The Caboto Project

...born in Venice,
went to Britain to navigate...

Emanuele Menegatti

MSc in Artificial Intelligence
Division of Informatics
University of Edinburgh
2000

Abstract

This dissertation describes the implementation of an omni-directional vision system as sole sensor for a robot building a topological map of an indoor environment, using the Spatial Semantic Hierarchy created by Benjamin Kuipers [Kuipers 00]. So far, no attempt to implement the SSH using a vision sensor has been made. We will show that an omni-directional vision system is a good sensor for the SSH.

We built a simulator for the omni-directional vision system. Using this simulator, we created typical omni-directional images of an indoor man-made environment. From a qualitative analysis of these images, we identified the transitions in the image sequence and the features in the images them-selves which are strictly related to the SSH representation. We designed algorithms to extract this information from the images. Finally, we tested these algorithms with simulated experiments and using a real robot fitted with an omni-directional vision sensor.

The omni-directional sensor is realized with a multi-part omni-directional mirror and a colour camera under the mirror. We used the colours to solve the correspondence problem in the frame sequence. Our approach worked properly in simulation but was not robust enough to cope with very noisy images in the real world. More sophisticated solutions are outlined. We present detailed descriptions of the image processing and of the matching algorithm used for the solution of the correspondence problem.

Al mio Amore Maria Grazia, "grazie di esistere" e di avermi aspettato per un anno.

Alla Mia Mamma e al Mio Papà, grazie perché tutto ció che sono oggi è merito loro.

Acknowledgement

- To Prof. Bob Fisher for his advices and his support.
- To Dr. Mark Right for the usefull suggestions.
- To Dr. John Hallam for his human touch.

Contents

List of Figures

Chapter 1

Background

1.1 Mobile Robotics

A mobile robot is a robot able to perform its tasks in different places and to move from place to place. Often, the robot does not need a map to perform a simple displacement, especially if the environment it travels through is very simple or highly engineered. But, if the robot's task requires an *understanding of the world*, the robot has to answer the three questions posed by Levitt and Lawton [Levitt & Lawton 90]:

- Where am I?
- How do I get to other places from here?
- Where are other places relative to me?

In other words, it needs a map of its *world*. With the term map we do not refer only to the geographical maps humans are used to. There are a wide range of different maps a robot can use. Two examples that are poles apart are metric and qualitative maps. In metric maps the space is represented in a single global coordinate system. In the qualitative maps, the environment is represented as a set of *places* and *paths* connecting these places. There

is no metric or geometric information, such as distances, angles, etc. but only the notion of proximity and order [Kortenkamp *et al.* 98]. Different kinds of maps answer the three basic questions using different properties of the environment. We have to keep in mind that the distance which separates two objects is only one of the properties of the space in which the two objects are embedded. The choice of which property to exploit and therefore which kind of map to use depends on the task the robot will be required to perform. The task of moving the robot using a map is called *navigation*.

What if there is no map? Can the robot be induced build its own map? This idea opens another stream of research: the map building task. In this case, the robot has to travel through an unknown and unexplored environment and to construct a map of it. Again, depending on the task these maps can be very different. If the robot has to produce a map useful for humans, a metric map will have to be composed at the last stage of the process . Moreover, the robot travelling through an environment could also gather data which would enable it to build a 3-D model and not just a map of the environment [Ishiguro *et al.* 95]. On the other hand, for tasks in which the map is going to be used only by the robot, the human readability is not an issue, a wide range of possibilities using the more cryptic features have been investigated [Bianco & Zelinsky 99] [Franz *et al.*]. One of the most effective representations of an environment is the so called *topological map*. This is a qualitative map which extracts from the environment the topological relationships between the different places and paths. The advantages of such a representation are its compactness, because it represents only interesting places and not the whole space, and its intrinsic solution to the problem of movement uncertainty. This is because when a robot goes to places, its position is known with a certain error and this error accumulates while it moves. Because topological maps do not rely on a global coordinate system, the error in position is reset whenever the robot reaches one of the distinct

places identified within the space. One of the key issues in the generation of topological maps is the abstraction from the continuous sensory experience, to a discrete set of distinct places. These are the basic elements on which to build the topological map. Topological maps have been used in other areas of robotics. One example which has been influential on this research is work of Wright and Deacon [Wright & Deacon 00]. Topological maps can be upgraded to metric maps by adding metric information to the places and to the relationship between paths and places. Therefore, a map can be seen as a hierarchal structure built layer by layer. Benjamin Kuipers created a formalisation of this intuition: the Spatial Semantic Hierarchy (**SSH**).

1.2 The Spatial Semantic Hierarchy

The SSH is a model of the knowledge of large-scale spaces of humans, intended to serve as a "method for robot exploration and map building" [Kuipers 00]. The SSH is made up of several layers: the sensory level, the control level, the causal level, the topological and the metrical level. Each layer can be implemented independently, even if they strongly interact. Let us see in details what each layer is about:

- The *Sensory Level* is the interface with the agent's sensory system. It extracts the useful environmental clues from the continuous flux of information it receives from the robots' sensors.

- The *Control Level* describes the world in terms of continuous actions called control laws. A control law is a function which relates the sensory input with the motor output. Each control law has conditions for its appropriateness and termination. A selected control law is retained until a transition of state is detected. These transitions can be detected with a function called a *distinctiveness measure*. The distinc-

tiveness function must be identified depending on the sensor used and the features which are to be extracted from the environment.

- The *Causal Level* abstracts a discrete model of the environment from the continuous world. This discrete model is composed of *views*[1] , *actions* [2] and the causal relations between them. At this stage causal maps and planning are possible using these three basics elements. For this purpose, it is convenient to classify actions into two categories: *travels* and *turns*. "A **turn** is an action that leaves the agent at the same place. A **travel** takes the agent from one place to another" [Pierce & Kuipers 97].

