localization

The localization process during navigation is depicted in Fig. 3. The navigation process 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. The wide-baseline matching is done using a correlation based approach [42] and by matching SIFT descriptors [21] applied to DoG [21], multi-scale Harris [25] and MSER [23] keypoints. The estimation of the camera pose using the matched points enables 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, previous reference and next reference image (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 one to restart the tracking of points currently not tracked and to stop the tracking of points which 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 next-next reference image (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. Similarly to forward transitions, a topological transition is performed backwards if the current image precedes the previous reference image.

To achieve seamless switching between nodes, points from next-next, previous and previous-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 for example due to a total obstruction of the camera's field of view. In such case the robot stops and automatic re-initialization is carried out by matching with the nearest reference images.

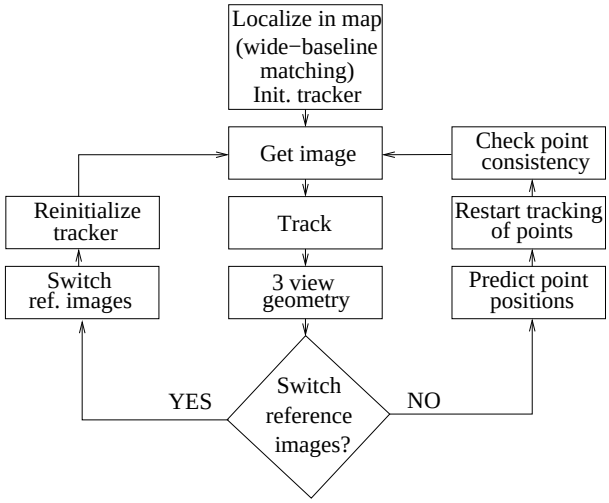

Fig. 3. Visual localization during navigation.

C. Motion Control

In the motion control scheme the robot is not required to accurately reach each reference image of the path, nor to follow accurately the learned path since this may not be useful during navigation. In practice, the exact motion of the robot should be controlled by an obstacle avoidance module that will constitute future work. Therefore a simple control algorithm was implemented where the difference in the x-coordinates (assuming the forward facing camera's horizontal

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

[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]

axis is orthogonal with the axis of robot rotation) of the centroid of features in the current (x_c) and next reference image (x_n) are fed back into the motion controller of the robot as the steering angle Φ:

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

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

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

where b is the feed forward 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 next-next image.

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

III. Vision techniques and algorithms

This section describes the main details about the employed vision techniques and algorithms. A key ability in vision-based path following is to correctly locate mapped features in images acquired during navigation. We therefore first introduce two basic approaches for establishing point correspondences between images: wide-baseline matching in III-A and tracking in 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 3D reconstructions (besides being several times faster to obtain). Consequently, we employ wide-baseline matching only for obtaining initial localization at the beginning of the navigation session (cf. the top box of the flowchart in Fig. 3). Other localization steps and the whole mapping stage employ tracking, as detailed in II-B and II-A.

The proposed localization subsystem often needs to (re-)-start the tracking of features which for various reasons were not tracked in the previous frame (cf. 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 ones. Additionally, 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 for predicting the locations of mapped features which are currently not tracked. This procedure is described in 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 (cf. Fig. 1), while the actual prediction is employed during navigation (cf. the box "Predict point positions" in Fig. 3).

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 so that we especially require robustness against 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]. Usually the detected keypoints are locally distinctive with respect to position, scale and rotation, while some approaches even address affine invariance [25]. The obtained descriptors are exhaustively compared against the descriptors from the other image, typically with respect to 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 three keypoint detectors: the maxima of the difference of Gaussians [21], multi-scale Harris corners [25], and maximally stable extremal regions [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. The two main point feature tracking approaches are iterative first-order differential approximation [36], [17], and exhaustive matching of light-weight point features [28], [22] such as Harris corners. We believe that the former approach is better suited to appearance-based navigation since it tends to be less susceptible to association errors and provides more accurate point tracks.

In order to tolerate significant inter-frame displacements, the features are first tracked between the previous and the current image across a multi-level resolution pyramid[5]. Then, in order 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 2D affine transform. An extended warp which additionally compensates for affine photometric deformations of the grey level value in the image has been proposed in [17].

In the rest of the subsection, we first provide a formulation of the general point feature tracker [36], [17], [1] in III-B1,

[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.

and then in 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 III-B3, while in 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:

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

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 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 appearing in (3) can now be represented as:

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

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:

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

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 is easy to perform:

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

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):

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

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, or when the new feature position falls outside the image bounds, or when the determinant $|\mathbf{g}\mathbf{g}^\top|$ becomes too small.

