5

A Visual Navigation Strategy Based on Inverse Perspective Transformation

Francisco Bonin-Font, Alberto Ortiz and Gabriel Oliver
University of the Balearic Islands
Spain

1. Introduction

Vision-Based Navigation Techniques can be roughly divided in map-based and mapless systems. Map-based systems plan routes and their performance and are labeled as deliverative, while mapless systems analyze on-line the environment to determine the route to follow (Bonin et al., 2008). Some reactive vision-based systems include the implementation of local occupancy maps that show the presence of obstacles in the vicinity of the robot and try to perform a symbolic view of the surrounding world. The construction of such maps entails the computation of range and angle of obstacles in a particularly accurate manner. These maps are updated on-line and used to navigate safely (Badal et al., 1994) (Goldberg et al., 2002).
Many of the local map-based and visual sonar reactive navigation solutions are vulnerable to the presence of shadows or inter-reflections and they are also vulnerable to textured floors, since they are mostly based on edge computation or on texture segmentation. Solutions based on homography computation fail in scenarios that generate scenes with multiple planes. Some road line trackers based on Inverse Perspective Transformation (*IPT*) need to previously find lines in the image that converge to the vanishing point. Some other *IPT*-based solutions project the whole image onto the ground, increasing the computational cost.
This chapter presents a new navigation strategy comprising obstacle detection and avoidance. Unlike previous approaches, the one presented in this chapter avoids back-projecting the whole image, presents a certain robustness to scenarios with textured floors or inter-reflections, overcomes scenes with multiple different planes and combines a quantitative process with a set of qualitative rules to converge in a robust technique to safely explore unknown environments. The method has been inspired on the visual sonar-based reactive navigation algorithms and implements a new version of the Vector Field Histogram method (Borenstein & Koren, 1991) but here adapted for vision-based systems.
The complete algorithm runs in five steps: 1) first, image main features are detected, tracked across consecutive frames, and classified as obstacle or ground using a new algorithm based on *IPT*; 2) the edge map of the processed frames is computed, and edges comprising obstacle points are discriminated from the rest, emphasizing the obstacle boundaries; 3) range and angle of obstacles located inside a Region of Interest (*ROI*), centered on the robot and with a fixed radius, are estimated computing the orientation and distance of those obstacle points that are in contact with the floor; 4) a qualitative occupancy map is performed with the data computed in the previous step; and 5) finally, the algorithm computes a vector which steers the robot towards world areas free of obstacles.

This chapter is structured as follows: related work is presented in section 2, the method is outlined in Sections 3, 4 and 5, experimental results are exposed and discussed in Section 6, and finally, conclusions and forthcoming work are given in Sections 7 and 8, respectively.

2. Related Work

2.1 Feature Detection and Tracking

Visual techniques for detecting and tracking significant elements of the scene, so called features, have been extensively improved over the last years and used for localization and/or navigation purposes. Significant scene features are categorized using the concept of distinctiveness. The distinctiveness notion is related with the size of the window used to define the neighborhood captured by the feature descriptor and with the amount and type of information processed and extracted from the feature neighborhood. Distinctive features can be characterized by a vector of values including location, scale, orientation, intensities and gradients in several directions. Distinctive features can be tracked without using a motion model and more accurately than the nondistinctive features.

Harris (Harris & Stephens, 1988) and Shi and Tomasi (Shi & Tomasi, 1994) are early and fast techniques to find and/or track little discriminative features. Zhou and Li (Zhou & Li, 2006) detected features located on the ground plane grouping all coplanar points that have been found applying the Harris corner detector. Nister *et al.* estimated the motion of mobile robots tracking nondistinctive Harris corners. Saeedi *et al* (Saeedi et al., 2006) presented a stereo vision-based 3-D trajectory tracker for localization of mobile robots in unexplored environments. This work proposed a new corner detector to extract image features. Features were matched in consecutive frames minimizing the mean-squared error and searching for the highest cross correlation as a similarity measure. Rabie *et al* (Rabie et al., 2001) used a feature-based tracking approach to match in consecutive images nondistinctive points detected using the Shi and Tomasi algorithm. The algortihm was applied for traffic flow speed estimation in a system addressed to automatically monitor traffic conditions.

Lowe (Lowe, 2004) developed the Scale Invariant Feature Transform (SIFT) method to extract high discriminative image features. SIFT is robust to image scaling, rotation, illumination changes or camera view-point changes. Stephen *et al.* performed global simultaneous localization and mapping in mobile robots tracking distinctive visual SIFT landmarks in static environments (Stephen et al., 2005). Rodrigo *et al.* (Rodrigo et al., 2009) combined a set of selected relevant distinctive SIFT features with a collection of nondistinctive features to perform a robust navigation algorithm based on feature or landmark tracking. SIFT features were used to compute the homographies of the different planar structures in the scene. Nondistinctive features were used to refine the motion model. Once the homographies of the different planes were computed it was possible to determine the motion of nondistinctive features across consecutive images.

Mikolajczyk and Schmid (Mikolajczyk & Schmid, 2005) compared the performance of different descriptors for image local regions. Experimental results of different matching approaches used to recognize the same region in different viewing conditions showed that SIFT yields the best performance in all tests.

2.2 Inverse Perspective Transformation-based Obstacle Detection

To detect obstacles, Mallot *et al* (Mallot et al., 1991) analyzed variations on the optical flow computed over the Inverse Perspective Mapping (*IPM*) of consecutive images. Bertozzi and Broggi (Bertozzi & Broggi, 1998) projected two stereo images onto the ground applying the

IPM concept. Then, they subtracted both projections to generate a non-zero pixel zone that evidenced the presence of obstacles. Batavia *et al* (Batavia et al., 1997) used the *IPT* and the camera ego-motion to predict future frames and compare them with the corresponding new real frames. Differences between the predicted and real images showed the presence of obstacles. The system was designed to detect vehicles in the blind spot of the cars rear-view mirror. Shu and Tan (Shu & Tan, 2004) also employed the *IPM* to detect road lanes for self-guided vehicles. Ma *et al* (Ma et al., 2007) presented an automatic pedestrian detection algorithm based on *IPM* for self-guided vehicles. The system predicted new frames assuming that all image points laid on the floor. The distorted zones of the predicted image corresponded to objects. Simond combined the *IPM* with the computation of the ground plane super-homography from road lines to discriminate obstacles from road in an autonomous guided vehicle application (Simond & Parent, 2007).

2.3 Visual Sonar

Some other researchers explored the idea of Visual Sonar. Visual Sonar techniques provide depth and orientation of elements in the scene, using visual sensors in an analogous way to ultrasonic sensors (Horswill, 1994) (Choi & Oh, 2005) (Martin, 2006). In some cases, the application of the visual sonar concept results in the computation of local occupancy maps (Lenser & Veloso, 2003) (Fasola et al., 2005).

3. Obstacle Detection: Overall Description

3.1 The Inverse Perspective Transformation

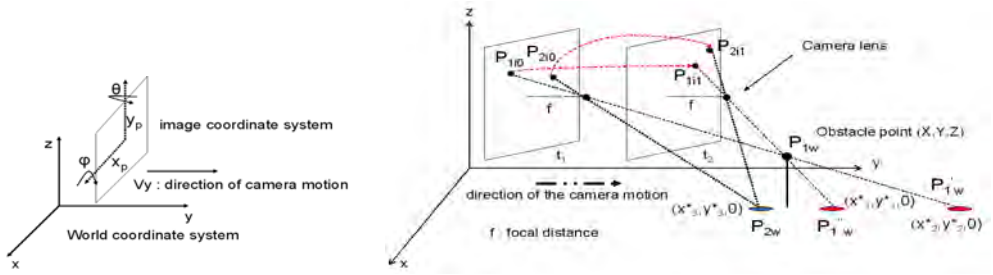
The Perspective Transformation is the method for mapping three-dimensional points onto a two-dimensional plane called the plane of projection. This Perspective Transformation models the process of taking a picture, being the image plane the plane where the spatial scene is projected. The line that connects 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 Inverse Perspective Transformation. 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 modelled assuming a pinhole camera and a flat ground (Duda & Hart, 1973) (Hartley & Zisserman, 2003). Three coordinate systems are involved: the world, the camera and the image coordinate systems. The linear mapping between world to image points, both expressed in homogeneous coordinates, can be written as (Hartley & Zisserman, 2003):

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

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 from world to the camera coordinates.

It is possible to compute the scene point world coordinates corresponding to an image point knowing either the distance between the camera 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 expressions



(a) (b)

Fig. 1. (a) Coordinate frame conventions. (b) The IPT-based obstacle detection.

in closed form to calculate the world coordinates for points lying on the floor are (Duda & Hart, 1973):

$$x^* = X_0 - \frac{Z_0 x_p cos\theta + (y_p sin\varphi - f cos\varphi)(Z_0 sin\theta)}{y_p cos\varphi + f sin\varphi} \tag{2}$$

$$y^* = Y_0 - \frac{Z_0 x_p sin\theta - (y_p sin\varphi - f cos\varphi)(Z_0 cos\theta)}{y_p cos\varphi + f sin\varphi} \tag{3}$$

where (x^*,y^*) are the ground point world coordinates, (X_0, Y_0, Z_0) are the camera world coordintes at the moment in which the frame has been taken, and θ and φ are the camera yaw and pitch angles, respectively. Coordinate system conventions and notation are illustrated in figure 1-(a).