- The *Topological Level* represents the environments as places, paths and regions, with details of how they are connected or contained one in the other. To use Kuipers words:

  > *The topological model of the environment is constructed by the non monotonic process of* ***abduction****, positing the minimal set of places and paths needed to explain the regularities observed among views and actions at the causal level.*

- The *Metrical Level* augments the topological representation of the environment by including metric properties such as distance, direction, shape, etc. At this stage, it is possible to build a global geometric map of the environment in a single frame of reference. This may be useful, but is seldom essential.

So far, the SSH has only been implemented on either simulated robots or real robots with very simple sensors as sonars. No attempt to use a vision sensor has been made.

[1] A view is defined as the sensor's reading at a place where a transition of state is detected.

[2] An action is defined as the application of a sequence of control laws.

1.3 The aim and approach of the project

The purpose of this project is to explore the possibility of using an omni-directional vision system as sole sensor for a robot building a topological map of a man-made environment [3] , using the Spatial Semantic Hierarchy.

We started by building a simulator of the omni-directional vision system. This simulator was used in a virtual environment, an office-like building, to generate simulated omni-directional images. We performed a qualitative analysis of these simulated images. From this analysis, we identified transitions in the image sequence and features in the images them-self, strictly related to the SSH representation. We designed algorithms to extract this information from the images. Finally, we tested these algorithms both with simulated experiments and the use of a real robot fitted with an omni-directional vision sensor.

1.4 Omni-directional Vision

The term *omni-directional vision* refers to a vision system able to see in all directions. It is possible to implement this *multi-directional vision* in several ways. The term omni-directional vision has been used to indicate very different realization of *vision in all directions*, generating a mix-up in the terminology. Sometimes, it was used as synonym for panoramic vision. In the past few years the two terms have been differentiated. The term **omni-directional vision** is presently used for vision systems which able to capture a 360^o image in one shot (See Figure 1.1 and 1.2). **Panoramic vision** is used for systems which are able to produce panoramic images by panning a camera around its vertical axis and merging the single images in a panoramic cylinder, see Figure A.1. An omni-directional sensor has usually a wider angle of view in the vertical direction than a panoramic one.

[3] like an office or a building

In this dissertation, we only considered panoramas about the vertical axis. Strictly speaking, to cover exactly all directions of view, directions around an horizontal axis also have to be considered. However, the applications for such a truly omni-directional vision sensor are still limited and only a small number of researchers have created such sensors. A couple of examples can be found on the following web pages

- www.cfar.umd.edu/ larson/EyesFromEyes.html
  for an example of a 360^o eye;

- www.cs.columbia.edu/CAVE/omnicam/prototypes.htm
  for an example of a "*single sensor that has a complete sphere of view*".

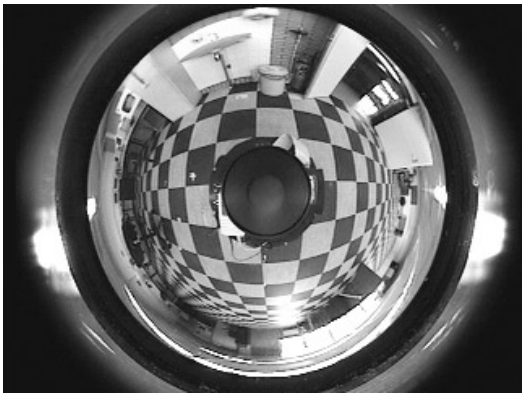


Figure 1.1: Omni-directional image. (Courtesy of B. Kröse at IAS group, University of Amsterdam)

Figure 1.2: Panoramic cylinder built from the omni-directional image.(Courtesy of Prof. B. Kröse at IAS group, University of Amsterdam)

From here on, we will therefore use the term *omni-directional* to refer to sensors with a 360^o field of view about the vertical axis.

1.5 Our specific realization

There are three basic classes of sensors which can see in all directions. They make use of different techniques:

- use of multiple images;
- use of special lens;
- use of convex mirrors;

In this dissertation we will only consider omni-directional vision sensors fitted with convex mirrors. For a review on the other type of realizations see Appendix 6.1. omni-directional sensor fitted with a convex mirror is composed of a perspective camera pointed upwards at the vertex of the convex mirror. The optical axis of the camera and the geometrical axis of the mirror are aligned. This system is usually fixed on top of a mobile robot as in Figure 1.3. The robot depicted in the picture is the goal keeper of the Azzurra Robot Team (ART)[4] . This is the robot we used for our experiments.

1.6 Why is omni-directional vision good for map building?

Omni-directional vision produces images with a wide angle of view but with low resolution. This is not a problem, because most of the task which require an understanding of what is happening in the surroundings do not need high

[4]Azzurra Robot Team (ART) is the Italian team at Robocup Championship, the robot football competition.

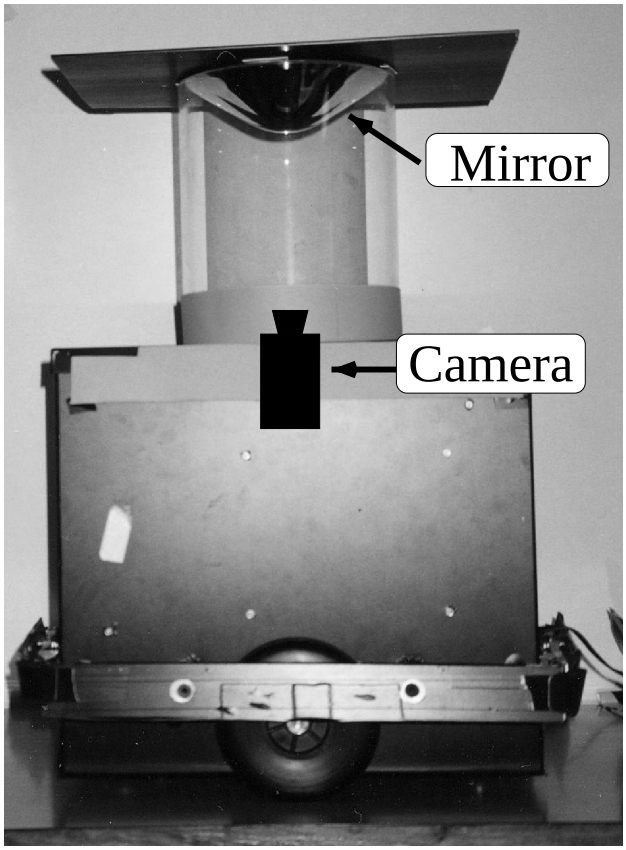


Figure 1.3: The robot with its omni-directional vision sensor.