2) Differential tracker with isotropic scaling and contrast compensation: In order to mitigate the danger that a physically unrelated image patch might be well transformed towards the reference, a trade-off between modelling power and tracking security should be carefully chosen. For our application, a good balance is obtained by a 5-dimensional warp consisting of a 2-dimensional 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:

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

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

$$\frac{\partial U}{\partial \mathbf{r}}(I(T(\mathbf{x},\mathbf{q})),\mathbf{r}) = \begin{bmatrix} I_T & 1 \end{bmatrix} \,, \tag{10}$$

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:

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

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, (11) and (10), 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 Stan Birchfield at Clemson university[7]. We performed several modifications to the original code, but the most important among them are contrast compensation warp extensions such as the one described in III-B2 and evaluated in III-C. Additionally, we provided code 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 which 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 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.

[7]URL: http://www.ces.clemson.edu/~stb/klt/ [Accessed: February 22, 2011]
[8]We employed GNU profiler gprof.

C. Performance evaluation of the correspondence approaches

Evaluating the correspondence performance for real scenes is tricky since ground-truth correspondences typically can not be recovered in experiments with real 3D 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, since the existing correspondences are employed to predict locations of previously unseen features. Bad predictions can be exceptionally troublesome, since 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. The four evaluated correspondence approaches are:

- 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)

The experiment is designed as follows. For each of the 23 key-images of the sequence referenced in 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 $(i,i+1)$, $(i,i+2)$ etc. For tracking we initialize the tracker at index i, and then track 5 frames forward. In each case we record 5 reprojection errors, for geometries from $(i,i+1)$ to $(i,i+5)$. The results are summarized in figure 4, as means of the five reprojection errors.

The results illustrate that the correspondences obtained by tracking with contrast compensation yield overall better and significantly more stable two-view geometries than the correspondences obtained by matching (track5 vs match). The figure also shows that contrast compensation provides a significant performance gain when tracking outdoors (track5 vs track6). Finally, the figure suggests that track5 is somewhat better than track8 (track5 vs track8). Our result regarding tracking performing 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 which are currently not tracked. After an approximate feature location is 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 which 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 2D point in the current image by employing i) the known projections of the same 3D point in two other images, and ii) some additional correspondences across the three images. This problem is illustrated in Fig. 5. In order to perform the point transfer, one needs to recover the three-view geometry between the current image and two key-images from the map.

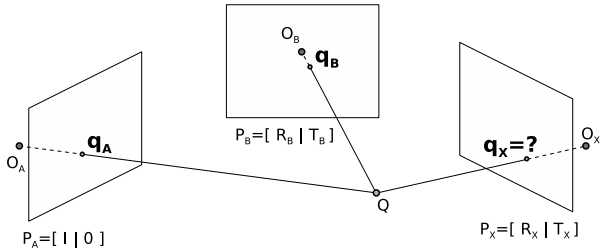

Fig. 5. The point transfer problem: given two known projections of the same point **Q** onto key-images A and B, find its projection in a current view X. The decomposed solution of that problem is: (i) image correspondences are used to recover the two-view geometry (A,B); (ii) the two known projections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

There are many ways to compute 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 many 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 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] corresponding to the case of unit focal distance [22]. Some implementation details of our solution will be described in the following paragraphs.

Each of the two geometries (A,B) and (A,X) (cf. Fig.5) is recovered independently. 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 employed implementation has been provided within the library `VW34`[10] maintained at the Imperial College in London, UK. The decomposition of the essential matrix into motion components is performed next, followed by the triangulation of 3D points [16].

[9]We employ the usual model for transforming pixels into normalized coordinates comprising of 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 described in [41].

