Abstract—Autonomous robot navigation has many applications such as space exploration (see the Mars rovers), military purposes (UGV and UAV) and autonomous vehicles (DARPA challenges). Usually such navigation is ensured by the use of multiple sensors. In this article we propose a solution for navigating a robot in an unknown environment using only monocular vision algorithms.

I. INTRODUCTION

AUTONOMOUS navigation is key to most applications of robots. It provides larger possibilities for planetary exploration (such as the Mars Rovers [1]), higher degree of interaction with humans (such as Toyota partner robot “Robina” [2]) or automatic vehicles [3][4][5][6]. Usually, autonomous systems rely on constrained environment knowledge (from type of objects to map of the floor) and/or use of a large number of sensors for variable environment detection as shown in Figure 1.

In this paper we propose a combination of monocular-vision algorithms to detect obstacles (of both known and unknown type) in the close unknown environment of an autonomous system (either a robot platform or a vehicle).

We don’t deal with the classical problem of SLAM but only with obstacle avoidance for autonomous vehicles. For this problem, we use the vision sensor to detect obstacles. Robot positioning is performed through odometry dead reckoning supposing planar ground. This hypothesis is verified (at least locally) in the conditions we run our algorithms (indoor or low speed movements on road).

We successfully demonstrated use of these algorithms in an in-vehicle anti-collision system as well as to perform autonomous navigation in an unknown indoor environment. In both situations many different obstacles were present: vehicles, humans and other random objects.

II. THE PROTOTYPES

A. Prototype vehicle

For a couple of years now, we have been using a prototype vehicle equipped with a monocular vision sensor and odometry sensors offering 3D vision reconstruction of the surrounding static environment. This vehicle is a Toyota Picnic which we already presented in [8]. It is equipped with a standard NTSC camera, odometry and steering-wheel angle sensors (Figure 2).

B. Autonomous robot

The autonomous robot platform used is based on MobileRobots, Inc. PeopleBotTM platform [1]. This platform offers a quickly controllable platform tall enough (115 cm) to reproduce the camera position in vehicles. The platform was modified to fit our needs (Figure 3):

1) Ultrasonic and IR sensors were disabled as we only use vision for obstacles detection.
2) Two standard IEEE1394 CMOS cameras (with NTSC resolution) were mounted on top of the platform, one looking down used to detect small obstacles, the second one looking up to detect tall obstacles such as humans. This two cameras setup was required to cope with the limited vertical field of view of the cameras lenses. Each camera was used independently (i.e. no stereo matching was performed).
3) An additional high power PC was added to perform all vision and navigation functions leaving basic controls to

Figure 1: Urban challenge winner obstacles sensors equipment

Figure 2: Prototype vehicle setup

Figure 3: Autonomous robot setup
the embedded microcontroller (transmission of left and right wheels encoders’ data to PC, application of direct movement commands received).

4) Power circuitry was also modified so that it could power the PC.

5) A hull was designed to cover the entire robot to clearly show that only cameras were used for obstacles detection.

---

**C. Prototypes equipment**

Both prototypes are equipped with a high speed computer equipped with a quad core 3GHz CPU and 2 GB of memory. Image acquisition is performed using standard IEEE 1394a ports on the robot whereas it is performed by a Matrox Meteor II frame grabber on the vehicle. In both cases full NTSC frames are captured at 30 frames per seconds. Odometry is grabbed through NI acquisition board in the vehicle. In the robot, wheels sensors encoders’ data is received through RS232 port as imposed by the platform design. A homemade dead reckoning algorithm is then used for precise positioning. On the robot, the RS232 link is used to control the robot using direct movement commands (mainly angular and longitudinal speed orders) sent to the robot microcontroller.

---

**III. MONOCULAR VISION-BASED OBSTACLES DETECTION**

We combined multiple monocular vision algorithms to make detection more robust to the different kind of obstacles that can be encountered on the road or indoor. Three categories of algorithms were combined to make final decision:

1) Emergency control algorithm to check that there is no obstacle on vehicle path (free space detection algorithm).

2) Generic algorithms to detect, position and characterize (in terms of shape) static and moving obstacles (shape from motion and movement detection algorithms).

3) Dedicated algorithms to recognize specific objects most commonly found (human and vehicles recognition).