resolution images. It is better to gather the biggest amount of information from the highest number of directions than to have a detailed analysis of a small area. In robotics several tasks do not require a detailed analysis of the scene but high speed and rough understanding of the region around the robot. Examples of these are:

- navigation;
- obstacle avoidance;
- exploration;
- map building;

- surveillance;
- telepresence;

In the case of a map building robot, the advantages of omni-directional vision are clear. For this task a vision sensor with high acuity is useless, it is not necessary to capture the details of objects and surfaces, but only to estimate their positions and dimensions. An omni-directional image captures at once all the objects visible from the robot location. This image has a strict connection with the *views* introduced in the causal level of the SSH, i.e. with the sensor reading at a distinct place. With an omni-directional sensor, the robot does not need several shots to understand the surroundings. It does not need to turn and take a look around. It does not need to be fitted with moving parts (camera or mirrors) to increment its field of view. These are just implementational considerations, there are deeper aspects supporting the use of an omni-directional sensor in the process of building a map with the SSH.

To understand these considerations we have to take a look at how a omni-directional sensor maps the scene into the image plane. Consider Figure 1.4. In this figure a conical mirror is represented, but the properties which are illustrated apply to any kind of omni-directional sensor.

The vertical edges in the scene are mapped in the image plane as radial lines originating from the point corresponding to the tip of the mirror. Therefore, to extract the vertical edges from the images, the image is searched for radial lines. The azimuth of a radial line in the image corresponds to the azimuth of the vertical edge in the scene, as viewed from the optical axis of the camera. The vertical lines in the environment provide an optimal clue to divide the environment into topologically different places and they can be used to generate a *distinctiveness measure* needed to distinguish transition of state in the robot ontology. Another advantage of the omni-directional vision

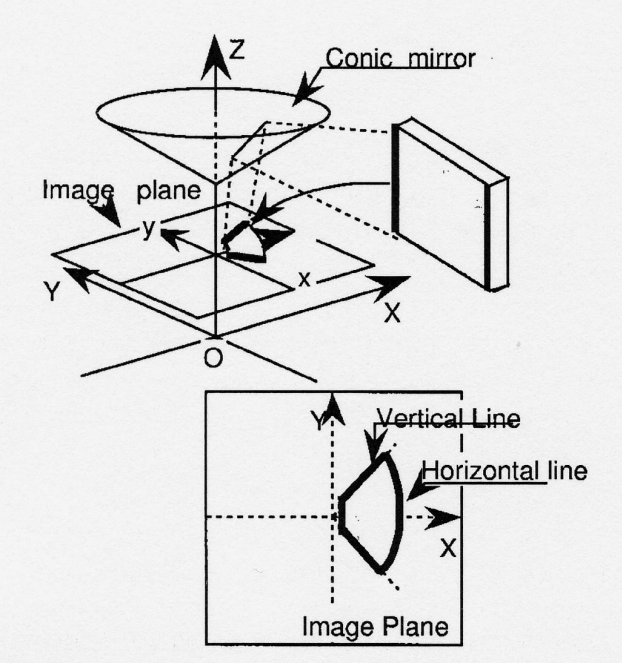


Figure 1.4: The conic projection. (Courtesy of Prof. Y. Yagi at Osaka University)

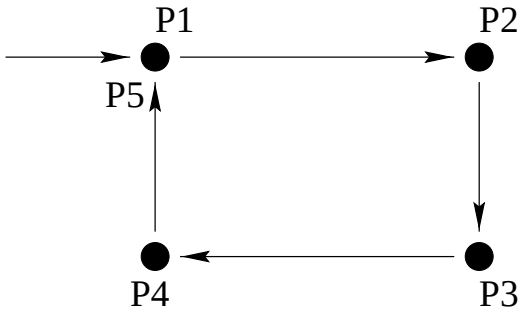


Figure 1.5: The "exploring around the block" problem. The problem of recognising the same place under different state labels.

is its rotational invariance. If the robot rotates of a certain angle about the optical axis of the camera, the relative position of the objects in the image does not change. The image is only rotated and the objects appear to have experienced an azimuthal shift equal to the angle of rotation. This permits a straightforward solution to the problem of *exploring around the block*, i.e. of recognising the same place under different state labels. See Figure 1.5. Using

the SSH terminology, it is easy to spot whether the current **view** is the same as has been experienced before and therefore to consider this view not as a different **place** but as the same place reached from a different direction.

Another problem which is easily solved by omni-directional vision is to discriminate the type of movement the robot is performing at a given time. Using optical flow techniques, Svoboda showed that with an omni-directional vision system it is very easy to discriminate between a small rotational movement and a small translational movement [Svoboda *et al.* 98]. This task is very difficult for a vision system fitted with a perspective camera. Moreover, using active vision on an omni-directional vision system it is possible to estimate precisely the motion of a robot. See again [Svoboda *et al.* 98] for a literature review.

The main disadvantage of omni-directional vision with respect to perspective vision is the already mentioned poor resolution. This is because with an omni-directional sensor light is gathered from a much wider area than with a perspective camera. Therefore, if the sensitive surface of the CCDs is the same, more points have to be mapped into the same pixel. The shape and size of the mirror influences the resolution of the sensor. Within certain limits, it is possible to design mirrors that maximise the image resolution in the most interesting regions of the scene.

Other disadvantages of the omni-directional images are the high distortion introduced into the image and the poor human readability. A mirror with a single focal point, like a hyperboloidal mirror, overcomes these difficulties. In fact the geometry of such a mirror permits transformation of portions of interest of the omni-directional image into a perspective image. See Figure A.5.