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

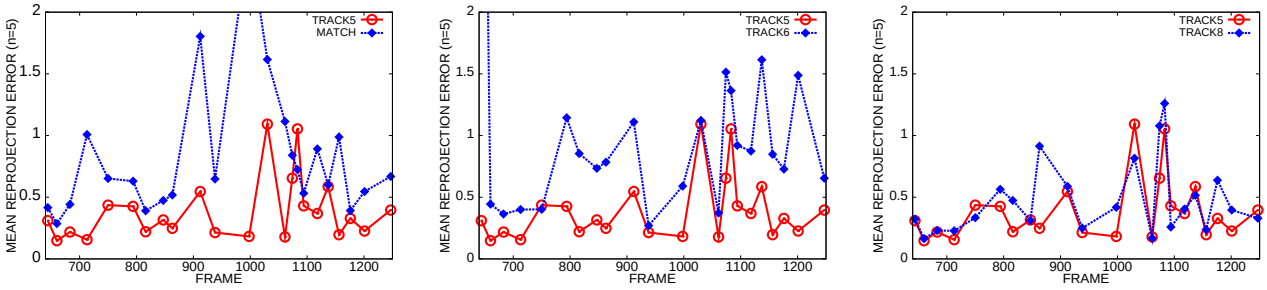

Fig. 4. Performance evaluation of four approaches for establishing correspondences between a given image and five subsequent images in the sequence. The horizontal axis holds the sequential number of the frame, while the vertical axis shows the mean reprojection error.

Consequently, the geometries (A,B) and (A,X) (cf. Fig.5) are expressed in the common frame. In the calibrated context, the adjustment involves estimation of only one parameter (scale), while in the projective context the ambiguity has 4 degrees of freedom [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.

3D coordinates of the desired point **Q** are obtained by triangulating its projections onto the two key-images A and B (cf. Fig.5). This can be performed offline, during mapping. The desired prediction $\mathbf{q_X}$ of the triangulated point **Q** to the current image X is finally obtained by simple projection.

The described 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 the presented work 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 on-line mapping and autonomous parking is also given. The results are evaluated quantitatively.

In all but the last experiment a CyCab, a French-made 4 wheel drive, 4 wheel steered intelligent vehicle designed to carry 2 passengers was used. On our CyCab all computations except the low-level control were carried out on a laptop with a 2GHz 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 65cm height. Except in experiment 3 the camera was used in auto shutter mode, with the rest of the settings constant.

During all experiments (except the last), no software parameters were changed except that of the forward and turning speed. Mapping has been performed off-line, except in experiment 6. The image resolution in the experiments was 320x240. Tracked feature patch sizes were 15x15 pixels.

A. Experiment 1: Basic experiment

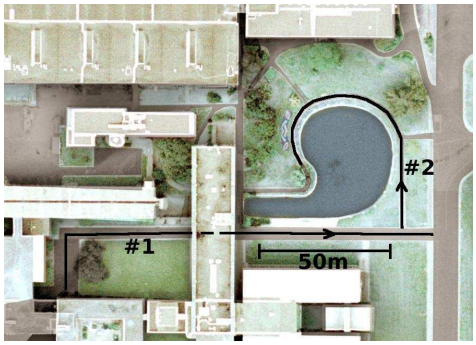

Fig. 6. Paths for experiments 1 and 2.

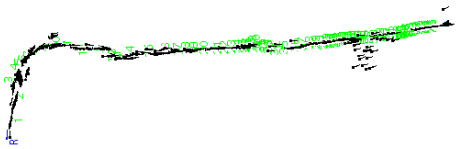

Fig. 7. Navigation results in experiment 1 shown as reconstructed robot poses (black) overlaid on 77 reconstructed reference image poses (lighter colored dots and barely visible sequence numbers). The first reference image pose is shown at the bottom left.

Experiment 1 (see Fig. 6) was conducted on an overcast day with a short time between mapping and navigation. Most views on the 158m long path contained buildings which provided stable image features. The main potential challenges in this experiment were (i) motion blur in the teaching sequence caused by fast driving for the used exposure times, (ii) driving under a building which caused a quick illumination change and (iii) people (more than 10) and cars covering up features during navigation.

In the teaching phase, 958 logged images were reduced into 77 reference images in 257s (3.7fps). While the robot was

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.

moving at 50cm/s in turns and at 90cm/s otherwise during navigation, 934 images were processed at 4.1fps on average. Statistics regarding mapping and navigation are shown in tab. I. Reconstructed robot and reference image poses shown in Fig. 7 were only used 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 re-appearing after occlusion were restarted immediately due to the feature reprojection scheme. Figure 8 contains images processed at the end of the navigation. They describe an interesting situation where a moving car progressively occludes most features. It can be seen that the tracking of re-appearing features is restarted, as there were enough good features tracked for the camera pose estimation used in point reprojection.

