

# 3D reconstruction of a static indoor environment by fusion of sonar and video data

J.M. Leiva, P. Martínez, E.J. Pérez<sup>?</sup>, C. Urdiales and F. Sandoval

Departamento de Tecnología Electrónica  
E.T.S.I. Telecomunicación, Universidad de Málaga  
Campus de Teatinos, 29071 Málaga  
Spain

**Abstract.** This paper proposes a new system to construct a 3D model of a real indoor environment by using merging sonar and video data. Data is collected by a moving robot exploring the environment and the 3D model is constructed on-line. Hence, while the robot navigates in the real environment, any remote PC may navigate in a virtual copy of such environment. The main advantage of the proposed system is that the combination of two different sources of information allows fast and computationally cheap construction of the model. The system has been successfully tested in unstructured real environments.

## 1 Introduction

In many applications, a exploring robot must send information about its environment to a remote machine. Human operators typically rely on visual information to control the agent. Even if robots work in an autonomous way, video information may be desired for several purposes. Unfortunately, video sequences typically present a large data volume. Transmission of such sequences through a low bandwidth channel implies large delays and low frame rates. Spatial and temporal compression techniques allow an increased transmission rate by decreasing the quality of the image. However, if a high quality is required, the compression rate is quite low and few images are transmitted per second.

Some applications rely on creating a 3D virtual model of the environment in the local PC. Video information is extracted from the model while odometric information is received from the robot through a radio link. Hence, the operator navigates through the virtual environment in the same way the robot navigates in the real one. Reconstruction of 3D environments from sensed data is a computer vision problem with important application to the area of virtual reality. Typically, 3D reconstruction relies on collecting range data from a single object, generally from a laser scanner or light striping system. Then a 3D computer model of the object is generated by: i) merging multiple views into a single registered data set; and ii) representing the data set in a compact computer representation. In this case, problems like scene segmentation and image understanding are avoided. 3D modelling of entire environments is a more complex problem because: i) there are multiple objects and, therefore, image segmentation must be performed; ii) color and textures need to be recovered; iii) all views of a given object might not be available. Typically, in this case a video-camera and a positioning device are required. Some of these systems [9] [6] achieve an excellent accuracy by using GPS localization techniques and digital CCD stereo cameras to locate the environment features. Obviously, these systems are not valid for indoor mapping. If localization relies uniquely in odometry, localization errors propagate relatively fast in absence of GPS information. Other methods [5] do not require GPS information, but they do work with a large number of cameras. Also, several cameras or, at least, several frames are required to recover a 3D shape from a video sequence (i.e. [13]).

This paper proposes a new fast and simple mapping system which combines sonar sensor information and video snapshots to build a 3D model of the environment. Sonar information and odometry are used to estimate the distance of the robot to the different objects in the environment. When such an information is combined with image data, odometric errors can be corrected. The system relies on dividing a 2D probabilistic map of the environment into segments. These segments become planes in 3D. When the robot receives a snapshot,

<sup>?</sup> Email: edu@dte.uma.es

it assigns a texture to each plane in its field of view. Obviously, objects are simplified into rectangular blocks but, in exchange, the method becomes fast enough to work on-line. The main advantage of merging sonar and video information is that a rough 3D representation may be constructed by using a single frame. It must be noted that there exist more sophisticated methods based on more complex technologies which provide better results in 3D mapping [3]. However, our goal is to achieve a basic representation in a fast and easy way, even though results are poorer than in other cases.

The mapping system has been included in an autonomous robot which explores an unknown environment. This system is controlled by the architecture presented in section 2. Section 3 describes the different stages of the 3D model generation. Section 4 presents some examples of 3D views generated from a probabilistic map and a single image. Finally, section 5 presents conclusions and future work.

## 2 The robot architecture

This work has been developed for a Nomad 200 robot from Nomadic. The robot presents a ring of 16 polaroid ultrasonic sensors. It also presents a video camera attached to a C44 frame grabber. The frame grabber is connected to an on-board 133 MHz Pentium PC by means of the ISA bus. The PC is connected via a radio Ethernet link to a local area network (LAN). The supported protocol is TCP/IP. The practical measured bandwidth is approximately equal to 60 Kbytes/s. The robot works in an autonomous way, but any remote PC connected to the LAN may receive the readings of its sensors.