3.2 Discrimination Between Obstacle and Ground Points

The (x^*,y^*) values computed by means of equations (2) and (3) for an image point that corresponds to a point lying on the floor are equal when they are computed from two consecutive images, and exactly correspond to the point (x, y) world coordinates. However, for an image point that belongs to an object protruding vertically from the floor, the assumption $z = 0$ is incorrect and the (x^*,y^*) values turn out to be different when they are calculated from two consecutive images, and different to the object point real (x, y) world coordinates. Hence, one can distinguish if the point belongs to an obstacle or to the floor assuming $z = 0$ and comparing the distance between the resulting (x^*,y^*) values calculated for two consecutive images:

$$(discrepancy) \quad D = \sqrt{(x_2^* - x_1^*)^2 + (y_2^* - y_1^*)^2} \Rightarrow \begin{cases} if D > \beta \Rightarrow \text{obstacle}, \\ if D \leq \beta \Rightarrow \text{ground}. \end{cases} \tag{4}$$

where (x_1^*,y_1^*) and (x_2^*,y_2^*) correspond to 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 figure 1-(b). Two frames of a scene are taken at instants t_1 and t_2. Point P_{2w} is on the floor. Its projection into the image plane at instants t_1 and t_2 generates, respectively, the image points P_{2i0} and P_{2i1}. The Inverse Transformation of P_{2i0} and P_{2i1} generates a unique point P_{2w}. P_{1w} is an obstacle point. Its projection into the image plane at t_1 and t_2 generates, respectively, the points P_{1i0} and P_{1i1}. However, the Inverse Transformation of P_{1i0} and P_{1i1} back to the world assuming $z = 0$ (e.g. projection onto the ground plane), generates two different points on the ground, namely, P'_{1w} and P''_{1w}.

3.3 Feature Detection and Tracking

The first key step of the algorithm is to detect a sufficiently large and relevant set of image features and match them across consecutive images. SIFT features (Lowe, 2004) have been chosen as the features to track because of their robustness to scale changes, rotation and/or translation as well as changes in illumination and view point. Wrong correspondences between points in consecutive frames are filtered out in four steps, using RANSAC and imposing the epipolar constraint: $x'_p F x_p = 0$, where x'_p and x_p are the point image coordinates in two consecutive frames, and F is the fundamental matrix (Hartley & Zisserman, 2003):

1. Compute SIFT features and match them in the two consecutive images,
2. starting with 7 correspondences randomly generated, compute the Fundamental Matrix taking the one with the lowest standard deviation of inliers (RANSAC robust estimation),
3. re-compute F from the correspondences classified as inliers,
4. update the correspondences using the updated F.

Steps 3 and 4 can be iterated until the number of correspondences is stable.

4. Obstacle Detection: Enhanced Feature Classification

4.1 Direct Identification of Some Obstacle Points

The set of scene points that map to a given image point can be written as (Duda & Hart, 1973):

$$p = p_l + \lambda(p_p - p_l) \tag{5}$$

where p is the scene object point (x, y, z), p_l is the camera center (X_0, Y_0, Z_0), p_p is the image point corresponding to the scene point p, expressed in the world coordinate system,

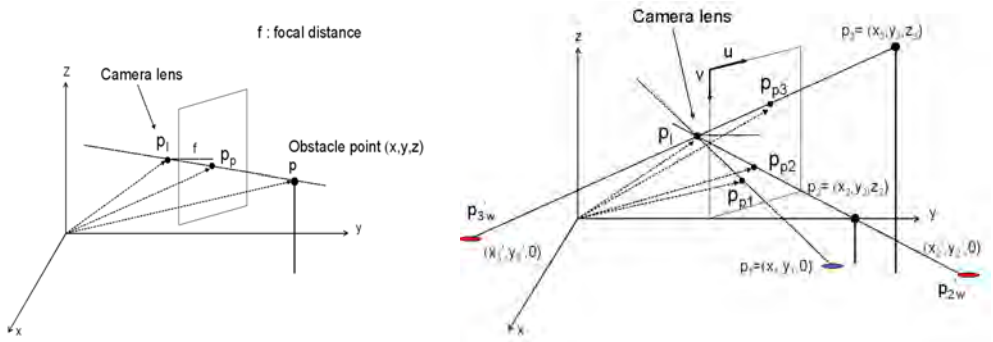
$$p_p = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} x_p cos\theta - f cos\varphi sin\theta + y_p sin\varphi sin\theta \\ x_p sin\theta + f cos\varphi cos\theta - y_p sin\varphi cos\theta \\ f sin\varphi + y_p cos\varphi \end{bmatrix}, \tag{6}$$

and λ is a free parameter leading to a certain world point somewhere on the image point-lens line. The idea is illustrated in figure 2-(a).

The back-projection onto the ground of all these image features (p_p) which correspond to scene points (p) located below the plane parallel to the flat ground and that contains the lens center (p_l) requires a positive λ, while λ has to be negative for all image features corresponding to scene points located above the mentioned plane. The idea is illustrated in figure 2-(b). p_1 is a point lying on the floor, and its corresponding image point is p_{p1}. In equation (5) 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.

The goal then is to find which is the location in the image of all these features that can be directly classified as obstacle points. Doing some manipulations on equation (5), the z world coordinate of a scene point can be calculated as (Duda & Hart, 1973):



(a) (b)

Fig. 2. (a) The *IPT*: $p = p_l + \lambda(p_p - p_l)$ (b) $\lambda's$ positives and negatives

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

where λ determines the exact position of the scene point in the inverse perspective projecting ray.
If $z = 0$, we can solve for λ:

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

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 (see figure 3), we obtain that:

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

where k_v factor is the relation $[pixels/length]$ of the used images.
Therefore, 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.

4.2 Selection of Threshold β

Those image features that do not admit a direct identification as obstacles must be classified using equation (4). In order to select an appropriate value for β in this equation, we have studied the behavior of discrepancy D for all the pixels of one image under reasonable conditions of application in the navigation strategy. More precisely: a) the distance between the camera and a scene point (DST) was fixed to a constant value for all pixels of one image, and tested with 1000mm, 1500mm and 2000mm, b) according to the experimental setup, the camera separation between two consecutive images was 31mm along the direction of motion, the image resolution was set to 256×192 pixels, the focal length was set to 3.720mm, and the camera angles φ and θ were set to -9° and 0°, respectively, c) the rotation matrix R representing the orientation of the camera in the world coordinate system was defined for rotation only around the x_p axis.

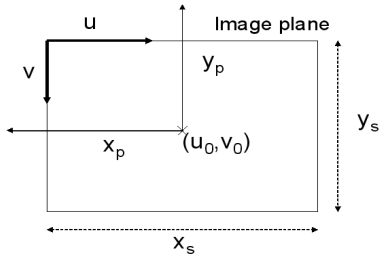


Fig. 3. Image plane coordinate system, where x_s and y_s are respectively, the horizontal and vertical image resolution

This section describes next the algorithm to compute the discrepancy D for each pixel of one image, under the aforementioned conditions.
I) We start from two consecutive images. First, given a point in the first image with coordinates (x_{p1}, y_{p1}), the world coordinates of its corresponding scene point are calculated assuming that the distance between the camera and the point of the scene (DST) is known. The relation between the coordinates (X_{c1}, Y_{c1}, Z_{c1}) of a scene point expressed in the coordinate system attached to the camera and its corresponding image point (x_{p1}, y_{p1}) is (Hartley & Zisserman, 2003):

$$X_{c1} = \frac{x_{p1} Z_{c1}}{f} \qquad Y_{c1} = \frac{y_{p1} Z_{c1}}{f}. \tag{11}$$

The distance between the camera and the point of the scene (DST) can be calculated as:

$$DST = \sqrt{Z_{c1}^2 + Y_{c1}^2 + X_{c1}^2}, \tag{12}$$

combining (12) with (11), we obtain:

$$Z_{c1}^2 = \frac{DST^2}{1 + (\frac{y_{p1}^2}{f^2}) + (\frac{x_{p1}^2}{f^2})}. \tag{13}$$

The euclidean transformation between the world and camera homogeneous coordinates can be expressed as:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = T_c^w \begin{bmatrix} X_{c1} \\ Y_{c1} \\ Z_{c1} \\ 1 \end{bmatrix}, \tag{14}$$

where

$$T_c^w = \begin{bmatrix} R & T_1 \\ 0^T & 1 \end{bmatrix}, \tag{15}$$

being R the 3×3 rotation matrix, and $T_1 = (X_{01}, Y_{01}, Z_0)$ (the camera position at the first image). Knowing R and T_1, we can obtain the world coordinates (x, y, z) of the scene point corresponding to the chosen image point.