1.7 The dissertation overview

In Chapter 2 we describe the implementation of the simulation of the omni-directional sensor and the virtual environment. The qualitative analysis performed on these simulations is described and the results are presented. In Chapter 3, we introduce our idea for the vision system software and we expand it in detail, explaining which information this software is able to extract from the images and how it performs this task. In Chapters 4 and 5 we perform experiments with this software respectively on a simulated sequence of omni-directional images and on a sequence acquired with a real robot fitted with an omni-directional vision system. Eventually, Chapter 6 draws the conclusions from the project, listing what this project has achieved and what it failed to achieve. Appendix 6.1 presents a survey on the different ways of building an omni-directional vision sensor.

To pay tribute to Venice, home town of the author, this project and the real robot we will use are called **Caboto**.

Giovanni Caboto was *explorer* and *cartographer*. He lived in the second half of the XV century. Like the author, he was born in Venice and he came to Britain. He had a ship from Henry II King of England to *"navigate and discover all the unknown lands"*.

Chapter 2

The Simulator Visualisation

As reported in the previous chapter, omni-directional images are not easy to understand. They present a point of view (and a field of view) we are not used to.

The first part of the project consisted of building a simulator to generate omni-directional images of a virtual environment. The aim of this simulator was to gain an intuitive understanding of the dynamic of the sensor (i.e. how the image changes when the robot moves), and of the proprieties of omni-directional images. The qualitative analysis of the images permitted us to extract clues about the distinctiveness measure and features we should use in the process of building the map of the environment. We used a simulator for two main reasons. First, the robot on which we implemented the omni-directional vision system was not available at the University of Edinburgh but only at the University of Padova (Italy) and it was therefore not possible to carry out extensive preliminary tests. Second and more essential, the simulations provided an environment easily controllable and reconfigurable. Moreover, the absence of noise permitted us to focus on the understanding of basic features rather than on implementational problems.

2.1 Building the simulations

To generate simulation of omni-directional images we used a ray-tracing program called **POV-Ray**. This is a free program for creating three-dimensional graphics. It is downloadable at the web site **www.povray.org**. POV-Ray permits the creation of a complex 3D virtual environment from a simple text file containing the description of the scene. In POV-Ray the basic elements of the world are: the camera, the lights and the objects. The camera represents the view of the observer of the scene. The lights are necessary to illuminate the scene and the objects present in there. The objects can be simple primitives (like spheres, cylinders, boxes, etc.) or compound objects of these primitives. An object can be defined with several attributes. Some of them are required, like its position and its colour, others are optional like the amount of reflection or the roughness of its surface. The POV-Ray language allows the application of transformations to the objects. The transformations can be both spatial (rotation, translation, etc.) or morphological (scale, ratio, etc.). The POV-Ray language provides also some commands to create sequences of images. In fact, specifying the position of an object as a function of the time, the object can be displaced from frame to frame.

Using POV-Ray we created a virtual environment in which to carry out the simulations. A simulated robot moves in this environment taking snapshots with its omni-directional viewer. The environment is a basic model of a man-made environment like an office or a building. This is a maze composed of boxes of different colours. See Figure 2.1. The maze is designed to present to the robot typical views of an indoor space. Typical views are corners, doors, corridors and convex objects (like cabinets or boxes). The robot is represented as a red square with a white sphere on top of it. The robot moves through the maze along a predefined path to encounter all the typical views. We generated a sequence of simulated images captured by the

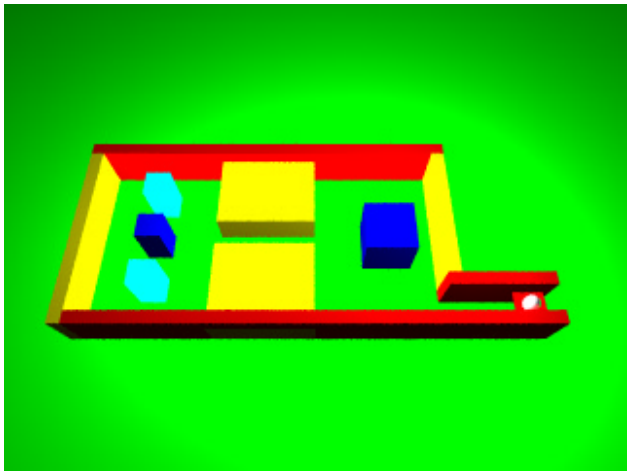


Figure 2.1: The virtual environment generated for the simulations of the vision system. The robot is the red square with the white sphere on top of it.

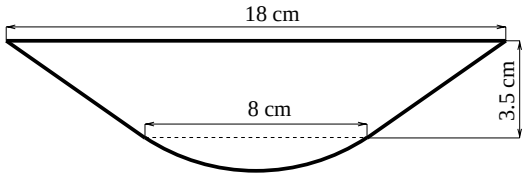


Figure 2.2: The Mirror Profile.

omni-directional viewer while the robot moves in the maze.

To simulate the omni-directional viewer we built a model of the omni-directional sensor present on the real robot of the University of Padova. The mirror of the real sensor is a multipart-mirror composed of a cone intersecting a sphere. It has a hybrid shape with respect the ones reviewed in Section A.3. The dimension of the cone, the radius of the sphere and the position of the point of intersection are calculated such that the cone and the sphere are tangential at the intersection point. If they were not tangential, a discontinuity would be present in the image. Figure 2.2 depicts a sketch of the