The robot relies on a behaviour based control architecture. The architecture presents several modules that work in parallel and exchange data. Our architecture (Fig. 1.a) is based on the one proposed by Brooks [2], but it does not use inhibitors and suppressors. The proposed architecture is implemented in C for a Linux operating system. It follows a client/server model similar to the model in [4] (Fig. 1.b). The main difference between both architectures is that our architecture includes a specific server for each connection. Thus, parallelism is increased. This improvement is specially useful if connections are exchanging a large amount of data. In autonomous architectures, most data exchanges are usually concurrently performed. Because of our improvement in our architecture modules transmitting a low data volume do not need to wait for modules transmitting a large data volume to finish. A single server controls all servers in an automatic way. Hence, this control process is transparent to the user.

The robot may send two types of data through the link: i) the values of its 16 sonar sensors, which are coded into 16 bytes; and ii) RGB images of 256x256 pixels. If a JPG compression standard is used, we can send approximately 1 image/second. However, we need to compress and decompress each image and the frame rate is still low. Hence, we send raw images through the link.

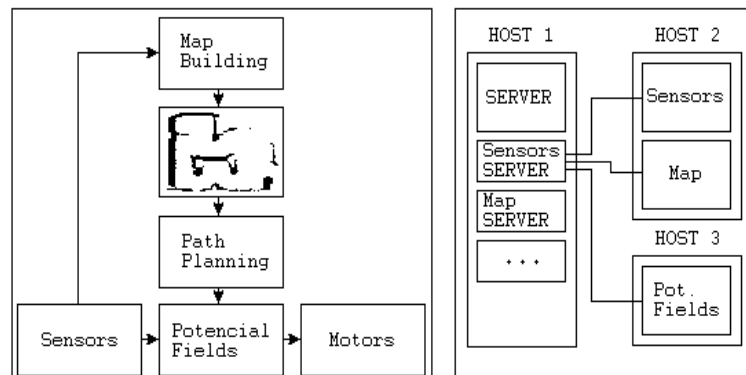


Fig. 1.: The robot architecture: a) modules of the implemented architecture; b) client-server model for data exchange.

### 3 Generation of a 3D model

In order to build a 3D model of the explored environment on-line, a client PC requests sonar and video data from the robot. In our system the robot may transmit only 1 image each 5 seconds. It must be noted that we could have used a higher frame rate, but in this case, it is not necessary to. Sonar readings are coded into 16 bytes. Hence, they may be transmitted more frequently.

The proposed control architecture keeps constantly updating a representation of the environment using the forementioned sonar readings. This representation is not built on the robot, but on the other side of the link. Thus, instead of transmitting the whole representation through a low bandwidth channel, only 16 bytes are transmitted. When a 3D model of the environment is going to be built, the client PC request the most recently updated representation of the environment to map building module of the architecture.

When a representation of the environment is available at the client PC, a planar map of the environment is extracted from it by using image processing techniques. Then, a 3D non-textured model of the environment is built from the planar map. Finally, images received each 5 seconds are used to calculate the textures and colours of the 3D objects of the model in the field of view. The following subsections present the different steps of the process.

#### 3.1 Construction of an evidence grid

Evidence grids were originally proposed by Moravec [7] to construct an internal model of the environment based on sensor data. The method deals with the uncertainty of the sensor data by working with certainty probabilities. Evidence grids are regular grids mapped over an environment where each cell holds a probability of being occupied. Such a probability is extracted from sonar sensor readings.

We use a very fast evidence grid model [14]. Each sensor presents an uncertainty arc of 15 degrees. When the sonar detects an obstacle: i) we decrease the probability of all the cells included in the sector delimited by the sensor and the arc at the detection range; ii) we increase the probability of all cells included in the uncertainty arc at the detection range. Although this system is very simple, it works correctly if sensors are read very often. In our case, sensors are read as often as possible -according to the hardware limitations- and their readings are immediately included in the grid. Its main advantage is its low computational load. Fig. 2.b presents the evidence grid built over the environment in Fig. 2.a. To eliminate sonar reflections, data values sharply different from their neighbours in a reading are ignored.

