Robotica (2013) volume 31, pp. 225–249. © Cambridge University Press 2012
doi:10.1017/S0263574712000252

A monocular mobile robot reactive navigation approach based on the inverse perspective transformation

Francisco Bonin-Font*, Antoni Burguera, Alberto Ortiz and Gabriel Oliver

Department of Mathematics and Computer Science, University of the Balearic Islands, Palma, Balearic Islands, Spain.
E-mails: {francisco.bonin,antoni.burguera,alberto.ortiz and goliver}@uib.es

(Accepted April 17, 2012. First published online: May 22, 2012)

SUMMARY

This paper presents an approach to visual obstacle avoidance and reactive robot navigation for outdoor and indoor environments. The obstacle detection algorithm includes an image feature tracking procedure followed by a feature classification process based on the IPT (*Inverse Perspective Transformation*). The classifier discriminates obstacle points from ground points. Obstacle features permit to draw out the obstacle boundaries which are used to construct a local and qualitative polar occupancy grid, analogously to a visual sonar. The navigation task is completed with a robocentric localization algorithm to compute the robot pose by means of an EKF (*Extended Kalman Filter*). The filter integrates the world coordinates of the ground points and the robot position in its state vector. The visual pose estimation process is intended to correct possible drifts on the dead-reckoning data provided by the proprioceptive robot sensors. The experiments, conducted indoors and outdoors, illustrate the range of scenarios where our proposal has proved to be useful, and show, both qualitatively and quantitatively, the benefits it provides.

KEYWORDS: Mobile Autonomous Robots; Visual Navigation; Localization; Feature Traking.

1. Introduction

Obstacle avoidance is a crucial capability which autonomous robots must be endowed with in almost any task they have to accomplish. Range sensors, basically ultrasonic and laser, have been traditionally used for obstacle avoidance and mapless navigation in structured and nonstructured scenarios.[2,25] However, the former are only able to provide sparse sets of readings and, consequently, grouping a set of them is necessary to cover a relatively wide field of view, while the latter are still expensive if one wants to scan also a relatively wide range of orientations. Lately, visual solutions have emerged as competitive alternatives because of the low cost of cameras, the richness of the provided sensor data, and the larger spatial and temporal resolution available.

Another crucial task in mobile robotics is the continuous estimation of the robot pose, or self-localization for short. This task usually involves the use of proprioceptive sensors such as wheel odometers, gyroscopes, or accelerometers. However, the information that is obtained by these sensors can not be entirely trusted during long paths since they are liable to drift. Then, exteroceptive sensors have to be used to correct position data and to close the localization loop. Similarly to navigation, range sensors constitute the traditional approach,[9,14] but vision-based localization solutions have progressively been emerging over the rest, principally visual odometry approaches (e.g., see ref. [29]) and, more sophisticated, visual SLAM (*Simultaneous Localization and Mapping*) (see, e.g., refs. [28 and 29] among many other solutions). In general, SLAM solutions store all the observed map data in order to be able to close loops, but their execution time increases as the robot moves in the environment and accumulates information.

Localization approaches can be either based on data represented with respect to a world fixed coordinate frame, which usually coincides with the initial position of the robot, or based on data represented with respect to a moving coordinate frame attached to the robot. The former are commonly known as world-centric and the latter as robocentric.

Extended Kalman Filters(EKF) have been extensively used in the robotics context, in applications such as visual localization, mapping, or visual SLAM. These solutions integrate the environmental information observed by the onboard sensors (usually landmark positions) in the filter state together with control information such as the estimated pose or the velocity of the robot.[36]

World-centric EKF-based localization algorithms present inconsistencies in the pose estimations. These inconsistencies are caused by errors accumulated due to the linearization process inherent to the EKF.[20] In robocentric approaches, all coordinate measurements are computed with respect to the robot position and this allows to deal better with linearization errors than traditional filter-like world-centric systems.[11]

If a camera is already used for navigation and obstacle avoidance, it is desirable to take advantage of it when estimating the robot pose, so that the robot sensorial equipment can be reduced. Moreover, if the same data gathered to perform one of these tasks are also utilized to perform the other, then, the system might also simplify its software design and its computational charge. However,

* Corresponding author. E-mail: francisco.bonin@uib.es

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

solutions are normally proposed so as to solve either the obstacle detection or the localization problem, but not trying to solve both problems at the same time using a unique sensor and a unique visual information source.

This paper presents a system that can perform safe reactive navigation and robocentric self-localization from the output of a single visual data module, obtained from images consecutively captured by a single camera, mounted on a moving robot platform. To this end, the system starts detecting and tracking image main features in consecutive frames and classifies them either as obstacle or ground points using a new algorithm based on the IPT (*Inverse Perspective Transformation*). Obstacle points are used for obstacle detection and avoidance while ground points are used as natural landmarks for robocentric localization.

Next, the obstacle detection and avoidance procedure continues as follows: (a) the edge map of the processed frames is computed, and edges comprising those points classified as obstacle are discriminated from the rest emphasizing the obstacle boundaries; (b) range and angle of obstacles located inside a ROI (*Region of Interest*) are estimated by computing the orientation and distance of those obstacle points that are in contact with the floor; (c) a qualitative occupancy map is obtained with the data computed in the previous step, and finally the system computes the steering vector toward world areas free of obstacles. The method has been inspired on the visual sonar-based reactive navigation algorithms and implements the Vector Field Histogram method,[7] but here adapted for a vision sensor.

In parallel with the obstacle detection and avoidance procedures, the robocentric localization algorithm fuses all those features classified as ground in an EKF context together with the robot position obtained by dead-reckoning. Ground landmarks are added to the EKF as they are seen, or removed if they disappear from the field of view. The continuous release of the EKF vector state stabilizes the ground landmark coordinates and gives the consecutive robot positions. The aim of this localization process is to correct the errors in the robot pose data given by the proprioceptive sensors, which are especially significant in long routes.

The rest of this paper is structured as follows: Section 2 surveys different visual reactive navigation and localization methods, contrasting their advantages and inconveniences with our proposal, and finally highlighting the main benefits and contributions of the new algorithms presented in this paper; Section 3 gives a brief outline about the direct and the inverse perspective projections in the context of imaging geometry; Section 4 describes the IPT-based feature classification algorithm followed by the process of obstacle detection and creation of the local occupancy map; Section 5 states our proposal to achieve localization fusing dead reckoning and ground points by means of an EKF; finally, the conclusions and an insight for forthcoming work is given in Section 6.

2. Related Work

An extensive survey of the most meaningful visual navigation and some localization approaches can be found in ref. [5].

Some reactive visual navigation or obstacle detection solutions are based on textures or model appearances. Fasola and Veloso[16] proposed to use image color segmentation techniques for discriminating obstacles from the field in a ROBOCUP soccer match. Afterward, a gray-scale image processing algorithm was proposed for classifying the obstacles as being a robot or not. This approach assumed that the color on the field would not change, and that the color of the rest of obstacles would significantly differ from the field color. The classification of obstacles between robots and others was based on comparing the original image windows containing the obstacles, converted into gray-scale, with gray-scale window models. This type of solutions can fail in the presence of bad illumination conditions, shadows, highly textured floors, specularities, or inter-reflections, since these aforementioned optical effects can easily alter the texture levels (color, brightness, mean gray-level, etc.) or certain parts of the image.

Zhou *et al.*[37] utilized histograms to describe the appearance of prerecorded indoor images. Color, gradient, edge density, and texture histograms were extracted from images, and stored in a multidimensional histogram database. The navigation and localization algorithm was based on the recognition of the environment online by matching the multidimensional histogram of the current image with the multidimensional histogram of the stored templates. Working with histograms has two main advantages: (i) it saves computation resources and (ii) it is simpler and quicker than entire image-based correlation processes. Although the use of several information such as color, gradient, textures, and edges increases the power of the image matching process with respect to those solutions based on a one-dimensional histogram, there still remains a certain vulnerability to changes on the environmental conditions.

Contrarily, our theoretical model for obstacle detection is based only on image geometric considerations and it is unconstrained to colors, textures, or gradients. Furthermore, our experimental results show a considerable hability to overcome scenarios with changes on illumination, specularities, shadows, or different textures on the ground.

The IPT has also been the core of several mapless navigation systems. For example, in ref. [4], the subtraction of the two stereo images back-projected onto the ground generated a nonzero pixel area that evidenced the presence of obstacles. Simond and Parent[34] combined the IPT with the computation of the ground plane super-homography to discriminate obstacles from road in an autonomous guided vehicle application. Ma *et al.*[22] presented an automatic pedestrian detection algorithm based on IPT for self-guided vehicles. The system predicted new frames assuming that all image points laid on the floor. In all these three examples, the process computes the IPT of the whole image. However, not all the image points may contain relevant information for obstacle detection. The obstacle detection algorithm exposed in this paper only computes the IPT of a certain number of relevant image points saving computation complexity and time.

Many of the existing navigation approaches are based on detecting and tracking the environmental data perceived by the onboard sensors. When using range sensors, the readings

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

themselves can be used as the tracking data. However, when using a camera, the process for obtaining such trackable data usually starts by tracking the most stable and significant points in the image, so called *features*, or *corners* in the early techniques. As a matter of fact, detection and tracking of visual features is a mature but still growing area in the computer vision community nowadays.

Visual features are typically labeled with a descriptor of *n* components, where *n* changes for each detector algorithm, and they are categorized using the concept of distinctiveness. Distinctiveness depends on the size of the neighborhood window captured by the feature descriptor and also on the amount and type of data extracted from it. Harris[18] and KLT (*Kanade-Lucas-Tomasi*)[33] algorithms are early and fast techniques to respectively detect and track little discriminative features. Harris corners have a significant repeatability rate, but the tracking fails when there are scale changes between images or significant rotations.

Later, Lowe developed SIFT (*Scale Invariant Feature Transform*) to extract high discriminative image features.[21] SIFT is robust to scale, rotation, illumination, and camera viewpoint changes. It has a high degree of repeatability and it is remarkably robust to noise.

Other feature detectors such as SURF (*Speeded-Up Robust Features*),[3] FAST *(Features from Accelerated Segment Test),*[31] and CenSurE (*Center Surround Extremas*)[1] intend to increase speed and keep or improve the same level of performance. SURF generates less interest points and outperforms SIFT in terms of computation time. FAST is quick but it is vulnerable to noise. CenSurE features are distinctive, stable, and repeatable in changes of viewpoint. CenSurE authors claim in ref. [1] that their features have better computational characteristics than other scale-space detectors, and that they are better for visual odometry than other detectors since they result in longer track lengths and present better motion estimates.

Mourikis *et al.*[26] presented a visual navigation algorithm for planetary landing which tracked Harris Corners from consecutive frames taken during the descent. The Harris corners were integrated into an EKF together with data provided by an IMU (*Inertial Measurement Uni*) to infer the relative position of the lander, attitude, and velocity.

Çelik *et al.*[12] introduced a novel monocular SLAM method addressed to indoor Micro Aerial Vehicles (*MAV*). The system was able to compute absolute range and bearing without any active range sensor assuming that the altitude above the ground is known and the landmarks are static. 3D reliable landmarks were extracted from the set of features detected and tracked using the Shi and Tomasi algorithm.[33] Features comprised in corridor (or ceiling) lines were potential good landmark candidates for SLAM. Corridor or ceiling lines are parallel in the scene, but in the image plain they intersect in the *infinity point*. Depth and bearing were computed by exploiting the geometrical properties of these lines, using the features that were detected on them and using the vehicle altitude provided by an altimeter.

Some visual navigation solutions are based on the computation of homographies between consecutive scene images. Homographies are invertible transformations that map coplanar points between images. Thus, a single-homography-based algorithm is unwise to calculate the motion model in scenes with different planes. In general, homography-based navigation solutions need to discriminate the ground plane from other planes of the scene, being this one the real challenging point. Rodrigo *et al.*[30] constrained the locations in the scene of a large number of KLT (non-distinctive) features by several homographies which were strategically computed using SIFT (more distinctive) features. The several homographies presented in a set of consecutive frames were used to compute the robot motion. In ref. [27], Ollero *et al.* propose a new image tracking approach that computes homographies to compensate the UAV (*Unmanned Aerial Vehicle*) motion and to detect obstacles.

Our system uses, without distinction, all tracked features regardless if they are coplanar or not. There is no need of either computing homographies or discriminating the different planes of a scene, resulting conceptually simpler and quicker than the homography-based approaches.

Other approaches to visual obstacle detection and navigation use the properties of the optical flow to infer the presence of obstacles. Optical flow can be roughly defined as the pattern of apparent motion of features in a sequence of images caused by the relative motion of the camera with respect to the environment. Variations on direction and/or magnitude in the optical flow were used by some researchers to discriminate obstacles from the ground. Object boundaries appear as regions with significant variations in the optical flow while optical flow generated by ground points should be more homogeneous. However, specularities, inter-reflections or shadows on the ground, obstacles with textures similar than the ground, and nontextured or highly textured floors can produce either the detection of false obstacles or the lack of optical flow (see, e.g.,[35]). Since our approach is based only on geometrical considerations, as has been previously mentioned, it should not depend on any environmental or texture conditions, except from the fact than it needs features to detect the obstacles.

In recent years, *visual sonar* has become an original idea to provide range data and depth measurements for navigation and obstacle avoidance using vision in an analogous way to ultrasound sensors. The originality of this concept is in the way the range data is obtained. For example, in ref. [23], genetic algorithms were used to estimate the lowest nonground pixels (i.e., points where obstacles contact the ground) in successive columns of one image using environmental training data previously gathered with sonars instead of vision. The set of nonground points became the visual sonar information and it gave an approximate idea of the composition of the environment surrounding the robot.

In summary, the principal contributions and benefits of our work are:

1. A new image point classifier which discriminates obstacle points from ground points based only on the IPT.
2. An obstacle detector which has significant advantages with respect to other approaches: it avoids back-projecting the whole image but only a reduced number of points, it does not need to compute motion or optical flow, experimental results show a certain robustness to scenarios

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

with textured floors, shadows or inter-reflections and it overcomes scenes with multiple different planes.
3. A new method which uses the obstacle points found in the feature classification process to qualitatively represent the navigability of the area surrounding the robot.
4. A complete reactive navigation strategy which includes a robocentric localization algorithm to improve the robot pose data given by the proprioceptive sensors; both tasks, obstacle avoidance and localization use the same sensor and the same image data (obstacle points and ground points obtained from the feature classification process) reducing the robot sensorial equipment and the software complexity.
5. The use of the features classified as ground as natural landmarks reduces and simplifies the localization problem to a 2D problem (x and y on the ground plane), contrarily to, for example, ref. [13].
6. Not all the observed ground features are stored, only those closer to the robot, and if they are not seen during a certain number of frames, then, they are forgotten; this prevents our approach from closing loops, but, contrarily to classic visual SLAM approaches, it leads to a bounded and almost constant execution time, which is a very important issue to run the system online.
7. As will be shown in the experimental results, not many ground points are needed to estimate the robot pose, obviously, the more ground features are tracked, the better the performance.

3. Perspective Transformations

The direct perspective transformation maps three-dimensional points onto a two-dimensional plane called the plane of projection. This transformation models the process of taking a picture. The line that connects a world point with the camera lens intersects the image plane defining the corresponding and unique image point of that world point. The inverse process, that is, the projection of every image point back to the world is modeled by the IPT, so called *back projection*. The back-projected point will be somewhere in the line that connects the image point with the center of projection (camera lens).