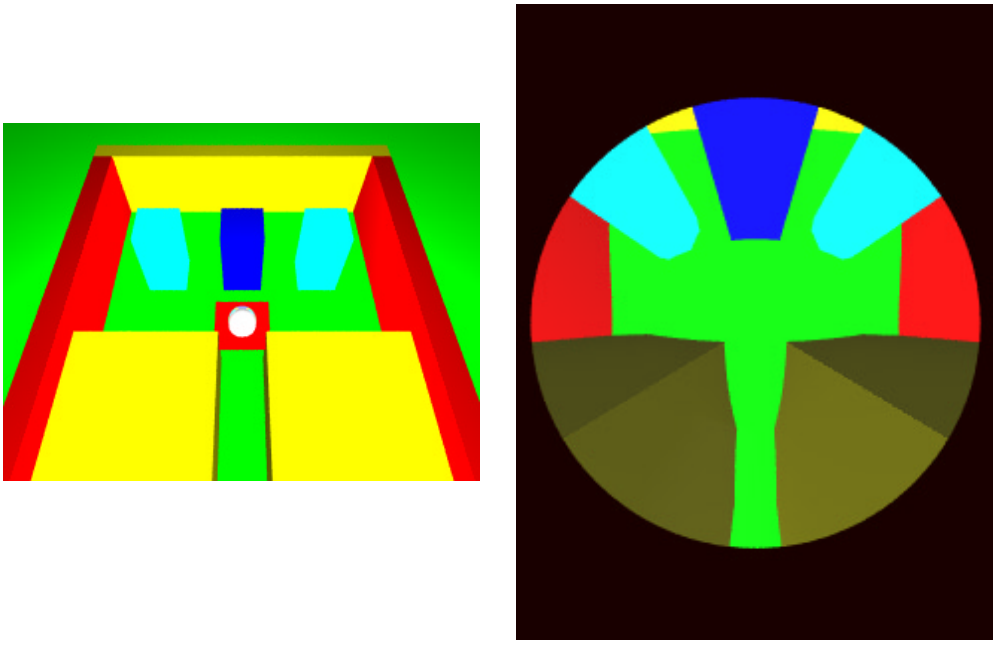
mirror profile with its dimensions.

We reproduced a model of this mirror in POV-Ray. The multi-part mirror is defined as the intersection of two primitives: a sphere and a cone. The surfaces of the two primitives are defined as totally reflective, so they behave like mirrors on which the image of the surrounding world reflects. This reflection is the image captured by a perspective camera placed under the mirror and pointing upward to the mirror.

In Figure 2.3.a we show a perspective view of the simulated environment and in Figure 2.3.b how the same scene is seen by the omni-directional viewer. Note that the body of the robot does not appear in the simulated images. We simulated only the viewer not the whole body. In the next paragraph, we will explain which kind of feature we extracted from the omni-directional images to understand the surrounding environment.

2.2 The feature selection

To extract from the images the information about where are the objects in the environment and where the robot is going, we need to select a feature, or a set of features, to search for. These features must be present in the environment or in the pictures of the environment and must be reliably, and possibly easily, detectable. First, we have to decide if we want to search for features naturally present in the scene or if we want to exploit the use of artificial landmarks. The task for which this robot is designed: map building, presumes the existence of an unknown and unexplored environment. Therefore, we have to discard the use of artificial landmarks. Several author selected features that strictly speaking are not present in the environment but only in the pictures of the environment, like brightness pattern or other features only loosely related to the objects in the world. Usually, these features are extracted from the pictures with the use of heavy mathematical tools



(a) (b)

Figure 2.3: (a) The perspective view of the virtual environment. The robot is the red square with the white sphere on top of it. (b) How the same scene is seen from the simulated omni-directional viewer. Note that the body of the robot does not appear in the image.

[Kröse *et al.* 00] [Winters & Santos-Victor] [Franz *et al.*]. We decided not to follow this approach, but to select features that humans can easily understand and that are strictly binded to the objects in the real world. This has a double beneficial effect: to collect easily interpretable data and to keep low the mathematical complexity of the project. The former is an important issue in the design and implementation of the vision system; especially because this was the first time the author designed and built a vision system. Without the help of experience, if the data are not easily interpretable, it is hard to spot eventual errors and bugs. The features we selected are the vertical edges present in the world. The vertical edges are diffusely present in the environment the robot is designed for: an indoor man-made environment like an office or a building. Examples of vertical edges are doors, the sides of

a cabinet, the legs of a chair, etc. A big advantage of choosing vertical edges as features is they are mapped by the omni-directional mirror as radial lines. Therefore, we search the images for radial lines. When the robot moves, the edges appear to move in the image. Analysing this movement it is possible to extract information both on the topology of the environment and on the robot's movements.

2.3 The assumptions

In the image analysis, we make use of some assumptions. It is worth to make them explicit here.

- The robot is moving in a indoor environment. This is a man-made environment like an office or a building;
- The lighting in the environment does not change during the motion of the robot;
- The objects present in the scene are static: they do not change their positions;
- The floor is almost flat and horizontal;
- The walls and the object present in the scene have vertical edges and surfaces;
- The axis of the camera and the mirror are vertical;
- The robot can only turn on the spot or move on a straight line. It cannot make more complex movements;

The last assumption is strong but permits to greatly simplify the vision system. Several constraints on the edge movements are based on this last assumption. Let us see why we made this assumption.

2.4 Running the simulations

We decided to move the robot in the maze with two basic movements: translations and rotation on the spot. This was done to mimic the distinction made by the SSH between the two type of actions: travel and turns. This distinction not only simplifies the construction of the movements in the virtual environment, it also generates a huge amount of simplifications in the process of building the vision system. If we force the robot to move only in straight lines and to turn on the spot to change the heading direction, it is possible to identify two separate sets of constraints on the apparent movements of the edges. The apparent motion of the edges depends on the movements of the robot and on the topology of the environment.

The topological events for translations

In Figure 2.4 is presented a simulation of the translation of the robot. For a better understanding of the scene two frame sequences are presented: the perspective view of the environment in which the robot is moving and the corresponding sequence acquired by the omni-directional camera of the robot. The situation depicted in the sequence is a typical situation faced by the robot: it exits a corridor and it aims for an object. In this sequence several thing happen. Focus your attention on the following:

- The robot approaches some objects and it moves away from others;
- Objects in the heading directions appear to expand;
- Objects in the escaping directions appear to contract;
- The vertical edges of the scene appear as radial lines that move changing their azimuth;
- There is an optical flow of radial lines with respect to the heading direction;

It is possible to identify some topological events in the frame sequence. These events happen at a single point in the space, therefore they can be used to identify distinct points in the space. This is the key that permit us to extract from the continuous world a set of distinct places as required by the Spatial Semantic Hierarchy.