---

**A. Free space detection**

Although we aim at detecting all obstacles soon enough to avoid them it is necessary to provide autonomous vehicles with a generic mean to immediately stop in order to avoid a very close obstacle (which hasn’t been detected before or which just entered the scene very close to the vehicle). Free space detection just provides this mean by detecting any obstacle very close to the vehicle without trying to characterize it (as the 3D shape of an obstacle is not important for an emergency stop).

Our free space detection algorithm is a variation of the classical flood fill algorithm used for road detection. The usual color similarity criterion is here replaced by a texture similarity criterion. A 3 bin HSV histogram is used to capture texture properties in 8x8 pixels blocks to produce results robust to camera noise. By detecting vertical edges (computed through probabilistic Hough transform [9]), large vertical objects are searched and used as a cue to limit free space texture propagation. Finally a 2D/3D discrimination algorithm is used to discriminate real obstacles from flat ground marks. This later algorithm relies on the analysis of shadows for each potential 3D objects. Objects with a shadow are considered as 3D obstacles (thus stopping free space propagation) whereas objects without shadows are considered as marks on the ground which can be run over (free space propagation is not stopped on such objects). Figure 4 illustrates how intensity variations are used to assess the existence of a 2D or 3D object. If object is only 2D its intensity profile is different from that of a 3D object which shadow locally reduces this profile.

---

**Figure 3: Design of the autonomous robot used**

**Figure 4: Illustration of 2D/3D discrimination technique for 2D (red) and 3D (blue) objects**

Eventually a map representing the free space in the image is generated. Analysis of the amount of free space in front of the vehicle combined with camera calibration provides information on the distance to obstacles. Too short distances to the obstacles in the vehicle path result in an emergency stop to avoid collisions (Figure 5). This algorithm has proved to be very efficient in areas with uniform/homogeneous road texture and visible shadows but is of limited use whenever such conditions are not available.

---

**Figure 5: Illustration of the results of free space detection run in an indoor environment. Amount of free space (blue area) is in direct correspondence with the proximity of obstacles. Color indicates estimated distance to obstacle (green: d>1m, yellow: 0.5m<d<1m, red: d<0.5m)**
B. Static obstacles detection using shape-from-motion

Over the recent years we have been developing a structure-from-motion algorithm as a way to gather accurate 3D obstacles information from the environment using a single camera [8][10]. Unsurprisingly we have used this technique to gather information about all the surrounding obstacles so that even unknown obstacles can be avoided. This is necessary as it is currently impossible (even using state of the art recognition technology) to recognize any type of obstacle that can appear in the path of a vehicle (as there is a quasi-infinite number of different objects that might be used as obstacles).

We have used an improved version of our shape-from-motion approach specifically developed for higher 3D data density and faster processing. This new version relies on the epipolar geometry computed from the camera movement to extract and track 3D points rather than using classical feature points detectors [11][12] which provide information only for corners in the image and can be difficult to track.

In this new algorithm feature points (characterized by color transitions – from black to white for example) are searched along “scanlines” (i.e. lines issued from epipole position in the image) in a first image and are matched to other feature points extracted from a previous image on the corresponding scanlines as shown in Figure 6 (i.e. this is made possible thanks to the epipolar constraint imposing a unique correspondence between each scanline of two images taken from different positions).

Once 3D points have been reconstructed, filtering is performed to remove outliers. These outliers can be generated by wrong matches of color transitions or degenerated edges. Filtering also removes points with too high precision uncertainty.

Filtered points are then grouped as 3D convex obstacles which characteristics (position, distance to vehicle, width and height) can be used for obstacle avoidance. Currently this grouping is performed solely using 3D measurements (distance between points, points inside obstacle convex hulls ...) and is updated with each new set of reconstructed 3D points. For obvious performance reasons, new reconstructed 3D points are streamed into former results to avoid having to compute the whole scene again each time a 3D point is added or refined.