It must be noted that the grid is constantly updated by a module of the architecture. If a PC needs to build a 3D model the module transmits the most recent grid to that PC.

#### 3.2 Extraction of a valid plant model

In indoor environments objects typically tend to lie in straight lines. Such lines form the 2D planar map of the environment. The lines may be extracted from evidence grids by searching for aligned cells presenting a high probability of occupation. If the map is treated as an image, such lines may be found by means of the Hough transform. This procedure has already been successfully used for position estimation in robotic applications [12]. In this work it is used to generate an untextured 3D model of the environment.

Our procedure consists of the following steps:

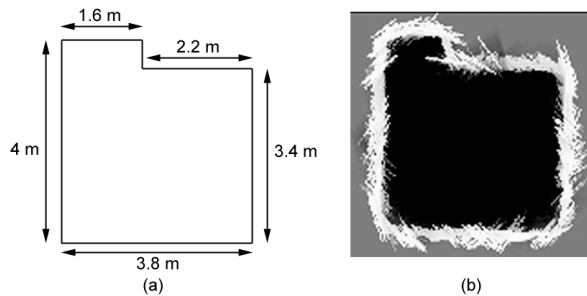


Fig. 2.: Test environment: a) layout; b) evidence grid.

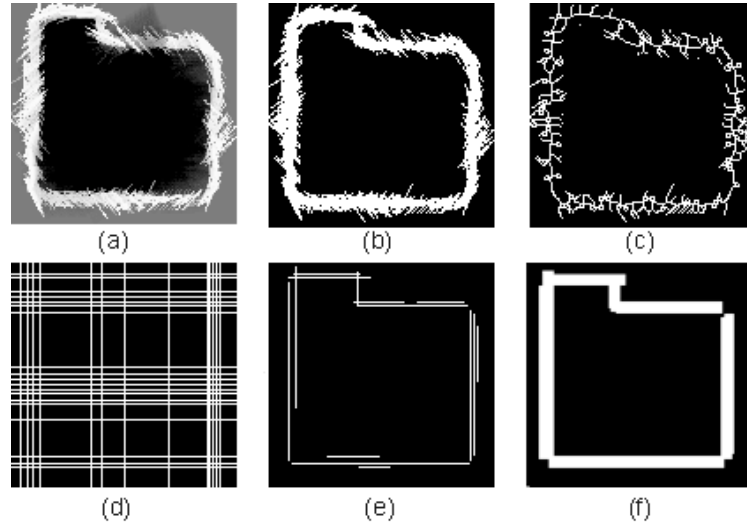


Fig. 3.: a) evidence grid; b) thresholded map; c) skeletized map; d) Hough transform of c); e) segment definition from b) and d); f) final results.

1. Binarization. This process is used to remove uncertainty areas from the map because only obstacles and free space are going to be represented in 3D. There are several methods to binarize a grid [11], but the simplest and fastest one relies on thresholding. In this case, all cells whose occupation probability is over a chosen threshold are set to 255. The rest of the grid is set to 0 (Fig. 3.b). It can be noted that the grid is now equal to a raw binary image.
2. Skeletization. This process is used to prepare the binarized map for the Hough transform. It erodes the map until lines become 1 pixel wide. To do that, every pixel belonging to a boundary is removed if no discontinuities are provoked by its removal [11] (Fig. 3.c).
3. Hough transform. The Hough transform [1] is used to detect straight lines in the skeletized binary map. A straight line may be described as  $y = ax + b$ . Such a line becomes point  $(a; b)$  of the Hough plane. Any given point may belong to infinite straight lines described as  $y = a_k x + b_k$ . The Hough table is equal to a matrix whose rows and columns are  $a_k$  and  $b_k$ . Resulting lines are given by the maxima of the matrix (Fig. 3.d). This process gets more computationally expensive when the number of slopes to be analysed is increased. Hence, instead of performing the procedure for different alignments, the global alignment of the map is estimated and the map is rotated so that walls are horizontal or vertical. This approach is valid for most indoor environments.
4. Segment detection. This process is achieved by performing the AND logic function between the Hough transform of the skeletized map and the original binarized map (Fig. 3.e).
5. Segment growing. To correctly represent walls and columns, their width must be extracted from the original binarized map. Hence, vertical segments resulting from the previous step are horizontally displaced until a given percentage of their length is out of the occupied area. The process is similarly performed for horizontal segments. When the process is finished, segments are located at the boundaries of walls and columns (Fig. 3.f).

