Building a Qualitative Local Occupancy Grid in a new Vision-based Reactive Navigation Strategy for Mobile Robots[*]

Francisco Bonin-Font and Alberto Ortiz
University of the Balearic Islands.
Cra de Valldemossa, km 7.5, 07122, Palma de Mallorca, Spain
francisco.bonin@uib.es; alberto.ortiz@uib.es

Abstract

Reactive navigation strategies for mobile robots base their success on the ability to discriminate obstacles from the ground in the vicinity of the robot and many of these strategies are based uniquely on the computation of quantitative information. In this paper we describe a new method to build qualitative local occupancy grids that are used in a new vision-based navigation strategy addressed to mobile robots to explore safely unknown environments. The process includes a new feature classifier based on the Inverse Perspective Transformation to discriminate object from ground points and a method to determine angle and range with respect to the camera in the world coordinate system.

1. Introduction

Vision-based navigation techniques can be roughly divided in map-based (normally labeled as deliverative) and mapless systems (labeled as reactive). Map-based systems plan routes and performance while mapless systems use the on-line analysis of the perceived environment to determine the route to follow [3]. Reactive vision-based systems can include the implementation of local occupancy grids, updated on-line and used to navigate avoiding obstacles located in the proximity of the robot [1][8]. Some researchers explored the idea of visual sonar, which provides range data and depth measurements using visual sensors but in an analogous way to ultrasound sensors [5]. In various cases the concept of visual sonar is strongly related with the computation of local occupancy grids to perform reactive navigation and obstacle avoidance strategies [7] [10]. This paper presents a new method to build occupancy maps to qualitatively register the presence of obstacles inside a Region of Interest (ROI), centered on the robot and with a fixed radius. The method has been inspired on the visual sonar-based reactive solutions and it implements a new version of the Vector Field Histogram [4] method, but here it has been adapted for vision-based systems. First, image main features are detected, tracked across consecutive frames, and classified as obstacles or ground using a new algorithm based on the Inverse Perspective Transformation (*IPT*). Next, the edge map of the processed frames is computed, and edges comprising obstacle points are discriminated from the rest, emphasizing the obstacle boundaries. Range and angle with respect to the robot are estimated computing the orientation and distance of those obstacle points that are in contact with the floor. This results in a qualitative occupancy map that displays the position of obstacles in the space with respect to the robot. Finally, the algorithm computes a vector which steers the robot towards the world areas into which it can move safely.

The rest of the paper is organized as follows: the method is outlined in Sections 2 and 3, experimental results are exposed and discussed in Section 4, and finally, conclusions and forthcoming work are given in Section 5.

2. Inverse Perspective Transformation, Feature Detection and Feature Classification

The process of taking a picture is usually modeled by the Perspective Transformation. The inverse process, that is, the projection of every image point back to the world is modeled by the *IPT*. The back projected point will be somewhere in the line that connects the image point with the center of projection. It is possible to compute the world coordinates of an image point either if the distance between the camera and the point in the space is known or, for example, if the point is lying on the ground. References [6] and later [9] described the Direct and the Inverse Perspective Transformations assuming a pinhole camera model and a flat ground.

Our obstacle detection method makes use of features matched across consecutive frames and the *IPT* to discriminate features lying on the ground from features corresponding to scene obstacles. More specifically, once a feature is matched between two frames, we assume it lies on the floor and we backproject the corresponding image points: the resulting world coordinates from both frames must coincide when the hypothesis is true, but they must be different when the feature comes from an elevated

[*]This work is partially supported by DPI 2008-06548-C03-02 and FEDER funding.

scene point. A threshold (β) can be defined as the maximum difference admissible between the backprojections on the floor [2].

The first step of the algorithm is thus to detect a sufficiently large and relevant set of image features, track them on consecutive frames and classify them using the aforementioned criteria. We chose SIFT features [11] 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 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).

3. The Navigation Strategy

3.1. Finding the Obstacle Profiles

SIFT features are usually detected at regions of high intensity variation [11]. Thus, it is likely they are near or belong to an edge. Besides, obstacle points are most likely to be contained or near a vertical edge belonging to an obstacle. Consequently, once SIFT points have been classified, the algorithm computes the edge map of the second frame of every pair of consecutive images, and then searches for all edge pixels which are inside a window centered at every obstacle point. Then, every edge is tracked down starting from the object point position until the last edge pixels or a ground point is found, considering them to be the point(s) where the object rests on the floor. This permits discriminating the obstacle boundaries from the rest of the edges.

3.2. Building the Local Occupancy map

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

$$dcp = \sqrt{(x - X_0)^2 + (y - Y_0)^2}$$
$$\phi = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right) \quad (1)$$

where $(x, y, 0)$ are the point world coordinates, (X_0, Y_0, Z_0) are the camera world coordinates at the moment of evaluating the distance between the camera and the ground point, $\vec{a}$ 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 1.

The orientation and distance of obstacles with respect to the robot can then be qualitatively estimated using (1) to compute the distance and orientation between the camera and those obstacle points in contact with the floor.