C. Moving obstacles detection using Ground Plane Transformation (GPT)

As autonomous vehicles usually encounter static as well as moving obstacles it was necessary to add a moving obstacles detector to the robot algorithms. Thus, we decided to use our approach (previously published in [13]) based on GPT (an inverse projective mapping on the ground plane) as it fitted nicely our needs in terms of detection precision and performance (real-time detection and localization with high positioning precision). Moving objects detection is therefore done according to the flowchart presented in Figure 7:

1) Input image is first warped using GPT to compensate for vehicle ego-motion.
2) Road detection is performed using a fast region growing algorithm.
3) Already detected static obstacles are used to create obstacles masks in which movement probability will be much reduced.
4) Image differencing is then performed between the warped image obtained at step 1 and the background warped image built by background-accumulation.
5) A movement map is then created by giving a movement probability to each pixel of the warped image. The probability is computed according to:
   a. Color difference between warped image pixel value and the value of the corresponding pixel in the background warped image.
   b. Pixel belonging or not to the detected road
   c. Pixel belonging or not to one of the static obstacles map
   d. Pixel value history (stable color history or very unstable) to reduce influence of very noisy parts of the image.
6) Connected regions segmentation is then performed to create movement blobs
7) Fuzzy filtering is applied to each movement blob to remove blobs with too low average probability or improbable shapes.
8) Moving obstacles position is then given by the closest
point of each validated blob.

As we have shown in [13], this algorithm has proved very efficient at detecting moving obstacles close to the vehicle (up to about 6 meters) with a per frame detection rate of 88% and false alarms rate of about 1%. All moving objects are detected (although some take some time to be detected which explains the 88% figure). Localization precision is also very good with moving obstacles precisely localized in both lateral (average angular error: 0.03 rad ± 0.03) and longitudinal (average distance error: -1% ± 4%) positioning.

D. Vehicles and humans recognition

At this point of development, the obstacle avoidance system already provides decent 3D information (i.e. it could already avoid obstacles in most cases) on its close environment but only up to a few meters. Moreover, the previously described algorithms may sometimes fail to completely detect obstacles with low visibility or low contrast with background. Finally, obstacles detected cannot be characterized (is this a human a vehicle or something else?). Such characterization is essential for obstacles such as humans and vehicles which display particular dynamic behaviors that can be used to select an appropriate avoidance strategy.

We have added a multiple objects recognition algorithm to cope with these issues. The algorithm is built according to a generic framework described in Figure 8. An additional advantage of such a recognition algorithm is that it can recognize specific object classes much farther than generic obstacle reconstruction methods and roughly position them using either flat ground assumption or object a priori size. The architecture proposed was created to keep computational complexity relatively low so that it could run in real-time. This architecture is a first step towards vehicle surroundings understanding architecture that we proposed in [14]. As shown in Figure 8 the architecture is split in two main components:

1) Fast filtering creates a pool of regions of potential interest (i.e. image areas where a searched object might be). Camera model is used to limit the search space according to the camera calibration (as it is not useful to search humans or vehicles in the sky for example). A Haar-like recognition process is then started to further limit the search space to areas presenting Haar features compatible with the searched objects. In these areas selected edges detection and optical flow computation are performed as additional cues for recognition.

2) Fast Recognition then accesses the pool of detected regions of interest and the corresponding edges detection and optical flow computation results in a one-by-one manner.

3) A first round decision tree, previously trained on a large number of data, is used to discriminate between different classes of objects and a background. This tree uses SIPINA algorithm [15] and is aimed at achieving 0.03 False Rejection Rate (FRR – valid objects wrongly rejected) and 0.5 False Acceptance Rate (FAR – invalid objects wrongly accepted). As a result of the discrimination process binary/fuzzy decision is made together with a corresponding confidence value that is used to update total score. Negative decision results in the rejection of the current region of interest as containing the searched for object.

4) Regions accepted by the previous decision tree are then fed to an object specific recognition process used to update the recognition score for each region of interest. Figure 8 describes this process for human recognition using full body and half body recognition (this module should actually be called partial body recognition as it aims to recognize parts of the body such as face, arms...) modules. Similarly for vehicles recognition a full vehicle recognition module is used as well as additional modules aimed at detecting wheels, license plate... Those modules are able to process image areas very fast yet achieving high recognition rates thanks to the use of cascaded classifiers trained by AdaBoost [16] based algorithm.

5) Shape descriptors are also used to ensure that object found has a potentially correct shape. This is done by classical Chamfer system based on distance transform. Additionally for pedestrian detection static gait pattern
is analyzed to take advantage of the motion information. In this later case the detector is trained (via AdaBoost) to take advantage of both motion and appearance information to detect a walking person.

