IEEE International Conference on Robotics and Automation (ICRA)
Orlando, Florida, May 2006

Qualitative Vision-Based Mobile Robot Navigation

Zhichao Chen and Stanley T. Birchfield
Electrical and Computer Engineering Department
Clemson University
Clemson, South Carolina 29634
Email: {zhichac,stb}@clemson.edu

Abstract— **We present a novel, simple algorithm for mobile robot navigation. Using a teach-replay approach, the robot is manually led along a desired path in a teaching phase, then the robot autonomously follows that path in a replay phase. The technique requires a single off-the-shelf, forward-looking camera with no calibration (including no calibration for lens distortion). Feature points are automatically detected and tracked throughout the image sequence, and the feature coordinates in the replay phase are compared with those computed previously in the teaching phase to determine the turning commands for the robot. The algorithm is entirely qualitative in nature, requiring no map of the environment, no image Jacobian, no homography, no fundamental matrix, and no assumption about a flat ground plane. Experimental results demonstrate the capability of autonomous navigation in both indoor and outdoor environments, on both flat and slanted surfaces, with dynamic occluding objects, for distances over 100 m.**

I. Introduction

The ability of a mobile robot to follow a desired trajectory is necessary for many applications. For example, a courier robot may need to deliver items from one office to another in the same building, or even in a different building; a delivery robot may need to transport parts from one machine to another in an industrial setting; a robot may need to travel along a prespecified route to give a tour of a facility; or a team of robots may need to follow the path taken earlier by a scout robot.

Traditional solutions to this problem involve building and using a map of the environment [15] or using artificial landmarks [7]. An alternate approach is that of visual servoing, in which the motions of the robot are determined by comparing a reference image with the current image taken by an on-board video camera [9], [6]. These techniques generally require a Jacobian that relates the coordinates of points in the world with their projected image coordinates, or a homography or fundamental matrix that relates coordinates between images. Approaches of this kind generally make assumptions such as a flat ground plane [5], [10], [8], [11], a calibrated camera [5], [3] (even uncalibrated systems often require some sort of calibration, e.g., for nonlinear lens distortion), or a man-made environment in which vertical straight lines [3], [8], [11] or the flat, parallel walls of a corridor [12] are present. Some of the systems require two or more cameras [3], [14] or omnidirectional cameras [2].

As Burschka and Hager [5] insightfully point out, the problem of following a predetermined path may not require a complicated approach. Intuitively, the vastly overdetermined nature of the problem (thousands of pixels in an image versus one turning command output) would seem to indicate that a simpler method might be feasible. In this paper we present a simple technique that uses a single, off-the-shelf camera attached to the front of the robot. The technique follows the teach-replay approach [5] in which the robot is manually led through the path once during a teaching phase and then follows the path autonomously during the replay phase. Without any calibration (even calibration for lens distortion), the robot is able to follow the path by making only *qualitative* comparisons between the feature coordinates computed during the teaching phase with those computed during replay. The technique does not involve Jacobians, homographies, or fundamental matrices, and it does not require an estimate of the focus of expansion. We demonstrate the technique on several indoor and outdoor experiments, with slanted surfaces and dynamic occluding objects, at distances over 100 m.

II. Qualitative mapping from feature coordinates to turning direction

Consider a mobile robot equipped with a camera whose optical axis is parallel to the heading direction of the robot. Suppose we wish to move the robot from location A to a previously encountered location B. The robot has access to a current image I_A, taken at A, and a *milestone image* I_B, taken previously at milestone B. Assuming there is enough overlap between the two images, then correspondence can be established automatically between feature points in the images using standard computer vision techniques. In this section we show that a rather simple qualitative mapping exists between the resulting coordinates and the turning commands that are necessary to guide the robot to the destination (milestone location). For ease of presentation, we will assume a pinhole camera model, but a similar analysis holds when the imaging rays are curved due to lens distortion.

A. When the robot is on the path

Suppose that the robot is at A facing B, and that I_B was taken at B from the same direction. As the robot moves from A to B, the feature points will move in the image plane away from the principal point (the intersection of the optical axis and the image plane) until they reach the locations of the corresponding feature points in I_B. This leads to an important observation:

©2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE.

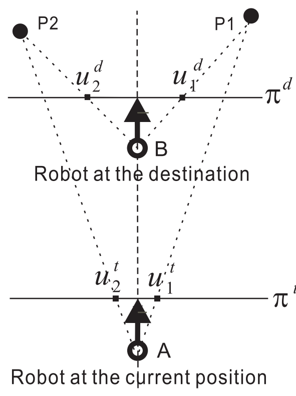


Fig. 1. The robot is on the path with the correct heading direction.