Once a valid 2D planar map of the environment is available, the untextured 3D model is generated by assigning the same height to every segment of the 2D model. Hence, each segment is now associated to a block. Each plane of the block presents a data structure including: i) its size; ii) its position; iii) its texture; iv) a vector normal to it. The texture field is initially empty.

### 3.3 Texture mapping

Each time an image arrives, it can be used to texturize the objects in the field of view. Texture estimation has often been used to segment real images into homogeneous texture areas. In this work no segmentation process is performed. Instead, the different planes to texturize are extracted from the planar model of the environment. Then, the different areas of each plane are studied to calculate their textures. It must be noted

that not every pixel of the image is studied, so that the process can be conducted in a fast way. This process consists of the following steps:

1. Estimation of the position of the robot in the environment. This position is given by the localization module of the control architecture. The localization module depends on odometry. Hence, the estimation may be wrong because of slippage. Initially, we know the relative position of the robot in the available map. Also, we have the most recently acquired image and actualised sonar readings. The  $(x;y)$  position of the robot may be corrected according to the sonar readings. However, rotational driftings are more difficult to correct. We use a process similar to the one proposed in [8], but in our case visual information is fused with the sonar based estimated layout of the environment. Hence, rotation driftings are corrected as follows:
  - (a) Perform an edge detection procedure to detect sharp horizontal color changes in the image. This procedure returns the vertical edges of the field of view.
  - (b) Calculate the vertical edge in the field of view closest to the robot in the 3D model.
  - (c) Search for a dominant edge in the received image in the vicinity of the edge calculated in the previous step.
  - (d) Correct the position data according to the difference between the edges closest to the robot in the virtual model and the real environment.
2. Selection of the planes to be texturized. The process works with all non-texturized planes in the 3D representation which are visible in the received image. Only planes which are parallel to the vision plane or present small perspectives are texturized. They are chosen by rejecting planes whose normal vector conforms an angle larger than 10 degrees with the vision axis.
3. Texture recognition in each studied plane. This process consists of the following steps:
  - (a) Crop the largest rectangular area of the received image contained into the studied plane. Note that small perspectives are allowed. Therefore, the cropped area is not necessarily equal to the area of the plane.
  - (b) Estimate local binary patterns and contrast patterns (LBP/C) [10] for each point of the cropped image.
  - (c) Divide the cropped image into 4 rectangular regions and compare their LBP/C distributions by using the G statistic [10]. G is used to detect if two or more regions present the same texture. When two or more areas present the same texture, only one of them is studied to extract that texture.
  - (d) If several areas presenting similar textures are identified, one of the areas is divided again into four new regions and their LBP/C distributions are compared again using G. This process is repeated until all the areas present different textures. The process also stops if areas are smaller than  $16 \times 16$  pixels. That size is the smallest one required in [10] for texture characterization.
  - (e) The portions of the image included in areas presenting no texture become snapshots. These snapshots are attached to the object, but it must be noted that they are not textures. Hence, they are not extended to the whole object. In this case, those parts of the object out of the field of view are not texturized.