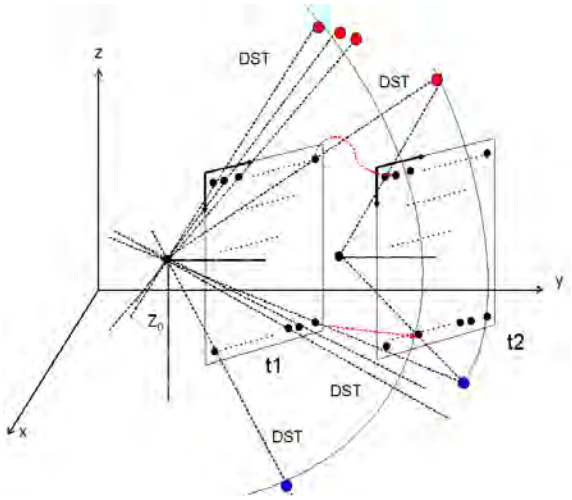
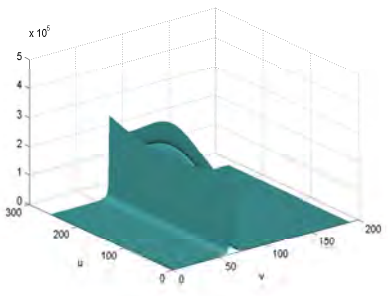
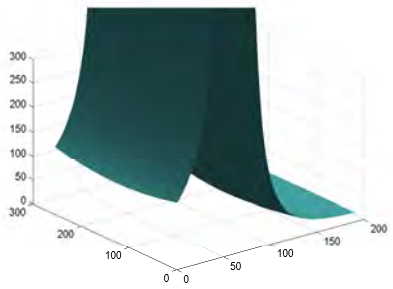


Fig. 4. All image points over a sphere of a defined radius DST are projected in the image plane, and then tracked in two consecutive images, at instants t_1 and t_2.

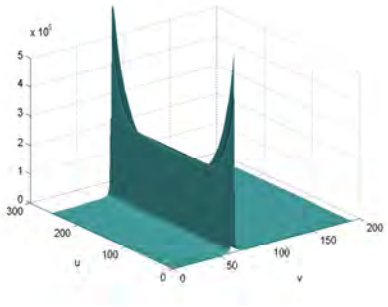
II) Next, this scene point (x, y, z) is projected onto the second image, obtaining the point coordinates (x_{p2}, y_{p2}) in the second frame. The camera position at the instant of taking the second image is $T_2 = (X_{02}, Y_{02}, Z_0)$. Assuming that R does not change, with the new camera position and the calculated world point coordinates (x, y, z), the scene point expressed in the new camera coordinates (X_{c2}, Y_{c2}, Z_{c2}) can be calculated using the equation (14). With (X_{c2}, Y_{c2}, Z_{c2}), the image coordinates (x_{p2}, y_{p2}) can be calculated using equations (11).
III) Finally, (x_{p1}, y_{p1}) and (x_{p2}, y_{p2}) are back-projected onto the floor using equations (2) and (3) to obtain (x_1^*, y_1^*) and (x_2^*, y_2^*), which are used to calculate the discrepancy D defined in equation (4).
The so far described process is illustrated in figure 4. Assuming a constant DST for all image points means that all points located in the surface of a sphere surrounding the camera, with a radius of DSTm are projected onto the image. If the radius is sufficiently long, the sphere will intersect the plane $z = 0$. All points of the sphere that intersect this plane are on the floor, and are projected at the bottom of the image.
Figure 5 plots three cases of function $D(x_p, y_p)$ for all pixels of one image of 256×192 pixels, using DST=1000mm, DST=1500mm and DST=2000mm, and setting the rest of parameters (f, φ and θ) as stated at the beginning of this section. Note in plots (d) and (f) how pixels at the bottom of the image present discrepancies equal to 0. These pixels correspond to those scene points located on the floor or below the $z = 0$ world plane.
Figure 6 plots $D(x_p, y_p)$ for images with a resolution of 1024×768 pixels and the same parameters as stated previously in this section. D values reach their maximum around v=50 pixels for images with a resolution of 256 × 192 pixels and v=200 for images with a resolution of 1024 × 768 pixels.
As can be observed in figures 5 and 6, discrepancy D exhibits different behavior depending on the image pixel location. The dependance of D with the position of the feature in the image suggests that the β threshold defined in equation (4) can be also dependent on the image feature position. Adjusting β to a low value for image features near the bottom or to a



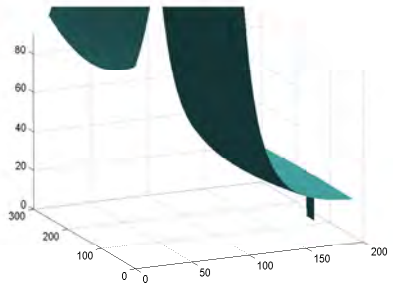
(a)



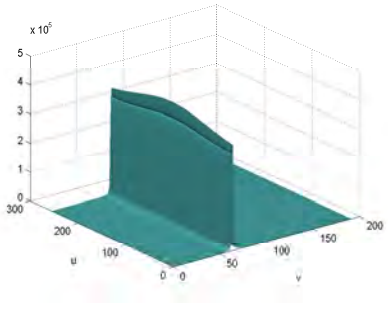
(b)



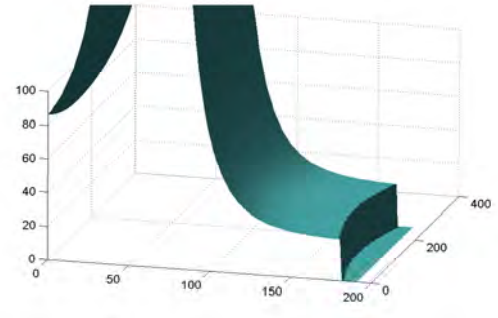
(c)



(d)

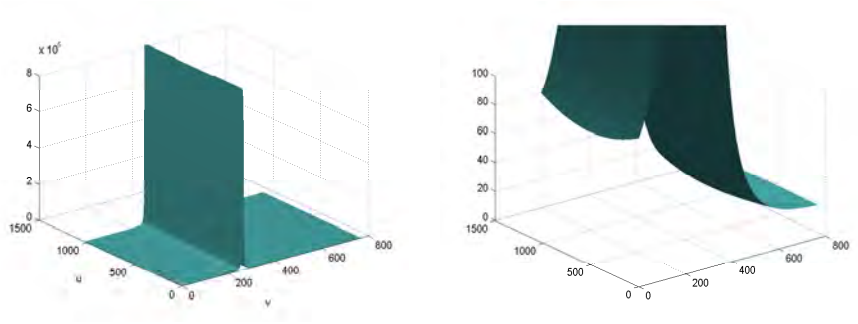


(e)



(f)

Fig. 5. (a), (c) and (e), plot of $D(x_p, y_p)$ function for DST=1m, DST=1.5m and DST=2.0m, respectively, for a resolution of 256×192 pixels. (b), (d) and (f): respectively, detail of (a), (c) and (e). In plots (c), (d), (e) and (f), $D = 0$ for world points lying on the sphere surface with radius DST and such that $z \leq 0$



(a) (b)

Fig. 6. (a) D graphic for DST=1.5m and image resolution of 1024×768. (b) detail of (a) for D (discrepancy) low levels

higher value for image features near the zone with maximum D should decrease the number of missclassified SIFT features.

5. The Navigation Strategy

5.1 Obstacle Profiles

SIFT features are usually detected at regions of high gradient (Lowe, 2004), thus they are likely to be near or belong to an edge. Besides, features classified as obstacles 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 real obstacle points. This process 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 runs in two steps (Canny, 1986):

1. First, the original image is convolved with a 1D gaussian derivative horizontal kernel. This permits detecting zones with high vertical gradient from smoothed intensity values with a single convolution.
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 every image feature classified as obstacle and then searches for all edge pixels which are inside a window centered at the feature image coordinates. Then, every edge is tracked down starting from the object point position until the last edge pixel or a ground point is found. This will be considered as to be the point(/s) where the object rests on the floor. This process permits isolating the obstacle boundaries from the rest of edges and to get a qualitative perception of the environment.

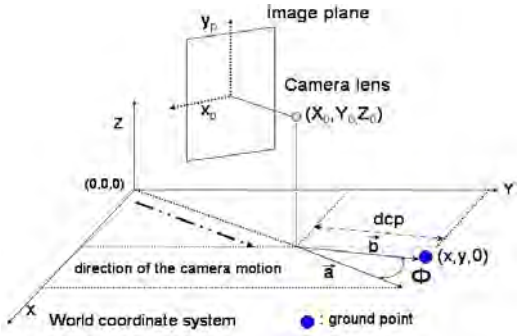


Fig. 7. Distance and orientation of an obstacle point with respect to the camera

5.2 Building the Local Occupancy Map

Knowing the camera position and the world coordinates of a point on the floor, the distance dcp between the vehicle and the floor point and the angle ϕ defined by the direction of motion and the relative orientation of this point with respect to the camera can be calculated as:

$$dcp = \sqrt{(x - X_0)^2 + (y - Y_0)^2}$$
$$\phi = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right) \tag{16}$$

where $(x, y, 0)$ are the world point coordinates, $\vec{a}$ is a vector with the same direction as the vector from the world coordinate system origin to point $(X_0, Y_0, 0)$, $\vec{b}$ is the vector from point $(X_0, Y_0, 0)$ to point $(x, y, 0)$, and $\vec{a} \cdot \vec{b}$ is the dot product between both vectors. The idea is illustrated in figure 7.