A histogram accounting for the number of obstacle-to-ground contact points at each polar direction of the ROI is used next to determine those polar directions free of obstacles. Our ROI is a semicircular area on the ground plane, of a fixed radius and centered at the robot position. Angles corresponding to regions occupied by a set of obstacle-to-ground contact points are labeled as forbidden and those free of obstacles are included in the set of possible next movement directions. The next movement direction is selected as the direction corresponding with the center of the widest polar obstacle-free zone.

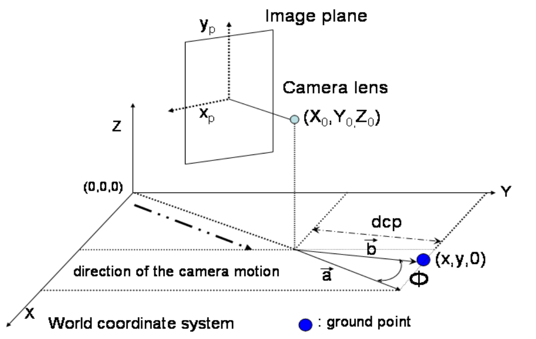

Figure 1. Distance and orientation of an obstacle point respect to the camera

4. Experimental results

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 with low illumination conditions. The ROI radius was 1.5m, the camera height was 430mm and the robot velocity was set to 40mm/sec. For each scene, the complete navigation algorithm was run over successive pairs of 0.86-second-separation consecutive frames. These frame rate was empirically found to be adequate so that the effect of the *IPT* was noticeable. Increasing the frame rate minimizes the *IPT* effect over the obstacle points, while decreasing the frame rate delays the execution of the algorithm. Frames were recorded and down-sampled to a resolution of 256×192 pixels, in order to reduce the computation time. All frames were undistorted to correct the error in the image point position due to the distortion introduced by the lens. For a varied set of scenes differing in light conditions and/or floor texture, the optimum β had a coincident value of 20mm [2]. The window used to find edge pixels near an obstacle point and to track down the obstacle contours is longer in the vertical direction to overcome possible discontinuities in the obstacle vertical borders.

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.

Figure 2 shows some examples of the navigation algorithm tested on the moving robot. Pictures (ai) and (ei)

($i = 1, 2, 3$ and 4) show, each one, the second frame of two different pairs of consecutive images. These two images are two examples extracted from the image sequence recorded and processed on-line during the route followed by the robot through scenarios 1, 2, 3 and 4, respectively. 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. Scenario 1 presents inter-reflections, specularities, and a lot of obstacles with regular and irregular shapes. Scenario 2 shows a route through a corridor with a very high textured floor, columns and walls. Scenarios 3 and 4 present bad illumination conditions, a lot of inter-reflections on the floor, and some image regions (walls) with almost homogeneous intensities and/or textures, resulting in few distinctive features and poorly edged obstacles which can complicate its detection.

Pictures (bi) and (fi) show the vertical contours (in orange) comprising obstacle points for scenes 1, 2, 3 and 4. Obstacle contours have been highlighted over the rest of the edges. The points where the obstacle rests on the floor are computed tracking down the obstacle boundaries from the position of an obstacle point until the edge finishes or a ground point is found. Then, the algorithm computes the world coordinates of these obstacle-to-ground contact points and their distance and orientation with respect to the camera. Only those points that are closer than 1'5m to the robot are highlighted in pink in the image.

Histograms of plots (ci) and (gi) account for the number of obstacle-to-ground contact points in each polar direction for scenes 1, 2, 3, and 4. Plots (di) and (hi) show the local occupancy grids in a bird's-eye view of a circular 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. The algorithm analyzes the polar histograms and defines the direction of the center of the widest obstacle-free polar zone as the next steering vector (shown in green). In all scenes, all features were well classified, obstacle profiles were correctly detected and the robot navigated through the free space avoiding all obstacles.

5. Conclusions and Future Work

We have presented a novel navigation strategy which employs an *IPT*-based feature classifier to distinguish between ground and obstacle points. The system finally builds a qualitative radial model of the robot's vicinity to determine the location of nearby obstacles. The algorithm was tested on a robot equipped with a wide angle camera and showed to tolerate scenes with shadows, inter-reflections, and different types of floor textures or light conditions.

Errors in the obstacle-to-ground contact point estimation are transmitted from the image to the range and angle calculation, and thus the qualitative local occupancy map can be slightly distorted. It might be necessary to analyze those errors to calculate the uncertainty of the range and angle estimation.

The exposed algorithm does not 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 execution time can also be different. It becomes necessary to explore different choices for detecting and tracking features to optimize our algorithm in terms of: a) number of necessary features, b) their location in the image, and c) execution time.

The exposed work is the base for a more complex robot system that has to be programmed for missions, like the exploration of unknown environments for map-building tasks, navigation between two defined points or even, for example, as a guiding robot.

References

[1] S. Badal, S. Ravela, B. Draper, and A. Hanson. A practical obstacle detection and avoidance system. In *Proc. of 2nd IEEE Workshop on Applications of Computer Vision*, pages 97–104, 1994.

[2] F. Bonin-Font, A. Ortiz, and G. Oliver. A novel image feature classifier based on inverse perspective transformation. Technical report, University of the Balearic Islands, 2008. A-01-2008 (http://dmi.uib.es/fbonin).