The direct and the inverse perspective projections are usually modeled assuming a pinhole camera.[15,19] Three coordinate systems are involved: the world, the camera, and the image coordinate systems. The linear mapping from world to image points, both expressed in homogeneous coordinates, can be written as:[19]

$$\begin{bmatrix} x_p \\ y_p \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} T_w^c \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

$$T_w^c = \begin{bmatrix} R_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}, \quad (1)$$

where (x_p, y_p) are the image point coordinates, f is the focal length, (x, y, z) are the corresponding scene point world coordinates and T_w^c is the 4×4 transformation matrix that relates, via a R rotational (3×3) matrix and a T translational (3×1) vector, both the world and the camera coordinate frames.

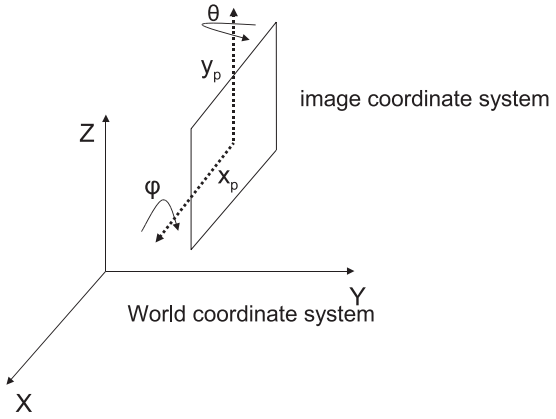


Fig. 1. Coordinate frame conventions.

The scene point world coordinates can be calculated from its corresponding image point coordinates knowing either the distance between the lens and the point in the space or any of the (x, y, z) world coordinates, as for example, for points lying on the floor ($z = 0$). The equations in a closed form to perform the IPT[15] are:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \lambda \begin{bmatrix} x_p cos\theta - f cos\varphi sin\theta + y_p sin\varphi sin\theta \\ x_p sin\theta + f cos\varphi cos\theta - y_p sin\varphi cos\theta \\ f sin\varphi + y_p cos\varphi \end{bmatrix}, \quad (2)$$

where (X_0,Y_0,Z_0) are the lens world coordinates at the moment in which the frame was taken, θ is the camera yaw angle, φ is the camera pitch angle, f is the focal length, and λ is a nonzero value that parametrizes the exact position of the world point on the inverse projective ray.

All points lying on the floor have $z = 0$. Solving for λ in z and substituting back, all performed over Eq. (2), the remaining world coordinates (x^*,y^*) turn out to be:

$$x^* = X_0 - \frac{Z_0 x_p cos\theta + (y_p sin\varphi - f cos\varphi)(Z_0 sin\theta)}{y_p cos\varphi + f sin\varphi}, \quad (3)$$

$$y^* = Y_0 - \frac{Z_0 x_p sin\theta - (y_p sin\varphi - f cos\varphi)(Z_0 cos\theta)}{y_p cos\varphi + f sin\varphi}, \quad (4)$$

where (x^*,y^*) are the ground point world coordinates. Coordinate system conventions and notations are illustrated in Fig. 1.

4. Obstacle Detection and Avoidance

4.1. Feature classification

4.1.1. Basic approach. Let us consider a single camera mounted on a moving platform running on a flat ground, either in translational or in rotational motion, and capturing frames at consecutive time steps. The relative motion

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

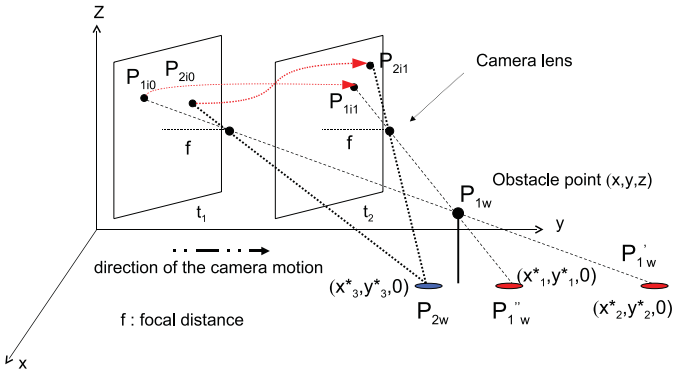


Fig. 2. (Colour online) The IPT-based obstacle detection.

between the camera and the scene causes the displacement of salient features in the image, changing their image coordinates (x_p, y_p), but still representing the same scene point. Given an image feature, its world coordinates can be computed by means of Eqs. (3) and (4) in two consecutive images assuming that it lies on the ground. If the assumption is correct, the two resulting world coordinates will coincide. However, if the feature does not correspond to a ground point, the resulting world coordinates turn out to be different to one another and different to the real world coordinates (x, y) of the scene point.

Hence, our algorithm starts by detecting a sufficiently large and relevant set of image features and tracking them continuously during the robot motion. All image features are back projected in consecutive images assuming $z = 0$ and classified as obstacle or ground points using the next criteria:

$$D = \sqrt{(x_2^* - x_1^*)^2 + (y_2^* - y_1^*)^2} \Rightarrow \begin{cases} if\, D > \beta \Rightarrow \text{obstacle}, \\ if\, D \leq \beta \Rightarrow \text{ground}, \end{cases} \tag{5}$$

where D denotes the *discrepancy*, (x_1^*,y_1^*) and (x_2^*,y_2^*) denote the (x, y) world coordinates assuming $z = 0$ (computed with Eqs. (3) and (4)) of the image feature identified in two frames taken at instants t_1 and t_2, respectively, and β is the threshold for the maximum difference admissible between (x_1^*,y_1^*) and (x_2^*,y_2^*) to classify the feature as ground point. Ideally β should be 0.

The idea is illustrated in Fig. 2. Two frames of a scene are taken at instants t_1 and t_2. Point P_{2w} is on the floor. Its projection onto the image plane at instants t_1 and t_2 generates the image points P_{2i0} and P_{2i1}, respectively. The inverse transformation of P_{2i0} and P_{2i1} generates a unique point P_{2w}. P_{1w} is an obstacle point. Its projection onto the image plane at t_1 and t_2 generates, respectively, the points P_{1i0} and P_{1i1}. However, the back projection of P_{1i0} and P_{1i1} assuming $z = 0$ generates two different points on the ground, namely, P'_{1w} and P''_{1w}.

Establishing good correspondences between image points, for example between P_{2i0} and P_{2i1} in Fig. 2, is of crucial importance as incorrect correspondences lead to wrong classifications. Wrong correspondences between image points in consecutive frames are filtered out using RANSAC and imposing the epipolar constraint.[1] It is important to emphasize that this proposal does not depend on any specific feature detector or matcher. In our particular implementation, SIFT features were used in the very first experiments conducted to assess the classifier performance and in the first online obstacle avoidance tests. However, later, the KLT feature detector and sparse tracker was chosen for the online autonomous navigation because: (a) it is faster than other detectors, (b) it generates a sufficiently large number of features with high repeatability, (c) feature descriptors are not significantly affected by translation, rotation, or changes in scale since differences in consecutive frames are negligible at relative hight frame rates, and (d) more than 90% of the tracked features are inliers and well classified.[6]

4.1.2. Direct identification of some obstacle points. The set of scene points that map to a given image point can be written as:[15]

$$p = p_l + \lambda(p_p - p_l), \tag{6}$$

where p is a point of the scene with coordinates (x, y, z), p_l is the camera center (X_0, Y_0, Z_0), and p_p is the image point corresponding to the scene point p. This expression is equivalent to Eq. (2). The idea is illustrated in Fig. 3(a).

All those image features (p_p) that correspond to scene points (p) located below the plane parallel to the flat ground and that contains the lens center (p_l) require a positive λ to be back-projected onto the ground, while λ has to be negative for all image features corresponding to scene points located above the mentioned plane.

The idea is illustrated in Fig. 3(b). p_1 is a point lying on the floor, and its corresponding image point is p_{p1}. In Eq. (6), p_1 is obtained from p_{p1} with $\lambda > 0$. Likewise, p_2 and p'_{2w} result from p_{p2} for $\lambda > 0$. However, p_{p3} leads to a point on the ground p'_{3w} for which λ is < 0.

Clearly, image points with λ negative necessarily correspond to scene points above the ground, while image points with λ positive can correspond either to elevated points or to points lying on the floor. Consequently, the process of inverse projection and discrepancy computation can be omitted for all these image features that need a $\lambda < 0$ to be projected onto the ground, as they can be directly classified as obstacle points.

Doing some manipulations on Eq. (6), the z world coordinate of a scene point can be calculated as:[15]

$$z = Z_0 + \lambda(f sin\varphi + y_p cos\varphi), \tag{7}$$

where λ determines the exact position of the scene point in the inverse perspective projecting ray.

If $z = 0$, λ can be easily solved as:

$$\lambda = \frac{-Z_0}{f sin\varphi + y_p cos\varphi}. \tag{8}$$

[1] $x_p'^T F x_p = 0$, where $x_p'^T$ and x_p are the point image coordinates in two consecutive frames, and F is the fundamental matrix[19]

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

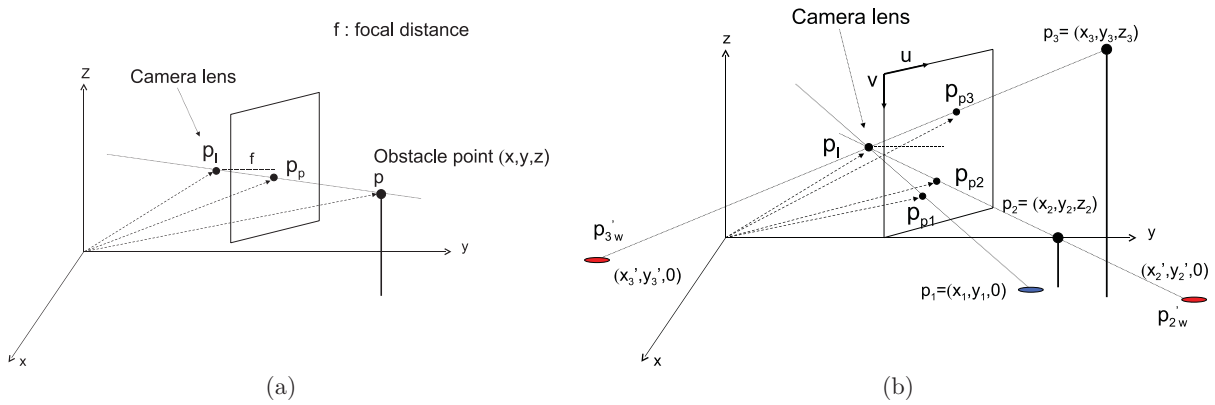


Fig. 3. (Colour online) (a) The IPT: $p = p_l + \lambda(p_p - p_l)$. (b) $\lambda's$ positives and negatives.

Making $\lambda < 0$ means that $(f sin\varphi + y_p cos\varphi) > 0$, then solving for y_p:

$$y_p > -f \tan\varphi. \tag{9}$$

Expressing the y_p image coordinate in pixels, and translating the origin of the image coordinate system from the image center to the upper left corner (from now on, the image coordinates with respect to the origin located at the upper left corner are denoted as (u, v)), we obtain that:

$$v < v_0 + k_v f \tan\varphi, \tag{10}$$

where v_0 depends on the image resolution and represents the constant factor used to convert vertical image coordinates with respect to the center of the image into vertical coordinates with respect to the upper-left corner of the image, and k_v factor is the vertical relation $[pixels/length]$ in the images.

All image points with a vertical coordinate v lower that $v_0 + k_v f \tan\varphi$ pixels correspond to obstacle points of the scene. Therefore, these image points can be directly classified as obstacle without being evaluated using the expression (5), and thus reducing the total execution time of the classifier.

4.2. Obstacle profiles

Image features or corners are usually detected at regions of high gradient, thus, they are likely to be near or belong to an edge. Besides, features classified as obstacle are most likely to be contained or near a vertical edge belonging to an obstacle. Hence, the next step of the algorithm is the computation of edge maps and the association of such edges with features classified as obstacle. This permits isolating the obstacle boundaries from the rest of edges and getting a qualitative perception of the environment.

In order to combine a high degree of performance in the edge map computation with a relatively low processing time, the edge detection procedure implements a reduced version of the Canny edge detector[10] running only these two basic steps:

1. First, the original image is convolved with a 1D Gaussian derivative horizontal kernel. This permits detecting, with a single convolution, zones with high vertical gradient from smoothed intensity values.
2. Next, a process of hysteresis thresholding is applied. Two thresholds are defined. A pixel with a gradient above the highest threshold is classified as edge pixel. A pixel with a gradient above the lowest threshold is classified as edge if it has in its vicinity a pixel with a gray value higher than the highest threshold. In this way, edge pixels with low gradient are not filtered if the threshold is defined too high, and noise is not considered as an edge if the threshold is defined too low.

The algorithm locates in the image all features classified as obstacle. Then, for each feature, the algorithm creates a window (a maximum of 10 pixels for x and y) centered on the feature image coordinates, and founds all edge pixels (white pixels) which are inside this window. Every edge is tracked down (in the direction of increasing the vertical image coordinates, supposing the origin in the image upper left corner) starting from the obstacle image point position until the last edge pixel or a ground point is found. The point where the algorithm finds the end of the edge or a ground point will be considered as to be the point or points where the object rests on the floor. This process permits isolating the relevant obstacle boundaries from the rest of edges (basically edges located on the ground) and getting a qualitative perception of the environment. In our implementation, the window used to find edges near obstacle points is longer in the vertical direction to overcome possible discontinuities in the computed obstacle vertical edges.

4.3. Computation of a local occupancy map for reactive navigation

Knowing the camera global position (X_0, Y_0, Z_0), and the world coordinates of a ground point $(x, y, 0)$, the distance and the relative orientation between the camera and the ground point can be easily computed. Since it is possible to calculate the world coordinates of those points where the obstacle touches the ground, the orientation and distance of obstacles with respect to the robot can be qualitatively estimated by computing the distance and orientation with respect to the camera of those obstacle points that are in contact with the floor. A semicircular area of a fixed radius, centered at the robot position and virtually located on the ground plane,

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

is considered to be the ROI used for the obstacle avoidance. Only obstacles detected inside this ROI are considered to be avoided.

The ROI is, in turn, divided in angular regions. Histograms of obstacle-to-ground contact points at each polar direction of the ROI are computed. Those polar directions corresponding to angular regions occupied by at least one obstacle-to-ground contact point are labeled as forbidden and those free of obstacle-to-ground contact points are included in the set of next possible movement directions. This process results in a qualitative polar occupancy map of free and occupied zones, that qualitatively represents the vicinity of the robot. Obstacle-free polar regions which are narrower than a certain threshold (determined empirically and depending on the robot size) are excluded from the possible motion directions. If all free angular regions are narrower than the defined threshold, the algorithm concludes that all the space ahead is occupied by obstacles and returns the proper order to stop the robot.

4.4. Overall performance of the classifier: experimental assessment

A total of 16 different image sequences, with more than 150 frames for each sequence, were recorded by a calibrated camera mounted on a moving Pioneer 3DX robot. The camera height was 430 mm, the camera pitch and yaw angles were -9° and 0°, respectively, and the focal length was 3.720 mm. Test recordings comprised several types of scenarios: with regular and irregular shaped obstacles, without obstacles, with textured and untextured floor, with specularities and with low illumination conditions. Frames were recorded during rectilinear motion in the forward direction and all frames were processed off-line to test the point classifier performance.

Just before running the feature tracking procedure, all images were (a) down-sampled to a resolution of 256×192 pixels, in order to reduce the feature tracking computation time, and (b) undistorted to correct the error in the image feature position due to the distortion introduced by the lens, and thus, to increase the accuracy in the calculation of (x^*, y^*). For each scene, the classifier algorithm was run over undistorted pairs of 0.5-second separation consecutive frames (one frame each 2 cm) so that the effect of the *IPT* was noticeable. This separation between frames was empirically determined. Increasing the frame rate decreases the *IPT* effect over the obstacle points, and decreasing the frame rate delays the execution of the algorithm.