The orientation and distance of obstacles with respect to the robot can then be qualitatively estimated computing the distance and orientation between the camera and those obstacle points in contact with the floor, using equations (16).

5.3 The Navigation Strategy

A semicircular area on the ground plane, of a fixed radius and centered at the robot position, is defined as the Region of Interest (ROI). 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 a set of obstacle-to-ground contact points are labeled as forbidden and those free of obstacle-to-ground contact points are included in the set of possible next movement directions. This process results in a qualitative polar occupancy map of free and occupied zones in 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 angular regions are narrower than the defined threshold, the algorithm concludes that all space in front is occupied by obstacles and returns a stop order.

The next movement direction is given as a vector pointing to the center of the widest polar obstacle-free zone. Positive angles result for turns to the right and negative angles for turns to the left.

6. Implementation and Experimental Results

6.1 Overall Performance of the Classifier

To test the proposed strategy, a Pioneer 3DX robot with a calibrated wide angle camera was programmed to navigate in different scenarios, such as environments with obstacles of regular and irregular shape, with textured and untextured floor, and environments with specularities or under low illumination conditions. The operative parameter settings were: robot speed=40mm/s; the radius of the ROI=1'5m; for the hysteresis thresholding, low level= 40 and high level= 50; camera height= $430mm$; $\varphi = -9^{\circ}$; initial $\theta = 0^{\circ}$, and finally, $f = 3.720mm$. For each scene, the complete navigation algorithm was run over successive pairs of 0.77-second-separation consecutive frames so that the effect of IPT was noticeable. Increasing the frame rate decreases the IPT effect over the obstacle points, and decreasing the frame rate delays the execution of the algorithm. Frames were originally recorded with a resolution of 1024×768 pixels but then they were down-sampled to a resolution of 256×192 pixels, in order to reduce the computation time. All frames were also 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 the point world coordinates. The implementation of the SIFT features detection and matching process was performed following the methods and approaches described in (Lowe, 2004). 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 ROC curves (Bowyer et al., 2001). These curves were computed for every pair of consecutive images and plot the *recall* of classified points vs the *fall-out*, varying the threshold β:

$$recall(\beta) = \frac{TP(\beta)}{TP(\beta) + FN(\beta)} \qquad fallout(\beta) = \frac{FP(\beta)}{FP(\beta) + TN(\beta)}, \tag{17}$$

where TP is the number of true positives (obstacle points classified correctly), FN is the number of false negatives (obstacle points classified as ground), FP is the number of false positives (ground points classified as obstacle) and TN is the number of true negatives (ground points classified correctly). For every ROC curve, its Area Under the Curve (AUC) (Hanley & McNeil, 1982) was calculated as a measure of the success rate. The optimum β value was obtained for every pair of images minimizing the cost function:

$$f(\beta) = FP(\beta) + \delta FN(\beta). \tag{18}$$

During the experiments, δ was set to 0.5 to prioritize the minimization of false positives over false negatives. For a total of 36 different pairs of images, corresponding to a varied set of scenes differing in light conditions, in the number and position of obstacles and in floor texture, a common optimum β value of 21mm resulted.

Figure 8 shows some examples of the classifier output. Pictures [(1)-(2)], [(4)-(5)], [(7)-(8)], [(10)-(11)] show several pairs of consecutive frames corresponding to examples 1, 2, 3 and 4, respectively, recorded by the moving robot and used as input to the algorithm. Pictures (2), (5), (8) and (11) 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 examples 1 to 4 (plots (3), (6), (9) and (12) of figure 8) suggest success rates of 97%, 94%, 92% and 95%, respectively. Notice that all scenes present inter-reflections, shadows and specularities, although they do not affect the classifier performance.

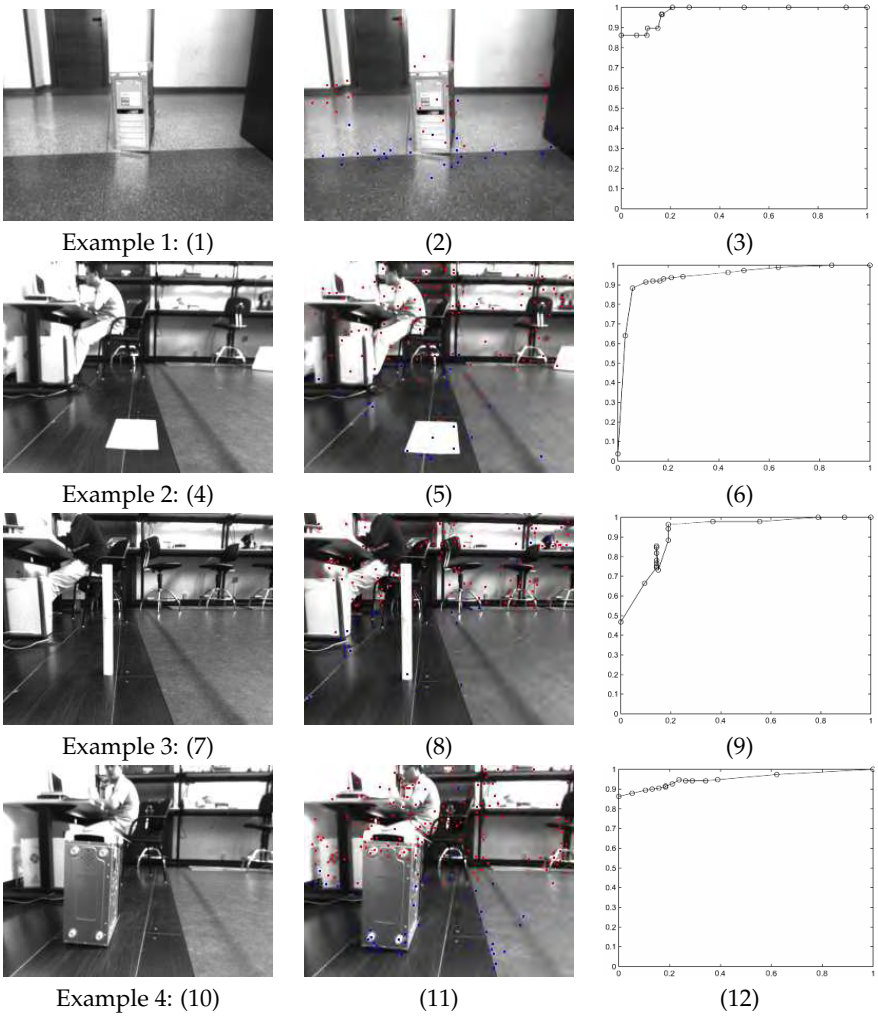


Fig. 8. (1),(4), (7) and (10): undistorted first frame of examples 1, 2, 3 and 4, respectively. (2), (5), (8) and (11): undistorted second frame. (3), (6), (9) and (12): ROC curves for examples 1, 2, 3 and 4, respectively (AUC_1=0'9791, AUC_2=0'9438, AUC_3=0'9236, AUC_4=0'9524).

6.2 The Classifier Refinement Routine

Features corresponding to points lying on the floor but classified as obstacle points can induce the detection of false obstacles. In order to filter out as much FPs as possible, the threshold β

was varied with the feature image location and according to the concepts and results outlined in section 4.2.

Taking the same values of f, φ, camera height, image resolution, robot speed, ROI and frame rate as stated in section 6.1, and with a k_v=1000/(4 ∗ 4, 65) (taking into account that 1 pixel=4.65μm for the original image resolution of 1024 × 768 pixels, then, for the down-sampled images with a resolution of 256 × 192 pixels, 1 pixel=4*4.65μm), from equation (10) resulted v <65 pixels. All features located between the top of the image and v=65 pixels were directly classified as obstacle points.

Since the yaw angle of the camera with respect to the direction of motion was 0 and the camera pitch angle was -9°, it was defined a rotation matrix corresponding to a unique rotation around the x_p camera axis. The transformation from camera to world coordinates T_c^w was set to:

$$T_c^w = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & \sin\varphi & \cos\varphi & Y_0 \\ 0 & \cos\varphi & -\sin\varphi & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The radius of the ROI was set to 1.5m, so the DST (see equation (12)) reference value was also set to 1.5m.

In a previous training phase, a number of image sequences were recorded in different scenarios with the moving robot remotely controlled. 36 image pairs were used to train the β adjustment. Every image was then virtually divided in four sectors, 1) zone 3, from v=0 to v=65, where all points were automatically classified as obstacle points; 2) zone 2, from v=65 to v=90, which is the zone where D reaches abruptly its maxima values; 3) zone 1, from v=90 to v=169, where D changes gradually with the image v coordinate and 4) zone 0, from v=169 to v=192, where D has a nearly constant value of 21mm, for a DST=1.5m. The threshold β used to determine the maximum discrepancy admissible for a feature to be classified as ground point was set differently for the different image zones: a) 21mm in zone 0, b) in zones 1 and 2, the β value was chosen to minimize the number of $FP(\beta) + 0.5FN(\beta)$ in each image zone, and for each different scenario. For example, scenario 2 required a higher β in zone 2 than scenario 1. In zone 1, βs resulted in a 20mm to 30mm range, and in zone 2, βs resulted in a 30mm to 150mm range.