- A new edge exits from occlusion, Figure 2.5;
- An edge disappear because occluded by another object, Figure 2.6;
- The two vertical edges are 180°apart in the vision sensor, Figure 2.7;
- The robots sees two pairs of vertical edges 180°apart, Figure 2.8. This identify a single point in the space. This point is the crossing point of the imaginary lines connecting opposites edges, as shown in the bird's eye view in Figure 2.9;

The movements of the edges in the frame sequence are subject to the following constraints. These constraints will be used by the algorithm that track the edges in the frame sequence.

- New edges exit from occlusion at a smaller[1] angle than the occluding edge;
- When an edge is occluded by another, the one that survives is the one with the smaller azimuth in the previous frame;
- The edges closer to the robot have a bigger azimuthal speed;
- Given a certain speed of the robot there is a maximum displacement an edge can experience from a frame to the next;
- The colours on the side of the edges change only slightly from a frame to the next one;

[1]We define an angle as "smaller" than another, when it is closer to the robot's heading direction

New edges exit from occlusion at a smaller angle than the occluding edge, referring to the absolute value of the angles. It is possible to look at it as if one edge splits in two when a new edge exits from occlusion, Figure 2.5.

When an edge is occluded by another, the one that survives is the one with the smaller azimuth in the previous frame (always referring to the absolute value of the azimuth). It is possible to look at it as two edges merge when one edge disappear because occluded by the other, Figure 2.6.

The two vertical parts of a door frame are 180°apart in the vision sensor when the robot is passing through a door, Figure 2.7;

The robots sees two pairs of vertical edges 180°apart, Figure 2.8. This identify a single point in the space. This point is the crossing point of the imaginary lines connecting opposites edges, as shown in the bird's eye view in Figure 2.9.

The topological events for rotations

Refer to Figure 2.10, in this sequence the robot is turning on the spot in front of the blue box where it stopped in the sequence of Figure 2.4. As before we draw from these pictures the considerations and the constraints that apply to the movement of the vertical edges in the image. Focus your attention on the following events:

- The robot turns on the spot;
- The distance of the robot from the objects does not change;
- Objects do not change their shape;
- The vertical edges of the scene appear as radial lines that move only by changing their azimuth;
- The number of visible edges is constant: no edges appear or disappear;

The last consideration comes from the fact that there is no relative displacement between the robot and the objects. Therefore, the occlusions do

not change. In other words, the image does not change, it appears only rotated around its centre.

The only topological consideration we can draw from the rotation sequence is that nothing changes and all the views the robot experiences are related to the same physical place. Therefore, in the implementation of the SSH all the *views* that differ only for a rotation around the centre of the image must be correlated to the same *place*.

We can extract the following constraints for the edges:

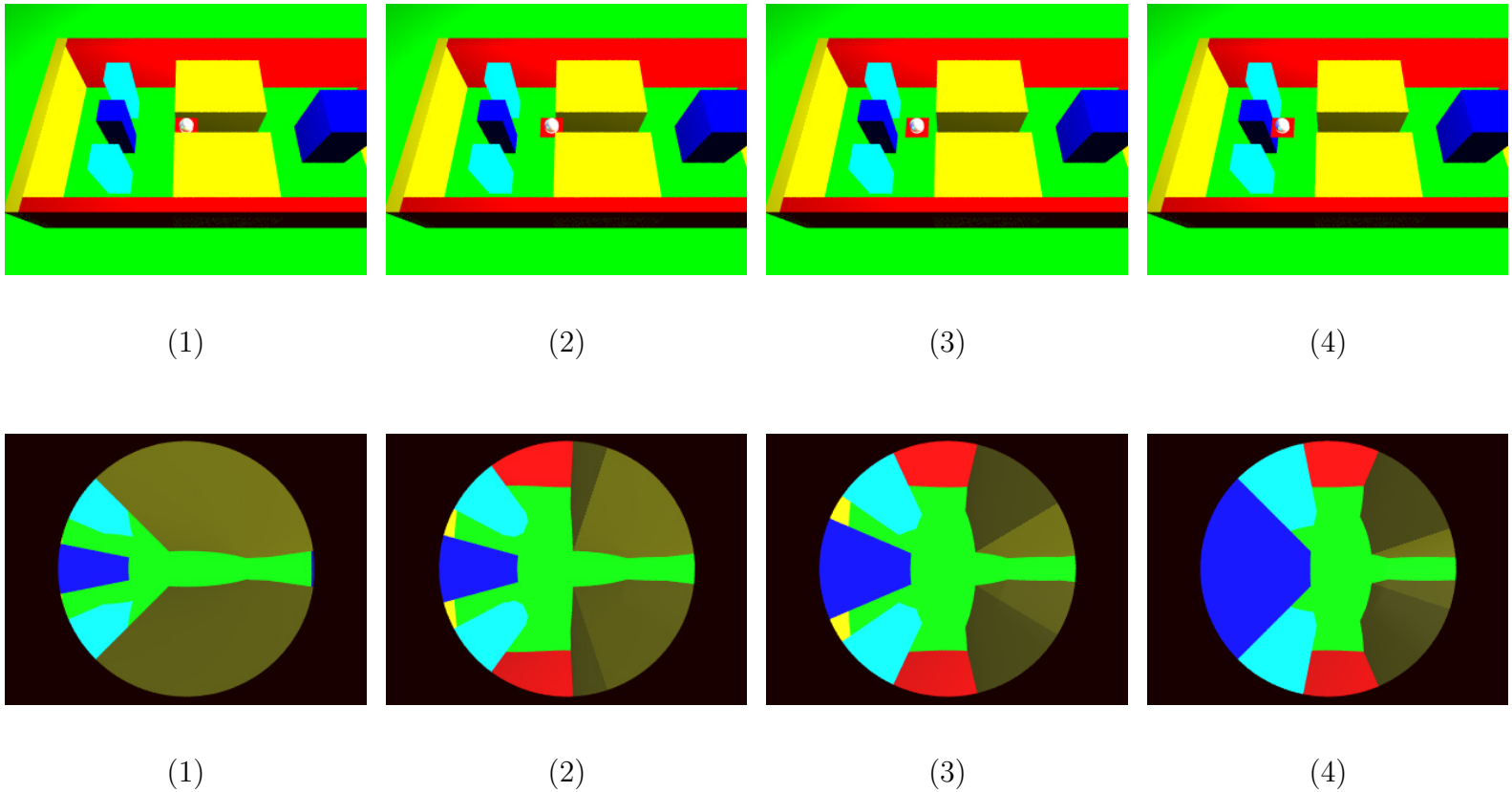
- All the edges experience the same azimuthal shift;
- The colours on the sides of the edges change only slightly from a frame to the next one;