In this assessment, SIFT features were used and the tracking algorithm was implemented according to the methods and approaches described in ref. [21]. The camera world coordinates were calculated for each frame by dead reckoning, taking into account the relative camera position with respect to the robot center.

First of all, the classifier performance was formally determined using Receiver Operating Characteristic (ROC) curves.[8] A ROC curve can be seen as a graphical plot of the binary classifier sensitivity evaluated under different environmental conditions. These different conditions are usually modeled by a threshold that determines the boundary used to decide whether an element belongs to a class or to another. The ROC curve space can be defined by the true positives rate (*recall*) in the y-axis and the false positives rate (*fall-out*) in the x-axis:

$$recall = \frac{TP}{TP + FN} \quad fall - out = \frac{FP}{FP + TN}, \quad (11)$$

where TP is the number of true positives (obstacle point correctly classified), FN is the number of false negatives (obstacle point classified as ground), FP is the number of false positives (ground point classified as obstacle), and TN is the number of true negatives (ground point correctly classified).

Both axes range between 0 and 1. A ROC curve close to the straight diagonal line that connects the points (0, 0) and (1, 1) defines a completely aleatory classifier, while a ROC curve close to the $y = 1$ line defines a classifier with a sensitivity close to 100%. In our case, these curves were computed for every pair of consecutive images. They plotted the *recall* of classified points in the y-axis vs. the *fall-out* in the x-axis for different values of the threshold β defined in Eq. (5). The cost function $f(\beta) = FP(\beta) + \delta FN(\beta)$ was calculated for every pair of consecutive frames, varying the β value. The optimum β was chosen so as to minimize $f(\beta)$. During the experiments, δ was set to 0.5 to prioritize the minimization of false positives over false negatives. For all tested scenes in all different scenarios, $f(\beta)$ minimized for a common β value around 21 mm.

The AUC (*Area Under the ROC Curve*) is commonly used as a summary measure, providing significant insights of the system performance in a very simple way: the closer it is to 1 the better, with 1 indicating a perfect classifier and 0.5 indicating a completely aleatory performance.[17] The AUC is equivalent to the probability that a randomly chosen member of one class (class A) have a smaller estimated probability of belonging to the other opposite class (class B) than a randomly chosen member of class B. Although some authors of some other disciplines claim that the AUC permits distinguishing between bad and good classifiers but not between good classifiers,[24] as a matter of fact, we have considered that classifiers with higher AUCs present better classification results.[32]

The AUC was calculated for every ROC curve as a measure of success classification rate.

The classification of those image features that present a discrepancy D (see Eq. (5)) close to the threshold β can be altered very easily with slight oscillations in β or in D.

In order to decrease the sensitivity of the classifier with regard to β, all these points are left unclassified. Besides, in a previous training phase conducted before the autonomous navigation, histograms of D values for misclassified points were built from images taken in all different testing scenarios. D ranges that presented a higher presence in these histograms were stored in a database. During the autonomous navigation phase, all points with a D included in any stored D interval were neither classified. Then, nearly all ground points classified as obstacles were eliminated reducing the risk of detecting false obstacles. Although some true obstacle points were also removed, the remaining ones were sufficient to permit the detection of those obstacles. The elimination of

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

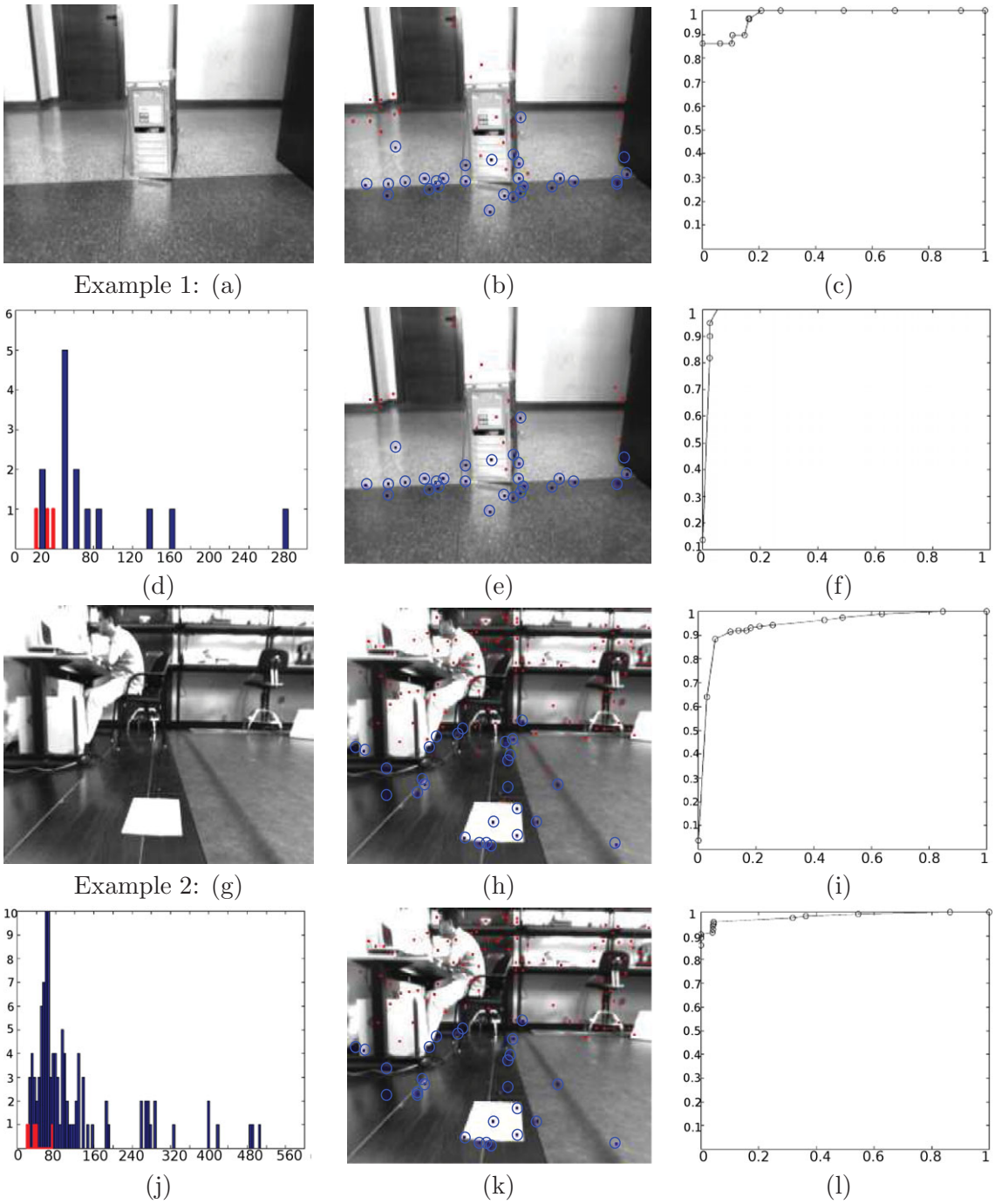


Fig. 4. (Colour online) Classifier Assessment. (a), (g) Undistorted first frame of examples 1 and 2, respectively. (b), (h) Undistorted second frames. (c), (i) ROC curves of scenes 1 and 2, respectively (AUC = 0'9791, AUC = 0'9438). (d), (j) Histogram of D values for TP (blue) and FP (red). (e), (k) Second frame with filtered SIFT points. (f), (l) ROC curves after the filter (AUC = 0'9843, AUC = 0'9821).

miss-classified features increases the *recall* and decreases the *fall-out*, improving the success rate of the classifier.

Figures 4 and 5 show some examples of the classifier algorithm. Pictures (a)-(b) and (g)-(h) of Fig. 4, and (a)-(b) of Fig. 5 show two consecutive frames corresponding to scenes 1, 2, and 3, respectively. Pictures (b) and (h) of Fig. 4 and picture (b) of Fig. 5 show obstacle points in red and ground points in blue. Although some ground points were wrongly classified as obstacles, the AUC of the ROC curves for scenes 1–3 (plots (c) and (i) of Fig. 4 and plot (c) of Fig. 5) were 0.98, 0.94, and 0.92, respectively. These values suggest considerably high success rates in the classification results. Plots (d) and (j) of Fig. 4 and plot (d) of Fig. 5 show the histograms of D values for true (blue) and false positives (red). D values for true positives predominantly range from 21 mm to 300 mm, and false positives from 21 mm to 50 mm. All positives with D values from 21 mm to 50 mm were filtered out, as shown in pictures (e) and (k) of Fig. 4 and in picture (e) of Fig. 5, where no false positives appear. The risk of detecting false obstacles is reduced but a sufficient number of true positives is maintained to detect the real obstacles. Plots (f) and (l) of Fig. 4 and plot (f) of Fig. 5 show the final ROC curves. Their AUCs increased to 0.98, 0.98, and 0.94, respectively, thus improving the performance of the

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

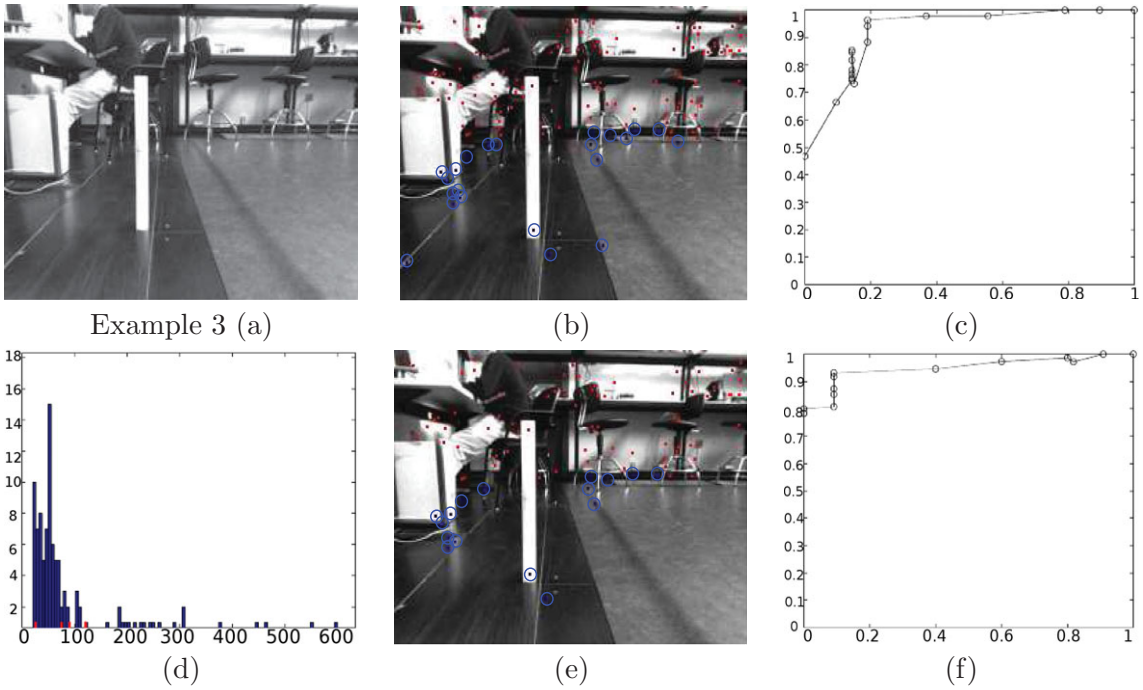


Fig. 5. (Colour online) Classifier Assessment. (a) Undistorted first frame. (b) Undistorted second frame. (c) ROC curve (AUC = 0'9236). (d) Histogram of D values for TP (blue) and FP (red). (e) Second frame with filtered SIFT points. (f) ROC curve after the filter (AUC = 0'9494).

Table I. Rates of false positives and AUC for some analyzed scenes.

Scene	FP/(Total N_{br} of features)	AUC after the filter
scene 1	0.0129	0.9482
scene 2	0.0078	0.9412
scene 3	0.0847	0.9434
scene 4	0.0000	0.9554
scene 5	0.0069	0.9834
scene 6	0.0000	0.9376
scene 7	0.0069	0.9827
scene 8	0.0166	0.9900

classification procedure. Notice that all the scenes present inter-reflections and specularities, although they do not affect the classifier performance.

Table I presents some relevant data concerning various tested scenes. Twenty-one frames were analyzed for every scene. The total amount of features ranged between 105 and 260. The percentage of false positives with respect to the total number of features suggests failure ratios lower than 8%. Finally, the AUC values after the filtering process are all greater than 0.93, suggesting success rates near the optimum value of 100%. Although the classifier performs accurately, obstacle points near the floor, which may have low D values, have more probability of being miss-classified than others with a larger height.

4.5. Experimental results: autonomous navigation in an exploration task

In a first set of experiments the mission to be accomplished by the vehicle was merely wandering through the free space, detecting the obstacles present in the scenarios described in Section 4.4. Hence, there was no target point.

The direction of motion was given as a vector pointing to the center of the widest polar obstacle-free zone. Positive angles resulted for turns to the right and negative angles for turns to the left. SIFT features were also used in these experiments and the system worked with the same operating conditions described in Section 4.4, in terms of camera, robot speed, focal distance, lens height, frame rate, image resolution, and intrinsic camera parameters for image undistortion. The camera world coordinates were obtained compounding the robot world coordinates obtained from the dead reckoning data with the camera position with respect to the robot.

Figure 6 shows four examples of the obstacle contour discrimination algorithm applied over images of a sequence recorded during these navigation experiments. Pictures (a), (c), (e), and (g) are the second frame of four different pairs. Pictures (b), (d), (f), and (h) show the corresponding edge maps with the obstacle boundaries highlighted in orange. Although picture (e) shows a very high inter-reflection on the ground and a very granulated texture on the floor tiles, only real obstacle boundaries survived.