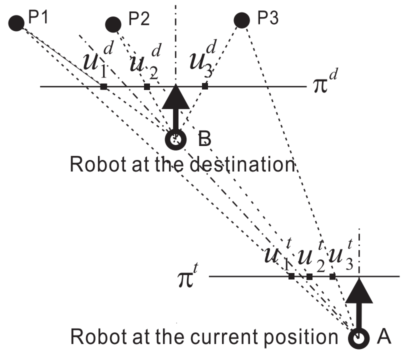


Fig. 2. The robot is facing the correct direction but has deviated laterally from the correct path.

Observation 1: If the robot is on the path from location A to location B and is facing B, then the image coordinates of a feature point in I_A lie between the image coordinates of its corresponding feature point in I_B and the principal point.

The situation is shown in Figure 1. For simplicity of illustration, the robot is assumed to be parallel to the ground plane, so that only the image coordinates parallel to the ground need to be considered. (As seen in the experimental results, this is not a fundamental assumption of the algorithm itself.) The value u_i^t is the x-coordinate of the ith feature point in the current image as the robot travels toward B, while u_i^d is the coordinate of the point in the destination image I_B. All coordinates are computed with respect to a coordinate system centered at the principal point, so that u_i^t and u_i^d are the signed distances from the principal point. In the drawing, the dark circle coincides with the focal point of the camera and indicates the robot's position, while the dark arrow indicates the robot's direction. The image plane is π^t at the current location and π^d at the destination.

As long as the robot is on the path and heading in the correct direction, then two constraints hold:

$$|u_i^t| < |u_i^d| \qquad \text{(Constraint 1)}$$
$$\text{sign}(u_i^t) = \text{sign}(u_i^d). \qquad \text{(Constraint 2)}$$

The converse, of course, is not true: Satisfying these two constraints does not guarantee that the robot is on the path with the correct heading. Thus, feature points for which the constraints hold provide no evidence about whether the robot is moving successfully. On the other hand, features for which one of the constraints is violated indicate that the robot needs to turn.

B. When the robot has deviated laterally

Now suppose that the robot has the correct heading but has deviated laterally from the correct path, as shown in Figure 2. Applying the observation of the previous subsection, three possibilities exist for any given feature point. First, both constraints might hold as in the case of feature 1. Such a feature does not provide any positive evidence about whether the robot is on course. Secondly, the coordinates of the feature point in the current image might exceed those of the destination image, thus violating Constraint 1, as in the case of feature 2 ($|u_2^t| > |u_2^d|$). Thirdly, the feature point might violate Constraint 2 by switching to the other side of the principal point, as in the case of feature 3 ($u_3^d > 0$ but $u_3^t < 0$). In the latter two cases the features not only indicate that the robot has deviated from the true path, but they also indicate the direction of the deviation — and hence the direction needed to turn in order to recover. This is summarized as follows:

Observation 2: When a feature on the left (right) of the destination image violates Constraint 1 in the current image, the robot needs to turn left (right).

Observation 3: When a feature on the left (right) of the destination image violates Constraint 2 in the current image, the robot needs to turn right (left).

C. When the robot has deviated angularly

Another situation to consider is when the robot is on the correct path but has deviated from the correct direction by a certain angle, as shown in Figure 3. As before, three possibilities exist: Both constraints might hold, as in feature 1; Constraint 1 might be violated, as in feature 2 ($|u_2^t| > |u_2^d|$); or Constraint 2 might be violated, as in feature 3 ($u_3^d < 0$ but $u_3^t > 0$). According to the observations of the previous section, feature 2 indicates that the robot should turn right since the feature is on the right side of the destination image. Similarly, feature 3 also indicates that the robot should turn right since it is on the left side. Thankfully, the observations of lateral and angular deviation are consistent.

D. Qualitative control algorithm

Combining the observations of the previous subsections yields an extremely simple control algorithm. For every successfully tracked feature point i, we compare the x-coordinate (u_i^t) of the point in the current image with the coordinate (u_i^d) of its corresponding point in the destination image:

- if $u_i^t > 0$ and $u_i^d < 0$, then turn right
- else if $u_i^t < 0$ and $u_i^d > 0$, then turn left
- else if $u_i^t > 0$ and $u_i^t > u_i^d$, then turn right
- else if $u_i^t < 0$ and $u_i^t < u_i^d$, then turn left
- else do not turn

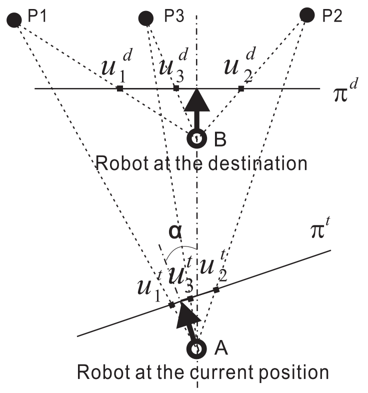


Fig. 3. The robot is on the correct path but has deviated from the correct direction by an angle α.