6) Shape information and recognition data are fed to a SIPINA based second round decision tree previously trained to achieve 0.03 FRR and 0.05 FAR.

7) A fast recognition decision tree then decides either to send the region as a valid object area if its score is very high, to tracking (to validate the target with time) or to terminate this region (if its score is too low).

8) Tracking is performed either on the current region or on a larger region if it could be merged with other neighboring regions (this is done to reduce processing complexity). A fast tracking decision tree (trained to achieve 0.03 FRR and 0.1 FAR) is then used to determine if region contains the searched object.

9) Correctly tracked regions are used to update gait pattern, optical flow and Kalman trackers that were used to validate the region.

Figure 8: Block diagram of the objects recognition algorithm used (human detection example)

This recognition strategy resulted in very good recognition results (as shown in Figure 9) throughout the tests performed with 0.1 FRR and 9.10^{-3} FAR. On average, this algorithm currently requires about 0.5 seconds to recognize a new object (as tracking must initialize on a few frames to be efficient) while managing correct recognition up to 20 meters using standard grade in-vehicle rear camera (NTSC camera with wide field of view ~130°).

Figure 9: Illustration of vehicles and humans recognition results

IV. OBSTACLES AVOIDANCE STRATEGY

As presented previously, automatic avoidance can only be performed on the robot as the prototype vehicle is not equipped with the necessary actuators.

A. System specific constraints

There exists a huge amount of strategies for automatic obstacles avoidance in the literature. Each of these strategies was designed according to some specific constraints of the system (CPU load, robot dynamics, sensors range…). Such constraints also exist in our robot, induced by the hardware platform or by the obstacle detection algorithms used.

1) Hardware platform induced constraints

As presented in section II.B, the robot is controlled by a microcontroller receiving a limited set of commands at a fixed frequency. This reduced set of commands results in a limited set of potential trajectory elements (straight lines, arc circles and heading changes) whose characteristics can only be changed at fixed intervals (every 100 ms usually). Finally, a small trajectory drift exists caused by wheels slipping and slightly different wheels diameter. Obstacles avoidance should thus use only trajectories made of those basic elements and support not exact trajectories as movement orders are not processed immediately.

2) Vision algorithms induced constraints

The generic obstacle detection algorithm used relies on the shape-from-motion paradigm. It thus requires the cameras to move sufficiently to enable correct obstacles reconstruction. Moreover, images need to have a sufficiently large overlap to provide information on most of the environment rather than on a very limited area. Additionally, the obstacles avoidance strategy should keep in mind that only obstacles in the camera field of view can be detected, the robot should thus avoid large fast heading changes. Finally, the chosen strategy should be able to cope with streamed obstacles (i.e. all obstacles are not detected at once and detected obstacles can be refined with time).

B. Environment map building

In order to successfully navigate in its unknown environment, while respecting the previous constraints, the robot has to build a map of its close surroundings. This is performed prior to each new trajectory check or computation to ensure that robot trajectory is always compatible with the detected obstacles.

Each detected obstacle occupies an area in the environment map that corresponds to its position relatively to an initial reference position. Robot position is also computed according to this reference position. Each such obstacle area is surrounded by a “forbidden area” which takes into account robot size as well as the uncertainty inherent to the vision algorithm that produced the obstacle. Obstacles recognition is given the lowest positioning confidence whereas movement detection is given the highest confidence. As a result, obstacles detected by object recognition have a larger forbidden area than those detected by static obstacles detector. Moving obstacles receive the smallest “forbidden area” thanks to the positioning precision of the movement detection algorithm.

As the robot displacement cannot be exactly defined,
forbidden areas are further increased slightly by a small “safety margin” that corresponds to the uncertainty of the robot control.

C. Path planning algorithm

Considering the different constraints induced by the robot and algorithms used, we decided to develop an algorithm that would ensure smooth trajectories around obstacles (to ensure that vision always has enough space to detect obstacles) and directly output trajectories as a sequence of basic robot movement commands compatible with the supported movements.