Figure 7 shows in plots (a)–(d) the trajectories followed by the robot, according to the odometry data, during the navigation through the scenarios of Figs. 8–11. All space coordinates shown in these plots are expressed in mm. The blue circle denotes the starting point and the red circle denotes the end point. Figures 8–11 show data corresponding to some navigation experiments. Pictures (a)–(d) in all four figures show the second frame of some pairs recorded and processed in scenarios 1–3. Every image was taken before the robot had to turn to avoid the frontal obstacles;

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

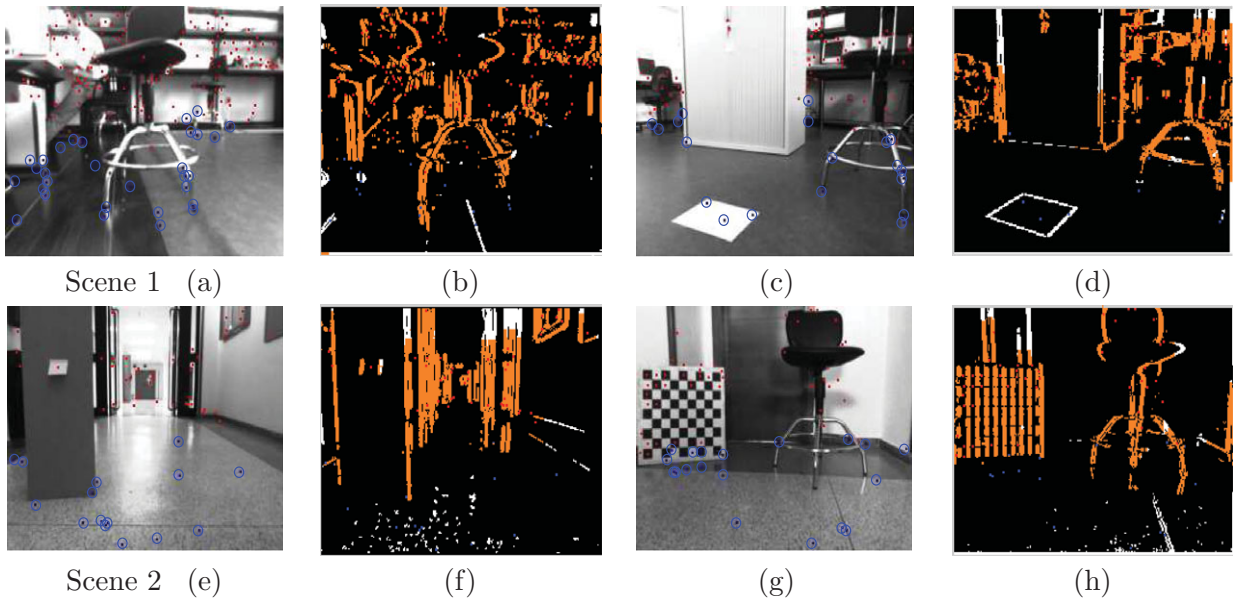


Fig. 6. (Colour online) Examples of obstacle contour discrimination. (a), (c), (e), (g): Undistorted second frame of different pairs of consecutive images of Scenes 1, and 2. (b), (d), (f), (h): Obstacle contours.

obstacle points are shown in red and ground points in blue. Figure 8 (scenario 1) shows a room full of obstacles with regular and irregular shape. This scene presents shadows and inter-reflections. Figure 9 (scenario 2) corresponds to a corridor with a very high textured floor, columns, walls, inter-reflections, and some specularities. Figures 10 and 11 (scenario 3) present bad illumination conditions, important inter-reflections, specularities on the floor, and some image regions (white walls, shelves, and lockers) with homogeneous intensities and/or textures. Regions with homogeneous textures result in few distinctive features and poorly edged obstacles, which can make their detection difficult. Pictures (e)–(h) in all four figures show the vertical contours (in orange) comprising obstacle points. As shown, obstacle contours were differentiated from the rest of the edges. Obstacle-to-ground contact points inside the ROI have been highlighted in pink.

Histograms (i)–(l) in Figs. 8–11 account for the number of obstacle-to-ground contact points detected in each polar direction. Plots (m)–(p) show the semicircular floor portion in front of the robot with all the obstacle-to-ground contact points, representing the local and qualitative occupancy maps used for reactive navigation.

In all these tests, the steering direction is shown in green. The experiments performed show a remarkable robustness against textured floors, bad illumination conditions, shadows or inter-reflections, and deals with scenes that include different planes. In all scenes, features were well classified with success rates greater than 90%, obstacle profiles were correctly detected, and the robot navigated through the free space avoiding all obstacles.

4.6. Experimental results: navigation to a goal point

In a later stage, the platform mission was to move from a starting point to a goal point, or to a set of subsequent goal points. According to Bonin-Font *et al.*,[6] in these experiments SIFT features were substituted by KLT features. The system used the same operating conditions as described in Section 4.4.

Figures 12 and 13 plot four different trajectories conducted by the robot, three of them in scenario 3 and one in scenario 1. In all these plots, the blue circle represents the starting point, the red circles represent the goal points and all space coordinates are expressed in *mm*. Besides, the red dots represent the robot trajectory according to the odometry

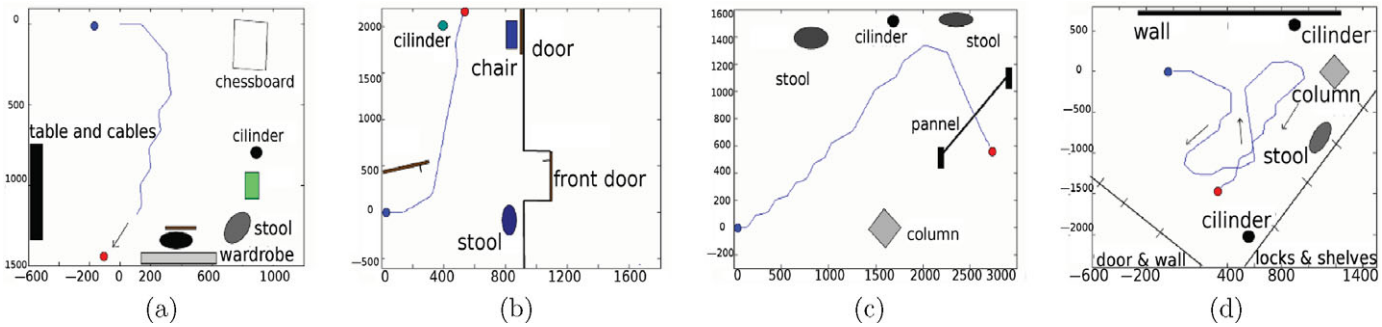


Fig. 7. (Colour online) (a), (b), (c) and –(d), robot trajectories for tests of Figs. 8, 9, 10 and –11, respectively. Length information referred in mm.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

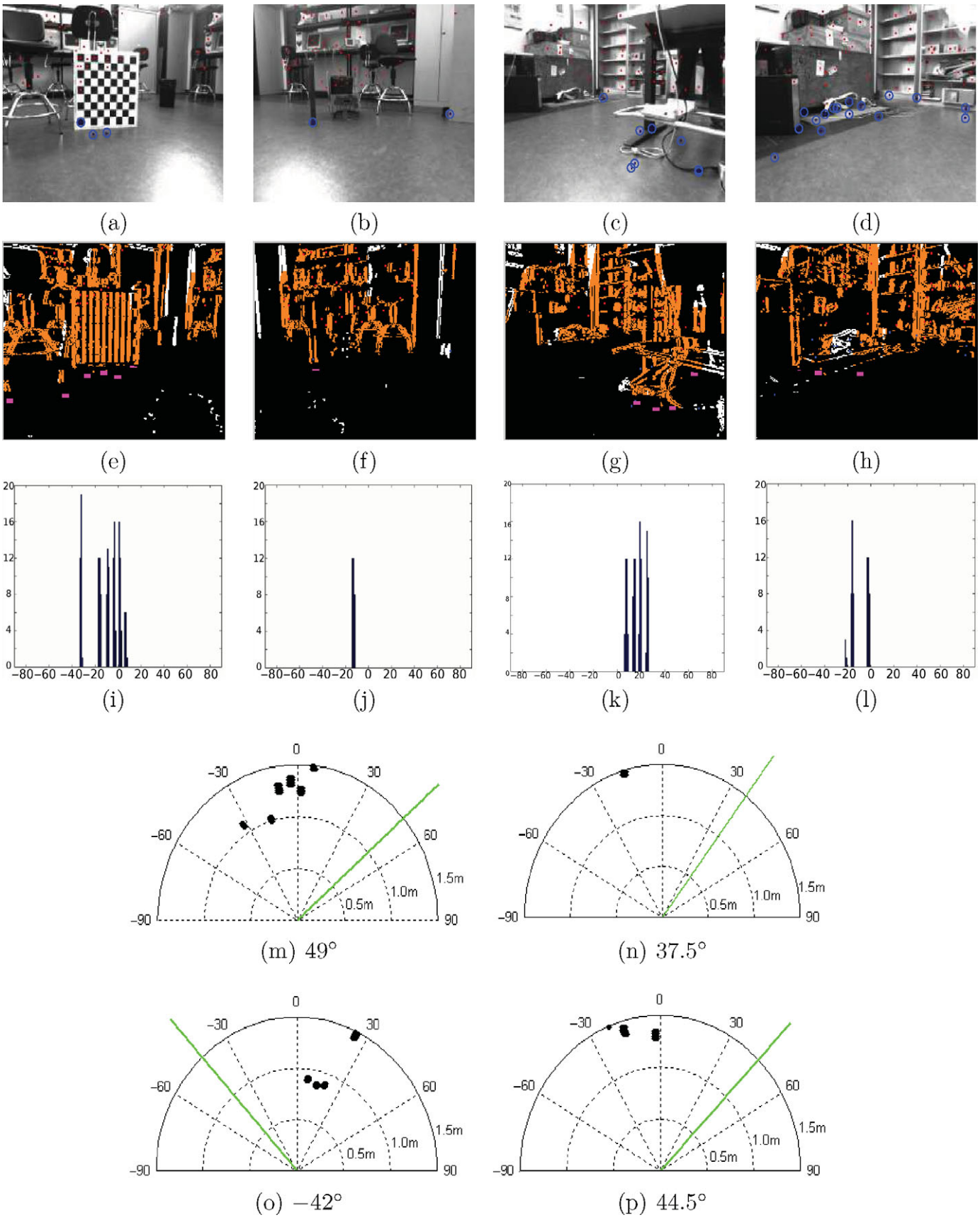


Fig. 8. (Colour online) Scenario 1. Experiment 1: (a), (b), (c) and –(d), undistorted frames; (e), (f), (g) and –(h), corresponding edge maps with obstacle borders highlighted in orange. (i), (j), (k), –(l), histograms of obstacle-to-ground contact points for each polar direction between -90° and 90°. (m), (n), (o) and –(p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and –(d), respectively.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

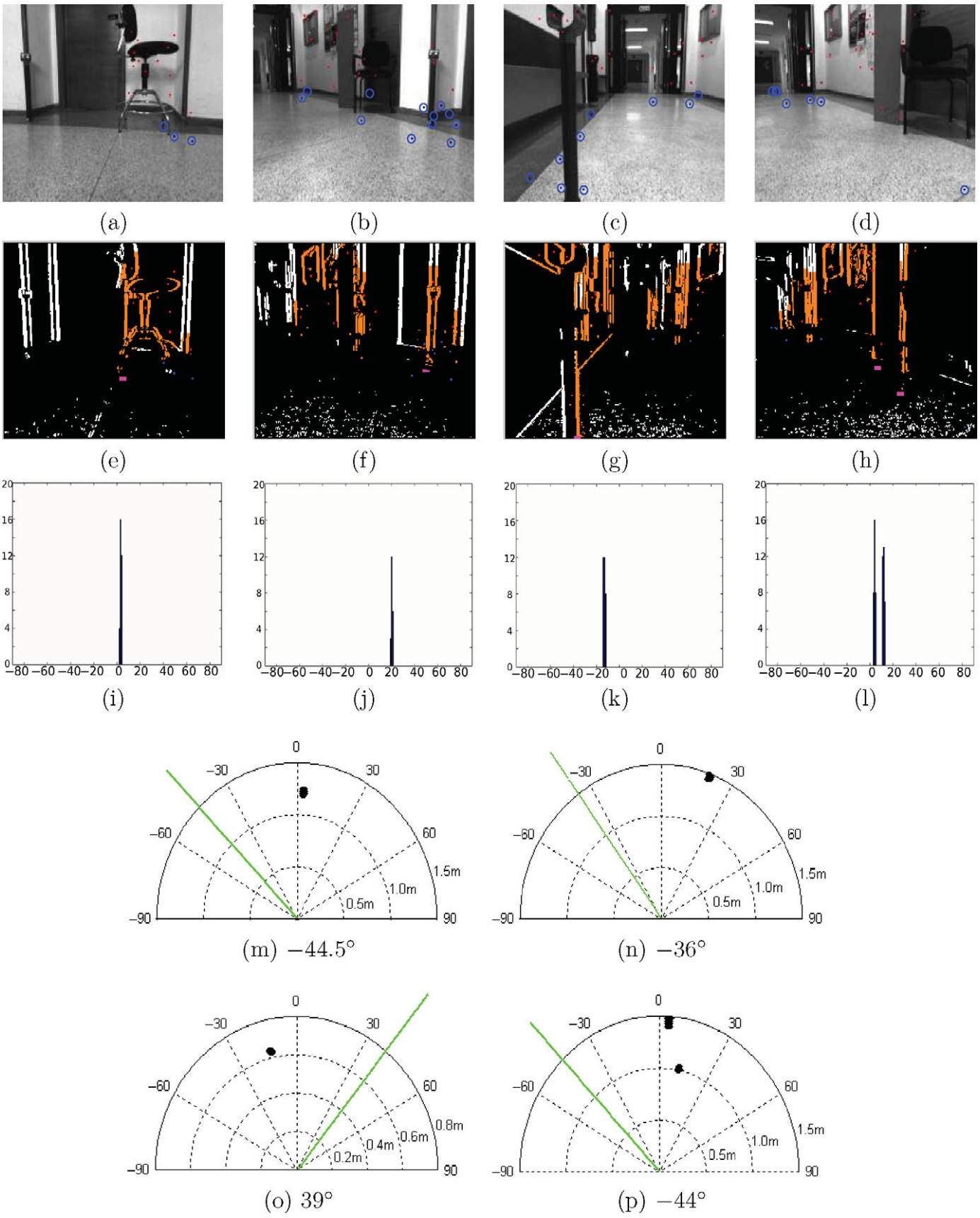


Fig. 9. (Colour online) Scenario 2. Experiment 2: (a), (b), (c), –(d), undistorted frames; (e), (f), (g) and –(h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k), –(l), histograms of obstacle-to-ground contact points for each polar direction between −90° and 90°; (m), (n), (o) and –(p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and –(d), respectively.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

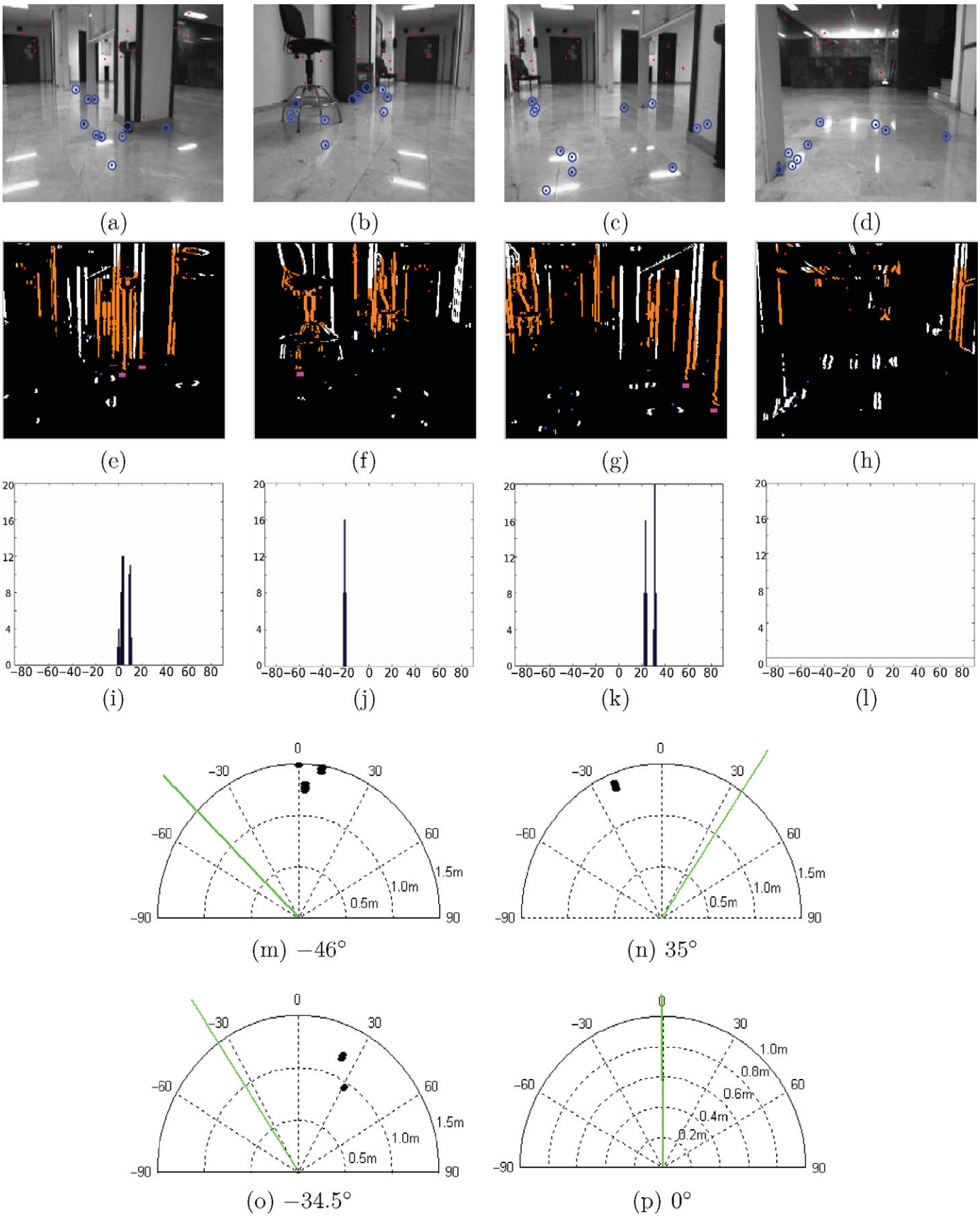


Fig. 10. (Colour online) Scenario 3. Experiment 3: (a), (b), (c) and –(d), undistorted frames; (e), (f), (g) and –(h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k) and –(l) histograms of obstacle-to-ground contact points for each polar direction between −90° and 90°; (m), (n), (o) and –(p), local occupancy map with the resulting steering vector, for images (a), (b), (c) and –(d), respectively.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