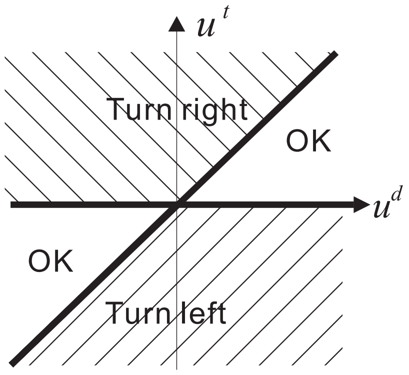


Fig. 4. Qualitative control decision space.

This algorithm is depicted graphically in Figure 4. The robot continually moves forward and, depending upon the relationship between the two coordinates, the robot either turns slightly in one direction or the other, or it continues moving forward without turning. The system is basically a bang-bang control in which the robot is always turning a constant amount to the right, a constant amount to the left, or not at all. Although one could extend this work by setting the turning speed based upon the amount by which the constraints are violated, we have found a constant turning speed to be sufficient for our purposes. Because the system is uncalibrated the actual principal point is not known but is estimated rather crudely as the center of the image. As a result, values that are close to the u^d axis are ignored to prevent the robot from turning based upon insufficient evidence (we use a threshold of 5 pixels away from the center).

The algorithm just described is for a single feature point. In order to guide the robot successfully, it is necessary to have multiple feature points distributed throughout both sides of the image. Each feature point votes for one of either "turn right", "turn left", or "do not turn". Those in the latter category are ignored. Letting N_R and N_L be the number of votes for turning right and left, respectively, the final decision can then be made based upon the following rule:

- if $N_R - N_L > 0$, then turn right
- else if $N_L - N_R > 0$, then turn left
- else do not turn

In practice the features are generally in agreement with each other, so that $N_R \gg N_L$ or $N_L \gg N_R$.

III. Tracking feature points

Feature points are automatically selected and tracked using the Kanade-Lucas-Tomasi (KLT) feature tracker [1], which computes the displacement $\mathbf{d} = [d_x \quad d_y]^T$ that minimizes the sum of the squared differences between consecutive image frames I and J:

$$\iint_W \left[I(\mathbf{x} - \frac{\mathbf{d}}{2}) - J(\mathbf{x} + \frac{\mathbf{d}}{2}) \right]^2 d\,\mathbf{x},$$

where W is a window of pixels around the feature point and $\mathbf{x} = [x \quad y]^T$ is a pixel in the image. This nonlinear error is minimized by repeatedly solving its linearized version by Taylor series expansion:

$$Z\mathbf{d} = \mathbf{e},$$

where

$$\begin{aligned} Z &= \sum_{\mathbf{x} \in W} \mathbf{g}(\mathbf{x}) \mathbf{g}^T(\mathbf{x}), \\ \mathbf{e} &= \sum_{\mathbf{x} \in W} \mathbf{g}(\mathbf{x}) \left[I(\mathbf{x}) - J(\mathbf{x}) \right], \end{aligned}$$

and $\mathbf{g}(\mathbf{x}) = \frac{1}{2} \partial [I(\mathbf{x}) + J(\mathbf{x})] / \partial \mathbf{x}$ is the spatial gradient of the average image. These equations are identical to the standard Lucas-Kanade equations [4], [13], [16] but are symmetric with respect to the two images. As in [13], [16], features are automatically selected as those points in the image for which both eigenvalues of Z are greater than a specified minimum threshold.

A. Handling lighting changes

The tracking algorithm just described relies on the well-known *constant brightness assumption* [17] in which image intensities are constant over time. As a robot moves about a real environment, however, the lighting conditions often change from one location to another. This problem is particularly acute in outdoor scenes during daylight hours when the robot moves in and out of shadows or when the sun is occluded or disoccluded by clouds. In such scenarios the standard algorithm often loses feature points prematurely. We present a simple extension of the KLT algorithm to handle illumination changes.