Also during the training phase, histograms accounting for the number of FP and TP for each D value where computed over a number of pre-recorded images of different scenarios. Figure 9 shows some examples of these histograms. TP located in zone 2 are shown in green, TP in zone 1 are shown in blue, FP in zone 1 are shown in red and FP in zone 2 are shown in magenta. The majority of TP are located in zone 2 and have high D values. Only a few obstacle points are located in zone 1. FP in the zone 2 do not affect our navigation algorithm since they are out of the ROI. FP in the zone 1 can be inside the ROI and have to be filtered out. For all the analyzed scenarios, all FP of zone 1 presented discrepancies (D) in a 20mm and 85mm range.

Once β had been configured for every image zone and scenario, and the filtering criteria had been defined, the algorithm could be run during the autonomous navigation phase. During this autonomous process and for all tested scenes, all features of zone 1 that presented a discrepancy between 20mm and 85mm were not classified. Combining the aforementioned filter with a β changing at each different image zone, nearly all ground points classified as obstacles were filtered out and some other points were well re-classified. This reduced the risk

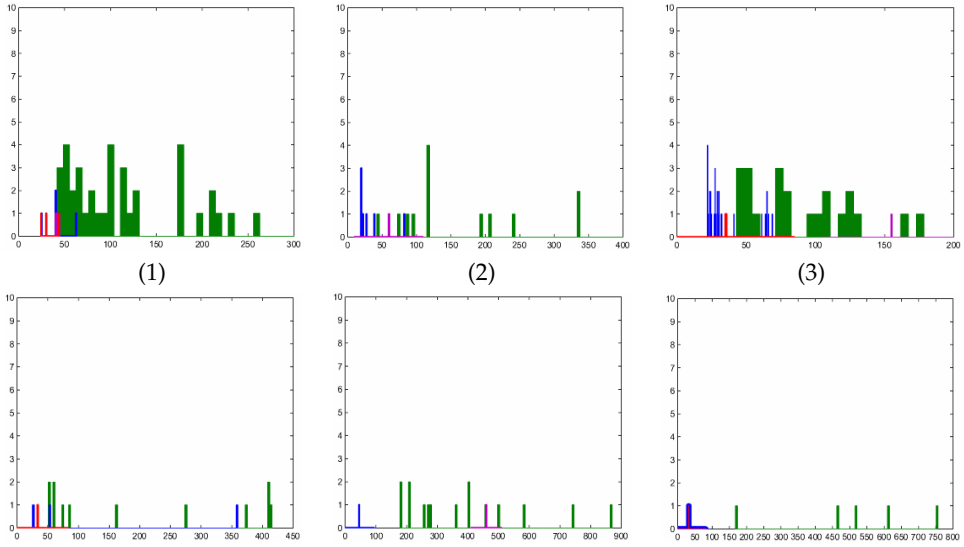


Fig. 9. (1), (2) and (3): Examples of histograms corresponding to scenario 1. (4) and (5): Example of histograms corresponding to scenario 2. (6): Example of histogram corresponding to scenario 3.

of detecting false obstacles, and although some true obstacle points were also removed, the remaining ones were sufficient to permit the detection of those obstacles.

Figure 10 shows several results of the refinement routine. Pictures (1), (3), (5), (7), (9) and (11) show images recorded during some navigation tests in different scenarios. Obstacle points are shown in red and ground points in blue. Pictures (2), (4), (6), (8), (10) and (12) show the corresponding images after the refinement routine was applied. See as in all these images false obstacles in zone 1 were filtered out.

Table 1 shows some numerical results to compare the classifier assessment using a single β and no filtering process vs the results obtained using a changing β and the filtering routine. Columns $FPAF/Nbr$ and FP/Nbr show the percentage of FP with respect to the total number of features at each scene, with and without the refinement process, respectively. In all cases this percentage either maintains the value or decreases. The column AUC shows the area under the ROC curve without the refinement process. All values suggest a classifier success rate greater than 90%. The *Fall Out* for the optimum β in each image zone, calculated when the refinement process was applied, decreases or maintains the value with respect to the *Fall Out* computed with the single optimum β (21mm) without the refinement process.

6.3 The Complete Navigation Strategy

After image features have been classified, the algorithm successfully identifies the relevant part of the obstacle contour. A 9×15 pixel window is used to find edge pixels near an obstacle point and to track down the obstacle contours. The window is longer in the vertical direction to overcome possible discontinuities in the obstacle vertical borders.

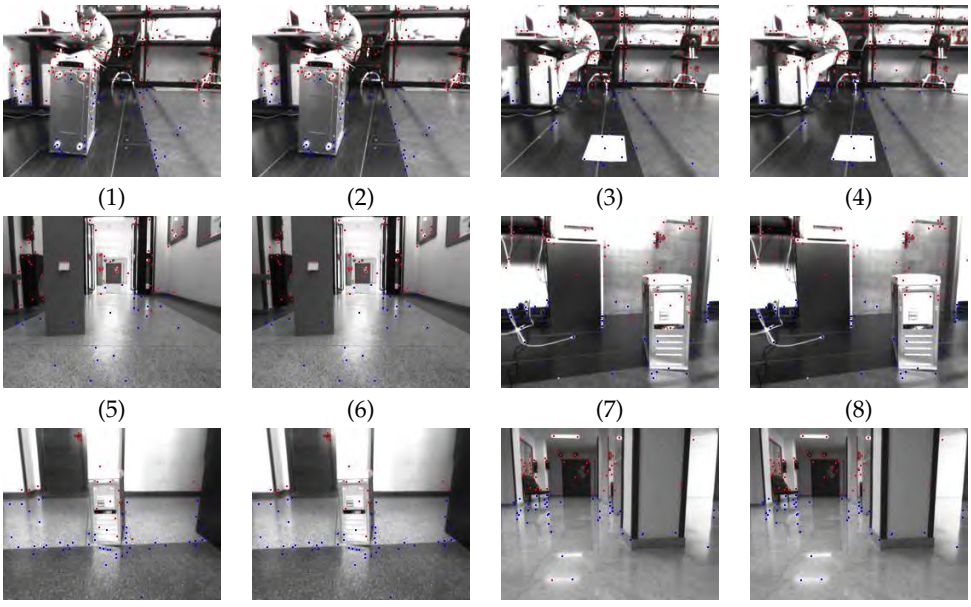


Fig. 10. (1), (3), (5), (7), (9) and (11): Image with SIFT features classified. (2), (4), (6), (8), (10) and (12): Image with SIFT features filtered and reclassified.

Scene	FP/N_{br}	AUC	*Fall-Out* for a unique β	*Recall* for a unique β	FPAF /N_{br}	*Fall Out* with refine-ment	*Recall* with refine-ment
scene 1	0.0078	0.9482	0.0600	0.9467	0.0042	0.0286	0.9415
scene 2	0.0275	0.9412	0.1500	0.8998	0.0096	0.0625	0.9034
scene 3	0.0313	0.9434	0.1156	0.9857	0.0108	0.0400	0.9850
scene 4	0.0081	0.9554	0.0416	0.7653	0.000	0.0000	0.7700
scene 5	0.0088	0.9834	0.0830	0.9010	0.0089	0.0833	0.9000
scene 6	0.0115	0.9376	0.0331	0.9818	0.0120	0.0357	0.9818
scene 7	0.0091	0.9827	0.0272	0.9315	0.0000	0.0000	0.9090
scene 8	0.0066	0.9350	0.0621	0.9700	0.0068	0.0625	0.9700
scene 9	0.0231	0.9100	0.1421	0.9459	0.0047	0.0294	0.9325
scene 10	0.0112	0.9036	0.0208	0.9047	0.0000	0.0000	0.9000

Table 1. Data results for some scenes. N_{br} is the number of scene SIFT features, FP: number of false positives; FPAF: number of false positives after the filter.

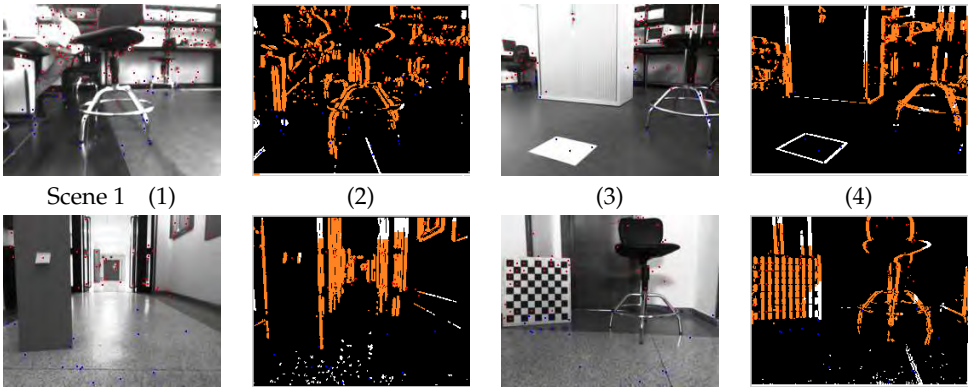


Fig. 11. (1), (3), (5), (7): Undistorted second frame of different pairs of consecutive images of Scenes 1, and 2. (2), (4), (6), (8): Obstacle contours.