[3] F. Bonin-Font, A. Ortiz, and G. Oliver. Visual navigation for mobile robots: a survey. *Journal of Intelligent and Robotic Systems*, 53(3):263–296, 2008.

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

[5] Y. Choi and S. Oh. Visual sonar based localization using particle attraction and scattering. In *Proc. of IEEE International Conference on Mechatronics and Automation*, pages 449–454, July 2005.

[6] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons Publisher, 1973.

[7] J. Fasola, P. Rybski, and M. Veloso. Fast goal navigation with obstacle avoidance using a dynamic local visual model. In *Proc. of SBAI'05, The VII Brazilian Symposium of Artificial Intelligence*, 2005.

[8] S. Goldberg, M. W. Maimone, and L. Matthies. Stereo vision and rover navigation software for planetary exploration. In *Proc. of IEEE Aerospace Conference*, pages 5–2025,5–2036, 2002.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2003.

[10] S. Lenser and M. Veloso. Visual sonar: Fast obstacle avoidance using monocular vision. In *Proc. of IEEE IROS*, pages 886–891, 2003.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

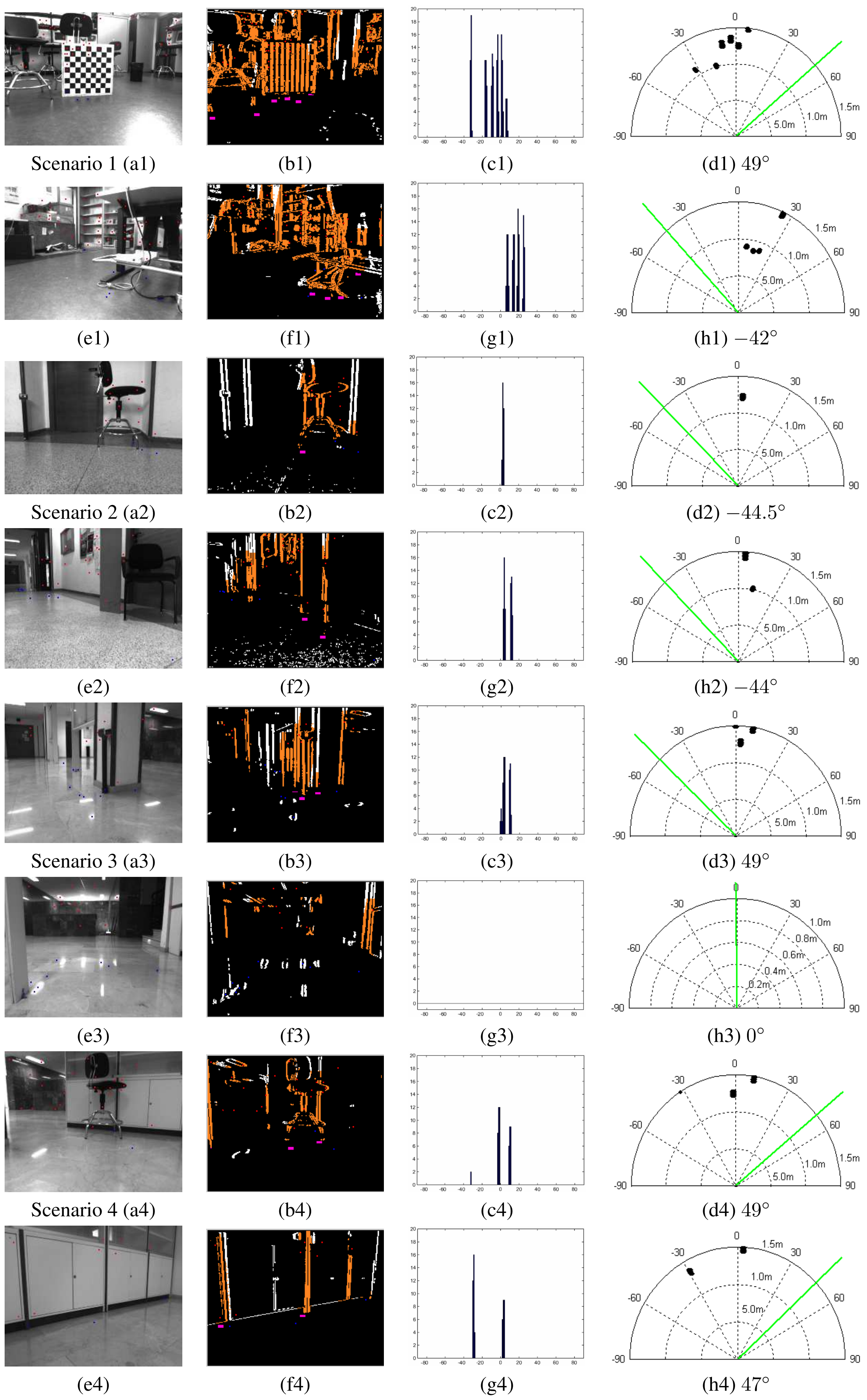

Figure 2. Examples of experiments in different environments