The residue equation defined above is augmented with a relative gain α and bias β describing the illumination relationship between the two images:

$$\iint_W \left[I(\mathbf{x} - \frac{\mathbf{d}}{2}) - \left(\alpha J(\mathbf{x} + \frac{\mathbf{d}}{2}) + \beta \right) \right]^2 d\,\mathbf{x}.$$

Applying a Taylor series expansion as above yields similar equations:

$$\begin{aligned} Z &= \sum_{\mathbf{x}\in W} \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x}), \\ \mathbf{e} &= \sum_{\mathbf{x}\in W} \mathbf{g}(\mathbf{x})\left[I(\mathbf{x}) - (\alpha J(\mathbf{x}) + \beta)\right], \end{aligned}$$

where $\mathbf{g}(\mathbf{x}) = \frac{1}{2}\partial[I(\mathbf{x}) + \alpha J(\mathbf{x})]/\partial\mathbf{x}$.

The values α and β are computed separately for each window by solving the following two equations:

$$\begin{aligned} E(I) &= \alpha E(J) + \beta \\ E(I^2) &= \alpha^2 E(J^2), \end{aligned}$$

where $E(I)$ is the mean intensity of the pixels in the window and $E(I^2)$ is the mean squared intensity of the pixels in the window; and similarly for $E(J)$ and $E(J^2)$.

B. Effects of dynamic objects

When a dynamic object (e.g., a person walking) comes into the view of the robot, two things may happen. First, feature points on the background may be lost due to occlusion. Because the algorithm treats all the features equally and because a new milestone image is taken frequently, this loss does not affect the performance of the algorithm in practice as long as the occlusion is fairly small (say, one-fourth of the image). If the occluding object is so large in the field of view that it causes a large number of feature points to be lost, then there is not enough information for the algorithm to continue. This problem can be solved by detecting the sudden loss of a large percentage of the features and commanding the robot to stop until it is able to reacquire the features once the object leaves the field of view. The second problem is that feature points on the background may be erroneously tracked using foreground texture, in which case it is difficult to recover. This problem is unlikely to occur because it would require the texture of the foreground to be similar to that of the background.

IV. Algorithm Overview

The approach involves a teaching phase and a replay phase.

A. Teaching phase

In the teaching phase, a person manually moves the robot along a desired path to gather training data. The path is divided into a number of non-overlapping segments defined by a constant amount of travel time. Within each segment, feature points are automatically detected in the first image and tracked throughout subsequent images. Feature points that are successfully tracked throughout a segment are stored in a database for use in the replay phase. For each feature its 25×25 gray-level intensity pattern is stored, along with its x-coordinate in the image for the first and last images of the segment. The last image of each segment is a milestone image. This is the extent of the learning that is performed in the teaching phase — no other information is stored.

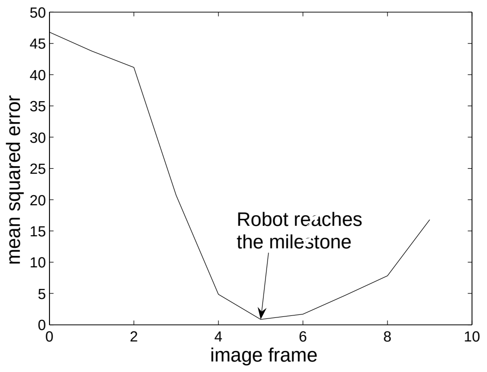


Fig. 5. A plot of the mean squared error between feature coordinates in the current image with those in the milestone image versus image frame, from an actual experiment.

B. Replay phase

In the replay phase, the robot is manually placed in approximately the same initial location as that of the teaching phase. The robot proceeds sequentially through the segments. At the beginning of each segment, correspondence is established between feature points in the current image and those of the first teaching image of the segment. Then, as the feature points are tracked throughout the incoming images, the feature coordinates are compared with those of the milestone image (i.e., the last teaching image of the segment) in order to determine the turning direction for the robot. The robot always moves forward at a constant speed, turning to the left or right at constant speed, or not turning at all, depending upon the output of the qualitative comparison of the feature points.

A remaining problem is to determine when the robot has reached a milestone position (i.e., the end of a segment) and needs to transition to the next segment. If there were no noise or error, one could simply wait until the feature coordinates in the current image were equal to those in the milestone image. In real images, of course, this will not work. A natural solution to the problem would be to introduce a threshold and then to wait until the mean squared error of the coordinates passed below the threshold. In practice, however, we have found it impossible to find a single threshold that works in all environments. The fact that the camera is completely uncalibrated contributes to this difficulty. We instead rely on the fact that the mean squared error tends to decrease over time as the robot approaches the milestone, and it increases afterwards. The point at which the error starts to increase indicates that the milestone has been reached, as shown in Figure 5. In our experiments, the monotonic nature of the error function as the milestone is approached is reliable, thus enabling this rather simple approach to work surprisingly well.