Figure 11 shows four examples of the obstacle contour discrimination algorithm applied over images of a sequence recorded by the mobile robot during the autonomous navigation phase in two different scenarios. Pictures (1), (3), (5) and (7) are the second frame of four different pairs of consecutive images. Pictures (2), (4), (6) and (8) show the corresponding edge map with the obstacle profiles highlighted in orange. Note how the paper in picture (3) has not been detected as an obstacle since all features lying on it were classified as ground points, as well as, although picture (5) shows a very high inter-reflection on the ground and a very granulated texture on the floor tiles, only real obstacle boundaries have survived.
Figures 12, 13, 14 and 15 show some examples of the complete navigation algorithm tested on the moving robot. Missions consisted of navigating through several environments with some special characteristics, avoiding the obstacles, including columns and walls. The navigation algorithm was run with a variable β and the filtering process, and with all the same settings reported at the beginning of this section. Pictures (1), (2), (3) and (4) in all four figures show the second frame of some pairs of consecutive images recorded and processed during the navigation through scenarios 1, 2, 3. Every image was taken before the robot had to turn to avoid the frontal obstacles; obstacle points are shown in red and ground points in blue. Figure 12 (scenario 1) shows a room full of obstacles with regular and irregular shape. This scene presents shadows and inter-reflections. Figure 13 (scenario 2) corresponds to a corridor with a very high textured floor, columns, walls, inter-reflections and some specularities. Figures 14 and 15 (scenario 3) present bad illumination conditions, important inter-reflections and specularities on the floor, and some image regions (white walls, shelves and lockers) have homogeneous intensities and/or textures, resulting in few distinctive features and poorly edged obstacles which can complicate its detection. Pictures (5), (6), (7) and (8) 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. Range and angle of the computed world points with respect to the camera coordinates were estimated using equations (16). Those obstacle-to-ground contact points closer than 1'5m were highlighted in pink.
Histograms (9), (10), (11) and (12) in figures 12, 13, 14 and 15 account for the number of obstacle-to-ground contact points detected in each polar direction. Therefore, they turn out

to be local occupancy maps in a bird's-eye view of a semicircular floor portion with a radius of 1'5m. These maps show the world polar coordinates, with respect to the camera position (which is in the center of the semicircle), of those obstacle points in contact with the floor. The grid gives a qualitative idea of which part of the robot vicinity is occupied by obstacles and the proximity of them to the robot.
The algorithm analyzes next the polar histograms and defines the direction of the center of the widest obstacle-free polar zone as the next steering direction (shown in green). The experiments performed suggest a certain level of robustness against textured floors, bad illumination conditions, shadows or inter-reflections, and deals with scenes comprising significantly 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.
Figure 16 shows in plots (1), (2), (3) and (4) the trajectories followed by the robot during the navigation through the environments of experiments 1, 2, 3 and 4 displayed in figures 12, 13, 14 and 15. The blue circle denotes the starting point and the red circle denotes the end point.

7. Conclusions

Reactive visual-based navigation solutions that build or use local occupancy maps representing the area that surrounds the robot and visual sonar-based solutions are sensitive to floor and obstacle textures, homogeneity in the color intensity distribution, edges or lighting conditions. The construction of local maps is a suitable way to clearly identify the presence and position of obstacles and thus to determine the direction to follow. But it is not essential to determine or to identify exact obstacle shapes, dimensions, colors or textures. In this chapter, a new navigation strategy including obstacle detection and avoidance has been presented. The algorithm shows a certain robustness to the presence of shadows, inter-reflections, specularities or textured floors, overcomes scenes with multiple planes and uses only a certain number of image points. The complete strategy starts with a novel image feature classifier that distinguishes with a success rate greater that 90% between obstacle features from features lying on the ground. The detection of points that belong to obstacles permits: a) discriminating the obstacle boundaries from the rest of edges, and b) the detection of obstacle-to-ground contact points.
By computing the world coordinates of those obstacle-to-ground contact points detected in the image, the system builds a radial qualitative model of the robot vicinity. Range and angle information are quantitatively and accurately computed to create a qualitative occupancy map. Navigation decisions are taken next on the basis of qualitative criteria. What is reflected in these maps is not the total area that the obstacle occupies or its exact shape or identification, but it is an evidence of the presence of *something* that has to be avoided in a determined direction and at a defined distance.
The experimental setup consisted of different scenarios with different characteristics, different obstacles, different illumination conditions and different floor textures. In all cases the mobile robot was able to navigate through the free space avoiding all obstacles, walls and columns.

8. Future Work

The proposed strategy can be applied as an obstacle detection and avoidance module in more complex robot systems, like programmed missions for exploration of unknown environments, map-building tasks, or even, for example, as a guiding robot. The algorithm depicted does not

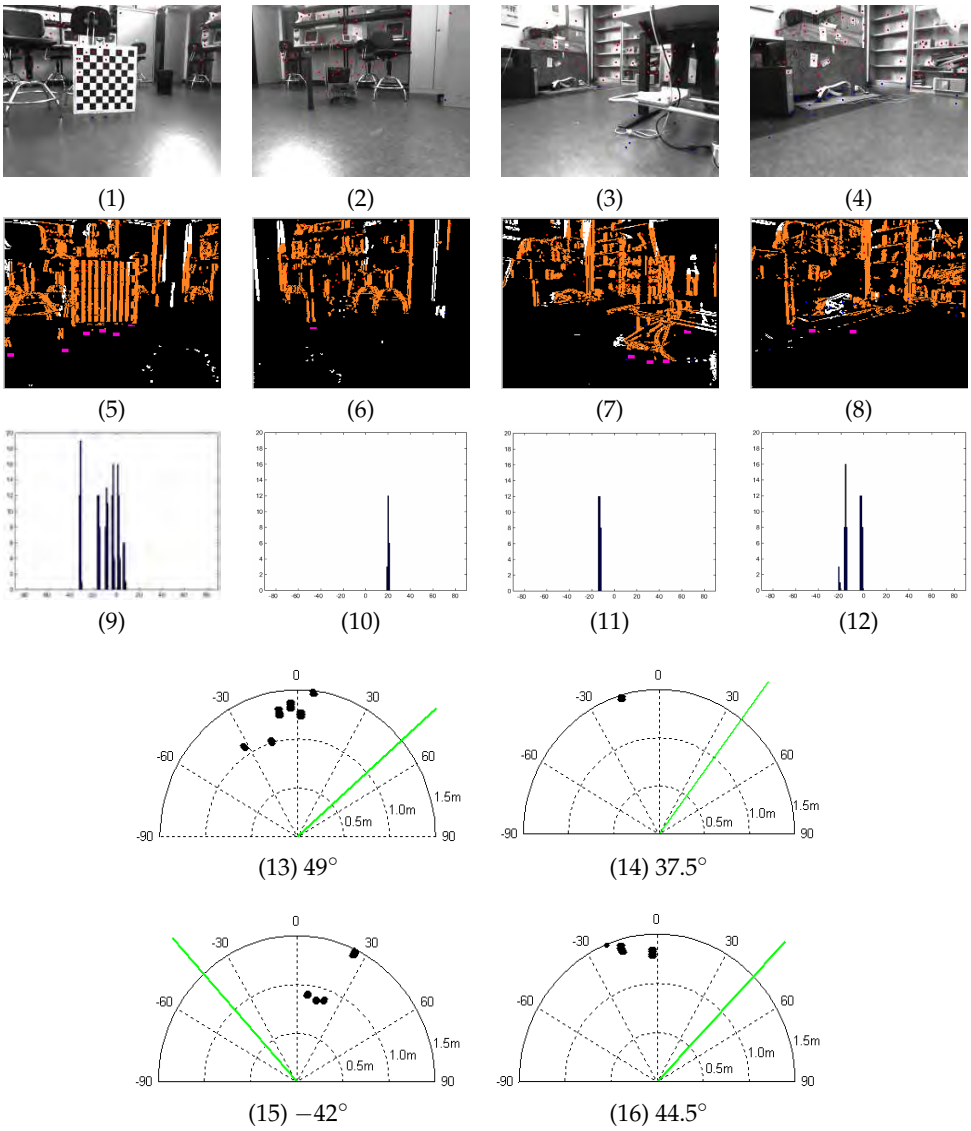


Fig. 12. Scenario 1. Experiment 1: (1), (2), (3) and (4), undistorted second frames; (5), (6), (7) and (8), corresponding edge maps with obstacle borders highlighted in orange. (9), (10), (11), (12), histograms of obstacle-to-ground contact points for each polar direction between $-90°$ and $90°$. (13), (14), (15) and (16), local occupancy map with the resulting steering vector, for images (1), (2), (3) and (4) respectively.