The basic concept of this path planning algorithm is to compute a direct trajectory to a given objective (which can be set randomly, by user input or by a supervisor program) using only straight lines and arc circles. Arc circles radii are forced to be larger than a given threshold value to ensure that heading changes are smooth enough for vision to always detect obstacles on the vehicle path. The trajectory is computed so that robot trajectory is tangent to obstacles’ “forbidden areas”. The trajectory is computed as follows:

1) Compute direct circular arc path to objective.
2) Check if computed path collides with an obstacle
3) If path collides with obstacle O, find tangents to O from current position
4) Check that these tangents don’t collide in another obstacles O’ otherwise repeat 3) with O’.
5) Repeat procedure from valid tangent points.

D. Robot control for obstacles avoidance

At a fixed frequency, current robot movement is checked against detected obstacles. This is performed to ensure that robot movement is done according to the planned trajectory but also to compensate the potential robot drift. If current movement is too far away from planned path or is on a collision course with an obstacle, robot is stopped in order to let path planner update the trajectory.

In parallel to this process robot trajectory is checked at another fixed frequency. As long as computed trajectory doesn’t collide with an obstacle safety area trajectory is not modified (this is to ensure smooth displacement of the robot). Similarly, if a part of the trajectory is now colliding with an obstacle trajectory is recomputed from this part (i.e. only colliding and subsequent parts of the trajectory are update). Figure 11 illustrates how trajectory is modified according to new detection results.

![Figure 10: Illustration of the way robot trajectory is computed](image1)

In order to optimize the computed trajectories the algorithm was designed to test all such possible trajectories until it finds the shortest trajectory as shown in Figure 10. Processing length is nevertheless controlled to ensure that algorithm doesn’t test too many possibilities (by forbidding trajectory loops, too long trajectories…). The optimal trajectory is then passed to robot control.

E. Robot control for obstacles avoidance

At a fixed frequency, current robot movement is checked
V. EXPERIMENTAL RESULTS

We have successfully performed autonomous navigation of our robot in many different indoor situations using mock vehicles, humans and many different objects such as card boxes, chairs, tables, bags and other kind of furniture usually found in office environment. We successfully managed 4 hours running session and the system encountered only 2 collisions with obstacles:

1) First one was caused by an incomplete detection of a small card box (cubix box 20 cm large) by the static obstacle detector and failure of the empty space detector to detect this box as a 3D obstacle (because of a too small shadow).

2) Second failure, a near collision with a chair, was caused by a delay in processing of the emergency stop order sent by the empty space detector and incomplete reconstruction of a chair’s tablet.

In-vehicle tests have also shown correct behavior. Only very small obstacles (soda cans) not being detected on-time to prevent collision. Some problems with movement detection have nevertheless been spotted with movements displaying large shadows closer to the vehicle which create too close moving targets resulting in too early collision warnings. Obviously this is not as critical as not detecting a pedestrian but we nevertheless plan on tackling this issue to make our anti-collision more acceptable to a driver. Further evaluation is required in reduced visibility conditions (nighttime or heavy rain for example) as we can expect lower performance.

The vision algorithms we have used prove that monocular vision only can provide a low-cost, easily integrated solution to accurate and reliable obstacle avoidance in limited conditions as sometimes obstacles are only partially detected. Nevertheless, many indoor environments don’t present these kinds of difficulties. Moreover, such system could actually be used outdoor in vehicles to realize a simple anti-collision system using larger safety margins. Considering the number of accidents [17] caused by improper checking during backing-up maneuvers, such a system might save many lives and avoid many crashes with other vehicles, pedestrians, trees and other objects.

VI. CONCLUSIONS AND PERSPECTIVES

We have demonstrated that it was possible to create a robust obstacles detector for low speed maneuvers using monocular vision algorithms only. Currently, the performance of this system is not good enough to compete in challenges such as the DARPA ones because of limitations of obstacles detection in difficult conditions. It nevertheless provides some very interesting applications (such as a low speed anti-collision system for commercial vehicles) while not requiring any expensive sensors (radars or laser scanners).

We now aim at improving the current system performances by making it more robust to variability of environment (cast shadows for example) and more sensitive to small obstacles to tackle our remaining issues. In parallel we plan to test our system in more difficult visibility conditions such as nighttime situations, heavy rain or snow to assess its performance in such hazardous situations. Finally, we should optimize the current algorithms so that they can be run on an embedded CPU to fully demonstrate the applicability of this monocular vision-based anti-collision system.

REFERENCES