B. Experiment 2: Robustness to environment changes

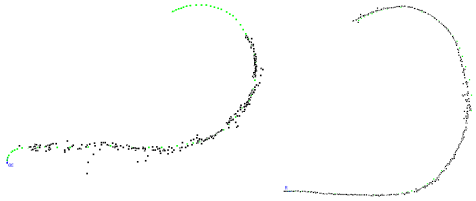

Fig. 9. Navigation results in experiment 2 (left) using a map created 4 months earlier. As can be seen from the proportion of black and lighter colored dots, CyCab completed about 80% of the path. A successfully repeated experiment (right) with a new map suggests the previous experiment did fail near the end because of large changes in the appearance of the environment.

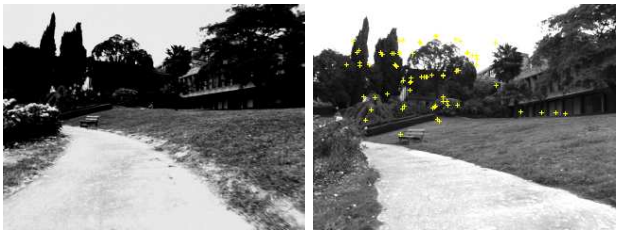

Fig. 10. Large difference in illumination and vegetation between a 4 month old reference image (left) and a current image used during navigation in experiment 2.

Fig. 11. Difference between the reference image (left) and current image (right) in experiment 2 which the vision system could not handle any more. Notice the missing flowers in the flowerbed.

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

Experiment 2 was conducted on a narrow path along a small lake (Fig. 6 and 12). Mapping was carried out in June, under the strong summer sun. Navigation took place in October, when vegetation and illumination conditions were very different (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 brushing 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 comes naturally with wide separation between reference images and the chosen control strategy. Without stopping the vision system, CyCab was moved 50cm 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 too large (see Fig. 11). Even though CyCab did not complete the whole path (see the left image in Fig. 9 where it failed), this experiment still represents a success because of the difficult conditions CyCab handled.

Shortly after CyCab got lost, we have repeated the experiment using a new map. As it can be seen in the right image of Fig. 9, CyCab completed the path without any problems and with smaller localization noise. Note that as image based visual servoing was used, localization noise had only an

indirect effect on the motion of the robot, as it only influenced 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 3D 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 tab. I) due to temporary implementation and processing platform limitations.

C. Experiment 3: Deterioration due to distant features

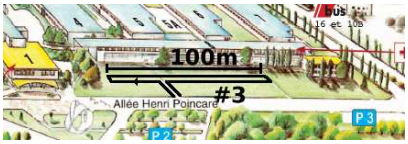

Fig. 13. The path for experiment 3.

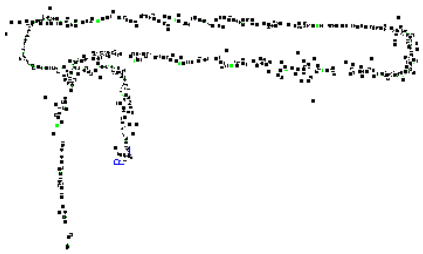

Fig. 14. Larger noise in the reconstructed robot poses where all features are far away in experiment 3.

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

In experiment 3 CyCab completed an approximately 304m track, where in some places (right side in Fig. 13), the closest features were more than 100m away. As the width of the footpath matched that 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 auto-shutter mode, the camera was used in its high dynamic range mode. The start and end positions were identical.

As one can expect, the error in the estimated pose during navigation was the largest at those places where there were no close features. Such large pose errors are represented by cluttered points in Fig. 14, for example at the right bottom part of the path. In this case the 3D pose error resulted in an early switching of a few reference images during turning, and subsequently following the learned path with a 1m lateral error for a short section of the path. Other than that, CyCab performed excellently even when the sun was shining into its camera as in Fig. 15. With seamless motion over the first and final reference frame, CyCab demonstrated that the framework does not require global consistency in the 3D reconstruction.

D. Experiment 4: Driving in a loop

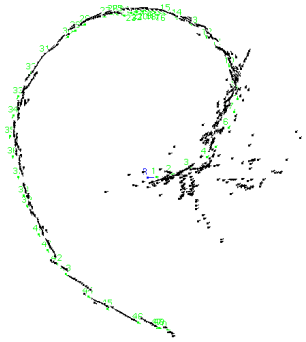

Fig. 16. Navigation results in the loop closing experiment (experiment 4).

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

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 approximately 119m circumference. The beginning and end of the loop were closed by matching the first and last image of the teaching sequence. If neighbouring nodes were connected with line segments, then the first and the last light colored dot in Fig. 16 were connected.