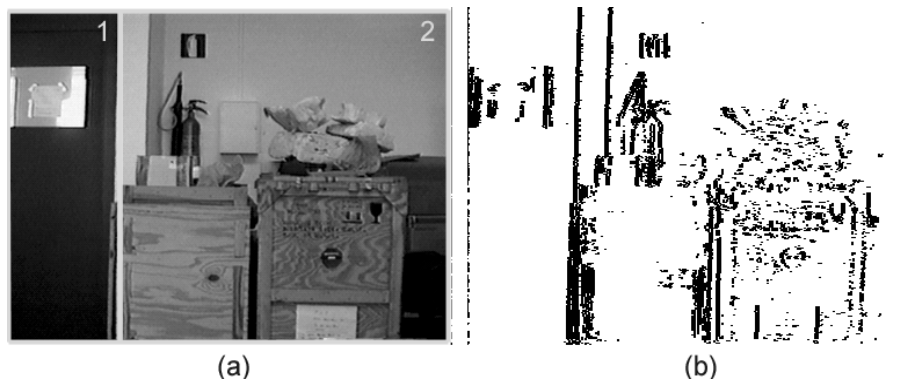


Fig. 4.: Texture mapping: a) original image and detected texture areas; b) edge detection for position correction.

Fig. 4.a presents a real image captured in the upper left corner of the environment in Fig. 2. First, edges are detected (Fig. 4.b). Then the robot position is corrected according to the 2D planar map (Fig. 3.f). When the planes to texturize are selected, textures are extracted from the image. Real environments tend to present too complex layouts to extract simple pattern textures. Hence, no textures are detected in the example in Fig. 4.

## 4 Experiments and results

The proposed algorithm has been tested in the robotic architecture described in section 2. The robot operates in a typical indoor laboratory. The room layout is presented in Fig. 2.a. Since the laboratory is not very large, the probabilistic map in Fig. 2.b is quickly constructed. When the robot is at the north-west corner of the room, it transmits a single image. That image is presented in Fig. 4.a. A client PC receives the image and request the map of the environment to the architecture server. When the map is available, it extracts a valid plant model from that map (Fig. 3). After correcting its position, the robot knows which 3D objects are perceived in the received image. Textures are only extracted from the planes of the environment which are orthogonal to the camera axis.

The proposed method can not find valid textures in the received image because no valid pattern is detected at that distance from the objects. Instead, two snapshots are captured: one from the visible door area on the left (1) and one for the boxes area on the right (2) (Fig. 4). It must be noted that it is not possible to perceive that there is a wall behind the boxes using only the 2D map information. Hence, the boxes and the wall are included in the same plane and they share a single snapshot. The snapshots are associated to the corresponding planes of the environment.

After this process is performed, no further sonar sensor readings nor images from the robot are received. The robot keeps on moving and its new position is transmitted to the client PC by the architecture server. Using the stored information, the client PC generates a new 3D view at such a position. The view is generated by using the OpenGL library.

Fig. 5.a is generated when the robot is at the east of the position where the real image was captured. It can be noted that the door is not texturized. Therefore, it appears in black in all three cases. It can also be noted that the boxes occlude partially the door plane. No depth information is available about the wall behind the boxes. Hence, the wall is assumed to be in the same plane than the boxes and it also occludes the door.

Fig. 5.b is generated when the robot is at the west of the position where the real image was captured. No texture information is available for the short plane segment between the door and the boxes. Therefore, it appears in black. Also, no texture information is available for the east wall, which appears in gray.

Similarly, Fig. 5.c presents the west wall in gray because no texture information is available for it. In this figure, an interesting feature may be noted. The system is aware that the door plane was not totally included in the received real image, because it can estimate the wall size from the acquired plant model. Thus, instead of mapping the door snapshot to the whole wall, it just maps it in a partial way. The rest of the door remains untextured. To texturize the whole wall, new real images are necessary.