V. Experimental results

The proposed algorithm was implemented in Visual C++ on a Dell Inspiron 700m laptop (1.6 GHz) controlling an ActivMedia Pioneer P3-AT mobile robot with an inexpensive Logitech QuickCam Pro 4000 webcam mounted on the front.

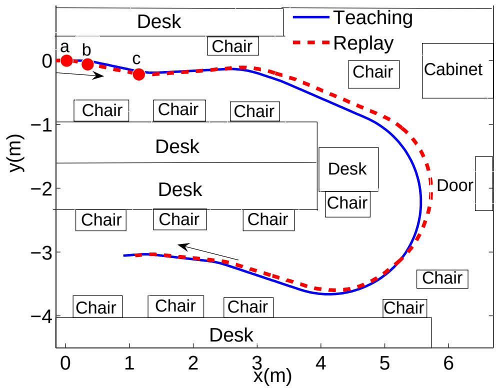


Fig. 6. TOP: The teaching and replay paths of the robot in an indoor environment (our laboratory). The locations a, b, and c are used in Figure 8. BOTTOM: Error versus distance traveled.

In all experiments the image sequence was divided into 0.8 sec segments, with up to 50 features being detected and tracked throughout each segment. The images were of size 320×240.

A. Indoor experiments

The algorithm was tested in an indoor environment, including our laboratory as well as a corridor of the hallway in our building. The maximum speed of the robot during the teaching phase was 100 mm/s and the turning speed was 4 degrees per second. Due to the delay caused by computational and response times, we found that the driving speed during the playback phase had to be smaller by a factor of approximately $1.5 - 2.5$ in order to avoid going off course, and the turning speed had to be slightly smaller as well. As a result, the playback speeds were 40 mm/s and 3 degrees per second, respectively. Figure 6 shows a typical run in which the robot successfully navigated between chairs and desks in our lab along a 10 m path. Trajectories displayed in the figure were computed by integrating the odometry readings (which are not used by the algorithm). The maximum error was 0.35 m (for 80% of the path the error was less than 0.2 m), and the final error was 0.03 m.

Figure 7 shows the decision process at two time instants during the replay phase. In one case the robot is pointing to the right of the current direction, so the features on the left half of the image violate either Constraint 1 or Constraint 2, thereby indicating the need for the robot to turn left. In the other case the features on the right half of the image violate one of the constraints, thereby indicating the need to turn right. In both cases notice the unanimity of the voting: Although many features simply plead ignorance, those features that do cast a vote are in agreement.

Figure 8 displays the decision process during three segments of the experiment. For display purposes, the feature coordinates are normalized so that the interval $y \in [0, 1]$ indicates "do not turn", while larger values ($y > 1$) indicate "turn right", and smaller values ($y < 0$) indicate "turn left", where y is

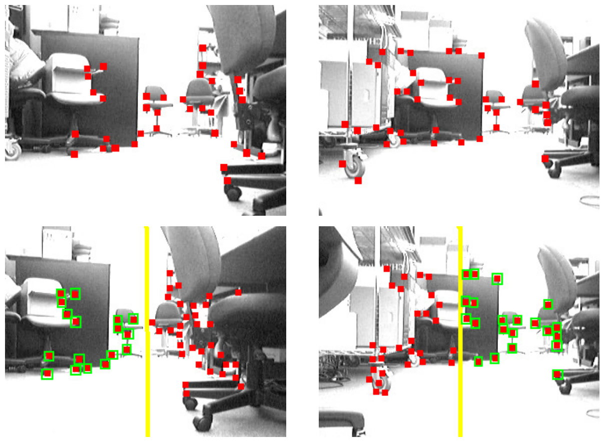


Fig. 7. TOP: Two milestone images from the indoor experiment, with all the feature points overlaid. BOTTOM: Two current images within each segment, as the robot moves toward the corresponding milestone location, with feature points overlaid. The features outlined by a rectangle (green in the electronic version of the paper) are the ones for which one of the constraints is violated. In the left column, the features on the left half of the current image tell the robot to turn left. In the right column, the features on the right half of the current image tell the robot to turn right.