CyCab managed to complete 1.25 loops even though 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 (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 seen 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 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 short straight section of black dots in Fig. 16 where it happened), however the experiment was a success because it did demonstrate that CyCab can connect the beginning and end of a loop and drive through the joint.

E. Experiment 5: Robustness to speed

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

Fig. 19. Navigation results in experiment 5.

This experiment investigates how fast can CyCab navigate on a straight path. On the track shown in Fig. 18 and 19, CyCab completed a 100m straight path at a 1.8m/s (6.5km/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 on-line mapping

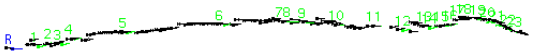

Fig. 20. Navigation results in experiment 6.

In this experiment on-line mapping (i.e. processing the images as they are grabbed) and a practical application is demonstrated. In the current state of the navigation system, i.e. without obstacle detection-avoidance, etc. 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 on-line while driving CyCab from the entrance of IRISA to the CyCab garage approximately 50m away (see Fig. 18 and 20) at about 50cm/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, while during navigation the sky was clear. CyCab even handled the transition from strong sunshine to the darkness of the garage.

G. Experiment 7: Robustness to lateral deviation

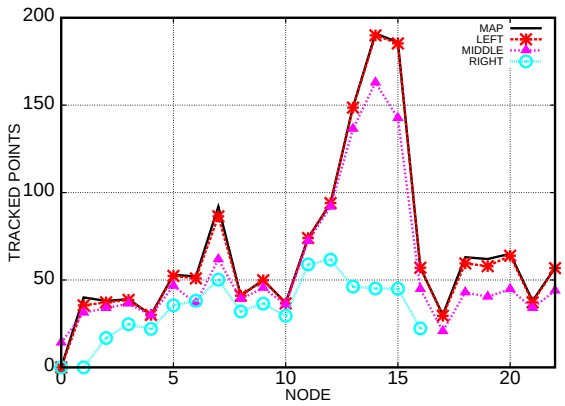

Fig. 23. The number of points in the map and the average number of tracked points for each node in experiment 7. With increasing lateral deviation the average number of tracked points decreases.

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 30Hz while the vehicle was traveling at approximately 5m/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 as 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 (Fig. 21), Such situations where the lateral deviation is large compared to the distance from the scene are challenging, as 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 just 2m during the mapping of the approximately 100m long path. The right side of the road was occupied by parked cars. During mapping, one car drove past. Data was gathered for localization in the subsequent runs, at estimated lateral deviations of 0m (left side), 2.5m (middle) and 5m (right side) from the reference path.

Off-line localization during the 0m deviation and the 2.5m deviation was successful, however the initial localization with wide-baseline matching at the 5m 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 just after the turn. One can observe

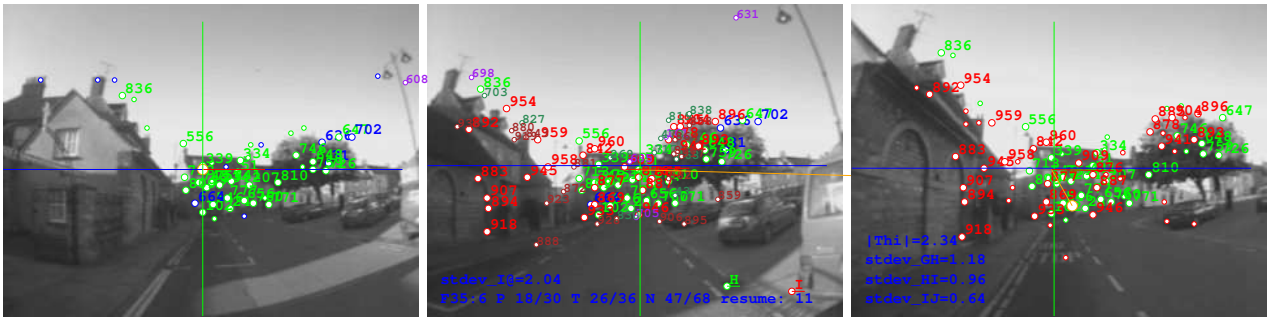

Fig. 21. View difference example in experiment 7. Left and right images are the previous and next image from the map captured on the left side of the road, while the middle image is the current one captured on the right side of the road. Notice the large separation between the reference images and the large lateral displacement of the current image from the reference images.