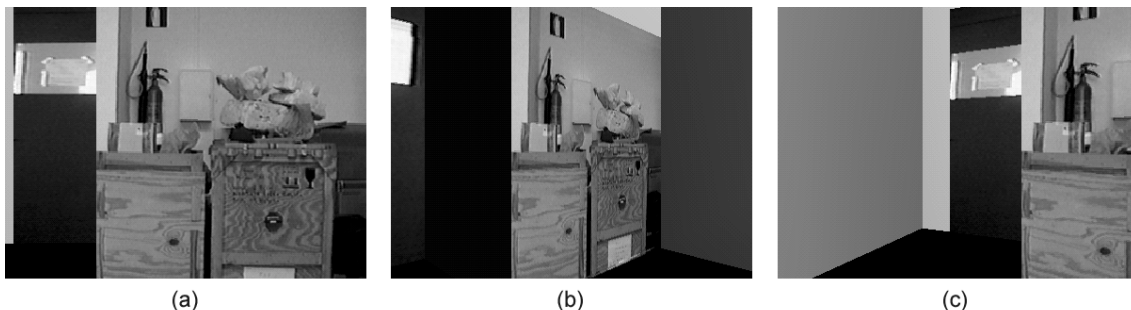


Fig 5.: Three different views of the generated 3D model.

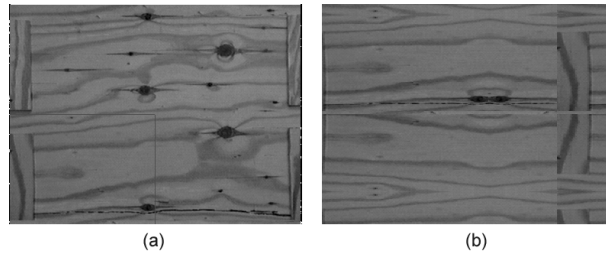


Fig. 6.: Texture mapping: a) received real image; b) texturized virtual object.

Also, the robot sent a second image while it was much closer to the wooden box (Fig. 6.a). In this case, a texture for the whole box is correctly calculated. Fig. 6.b shows the generated texturized virtual object. The former boxes plane model is corrected by adding a new square planar object superimposed to it. This new plane object presents the detected texture.

Further tests were performed in a corridor outside the laboratory. The complete probabilistic map of the corridor is shown in Fig. 7.a. After it has been processed, the resulting map is shown in Fig. 7.f.

It must be noted that it is not necessary to have the whole corridor explored to create a map. Fig. 8 shows the map available at different exploration stages and the associated 3D views from the viewpoints marked in Figs. 8.a to c.

It can be appreciated that in these examples no texture is mapped in the 3D world. As the robot starts to capture images, the model is improved with textures. First of all, the floor and the ceiling textures are captured. Fig. 9 shows three examples of corridor views after mapping the floor and the ceiling are mapped with textures.

Many items in the corridor do not yield a clear texture. In this case, as it has been previously commented, a snapshot of those items is captured and it is mapped as a whole in the 3D model. Fig. 10 shows an example of this mapping technique.

The 3D model becomes progressively complete as long as images arrive from the robot. Fig. 11.b shows how the 3D model is improved after adding the information in capture in Fig. 11.a. Then, a door texture is added in Fig. 11.c. Even though the texture mapping process is still very basic, it can be observed in Fig. 11.d that results are still adequate for different points of view.

Finally, Fig. 12 shows several views of the corridor after adding the texture information in 6 different images. It can be noted that the model is not complete yet. Nevertheless, with only 6 images the remote user can get an idea about the layout of the environment from any given position.

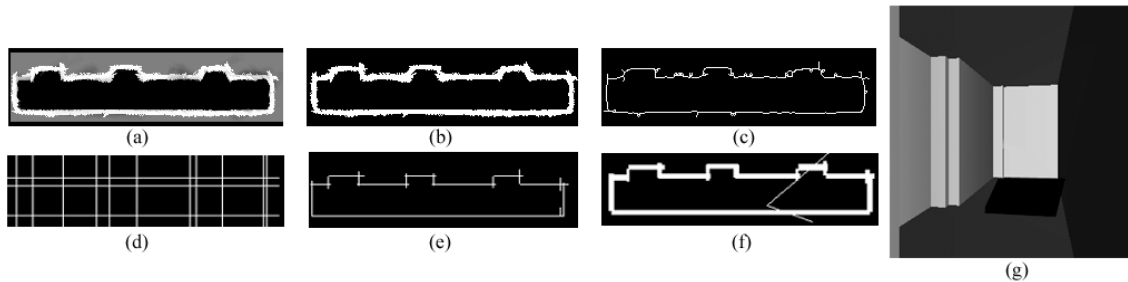


Fig. 7.: Map building: a) probabilistic map; b) binarized map; c) skeletized map; d) hough transform of the map; e) segment detection; f) segment growing; g) 3D view from the viewpoint in f.