defined as

$$y = \begin{cases} 1 + u_i^t / \left| u_i^d \right| & (u_i^d < 0) \\ u_i^t / \left| u_i^d \right| & (u_i^d > 0) \end{cases} . \qquad (1)$$

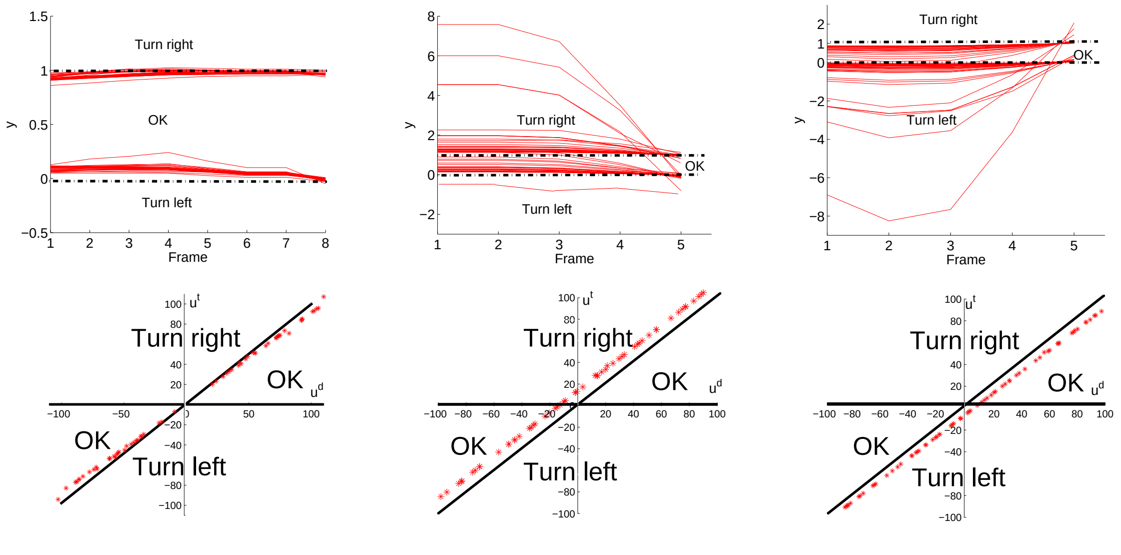
Notice again the near unanimity in voting (a lone feature in (b) votes incorrectly to turn left). Also notice that, as the robot turns (in (b) and (c)) the features move toward the OK region.

Indoor environments present a particular challenge for feature point tracking because of the lack of texture on the walls. Occasionally the robot failed to remain on course due to lack of texture in the scene that caused feature points to be lost. Difficulty was encountered primarily when the robot had to turn near the corner of a hallway containing no additional objects. We plan to address this problem by tracking vertical line features along with the feature points, whose coordinates can then be fed to the algorithm in the same manner.

B. Outdoor experiments

Dozens of experiments were also conducted outdoors, with the robot driving along sidewalks and parking lots of a university campus. With more room to maneuver, the driving and turning speeds of the robot were increased to 750 mm/s and 6 degrees per second, respectively. As before, the speeds during the replay phase were smaller: 350 mm/s and 4 degrees per second. Figure 9 show the results of a typical run in which the robot successfully followed a 140 m loop trajectory in a parking lot. The maximum error was 2.2 m, and the final error was 1.3 m.

As mentioned before, the algorithm makes no assumption about the ground being flat. We have conducted several experiments — all successful — in which the robot navigates up and down ramps with no modification to the algorithm. Figure 10 shows some sample images from a 40 m experiment.



(a) Do not turn (b) Turn right (c) Turn left

Fig. 8. TOP: The normalized feature coordinates of all the features plotted versus the image frame number for three segments of the indoor experiment. Features below 0 vote for "turn left", while those above 1 vote for "turn right". BOTTOM: A snapshot of the features from frame 2 of each segment plotted on the qualitative control decision space to show the instantaneous decision. The three segments correspond to the points a, b, and c from Figure 6.

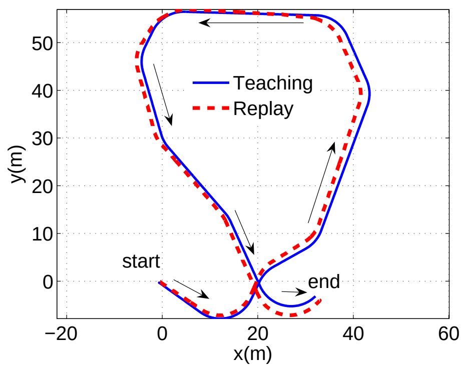


Fig. 9. TOP: The teaching and replay paths of the robot in an outdoor environment (parking lot). BOTTOM: Error versus distance traveled.