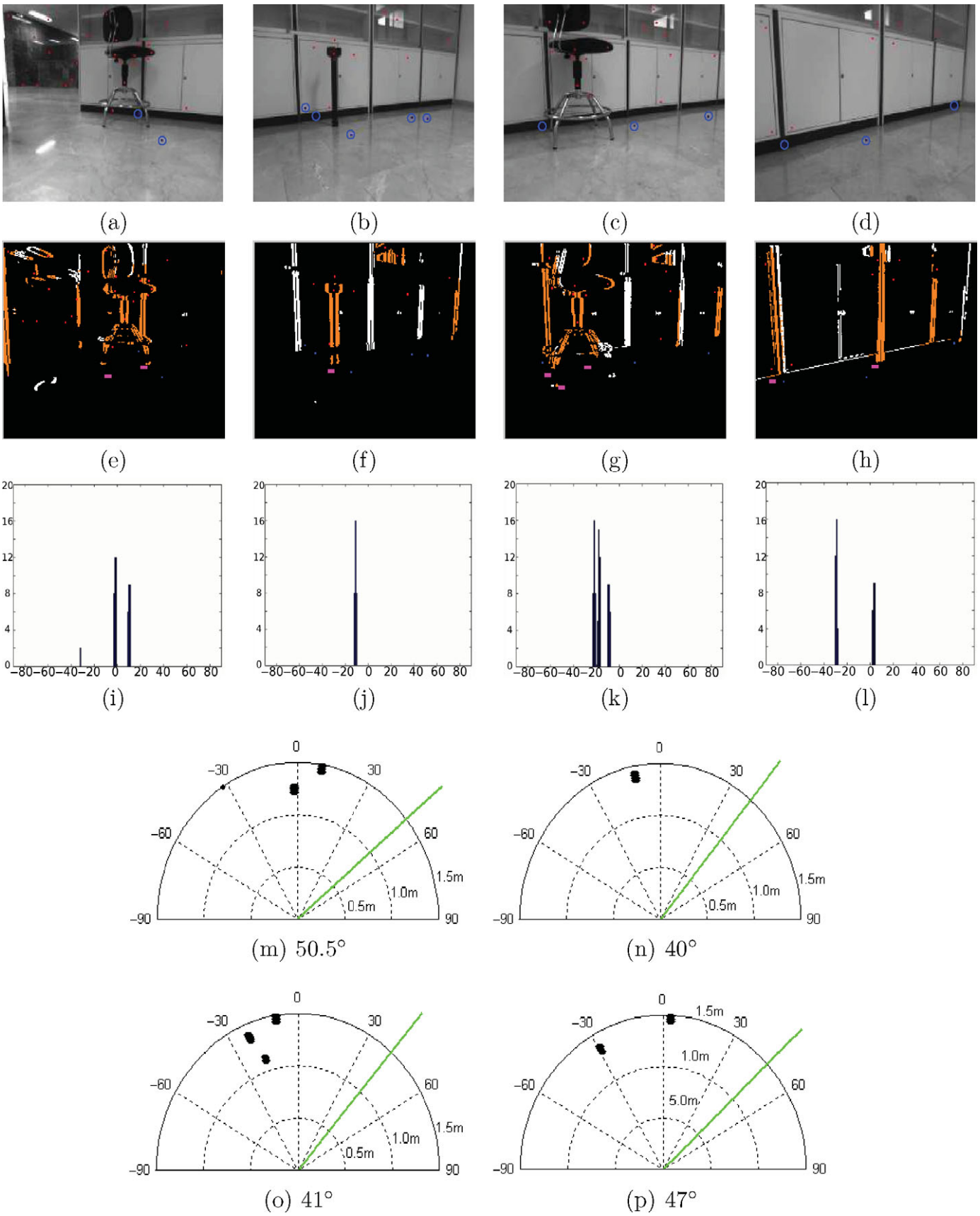


Fig. 11. (Colour online) Scenario 3. Experiment 4: (a), (b), (c), –(d), undistorted frames; (e), (f), (g) and –(h), corresponding edge maps with obstacle borders highlighted in orange; (i), (j), (k), –(l), histograms of obstacle-to-ground contact points for each polar direction between -90° and 90°. (m), (n), (o) and –(p), local occupancy map with the resulting steering vector, for images (a), (b), (c), and –(d), respectively.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

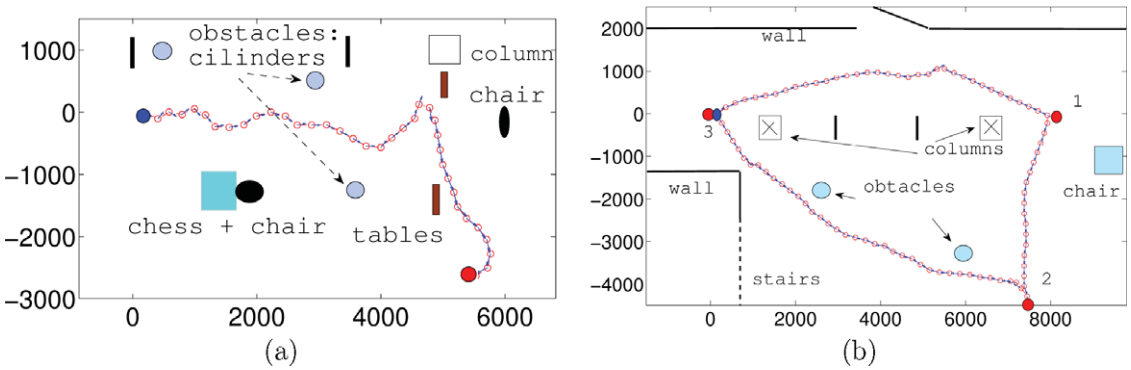


Fig. 12. (Colour online) Experiments in Scenario 3: (a) One fixed goal point. (b) Three consecutive goal points forming a closed loop. Length units expressed in *mm*.

information, and the blue vectors denote the steering orders computed by the navigation algorithm at each position.

In all these aforementioned experiments, the robot detected the obstacles and built the corresponding occupancy maps according to the procedures described in the previous sections. However, the process of generating the steering vector was complemented to perform a new and particular navigation strategy (not described in this paper) which combines the most relevant aspects of Bug-T2 (*Traversability and Tenacity*)[2] and ND (*Nearness Diagram*),[25] but adapted to visual systems.

Bug-T2 is particularly useful because: (a) with a relatively sparse set of sensor readings it performs an occupancy map sufficiently adequate to define the free and the occupied zones in the vicinity of the robot; (b) it guarantees the robot escaping from trapping zones and convergence in a reactive solution; and (c) it facilitates portability to other sensor-based systems. However, Bug-T2 is optimized to deal with very local information of the environment, which means that the robot is unlikely to anticipate potential undesired situations, but to escape from them. Using wider ROIs allows Bug-T2 to increase the probability of anticipating trapping zones. Nevertheless, wider ROIs trigger: (a) a higher number of missed space gaps which the robot can pass through, and consequently, (b) possible loss of efficiency in terms of course length and thus in mission execution time.

Nearness diagram permits navigating safely in cluttered environments, detecting all visible gaps between the robot and the target, and determining a route through them as direct as possible, until the goal point is reached. Considering in the ROI all the visible environment, from the robot pose to the target coordinates, considerably increases the range of the gathered data (obstacle and ground points) to deal with in the navigation module. The number of obstacle points included in the qualitative occupancy map increases, covering a wider zone, facilitating the anticipation of undesired circumstances, detecting all visible gaps and choosing more efficient paths toward the goal. However, ND does not guarantee convergence or escaping from trapping zones without applying the *Tenacity*[2] concept.

Figure 12(b) shows three goal points, enumerated according to the order in which they had to be reached. Figure 13(b) plots the trajectory followed by the robot going from the origin to the goal point, defined just behind a

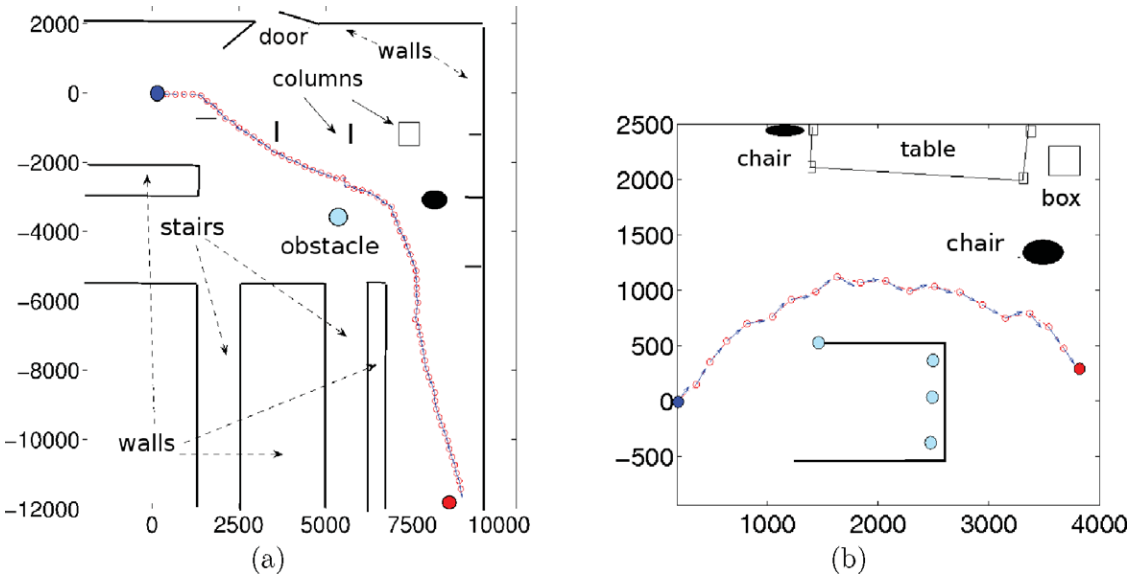


Fig. 13. (Colour online) (a) Scenario 3: One fixed goal point, longer trajectory. (b) Scenario 1: Avoiding a trapping zone. Length units expressed in *mm*.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

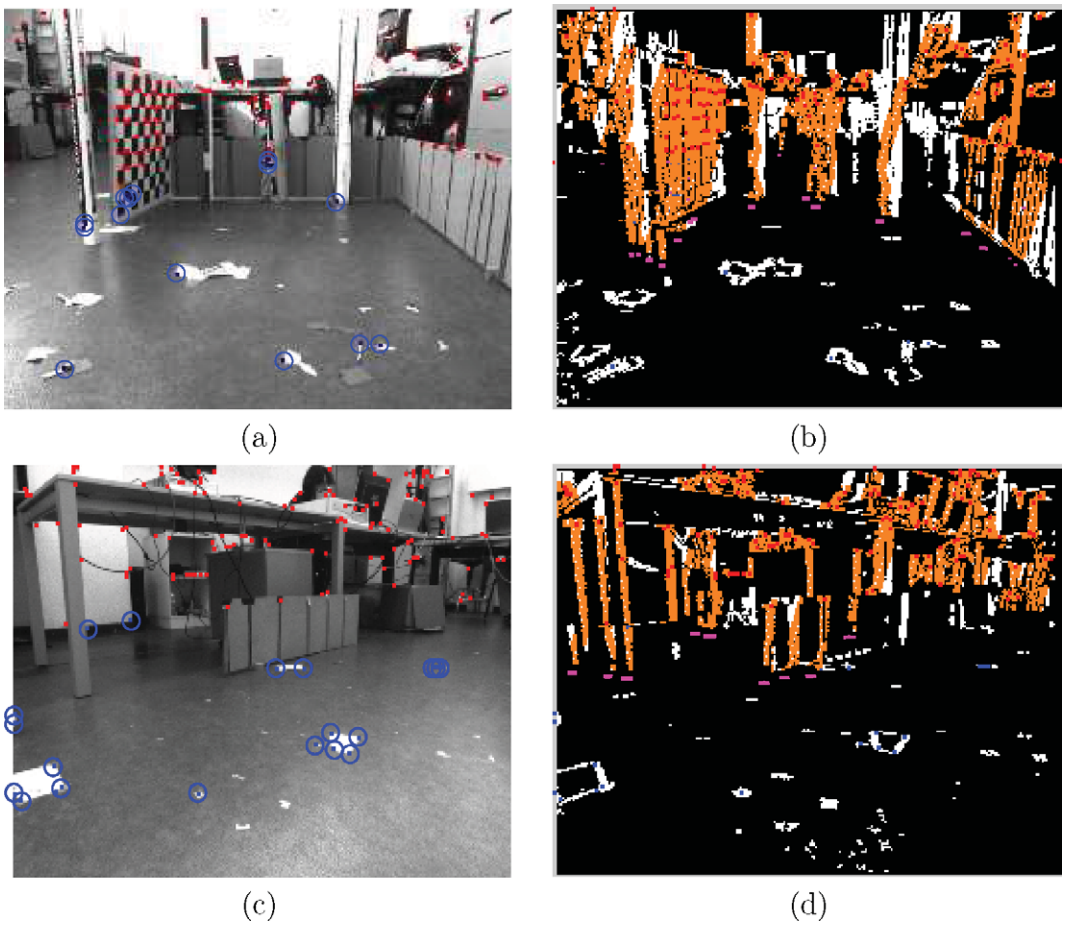


Fig. 14. (Colour online) Avoiding a trapping zone. (a) and (c): Frames with obstacle and ground points. (b) and (d): Edge maps with obstacle boundaries in orange and the obstacle-to-ground contact points in pink.

U-shaped obstacle in which the robot might stay trapped if it is always prone to take the goal direction. The robot was able to foresee the potential trapping zone and to choose the free path beside it. Obviously, if the trapping zone had been too big to be seen in a single or in a couple of consecutive frames, the obstacle would have not been avoided so soon. However, the application of the *Tenacity* concept[2] guarantees to escape from these kind of trapping zones in case the robot gets deeply into one of them.