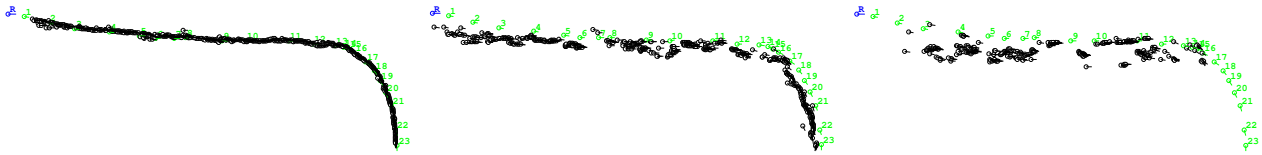

Fig. 22. Off-line localization result in experiment 7 whilst driving on the reference path (left), in the middle (center) and on the right (right) of the path. Notice the increase of noise in the reconstructed robot poses with increasing lateral deviation. The point of getting lost in the rightmost track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

the increase of jitter with increasing lateral deviation in the localization results (Fig. 22). A decline in the number of tracked points with increasing lateral deviation can be seen in Fig. 23. It is also visible from 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 decreases. The limit of the vision system for lateral deviation in the tested environment lies between 2.5 and 5m.

V. Discussion and Lessons Learned

A. Scalability and performance

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

By performing simple image-based visual servoing instead of position-based control of the robot, one can have many advantages. Because there is no need for an accurate robot pose during navigation, one can allow a larger 3D reconstruction error during mapping. Because of this, there is no need to perform a computationally costly global bundle adjustment and mapping can be done on-line. During the experiments it was noticed that, after the baseline between reference images increased beyond a certain distance, the 3D reconstruction error increased as well. Therefore if a larger 3D reconstruction error is allowed, one can have larger distances between reference images, and the memory requirement for storing the map is reduced. This can be seen for example in experiment 3 where the average distance between reference images was 3.1m. Sparse reference images improve not only scalability but performance as well as 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 grow linearly with the length of the path. The computational cost during each navigation step is approximately constant. As for the memory requirements, when calculating with 3.1m between reference images, a 1km long path can be represented using 25MB of storage if one uses 320x240 uncompressed images and neglects the stored feature point coordinates. As one can store the reference images on a hard drive, a 1TB drive may store approximately 40000km 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 12m. The detailed study of the effects of field of view constitutes future work.

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

B. Vision techniques

The implemented contrast compensation in the tracker is able to handle large affine changes of illumination between the reference and current images which was crucial for example during experiment 2 (Fig. 10). Even though the tracker was fairly resilient against illumination changes, the same is not true of the wide-baseline matching. Problems occurred from time to time when buildings holding the majority of the features reflected the sun light directly into the camera. The matching of overexposed features with well exposed ones using SIFT descriptors often failed even when the tracker was capable of tracking them. As initial localization or re-localization 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 3D information enables to restart the tracking of features just becoming visible after occlusion as can be seen in Fig. 8. This property is important in dynamic environments. Also, having 3D information enables the system to check the consistency of the tracked features. Tracked points which "jump" from the background onto a moving object in the foreground are discarded. Even though having 3D 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.

As only features that were reliably tracked are kept between two possibly distant reference images, the feature selection for 3D geometry estimation did not pose a significant problem. One may intuitively think, that maps built with an EKF based monocular SLAM implementation are more accurate due to a larger amount of information integrated into the maps. However the superiority of many EKF based monocular SLAM implementations is not so clear as unstable features or features located on slowly moving objects (for example 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 3D reconstructions always stay local in our framework, and do not affect other nodes of the map. Similar effect can be achieved with local SLAM maps as well. 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 many.

C. Limitations

As shown in experiment 7, the framework has handled lateral deviations in excess of 2.5m even when used with a noisy camera with a high radial distortion. This 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 3D pose to switch reference images. In cases where the 3D 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 less points, the reconstructed geometry is often less accurate, which further worsens reference image switching and also 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 are planning 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.

A further limitation is that of illumination. Extreme illumination changes such as the sun shining into the camera during mapping but not during navigation, or the lack of light may impair the performance of the framework, especially that 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 were not so successful.

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

Navigation frameworks for uncontrolled environments such as the one described in this paper should be able to detect and avoid obstacles. Since this is not implemented in the framework yet, it constitutes part of the future work.