We have also found the algorithm to be insensitive to dynamic objects as long as they do not cause too many of the feature points to be lost. Figure 11 displays sample images from an 80 m experiment in which a pedestrian walked by the robot and later a van drove by it. In frames 265 and 684, it is evident that many features have been lost due to the occlusion caused by the dynamic object. Because the milestone images change frequently, however, the algorithm quickly recovers, as can be seen by the new features in frames 283 and 692.

In our outdoor experiments, the algorithm failed occasionally. One cause of failure was the lack of contrast in the image when the robot was run at dusk, a problem which was no doubt made more acute because of the limited dynamic range of the CMOS sensor in the webcam. Another cause of failure was the lack of unique texture in trees when those objects took up a large percentage of the image. Although trees contain much texture, the similarity in appearance of features throughout a tree cause the feature tracker to frequently fail on these objects.

C. *Different cameras*

As an additional experiment illustrating the lack of calibration in the system, we ran two 50 m experiments outdoors. In the first run, we used the Logitech camera, as described before. In the second run, we used an Imaging Source DFK21F04 Firewire camera with an 8.0 mm F1.2 lens. In each case the same camera was used for both teaching and replay. Without changing any parameters between runs, the algorithm was able to successfully follow the teaching path, as shown in Figure 12.

VI. CONCLUSION

A simple and efficient algorithm has been presented for enabling a mobile robot to follow a desired path using a single off-the-shelf camera. Following a teach-replay approach, the robot navigates by performing a qualitative comparison of feature coordinates across the teaching and replay phases. As such, the algorithm does not make use of the traditional concepts of Jacobians, homographies, fundamental matrices, or the focus of expansion. It also does not require any calibration (even lens calibration). Experimental results on both indoor and outdoor scenes demonstrate the effectiveness of the approach on trajectories over 100 m, along with its robustness to effects such as dynamic objects and slanted surfaces.

We believe that visual sensing will be essential for mobile robots to progress in the direction of increased robustness, reduced cost, and reduced power consumption. Moreover, if robots can be made to use computationally efficient algorithms and off-the-shelf cameras with minimal setup (e.g., no calibration), then the door is opened for robots to be widely deployed (e.g., multiple inexpensive coordinating robots). The

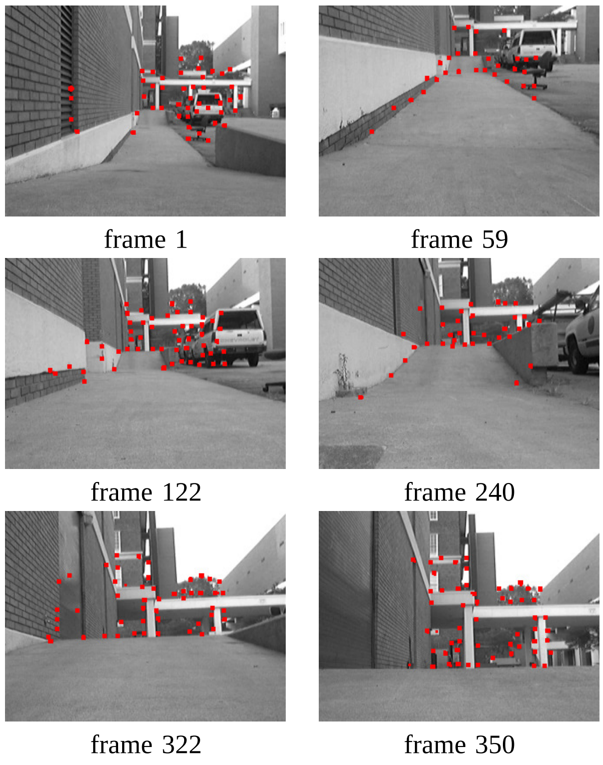


Fig. 10. Sample image frames from a sequence in which the robot traveled up and down an outdoor ramp.