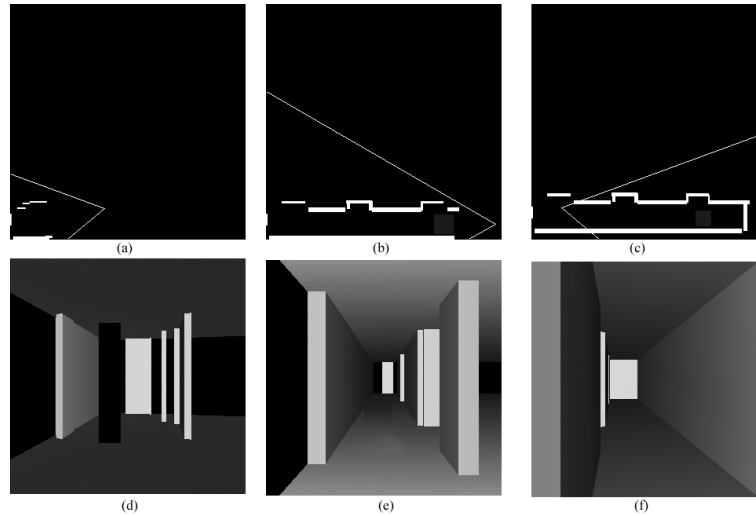


Fig. 8.: Progressively acquired maps and their associated 3D views.

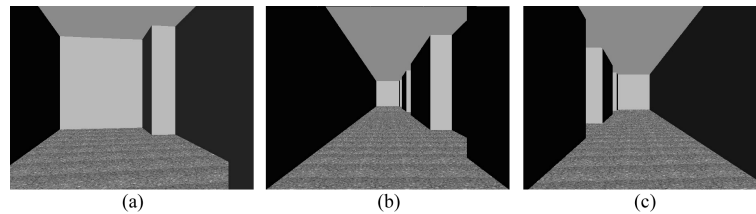


Fig. 9.: Three views of the corridor after mapping the floor and the ceiling.

## 5 Conclusions and future work

This paper has presented a new fast and easy method to build 3D models of real indoor environments. The main advantages of the proposed method are that: i) no prior knowledge about the environment is required; ii) no 3D object library is required; iii) the model is built on-line; iv) the method does not present a high computational load. The main disadvantages of the method are that: i) only regular blocks are generated; and ii) depth is only estimated at the height of the sonar sensors.

Future work will focus on the following points: i) depth estimation at different heights by using several frames or binocular vision in the robot extreme; ii) 3D object recognition to create a library of non-regular objects in the environment; iii) mapping of moving obstacles in the environment by implementing a better image capture strategy; iv) construction of a new module to move the head of the robot in an intelligent way to texturize as many planes as possible.

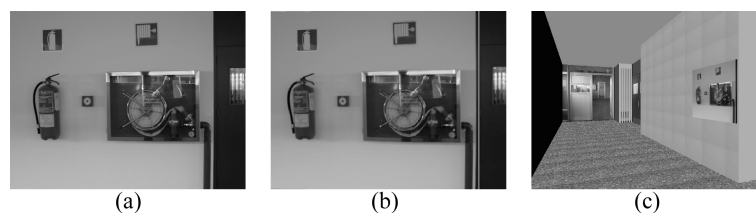


Fig. 10.: Snapshot mapping: a) image capture showing a fire extinguisher; b) 3D model after mapping the snapshot from the same point of view; c) 3D model after mapping the snapshot from a different point of view.