Figure 14 shows two images gathered during the experiment shown in Fig. 13(b) and the corresponding edge maps.

Figure 15 shows some images recorded during experiments shown in Figs. 12 and 13(a) with the obstacle and ground points and with the corresponding edge maps highlighting the obstacle contours. In all cases, all obstacle-to-ground contact points detected between the current robot pose and the goal point were included in the local occupancy map.

These tests evidence the capacity of the algorithm to provide a robust obstacle detection module suitable for being integrated in reactive navigation architectures.

All the position data used to plot robot trajectories and to calculate the world coordinates of the ground points was obtained, up to now, from the robot wheel odometers. But, as it has been previously mentioned in the Introduction section of this paper, the pose estimation obtained from proprioceptive sensors is unreliable at relatively long distances since it is prone to drift. Visual localization is a good alternative to compute the robot position and thus to reduce the errors inherent to dead reckoning. Next section details the visual robocentric localization algorithm. This is designed to compute the robot position by fusing in an EKF the world coordinates of those features classified as ground points. Ground points had been chosen to be used in the localization process since their 2D position in the ground plane with respect to the robot is known with reasonable accuracy. The aim is to use part of the features gathered in the obstacle detection module to obtain a reliable robot pose data along the whole trajectory and to stabilize the world coordinates of the ground landmarks.

5. Visual Localization

Our proposal is to use the ground points that are obtained from the classification process, and not used until now, to perform localization. As all these points are located on the floor, their z coordinate is known to be zero and their local coordinates $[x, y]$ can be calculated from the corresponding image coordinates and the camera angles. In the context of this section, the ground points will be referred to as landmarks.

The subsequent ground point observations are fused with dead reckoning by means of an EKF. The robot pose, as

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

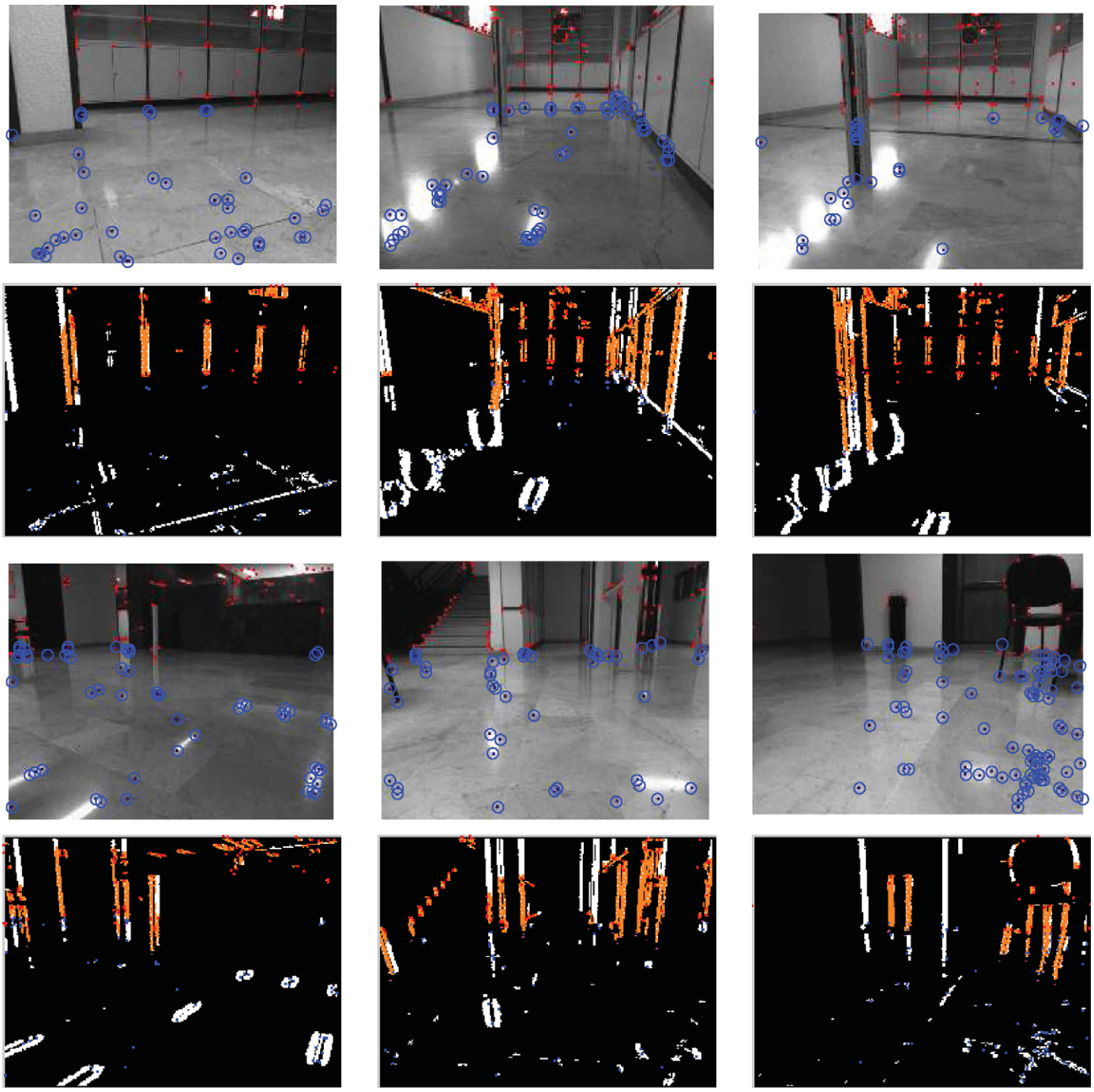


Fig. 15. (Colour online) Experiments on Scenario 3: Images recorded online during the route.

well as the landmarks themselves, is stored in the EKF state vector.

The EKF approach to localization involves the linearization of both sensor and motion models by means of a first-order Taylor series expansion. Due to the errors introduced by these linearizations, the filter ends up providing inconsistent pose estimates. Similar to ref [13], these inconsistencies are particularly problematic in our localization proposal as ground points are removed from the filter when they are not observed. Thus, no loop closure is performed and the uncertainty with respect to a world-fixed reference frame always grows. As large uncertainties (i.e., covariances) are responsible for large EKF linearization errors, one of our goals is to reduce these uncertainties.

Our proposal is based on the robocentric approach.[11] Under this approach, ground points are stored and managed with respect to a coordinate frame locked to the robot. Thus, the ground points that are close to the robot will have lower uncertainty and linearizations will be more valid. Our approach particularly benefits from this idea as only the visible ground points, which are close to the robot, take part in the localization process.

Next, the main steps involved in robocentric localization using ground points are described.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

5.1. Robocentric EKF prediction

The EKF prediction is in charge of estimating the state vector at time step k from the state vector at time step $k-1$ and the dead reckoning information. Let $x_k = N(\hat{x}_k, P_k)$ be the state vector at time k, being $\hat{x}_k$ its mean and P_k its covariance:

$$x_k = \begin{bmatrix} x_W^{R_k} \\ x_L^{R_k} \end{bmatrix}, \tag{12}$$

where $x_W^{R_k} = N(\hat{x}_W^{R_k}, P_W^{R_k})$ denotes the pose of a world-fixed coordinate frame with respect to the robot and $x_L^{R_k} = N(\hat{x}_L^{R_k}, P_L^{R_k})$ represents the positions of the landmarks with respect to the robot. The term landmark refers to the aforementioned ground points. Note that, as this is a robocentric approach, every item in the state vector is represented with respect to the robot. Initially, the state vector will consist only of $x_W^{R_k}$, with zero mean and covariance. Let us denote by $x_{R_k}^{R_{k-1}} = N(\hat{x}_{R_k}^{R_{k-1}}, Q_k)$ the dead reckoning estimate of the robot motion from time step $k-1$ to time step k.

At this point, the next logical step would be to use the dead reckoning information to represent each landmark position, $x_{L_j}^{R_k}$, in the state vector with respect to the current robot pose so that the state vector remains robocentric. This could be easily done as $x_{L_j}^{R_k} = \ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}}$, where $\ominus$ and $\oplus$ are the inversion and compounding operators, widely used in stochastic mapping and SLAM.[11] However, as dead reckoning is likely to be the less precise component in the system, performing this transformation may introduce significant linearization errors. Accordingly, the proposal in the robocentric approach is to delay this composition until the robot motion has been improved in the EKF update step. To this end, during the prediction step, the state vector comprises the state vector in the previous time step plus the dead reckoning estimate as an independent element:

$$\hat{x}_k^- = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{x}_{R_k}^{R_{k-1}} \end{bmatrix} \quad P_k^- = \begin{bmatrix} P_{k-1} & 0 \\ 0 & Q_k \end{bmatrix}, \tag{13}$$

where $x_k^- = N(\hat{x}_k^-, P_k^-)$ denotes the predicted state vector.

5.2. Robocentric EKF update

By matching the new observations against the corresponding landmarks in the state vector, the filter performs the measurement update. As stated previously, the vision modules not only provide the landmark coordinates, but also their associations between consecutive frames and, thus, between the observed and the ones stored in the state vector. Thus, from the EKF point of view, the data association is assumed to be solved. Let us assume that at time step k a set of M observed ground points has been associated with M landmarks previously included in the state vector. Let z_k^i denote the ith ground point observed at time step k, which is known to be associated with the landmark $x_{L_j}^{R_{k-1}}$, stored in the state vector.

In the EKF context, the observation function $h_{j,k}^i$ is in charge of predicting the measurement z_k^i from the state vector. Accordingly, $h_{j,k}^i$, is:

$$z_k^i \simeq h_{j,k}^i(\hat{x}_k^-) = \ominus \hat{x}_{R_k}^{R_{k-1}} \oplus \hat{x}_{L_j}^{R_{k-1}}. \tag{14}$$

The observation matrix $H_{j,k}^i$ which is the Jacobian matrix of $h_{j,k}^i$, is the following:

$$H_{j,k}^i = \left.\frac{\partial h_{j,k}^i}{\partial x_k^-}\right|_{\hat{x}_k^-} = \left.\left[\frac{\partial h_{j,k}^i}{\partial x_W^{R_{k-1}}}, \frac{\partial h_{j,k}^i}{\partial x_{L1}^{R_{k-1}}} \cdots \frac{\partial h_{j,k}^i}{\partial x_{Lj}^{R_{k-1}}} \cdots \frac{\partial h_{j,k}^i}{\partial x_{LN}^{R_{k-1}}}, \frac{\partial h_{j,k}^i}{\partial x_{R_k}^{R_{k-1}}}\right]\right|_{\hat{x}_k^-}. \tag{15}$$

It is easy to see that all the terms in the previous expression are zero except the two terms that depend on the landmark currently observed and on the dead reckoning estimate. The former can be obtained as follows:

$$\begin{aligned} \left.\frac{\partial h_{j,k}^i}{\partial x_{L_j}^{R_{k-1}}}\right|_{\hat{x}_k^-} &= \left.\frac{\partial \left(\ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}}\right)}{\partial x_{L_j}^{R_{k-1}}}\right|_{\hat{x}_k^-} \\ &= J_{2\oplus}\left\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_{L_j}^{R_{k-1}}\right\}, \end{aligned} \tag{16}$$

where $J_{2\oplus}$ is the second Jacobian matrix of the compounding operator, as described in ref. [11]. The second of the mentioned nonzero terms is:

$$\begin{aligned} \left.\frac{\partial h_{j,k}^i}{\partial x_{R_k}^{R_{k-1}}}\right|_{\hat{x}_k^-} &= \left.\frac{\partial \left(\ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}}\right)}{\partial x_{R_k}^{R_{k-1}}}\right|_{\hat{x}_k^-} \\ &= \left.\frac{\partial \left(\ominus x_{R_k}^{R_{k-1}} \oplus x_{L_j}^{R_{k-1}}\right)}{\partial \left(\ominus x_{R_k}^{R_{k-1}}\right)} \cdot \frac{\partial \left(\ominus x_{R_k}^{R_{k-1}}\right)}{\partial x_{R_k}^{R_{k-1}}}\right|_{\hat{x}_k^-} \end{aligned} \tag{17}$$

$$= J_{1\oplus}\left\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_{L_j}^{R_{k-1}}\right\} J_{\ominus}\left\{\hat{x}_{R_k}^{R_{k-1}}\right\}, \tag{18}$$

where $J_{1\oplus}$ is the first Jacobian matrix of the compounding operator and $J_{\ominus}$ is the Jacobian matrix of the inversion, also described in ref. [11].

The measurement vector z_k, observation function h_k, and observation matrix H_k can be constructed using the M values from z_k^i, $h_{j,k}^i$, $H_{j,k}^i$, and C_k, where $C_k = \{<\alpha_1, \beta_1>, <\alpha_2, \beta_2>, \cdots, <\alpha_M, \beta_M>\}$ denotes the set of data associations so that the measurements $z_k^{\alpha_1} \cdots z_k^{\alpha_M}$ are associated with the landmarks $L_{\beta_1} \cdots L_{\beta_M}$, respectively:

$$z_k = \begin{bmatrix} z_k^{\alpha_1} \\ \vdots \\ z_k^{\alpha_M} \end{bmatrix}, \quad h_k = \begin{bmatrix} h_{\beta_1,k}^{\alpha_1} \\ \vdots \\ h_{\beta_M,k}^{\alpha_M} \end{bmatrix}, \quad H_k = \begin{bmatrix} H_{\beta_1,k}^{\alpha_1} \\ \vdots \\ H_{\beta_M,k}^{\alpha_M} \end{bmatrix}. \tag{19}$$

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

The update step can be performed by applying the EKF update equations as follows:

$$\begin{gathered} K = P_k^- H_k^T \left(H_k P_k^- H_k^T + R_k\right)^{-1}, \\ \hat{x}_k^+ = \hat{x}_k^- + K(z_k - h_k(\hat{x}_k^-)), \\ P_k^+ = (I - K H_k) P_k^-, \end{gathered} \tag{20}$$

where R_k is a block diagonal matrix containing the error covariances associated to each measurement and $x_k^+ = N(\hat{x}_k^+, P_k^+)$ denotes the state vector after the EKF update.

5.3. Robocentric composition and state augmentation

At this point, the robot motion $x_{R_k}^{R_{k-1}}$, which was included into the state vector in the prediction step, has been improved. Thus, this improved robot motion can now be used to arrange the state vector so that it remains robocentric. This is the so called composition step. The resulting state vector is the following:

$$x_k = \begin{bmatrix} \ominus x_{R_k}^{R_{k-1}} \oplus x_W^{R_{k-1}} \\ \vdots \\ \ominus x_{R_k}^{R_{k-1}} \oplus x_{LN}^{R_{k-1}} \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_N \end{bmatrix}, \tag{21}$$

where $x_W^{R_{k-1}}$ is the pose of the world-fixed coordinate frame with respect to the robot after the EKF update, all $x_{LN}^{R_{k-1}}$ correspond to the landmarks pose with respect to the robot also after the EKF update, and all the f_i have been introduced just to ease notation in further explanations. It is straightforward to derive the expressions for the mean of the state vector. In order to obtain the covariance of x_k, the following Jacobian matrix has to be computed:

$$J \doteq \left.\frac{\partial x_k}{\partial x_k^+}\right|_{\hat{x}_k^+} = \left.\begin{bmatrix} \frac{\partial f_0}{\partial x_W^{R_{k-1}}} & \cdots & \frac{\partial f_0}{\partial x_{LN}^{R_{k-1}}} & \frac{\partial f_0}{\partial x_{R_k}^{R_{k-1}}} \\ \vdots & \ddots & \vdots & \\ \frac{\partial f_N}{\partial x_W^{R_{k-1}}} & \cdots & \frac{\partial f_N}{\partial x_{LN}^{R_{k-1}}} & \frac{\partial f_N}{\partial x_{R_k}^{R_{k-1}}} \end{bmatrix}\right|_{\hat{x}_k^+}. \tag{22}$$

The nonzero terms in this expression are computed similar to Eqs. (16) and (18):

$$\left.\frac{\partial f_i}{\partial x_i^{R_{k-1}}}\right|_{\hat{x}_k^+} = J_{2\oplus}\left\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_i^{R_{k-1}}\right\}, \tag{23}$$

$$\left.\frac{\partial f_i}{\partial x_{R_k}^{R_{k-1}}}\right|_{\hat{x}_k^+} = J_{1\oplus}\left\{\ominus \hat{x}_{R_k}^{R_{k-1}}, \hat{x}_i^{R_{k-1}}\right\} J_{\ominus}\left\{\hat{x}_{R_k}^{R_{k-1}}\right\}, \tag{24}$$

where $\hat{x}_i^{R_{k-1}}$ denotes $\hat{x}_W^{R_{k-1}}$ when $i = 0$ and $\hat{x}_{L_i}^{R_{k-1}}$ when $i > 0$. Using this Jacobian, the covariance is updated as $P_k = J \cdot P_k^+ \cdot J^T$.