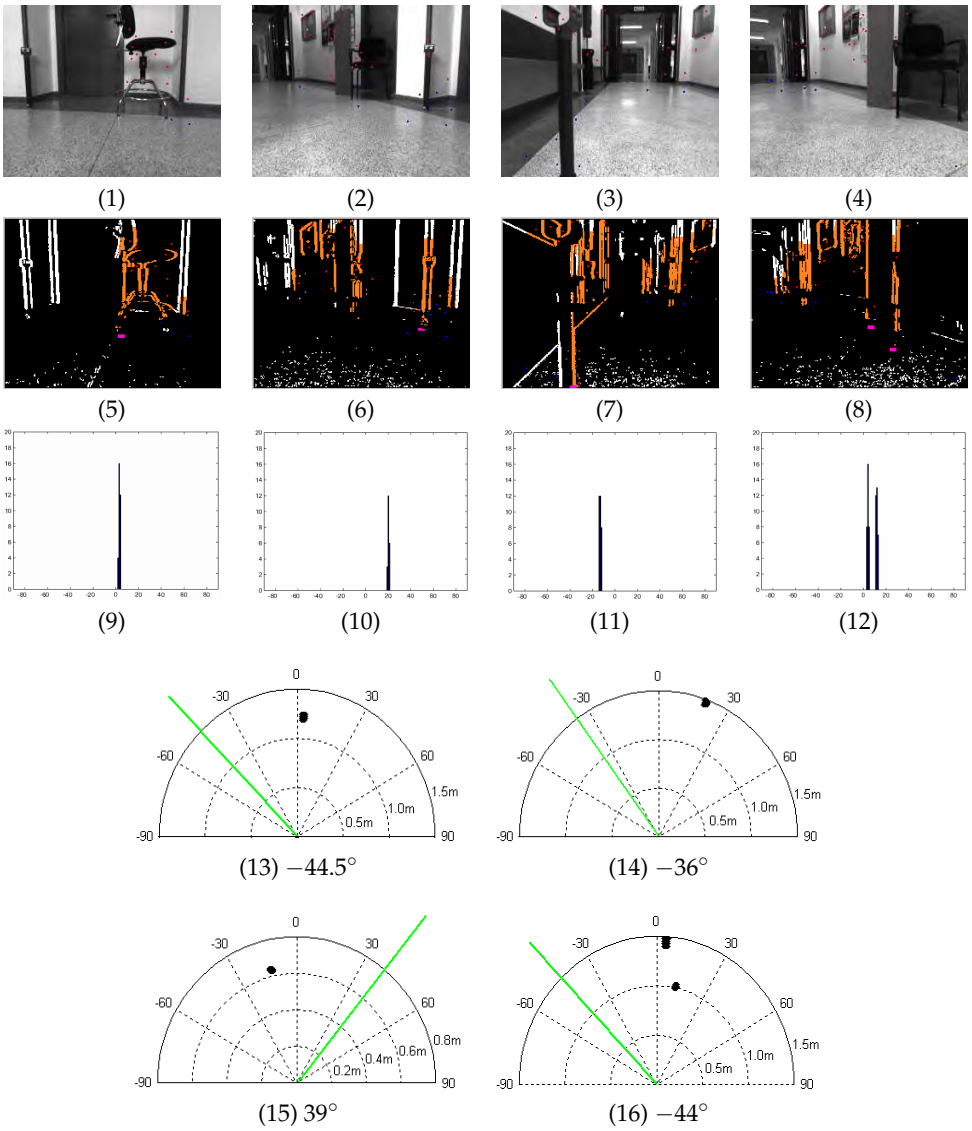


Fig. 13. Scenario 2. Experiment 2: floor with a very granulated texture. (1), (2), (3), (4), undistorted second frames; (5), (6), (7) and (8), corresponding edge maps with obstacle borders highlighted in orange; (9), (10), (11), (12), histograms of obstacle-to-ground contact points for each polar direction between -90° and 90°; (13), (14), (15) and (16), local occupancy map with the resulting steering vector, for images (1), (2), (3) and (4), respectively.

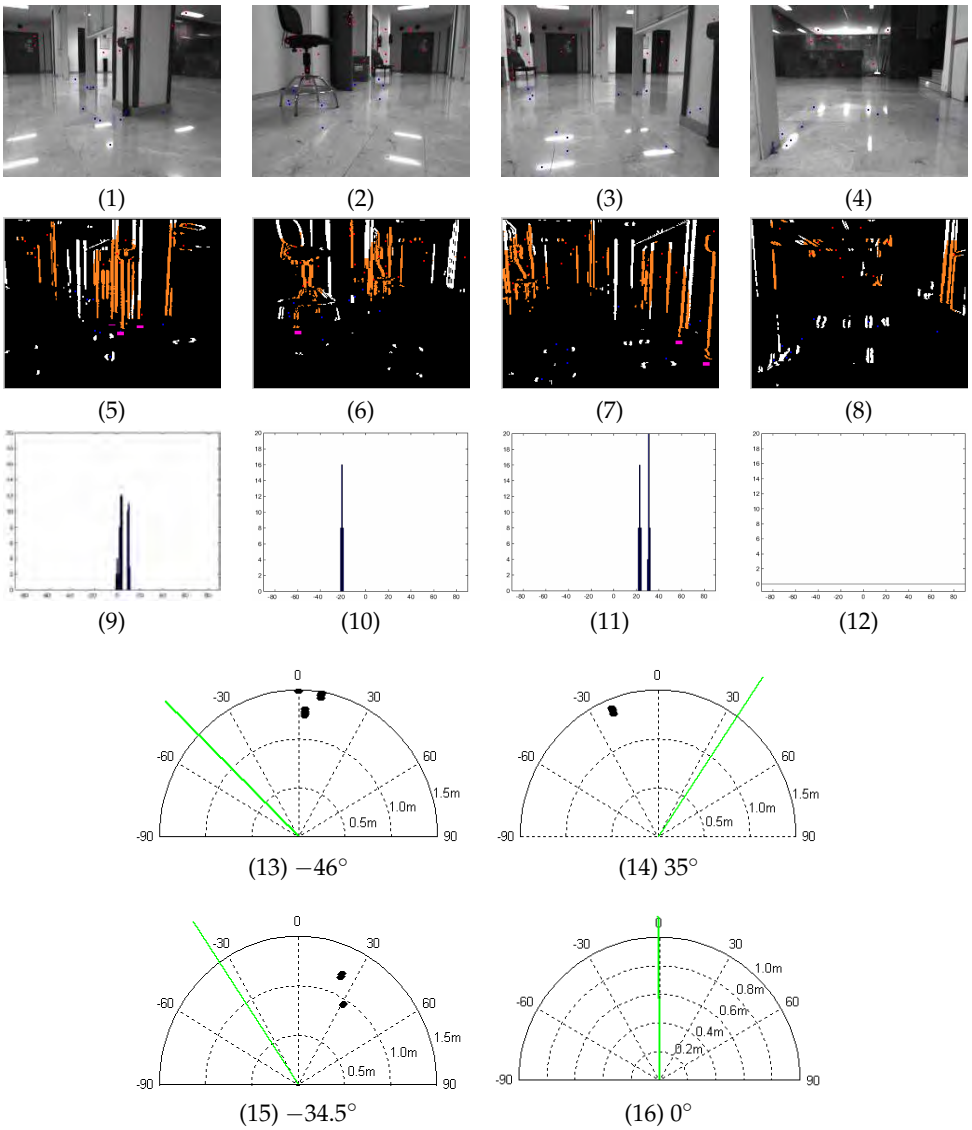


Fig. 14. Scenario 3. Experiment 3: some inter-reflections and bad illumination conditions. (1), (2), (3) and (4), undistorted second frames; (5), (6), (7) and (8), corresponding edge maps with obstacle borders highlighted in orange; (9), (10), (11) and (12) histograms of obstacle-to-ground contact points for each polar direction between $-90°$ and $90°$; (13), (14), (15) and (16), local occupancy map with the resulting steering vector, for images (1), (2), (3) and (4) respectively.