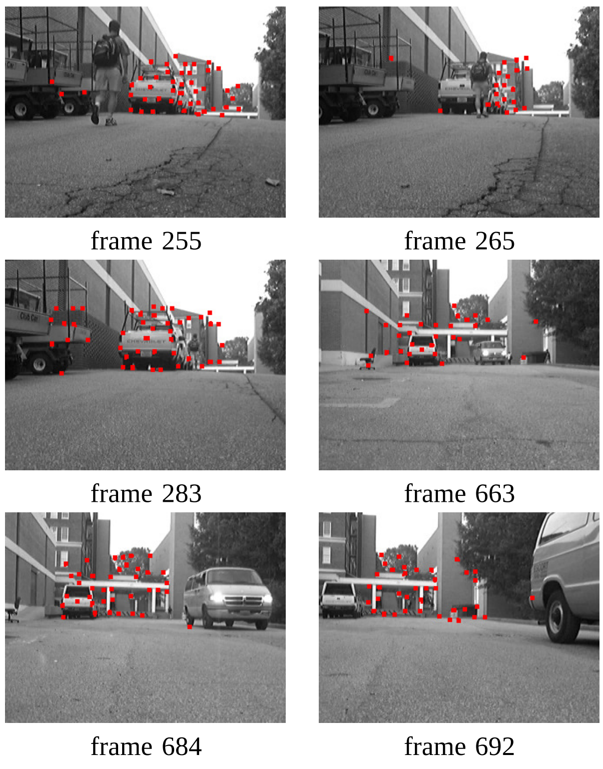


Fig. 11. Sample image frames from a sequence containing dynamic objects.

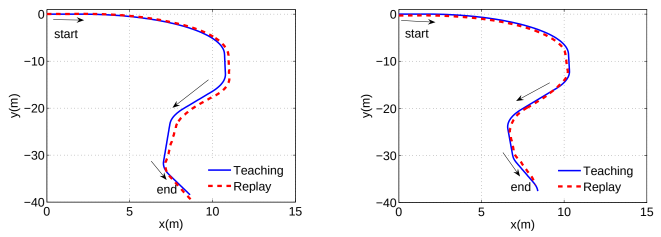


Fig. 12. Teaching and replay paths for the robot using two different uncalibrated cameras, with the same system parameters. LEFT: Imaging Source DFK 21F04 Firewire camera, RIGHT: Logitech QuickCam Pro 4000 USB webcam.

algorithm described in this paper is one small step in this direction. This algorithm is somewhat simplistic and limited in its view of the world, only enabling a robot to follow a predetermined trajectory, and only then when sufficient feature points are successfully tracked. In future research we plan to continue this line of inquiry to develop robust techniques to take advantage of the rich information available from video to solve problems such as obstacle avoidance and global localization.

REFERENCES

[1] S. Birchfield. KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker, http://www.ces.clemson.edu/∼stb/klt/.

[2] G. Adorni, S. Cagnoni, M. Mordonini, and A. Sgorbissa. Omnidirectional stereo systems for robot navigation. In *Proceedings of the Fourth Workshop on Omnidirectional Vision (Omnivis)*, 2003.

[3] S. Atiya and G. D. Hager. Real-time vision-based robot localization. *IEEE Trans. on Robotics and Automation*, 9(6):785–799, Dec. 1993.

[4] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *Intl. Journal of Computer Vision*, 56(3):221–255, 2004.

[5] D. Burschka and G. Hager. Vision-based control of mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 1707–1713, May 2001.

[6] F. Chaumette and E. Malis. 2 1/2-D visual servoing: A possible solution to improve image-based and position-based visual servoings. In *Proceedings of the International Conference on Robotics and Automation*, pages 630–635, Apr. 2000.

[7] J. M. Evans. HelpMate: An autonomous mobile robot courier for hospitals. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1695–1700, 1994.

[8] J. J. Guerrero and C. Sagues. Uncalibrated vision based on lines for robot navigation. *Mechatronics*, 11(6):759–777, 2001.

[9] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[10] B. Liang and N. Pears. Visual navigation using planar homographies. In *Proc. of the IAPR Intl. Conference on Pattern Recognition*, 2002.

[11] C. Sagues and J. J. Guerrero. Visual correction for mobile robot homing. *Robotics and Autonomous Systems*, 50(1):41–49, 2005.

[12] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi. Divergent stereo in autonomous navigation: from bees to robots. *International Journal of Computer Vision*, 14(2):159–177, Mar. 1995.

[13] J. Shi and C. Tomasi. Good features to track. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[14] Y. Shimizu and J. Sato. Visual navigation of uncalibrated mobile robots from uncalibrated stereo pointers. In *Proc. of the IAPR Intl. Conference on Pattern Recognition*, volume 1, pages 1346–1349, 2000.

[15] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[16] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.

[17] E. Trucco and A. Verri. *Introductory Techniques for 3D Computer Vision*. Upper Saddle River, NJ: Prentice Hall, 1998.