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

When considering navigation based on maps created a long time ago, one can expect vegetation to change significantly. This restricts the long term application of such system to places with a slower rate of change such as to urban areas where buildings are visible.[11] It seems to be reasonable to assume that the appearance of buildings changes slowly. 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 to capture parts of the scene which have not changed. To increase the robustness of the system even more, 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 change the facades of buildings sufficiently 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 this may be used one day on arbitrary systems which have to move on a previously completed track. Such systems are for example: people carriers, street cleaning robots, robots transporting goods between buildings of a factory, etc. 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 3D geometry estimation, one could likely use the framework on hovercrafts, blimps, helicopters and airplanes. However for aircrafts, 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 be able to handle rotated image patches, and the control algorithm changed to handle 3D motion. One could also envisage the use of such system on autonomous boats in places such as canals in some cities where many stationary features are visible.

As 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 which drive themselves back to their base after use.

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

[11] In a wider context one could also consider space objects with slowly changing landscapes as the Moon.

VI. CONCLUSIONS

An experimental evaluation of a framework for visual path following in outdoor urban environments using only monocular vision was presented in this paper. In the framework no other sensor 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 3D information, contrast compensation and image-based visual servoing can lead to a system capable of navigating in diverse outdoor environments with reasonable changes in lighting conditions and moving objects. On-line learning was also demonstrated.

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

VII. ACKNOWLEDGMENTS

The help of Fabien Spindler, Andrea Cherubini, Hai Tran and Fabien Servant during experiments are gratefully acknowledged. Mark Pupilli's loan of equipment enabling the conduction of experiment 7 is appreciated together with his comments on the manuscript. We are very grateful for the reviewers' comments.

REFERENCES

[1] Simon Baker and Iain Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, March 2004.

[2] O. Booij, Z. Terwijn, Z. Zivkovic, and B. Krosse. Navigation using an appearance based topological map. In *ICRA'07*, 2007.

[3] D. Bradley, R. Patel, N. Vandapel, and S. Thayer. Real-time image-based topological localization in large outdoor environments. In *IROS'05*, 2005.

[4] Z. Chen and S. T. Birchfield. Qualitative vision-based mobile robot navigation. In *ICRA'06*, Orlando, 2006.

[5] Zhichao Chen and Stanley T. Birchfield. Qualitative vision-based path following. *Trans. Rob.*, 25(3):749–754, 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. *Robotics & Automation Magazine*, 13(2), 2006.

[7] L. A. Clemente, A. J. Davison, I. Reid, H. Neira, and J. D. Tardos. Mapping large loops with a single hand-held camera. In *RSS'07*, 2007.

[8] J. Courbon, Y. Mezouar, and P. Martinet. Indoor navigation of a non-holonomic mobile robot using a visual memory. *Autonomous Robot*, 25:253–266, 2008.

[9] J. Courbon, Y. Mezouar, and P. Martinet. Autonomous navigation of vehicles from a visual memory using a generic camera model. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):392–402, 2009.

[10] M. Cummins and P. Newman. Highly scalable appearance-only SLAM-FAB-MAP 2.0. In *RSS'09*, 2009.

[11] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007.

[12] A. Diosi, A. Remazeilles, S. Segvic, and F. Chaumette. Outdoor visual path following experiments. In *IROS'07*, 2007.

[13] D Fontanelli, A. Danesi, P. Belo, F. A. W. amd Slaris, and A. Bicchi. Visual servoing in the large. *The International Journal of Robotics Research*, 28(6):802–813, June 2009.

[14] U. Frese and L. Schroder. Closing a million-landmarks loop. In *IROS*, Beijing, 2006.

[15] T. Goedeme, T. Tuytelaars, G. Vanacker, M. Nuttin, and L. Van Gool. Feature based omnidirectional sparse visual path following. In *IROS'05*, Edmonton, Canada, August 2005.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2004.

[17] H. Jin, P. Favaro, and S. Soatto. Real-time feature tracking and outlier rejection with changes in illumination. In *ICCV*, volume 1, pages 684–689, 2001.