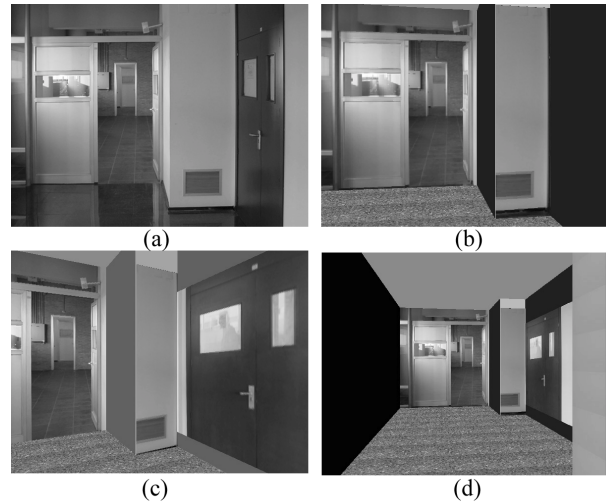


Fig. 11.: Combined mapping: a) image capture; b) 3D model after mapping a snapshot of the door and the wall; c) 3D model after mapping a door; d) 3D model in c from a different point of view.

## A cknow ledgements

This work has been partially supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT), project number TIC098-0562.

## R eferences

1. D.H. Ballard and C.M. Brown, Computer Vision, New Jersey: Prentice-Hall Inc. 1982.
2. Brooks, R.A., "A robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, R A -2 (1), pp. 14-23, 1986
3. Dias, P., Sequeira, V. Goncalves, J. and Vaz, F., "Automatic registration of laser reflectance and color intensity images for 3D reconstruction", Proc. of SIRS'2000, pp. 191-197, 2000
4. Dulimarta, H.S. and Jain, A.K., "A client-server control architecture for robot navigation", Pattern Recognition, 29 (8), pp.1259-1284, 1996
5. S.F., El-Hakim, P. Boulanger, F. Blais and J.A. Beraldin, "Sensor Based Creation of Indoor Virtual Environment Models", Proc. of Int. Conf. on Virtual Systems and Multimedia (VSM '97), Geneva, Switzerland, September, 1997.
6. Heister, H., Caspary, W., Hock, C., Klemm, H. and Sternberg, H., "The mobile mapping system - KISS", Integrated Acquisition and Interpretation of Photometric Data Workshop, Stuttgart, 1995.
7. H.P. Moravec, "Sensor fusion in certainty grids for mobile robots", AI Magazine, 9, pp. 61-74, 1988.
8. J. Neira, M. I. Ribeiro and J.D. Tardes, "Mobile Robot Localization and Map Building using Monocular Vision" 5th Int. Symp. on Intelligent Robotic Systems'97, pp. 275-284, Sweden, 1997.
9. K. Novak, "Mobile mapping technology for GIS data recollection", Photogrammetric engineering and remote sensing, 61 (5), pp. 493-501, 1995

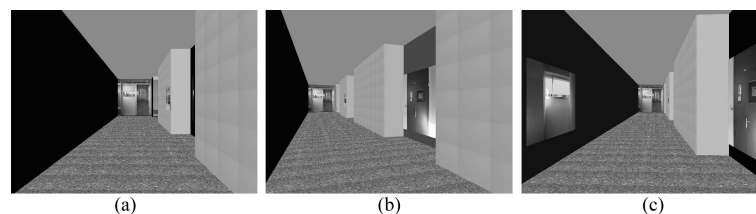


Fig. 12.: Three views of the corridor after mapping several textures.

10. T. Ojala, and M. Pietikinen, "Unsupervised texture segmentation using feature distributions", Pattern Recognition Letters, 32, pp. 477-486, 1999.
11. I. Pitas, Digital Image Processing Algorithms, New York: Prentice Hall, 1993.
12. B. Schiele and J. Crowley, "A comparison of position estimation techniques using occupancy grids", Proc. of the 1994 IEEE International Conference on Robotics and Automation, pp. 1628-1634, San Diego, CA, May 1994.
13. H-Y. Shum, M. Han and R. Szeliski, "Interactive construction of 3D models from Panoramic Mosaics", Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 427-433, Santa Barbara, (June) 1998.
14. C. Urdiales, A. Bandera, F. Arrebola and F. Sandoval, "Multi-level path planning algorithm for autonomous robots", Electronic Letters, 2 (34), pp. 223-224, 1998