Running the simulations, we discovered a flaw in the assumptions we made. This problem exists both in translations and in rotations. All the vertical edges in the scene are mapped into radial lines in the image, but the converse is not always true. It is not true that all the radial lines in the image correspond to vertical edges in the scene. There could be some radial lines in the image that are actually radial to the vision sensor. For an example see Figure 2.11. There, the right baseline of the corridor happens to be radial in the first frame. In fact, if the robot is moving on a straight line, an accidentally radial line will appear as radial only for few frames, unless the line lies in the direction of the motion. If the robot is turning on the spot, the accidentally radial line will not disappear until the robot moves away from that spot. This suggests that it is not enough to match the edges in two consecutive frames, but that we need a check over a longer period of time: a memory, as to say. For instance, we could require a minimum life time before confirming a candidate radial line as a vertical edge.

2.5 Conclusion

In this chapter we gained a qualitative understanding of an omni-directional image and how the vertical edges move in a sequence of these images. We identified two set of topological events, one for translations and one for rotations, to be used to build a distinctiveness measure. Once formalised, this measure can be used to abduct a discrete set of *places* from the continuous world. These *places*, and the *paths* connecting them, will be the elements of the topological map of the environment.

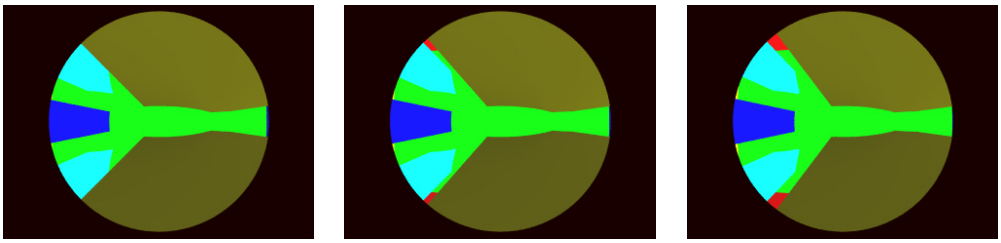
We also identified two set of constraints for the movements of the edges. These constraints will be crucial in the design of the matching algorithm of the software that tracks the edges along the omni-directional frame sequence, as we will see in the next chapter.



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

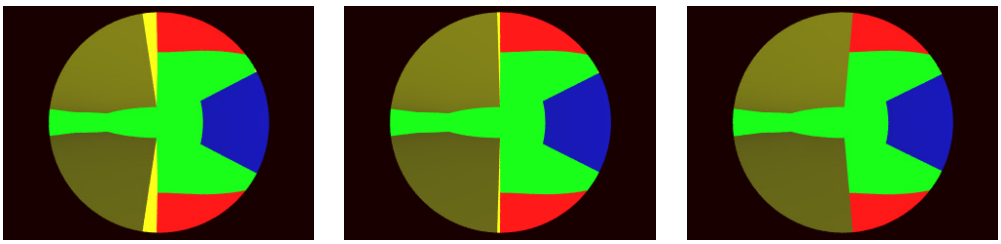
(1) (2) (3) (4)

Figure 2.4: (Top) Perspective view of the robot moving along a corridor. The robot is the red square with the white sphere on top of it. (Bottom) The corresponding sequence acquired by the omni-directional camera of the robot



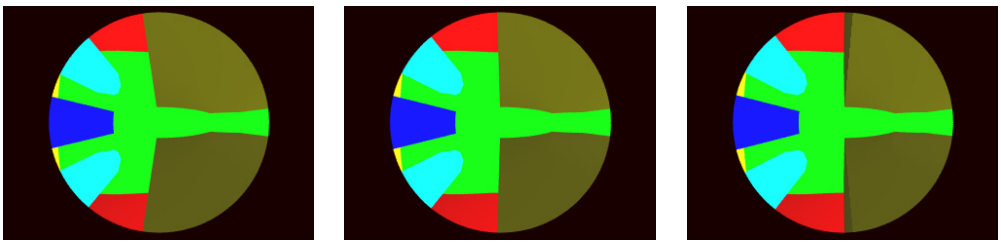
(1) (2) (3)

Figure 2.5: The simulation of an edge exiting from occlusion. Note that in the first frame a part of the two cyan boxes is occluded by the corridor walls. In the second frame all the edges of the cyan box are visible. In the third the cyan boxes are clearly visible.



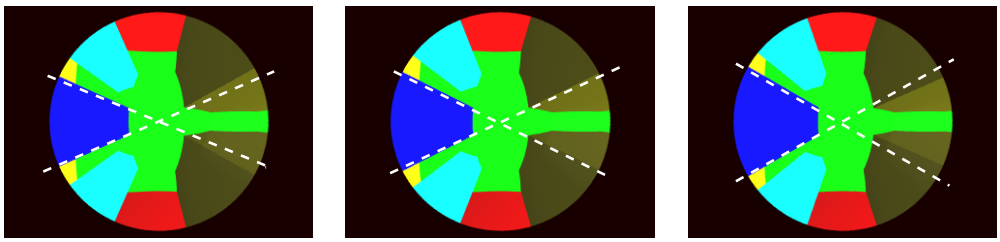
(1) (2) (3)

Figure 2.6: The simulation of an edge going to be occluded. In the first frame, the edge between the yellow corridor boxes and the red walls are clearly visible. In the second frame, they are just visible. In the third frame, they disappeared.