[18] K. Konolige, M. Agrawal, and S. Sola. Large scale visual odometry for rough terrain. In *International Symposium on Research in Robotics*, 2007.
[19] T Lemaire, C. Berger, I. Jung, and S. Lacroix. Vision-based SLAM: Stereo and monocular approaches. *IJCV/IJRR special joint issue*, 2007.
[20] M.I.A. Lourakis and A.A. Argyros. Fast trifocal tensor estimation using virtual parallax. In *ICIP*, pages 169–172, Genoa, Italy, June 2005.
[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
[22] Y. Ma, S. Soatto, J. Košecká, and S.S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, New York, USA, 2004.
[23] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, 2002.
[24] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *ICRA'96*, Minneapolis, April 1996.
[25] Krystian Mikolajczyk and Cordelia Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.
[26] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.
[27] M. Milford and G. Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics Special Issue on Visual SLAM*, 24(5), October 2008.
[28] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In IEEE *Conference on Computer Vision and Pattern Recognition*, pages 652–659, Washington, DC, 2004.
[29] D. Nister. An efficient solution to the five-point relative pose problem. *PAMI*, 26(6):756–770, June 2004.
[30] N. Nourani-Vatani, J. Roberts, and M. Srinivasan. Practical visual odometry for car-like vehicles. In *ICRA'09*, 2009.
[31] D. Pomerleau and T. Jochem. Rapidly adapting machine vision for automated vehicle steering. *IEEE Expert*, 11(2), 1996.
[32] A. Remazeilles and F. Chaumette. Image-based robot navigation from an image memory. *Robotics and Autonomous Systems*, 55(4):345–356, April 2007.
[33] E. Royer, J. Bom, M. Dhome, B Thuillot, M. Lhuillier, and F. Marmoiton. Outdoor autonomous navigation using monocular vision. In *IROS*, pages 3395–3400, Edmonton, Canada, August 2005.
[34] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC. In *ICRA'09*, 2009.
[35] S. Segvic, A. Remazeilles, A. Diosi, and F. Chaumette. A mapping and localization framework for scalable appearance-based navigation. *Computer Vision and Image Understanding*, 113(2):172–187, February 2009.
[36] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR'94*, pages 593–600, 1994.
[37] S. Thrun et al. Stanley, the robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9), 2006.
[38] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.
[39] Tinne Tuytelaars and Krystian Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., Hanover, MA, USA, 2008.
[40] A. Zhang and K. Kleeman. Robust appearance based visual route following in large scale outdoor experiments. In *ACRA'07*, 2007.
[41] Zhengyou Zhang. A flexible new technique for camera calibration. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.
[42] Zhengyou Zhang, Rachid Deriche, Olivier D. Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *AI*, 78(1-2):87–119, 1995.

Albert Diosi received his B.S and M.S. degrees in control systems engineering from the Faculty of Electrical Engineering and Computer Science of the Slovak University of Technology in Bratislava, Slovakia in 1999 resp. 2001. In 2006 he received his PhD degree in electrical engineering from Monash University in Melbourne, Australia. After a 10 month post-doc research at IRISA/INRIA in Rennes, France in 2006 and 2007, he works now in the industry as a robotic systems engineer.

Dr. Diosi's PhD thesis was awarded the Douglas Lampard Medal in 2007.

Siniša Šegvić received B.S., M.S., and Ph.D. degrees in electrical engineering and computer science at the University of Zagreb, Croatia. He spent one year as a post-doctoral researcher at IRISA/INRIA in Rennes, France in 2005 and 2006. In 2007, he completed an another one-year postdoc position at TU Graz, Austria. He is currently an assistant professor at the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia. His research and professional interests include various applications of computer vision, especially in the fields of robot navigation and analysis of video acquired from a moving vehicle.

Anthony Remazeilles graduated in 2001 with an engineering degree (Computer Science) from INSA, Rennes, France and with a M.S. of artificial intelligence and computer vision from University of Rennes I. From 2001 to 2006, he was with the INRIA Rennes, under supervision of Francois Chaumette (Lagadic group), and Patrick Gros (TeXMeX group). He received the PhD. degree in December 2004, and was then teaching assistant at the INSA Rennes (Computer Science Department). In 2006 and 2007, he worked as a post-doc at LIST/CEA, Fontenay aux Roses, France, on robotic assistance for injured people. Since 2008, he is with Fatronik-Tecnalia, Spain.

François Chaumette was graduated from École Nationale Supérieure de Mécanique, Nantes, France, in 1987. He received the Ph.D. degree in computer science from the University of Rennes, France, in 1990. Since 1990, he has been with INRIA in Rennes where he is now "Directeur de Recherches" and head of the Lagadic group (`http://www.irisa.fr/lagadic`). His research interests include robotics and computer vision, especially visual servoing and active perception.

Dr. Chaumette received the AFCET/CNRS Prize for the best French thesis in automatic control in 1991. He also received with Ezio Malis the 2002 King-Sun Fu Memorial Best IEEE Transactions on Robotics and Automation Paper Award. He has been Associate Editor of the IEEE Transactions on Robotics from 2001 to 2005 and is now in the Editorial Board of the Int. Journal of Robotics Research.