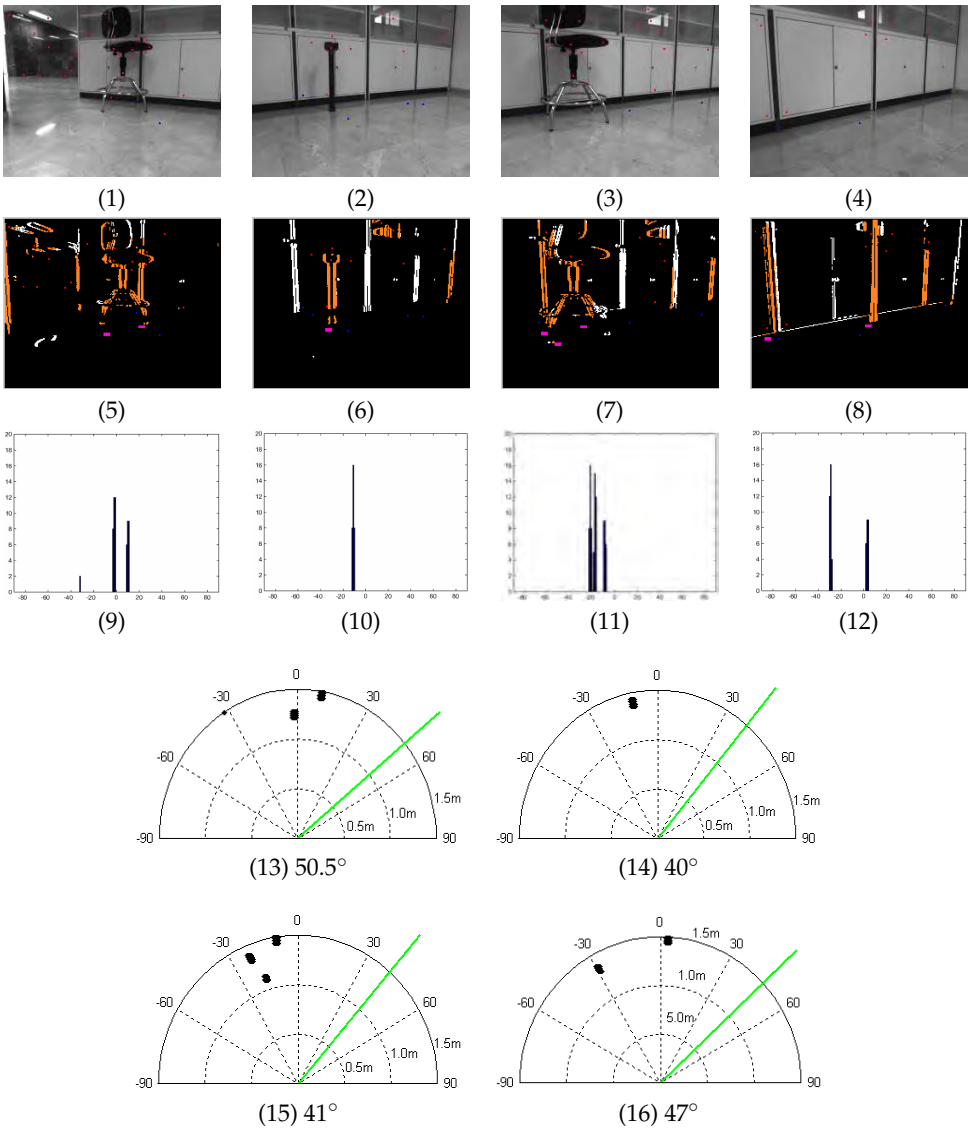
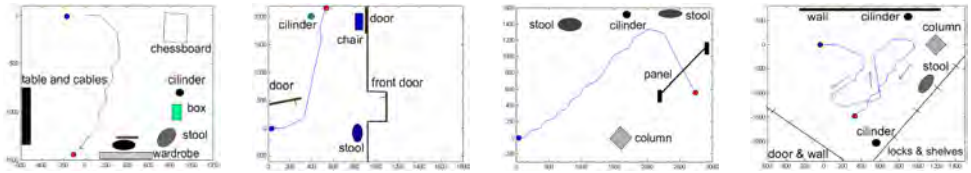


Fig. 15. Scenario 3. Experiment 4: few distinctive points, few borders, some inter-reflections and bad illumination conditions.(1), (2), (3), (4), undistorted second frames; (5), (6), (7) and (8), corresponding edge maps with obstacle borders highlighted in orange; (9), (10), (11), (12), histograms of obstacle-to-ground contact points for each polar direction between -90° and 90°. (13), (14), (15) and (16), local occupancy map with the resulting steering vector, for images (1), (2), (3) and (4) respectively.



(1) (2) (3) (4)

Fig. 16. (1), (2), (3) and (4), robot trajectories for tests of figures 12, 13, 14 and 15, respectively.

restrict the method used for feature detection and tracking. Depending on this method, the number of detected features can change, features can be detected in different image points, their classification can change and the algorithm time of execution can also be different. To explore different choices for detecting and tracking features becomes necessary to optimize our algorithm in terms of: a) number of necessary features, b) their location in the image, and c) time of execution

9. References

Badal, S., Ravela, S., Draper, B. & Hanson, A. (1994). A practical obstacle detection and avoidance system, *Proceedings of 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota FL USA, pp. 97–104.

Batavia, P., Pomerleau, D. & Thorpe, C. E. (1997). Overtaking vehicle detection using implicit optical flow, *IEEE Conference on Intelligent Transportation System*, Boston, MA, USA, pp. 729–734.

Bertozzi, M. & Broggi, A. (1998). Gold: a parallel real-time stereo vision system for generic obstacle and lane detection, *IEEE Transactions on Image Processing* **7**(1): 62–81.

Bonin, F., Ortiz, A. & Oliver, G. (2008). Visual navigation for mobile robots: a survey, *Journal of Intelligent and Robotic Systems* **Vol. 53**(No. 3): 263–296.

Borenstein, J. & Koren, I. (1991). The vector field histogram - fast obstacle avoidance for mobile robots, *Journal of Robotics and Automation* **7**(3): 278–288.

Bowyer, K., Kranenburg, C. & Dougherty, S. (2001). Edge detector evaluation using empirical roc curves, *Computer Vision and Image Understanding* **84**(1): 77–103.

Canny, J. (1986). A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **8**(6): 679 – 698.

Choi, Y. & Oh, S. (2005). Visual sonar based localization using particle attraction and scattering, *Proceedings of IEEE International Conference on Mechatronics and Automation*, Niagara Falls, Canada, pp. 449–454.

Duda, R. & Hart, P. (1973). *Pattern Classification and Scene Analysis*, John Wiley and Sons Publisher, USA.

Fasola, J., Rybski, P. & Veloso, M. (2005). Fast goal navigation with obstacle avoidance using a dynamic local visual model, *Proceedings of the SBAI'05 VII Brazilian Symposium of Artificial Intelligence*, Ao Luiz, Brasil.

Goldberg, S., Maimone, M. & Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration, *Proceedings of IEEE Aerospace Conference*, Big Sky, Montana, USA, pp. 2025–2036.

Hanley, J. A. & McNeil, B. (1982). The meaning and use of the area under a receiver operating charateristic (ROC) curve, *Radiology* **143**(1): 381–395.

Harris, C. & Stephens, M. (1988). Combined corner and edge detector, *Proceedings of the Fourth Alvey Vision Conference*, Manchester, UK, pp. 147–151.

Hartley, R. & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521623049, Cambridge, UK.

Horswill, I. (1994). Collision avoidance by segmentation, *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, Munich ,Germany, pp. 902–909.

Lenser, S. & Veloso, M. (2003). Visual sonar: Fast obstacle avoidance using monocular vision, *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, Pittsburgh, PA, USA, pp. 886–891.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* **Vol. 60**(No. 2): 91–110.

Ma, G., Park, S., Müller-Schneiders, S., Ioffe, A. & Kummert, A. (2007). 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, pp. 137–142.

Mallot, H., Buelthoff, H., Little, J. & Bohrer, S. (1991). Inverse perspective mapping simplifies optical flow computation and obstacle detection, *Biomedical and Life Sciences, Computer Science and Engineering* **64**(3): 177–185.

Martin, M. C. (2006). Evolving visual sonar: Depth from monocular images, *Pattern Recognition Letters* **27**(11): 1174–1180.

Mikolajczyk, K. & Schmid, C. (2005). A performance evaluation of local descriptors, *IEEE TPAMI* **27**(10): 1615–1630.

Rabie, T., Auda, G., El-Rabbany, A., Shalaby, A. & Abdulhai, B. (2001). Active-vision-based traffic surveillance and control, *Proceedings of the Vision Interface Annual Conference*, Ottawa, Canada, pp. 87–93.

Rodrigo, R., Zouqi, M., Chen, Z. & Samarabandu, J. (2009). Robust and efficient feature tracking for indoor navigation, *IEEE Transactions on Systems, Man and Cybernetics* **Vol. 39**(No. 3): 658–671.

Saeedi, P., Lawrence, P. & Lowe, D. (2006). Vision-based 3-d trajectory tracking for unknown environments, *IEEE Transactions on Robotics* **22**(1): 119–136.

Shi, J. & Tomasi, C. (1994). Good features to track, *Proceedings of the IEEE IntŠl Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600.

Shu, Y. & Tan, Z. (2004). Vision-based lane detection in autonomous vehicle, *Proceedings of the Congress on Intelligent Control and Automation*, Xi'an Jiaotong, China, pp. 5258–5260.

Simond, N. & Parent, M. (2007). Obstacle detection from ipm and super-homography, *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, California, Sant Diego, USA, pp. 4283–4288.

Stephen, S., Lowe, D. & Little, J. (2005). Vision-based global localization and mapping for mobile robots, *IEEE Transactions on Robotics* **Vol. 21**(No. 3): 364–375.

Zhou, J. & Li, B. (2006). Homography-based ground detection for a mobile robot platform using a single camera, *Proceedings of the IEEE Int'l Conference on Robotics and Automation (ICRA)*, Arizona, Tempe, USA, pp. 4100–4101.