Finally, the observed ground points that have no corresponding landmark in the state vector are considered to be new landmarks and included in the state vector. Because of the robocentric approach, adding new landmarks consists, simply, in including them in the state vector exactly as they are provided by the vision modules. Also at this point, the landmarks in the state vector that have not been observed during a certain time (i.e., three frames) are marginalized out and removed.

When needed, the robot pose with respect to the world-fixed coordinate frame can be recovered easily from the state vector as $x_{R_k}^W = \ominus x_W^{R_k}$.

5.4. Experimental results: localization

Concerning localization, the same robot Pioneer 3DX was used in the same scenarios as for the navigation experiments, and with exactly the same operative conditions, in terms of focal distance, frame rate, camera intrinsic parameters, camera height and tilt. All the length units shown in the incoming experiments are expressed in meters.

The camera motion between consecutive frames was obtained from the robot odometry and the relative position of the camera with respect to the robot center.

In the implementation, ground features are continuously tracked while they stay in the camera field of view. When a ground feature has not been seen during three frames, it is removed from the state vector, and when a new feature is fetched it is included in the filter. The system always tries to ensure a minimum number of observed landmarks in the state vector in order to reduce the computation time as much as possible.

Several experiments were conducted both indoors and outdoors. Indoor experiments have been performed both inside the lab and throughout corridors in a university building, making the robot follow several different trajectories. The outdoor experiments were performed in the university campus. In these environments, the rough terrain was responsible for bad odometry estimates.

Finding a proper ground truth became necessary to quantitatively evaluate the obtained pose estimates. For the experiments performed inside the lab, the ground truth was calculated from the images captured by two calibrated, world-fixed, wide angle cameras facing perpendicularly to the ground plane at a height of 3 m (see Fig. 16). The experiments were recorded using these two cameras and the ground truth trajectory was recovered from the gathered images.

A different approach was adopted for outdoor scenarios since no cameras were available to generate a ground truth. In these cases, the Pioneer 3-DX ultrasonic array was activated and the distances it provided were recorded. Then, the ultrasonic data were plotted according to the trajectory being evaluated. The better the plotted sonar data coincides with the reality, the better is the robot trajectory used to plot it. Thus, although no ground truth is available in outdoor tests, sonar data make it possible to qualitatively evaluate trajectories by visual inspection.

For all and each of the trajectories run inside the lab, five different levels of odometry noise were tested (namely, from now on, experiments 1–5) in order to check our approach depending on the dead reckoning error and to observe the obtained pose estimates in front of severe odometric errors. The odometric noise was additive, zero-mean Gaussian with covariance $\Sigma = diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2)$, where σ_x and σ_y ranged from 0.0032 m in the first experiment to 0.032 m in the

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press



Fig. 16. (Colour online) World-fixed cameras for ground truth calculation, on top of the picture

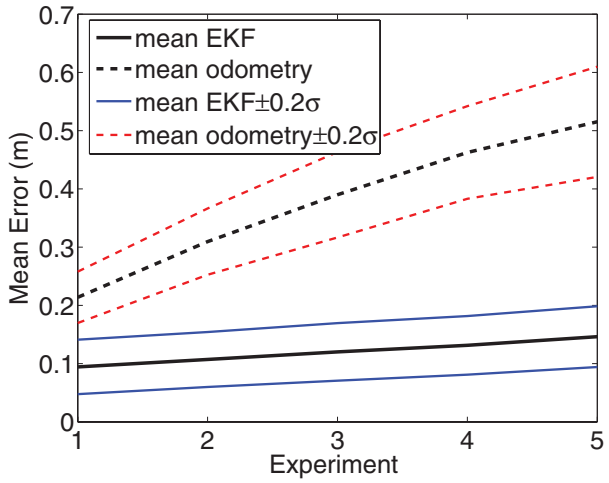


Fig. 17. (Colour online) Mean and Mean $\pm 0.2\sigma$ for the trajectory errors.

fifth experiment, and σ_θ ranged from 0.0032 rad in the first experiment to 0.032 rad in the fifth. For each one of these tests, with a certain trajectory and a certain noise level, 100 runs were executed in order to obtain statistically significant results. In all tests and for each robot pose estimate, the trajectory error was computed, both for dead reckoning and for our approach. The trajectory error is defined as the distance between the robot pose under evaluation and its corresponding ground truth.

Figure 17 shows a comparison of the mean errors corresponding to the corrupted odometry with those corresponding to our approach, as a function of the five different noise levels and computed for all the trajectories. As can be observed, the mean error when using our approach, labeled as EKF, is considerably below the dead reckoning error, and so does the standard deviation of the error. This correction is much more evident as the odometry noise grows. In the figure, σ has been scaled down to 20% to provide a more clear plot.

Figure 18 shows some data corresponding to two of the trajectories conducted in the laboratory and used to perform the plot of Fig. 17. The plots show the trajectories with different levels of odometry noise and the evolution of the corresponding trajectory error with time. Figure 19 shows some images captured from the robot during these trajectories. All images show the KLT features classified as obstacle points in red and those classified as ground points as circles in blue.

Figure 20 shows images and the plot of a rectilinear trajectory conducted along the center of a 2.50 m wide corridor with a high textured floor. This is one of the aforementioned indoor scenarios where no ground truth was available. Wheels were slightly deflated, resulting in odometric error, but no synthetic noise was added in this case. Notice how odometry trajectory significantly deviates from the forward direction whilst the EKF trajectory approximates better the right ahead course.

Figure 21 shows some data concerning two different outdoor experiments. Both tests consisted in moving the robot parallel to the walls of different buildings. It is important to emphasize that the proximity of walls is only needed to collect the aforementioned sonar data, which is only used for human inspection. In these experiments, stones, grass, and small clumps of soil over the floor influenced the robot odometry estimates.

The top row shows, for both experiments, the EKF trajectory in black with the 2σ bound uncertainty ellipses in the picture on the left, the odometry trajectory in red, and the sonar data with respect to every EKF robot position in blue. The middle row shows two images captured from the robot showing the KLT obstacle points in red and the features classified as ground circled in blue. The bottom row shows two images of the whole scene during the test. Notice how in outdoor environments, even if only a reduced number of landmarks is detected and tracked, the pose estimation is good.

Although no ground truth was available in these experiments, the plotted sonar data give a clear idea regarding the correctness of the estimated trajectories. As it can be observed in the bottom row pictures, the building walls and angles are straight, and so it is reflected by the sonar readings, which are plotted using the robocentric pose estimates.

The exposed experiments evidence the potential of the presented localization approach. It gives to the navigation system exposed in this work the possibility of using part of the image features for improving the robot pose provided by dead-reckoning.

6. Conclusions and Future Work

This paper focuses on the use of image features to perform reactive navigation and robocentric localization using a single camera. The former task is based on features classified as obstacle points, and the latter task is performed from features classified as ground points. The main advantages of our approach are basically: (a) the algorithm is able

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

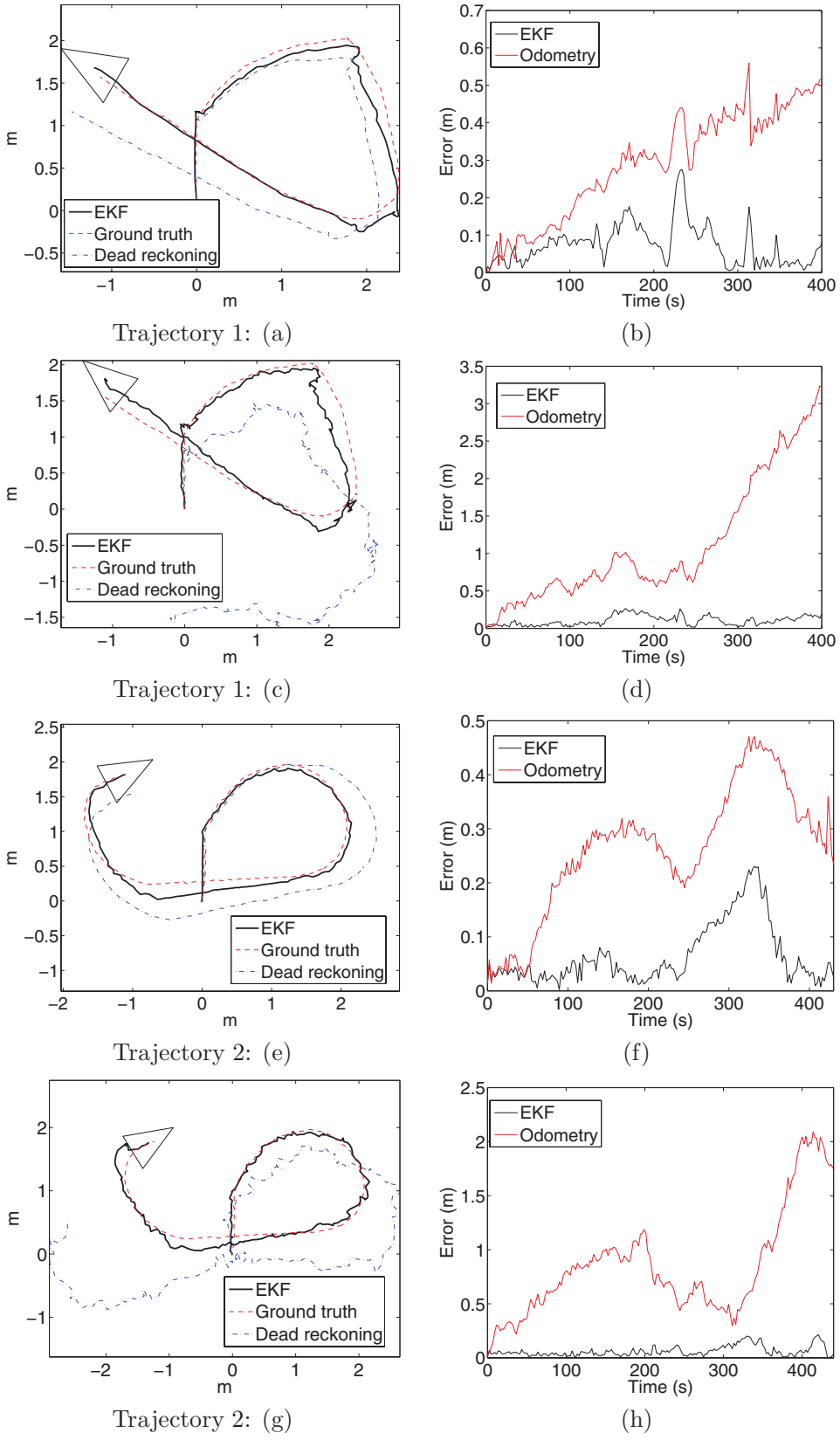


Fig. 18. (Colour online) (a), (e) Two different trajectories with $\sigma_x = \sigma_y = 0.0055$ m for the odometry Gaussian noise. (c), (g) Trajectories with $\sigma_x = \sigma_y = 0.031$ m for the odometry Gaussian noise. (b), (d), (f), and (h) Trajectory errors.

to concurrently perform two tasks, obstacle avoidance and localization, using data obtained from a single sensor and reducing both problems to 2D problems; (b) the obstacle detection algorithm employs a relatively sparse set of visual sonar points to qualitatively define which are the occupied and the free zones in the vicinity of the robot, without the need of a detailed mapped area; (c) it overcomes scenes with inter-reflections, shadows, highly or poorly textured floors

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

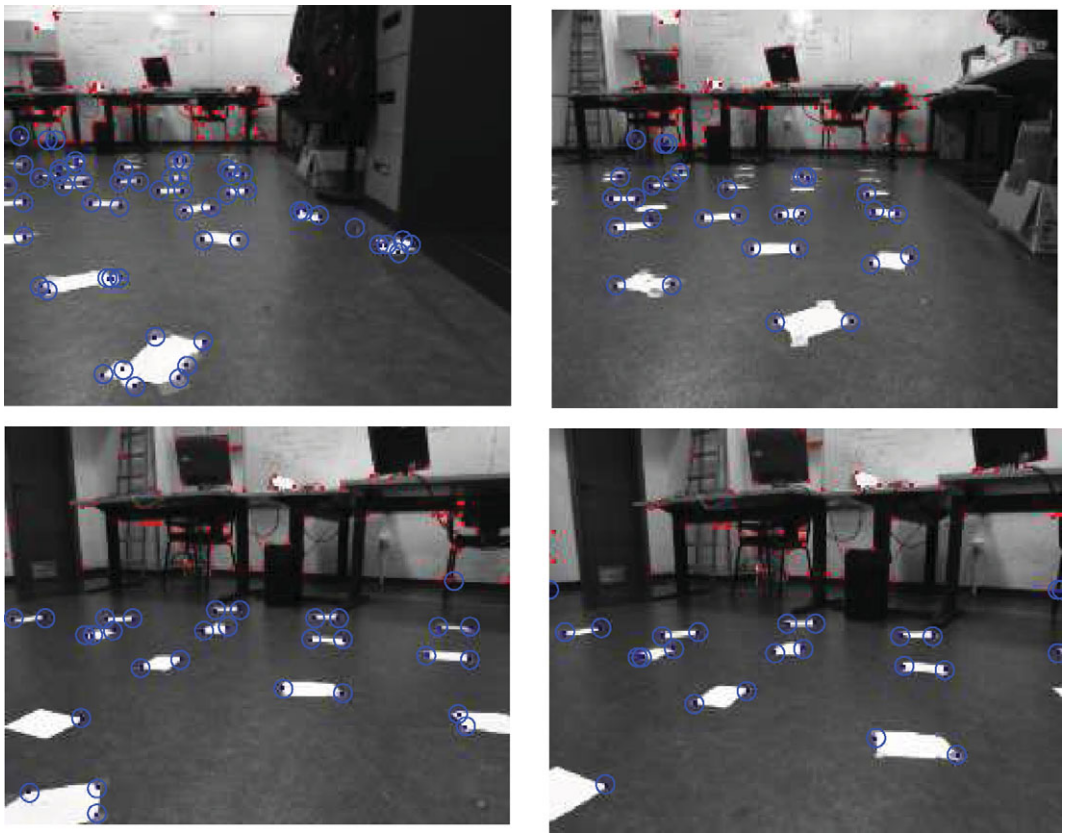


Fig. 19. (Colour online) Images taken from the robot during the trajectories 1 and 2 referred to in Fig. 18.

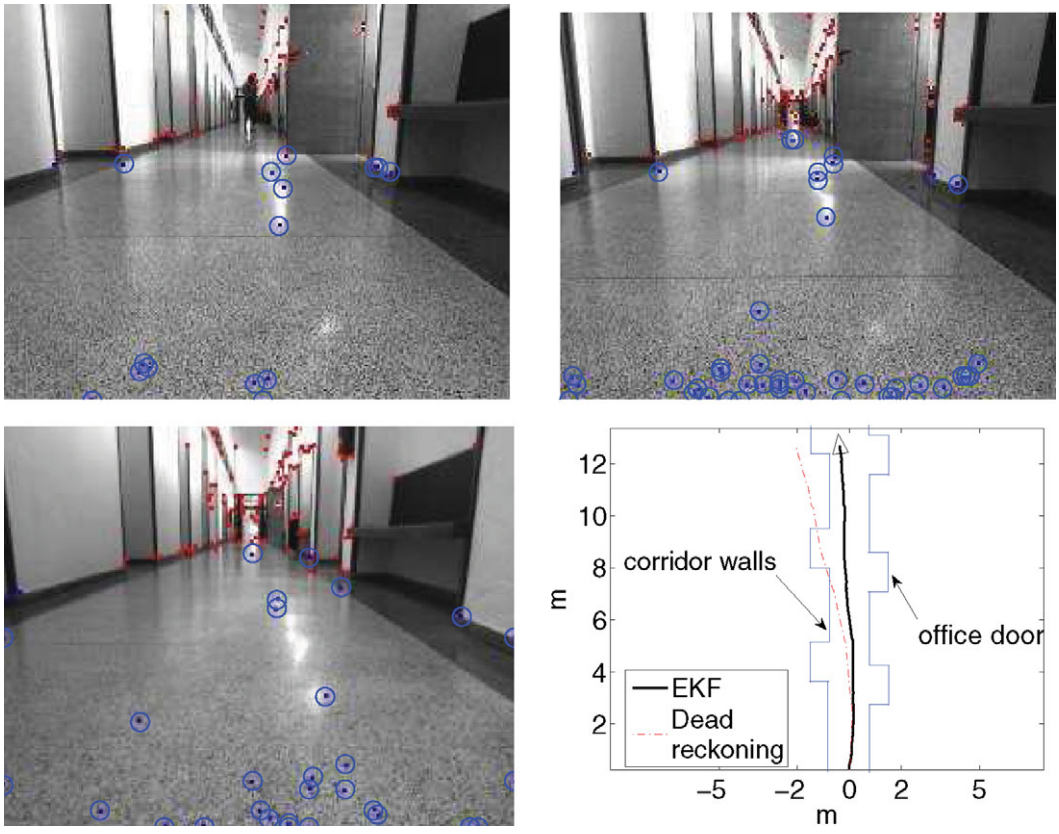


Fig. 20. (Colour online) Images and trajectory on a straight corridor.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

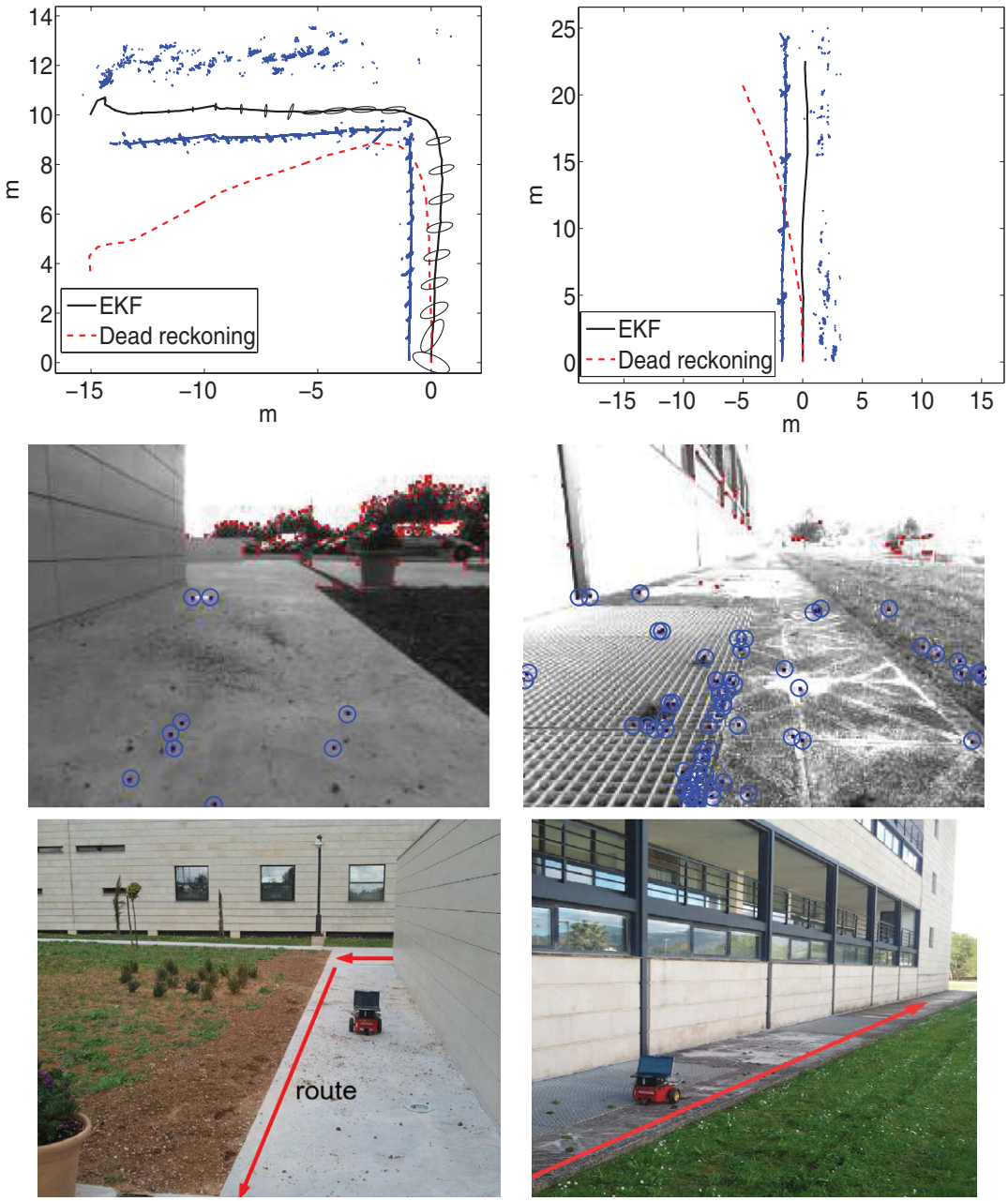


Fig. 21. (Colour online) Trajectories and some images of outdoor experiments.

and scenes with multiple planes; (d) it only processes a limited number of image points, reducing the computation time of the obstacle detection process and the calculation of the steering vector; (e) localization is performed in a robocentric framework to reduce the EKF linearization errors and provides good results also for a reduced number of ground features; and (f) as features are forgotten when they are unseen, the EKF-based localization is performed in bounded, almost constant, execution time.

Experimental results show how errors on the EKF estimated robot poses appear to be much lower than errors on positions given by dead-reckoning.

The navigation and localization algorithms have proved their efficiency online, in different types of scenarios, and with different specific associated problems. However, in order to increase the scope of applicability of our approach it is currently into consideration to investigate two important points: (a) the application of the algorithm in dynamic environments, and (b) lessening the flat ground constraint by incorporating an extension of the IPT to handle nonflat floors.

Although they are preliminary hypothesis, some considerations can be outlined:

(a) The motion of the camera causes the displacement of salient features in consecutive images, changing their image coordinates, but still representing the same scene point. The displacement model of one image point due to the camera motion will not be equal for features corresponding to static points of the environment than for features corresponding to dynamic points of the scene (moving obstacles). For static points, the motion of their image projections should be related with the camera motion. Instead, for points

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

corresponding to moving obstacles, the displacement of their image projections should combine the influence of the camera motion with the effects caused by the obstacle motion itself. Consequently, the challenging goal is the discrimination of image features corresponding to moving obstacles from those corresponding to static points of the environment by means of the analysis and comparison of their respective motion rules or optical flow.

(b) Concerning the application of the algorithm in irregular terrains or with changes of slope: when moving on these irregularities, the camera yaw (θ) and pitch (φ) angles will be altered and the roll angle can be different from 0, directly affecting the computation of the ground point world coordinates. Knowing, by some means (e.g., with an IMU), the rotation suffered by the robot, in roll, pitch and yaw, between two consecutive positions would permit the use of this information to either rectify the images before extracting the features or to adjust the IPT calculation in the second frame in terms of these camera rotations.

Acknowledgment

This work is partially supported by DPI2011-27977-C03-03, FP7-ICT-2009-4 GA-248497 and PTA2011-5077-I.

References

1. M. Agrawal, K. Konolige and M. R. Blas, "Censur E: Center surround extremas or realtime feature detection and matching," *Lecture Notes Comput. Sci.* **5305**(3), 102–115 (2008).
2. J. Antich and A. Ortiz, "Bug-based T2: A New Globally Convergent Potential Field Approach to Obstacle Avoidance," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China (2006).
3. H. Bay, T. Tuytelaars and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Comput. Vis. Image Underst.* **110,** 346–359 (2008).
4. M. Bertozzi and A. Broggi, "GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE Trans. Image Proc.* **7**(1), 62–81 (1998).
5. F. Bonin, A. Ortiz and G. Oliver, "Visual navigation for mobile robots: A survey," *J. Intell. Robot. Syst.* **53**(3), 263–296 (2008).
6. F. Bonin-Font, A. Ortiz and G. Oliver, "Experimental Assessment of Different Feature Tracking Strategies for an IPT-based Navigation Task," *Proceedings of IFAC Intelligent Autonomous Vehicles (IAV)*, Lecce (Italy) (2010).
7. J. Borenstein and I. Koren, "The vector field histogram - fast obstacle avoidance for mobile robots," *J. Robot. Autom.* **7**(3), 278–288 (1991).
8. K. Bowyer, C. Kranenburg and S. Dougherty, "Edge detector evaluation using empirical ROC curves," *Comput. Vis. Image Underst.* **84**(1), 77–103 (2001).
9. A. Burguera, Y. González and G. Oliver, "On the use of likelihood fields to perform sonar scan matching localization. *Springer Auton. Robots* **26**(4), 203–222 (2009).
10. J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.* **8**(6), 679 – 698 (1986).
11. J. A. Castellanos, R. Martínez-Cantín, J. D. Tardós and J. Neira, "Robocentric map joining: Improving the consistency of EKF-SLAM," *Robot. Auton. Syst.* **55**, 21–29 (2007).
12. K. Çelik, S. Chung, M. Clausman and A. K. Somani, "Monocular Vision SLAM for Indoor Aerial Vehicles," *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA (2009) pp. 1566–1573.
13. J. Civera, O. G. Grasa, A. J. Davison and J. M. M. Montiel, "1-Point RANSAC for EKF-Based Structure from Motion," *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA (2009).
14. A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *Int. J. Robot. Res.* **26**(10), 1125–1153 (2007).
15. R. O. Duda and P. Hart, *Pattern Classification and Scene Analysis* (John Wiley and Sons, USA, 1973).
16. J. Fasola and M. Veloso, "Real-Time Object Detection using Segmented and Grayscale Images," *Proceedings of IEEE Int'l Conference on Robotics and Automations (ICRA)*, Orlando, FL, USA (2006) pp. 4088–4090.
17. J. A. Hanley and B. McNeil, "The meaning and use of the area under a Receiver Operating Charateristic (ROC) Curve," *Radiology* **143**(1), 381–395 (1982).
18. C. Harris and M. Stephens, "Combined Corner and Edge Detector," **In**: *Proceedings of the Fourth Alvey Vision Conference*, Manchester, UK, vol. 15 (1988) pp. 147–151.
19. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision* (Cambridge University Press, Cambridge, UK, 2003) ISBN: 0521623049.
20. S. Huang and Dissanayake, "Convergence and consistency analysis for extended kalman filter based SLAM," *IEEE Trans. Robot.* **23**(5), 1036–1049 (2007).
21. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.* **60**(2), 91–110 (2004).
22. G. Ma, S. B. Park, S. Miller-Schneiders, A. Ioffe and A. Kummert "Vision-based Pedestrian Detection - Reliable Pedestrian Candidate Detection by Combining IPM and a 1D Profile," *Proceedings of the IEEE Intelligent Transportation Systems Conference*, Seattle, WA, USA (Sep. 2007) pp. 137–142.
23. M. C. Martin, "Evolving visual sonar: Depth from monocular images," *Pattern Recognit. Lett.* **27**, 1174–1180 (2006).
24. C. Marzban, "The ROC curve and the area under it as performance measures," *Weather Forecast.* **19**(6), 1106–1114 (2004).
25. J. Minguez and L. Montano, "Nearness Diagram (ND) Navigation: Collision avoidance in troublesome scenarios," *IEEE Trans. Robot. Autom.* **20**(1), 45–59 (2004).
26. A. Mourikis, N. Trawny, S. Roumeliotis, A. Johnson, E. Ansar and L. Matthies. "Vision-aided inertial navigation for spacecraft entry, descent and landing," *IEEE Trans. Robot.* **25**(2), 264–280 (2009).
27. A. Ollero, J. Ferruz, F. Caballero, S. Hurtado and L. Merino, "Motion Compensation and Object Detection for Autonomous Helicopter Visual Navigation in the COMETS System," *Proceedings of IEEE Int'l Conference on Robotics and Automation (ICRA)*, New Orleans, USA (2004) pp. 19–24.
28. L. M. Paz, P. Piniés, J. D. Tardós and J. Neira. "Large scale 6DOF SLAM with a stereo camera in hand," *IEEE Trans. Robot.* **24**(5), 946–957 (2007).
29. R. Roberts, H. Nguyen, N. Krishnamurthi and T. Balch, "Memory-based Learning for Visual Odometry," *Proceedings of IEEE Int'l Conference on Robotics and Automation (ICRA)*, Pasadena, CA, USA (2008).
30. R. Rodrigo, M. Zouqi, Z. Chen and J. Samarabandu, "Robust and efficient feature tracking for indoor navigation," *IEEE Trans. Syst., Man and Cybern.* **39**(3), 658–671 (2009).
31. E. Rosten and T. Drummond "Machine Learning for High-Speed Corner Detection," **In**: *Proceedings of the European Conference on Computer Vision (ECCV)*, vol. 1 (2006) pp. 430–443 .
32. A. J. Serrano, E. Soria, J. D. Martin, R. Magdalena and J. Gomez. "Feature Selection Using ROC Curves on Classification Problems," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Barcelona (Spain) (2010) pp. 1–6.
33. J. Shi and C. Tomasi, "Good Features to Track," *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, USA (1994) pp. 593–600.
34. N. Simond and M. Parent, "Obstacle Detection from IPM and Super-Homography," *Proceedings of IEEE International*

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press

Conference on Intelligent Robots and Systems (IROS), California, Sant Diego, USA (Nov. 2007) pp. 4283–4288.
35. A. Talukder and L. Matties, "Real-time Detection of Moving Objects from Moving Vehicles Using Dense Stereo and Optical FLow," **In**: *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan (Oct. 2004) pp. 3718–3725.
36. S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics*, Massachusetts (USA) (The MIT Press, 2005).
37. C. Zhou, Y. Wei and T. Tan, "Mobile Robot Self-Localization Based on Global Visual Appearance Features," **In**: *Proceedings of IEEE Int'l Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan (2003) pp. 1271–1276.

https://doi.org/10.1017/S0263574712000252 Published online by Cambridge University Press