(1) (2) (3)

Figure 2.7: The simulation of the robot passing through a door. In the first frame, the edges of the yellow corridor boxes are not yet at 180°. In the second frame, they are at 180°. In the third, they are not longer at 180°.



(1) (2) (3)

Figure 2.8: The simulation of the crossing point of the imaginary lines connecting opposites edges. In the first and third frame, the edges of the blue box and the corridor are not exactly 180° apart. In the second they are.

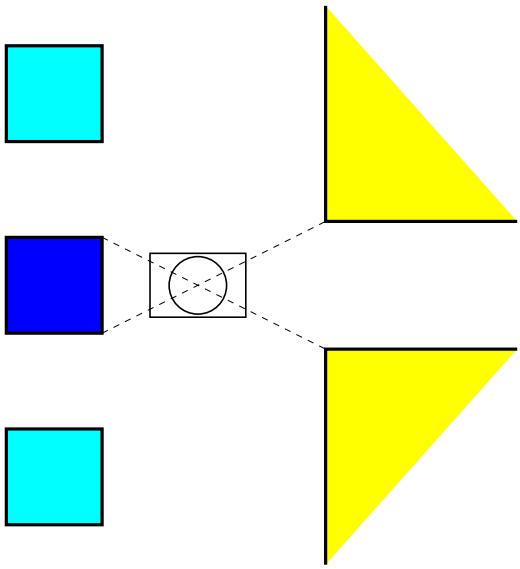
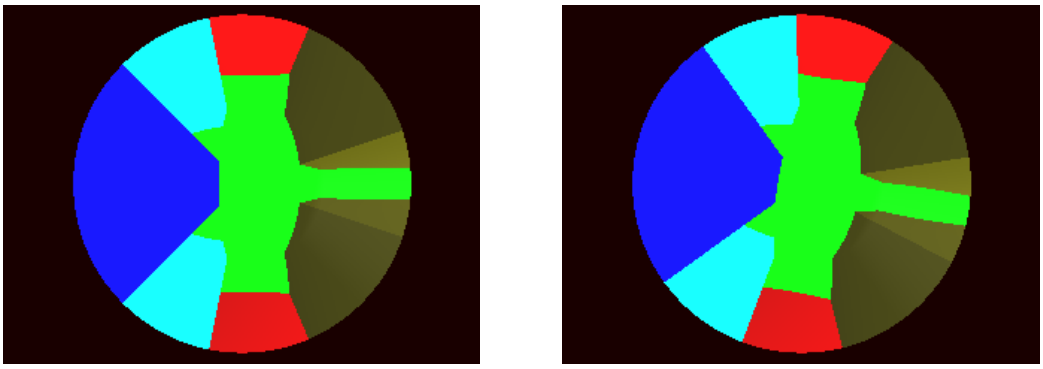
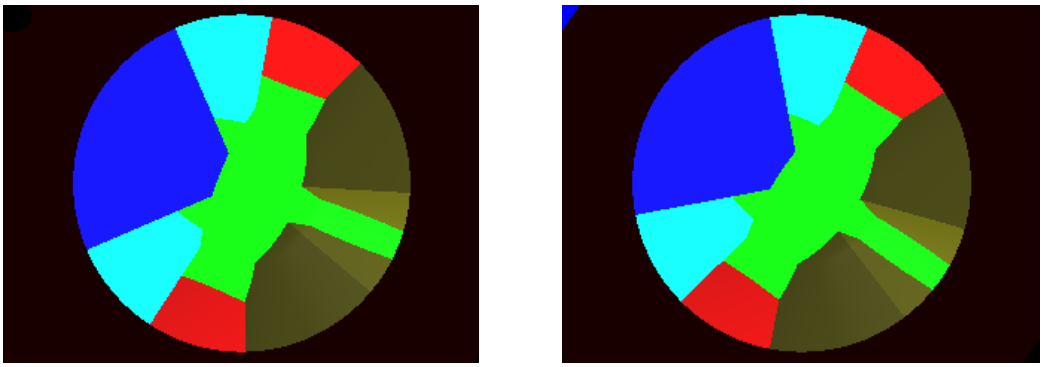


Figure 2.9: The bird's eye view of the crossing point of the imaginary lines connecting opposites edges.

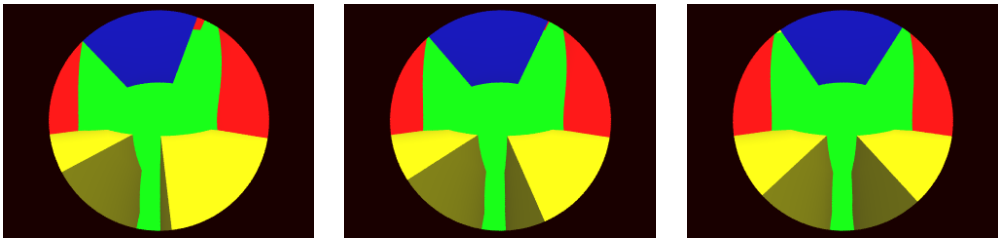


(1) (2)



(3) (4)

Figure 2.10: The simulation of a sequence acquired by the omni-directional camera of the robot while it turns on the spot



(1) (2) (3)

Figure 2.11: The simulation showing that not all radial lines are vertical edges. Notice how the baseline of the right yellow corridor box appears to be radial only in the first frame and in the others frame it is not.

Chapter 3

The Vision System Software

In the last chapter we created simulations to gain intuition about the features of the image and the events we could use in the map building process. In this chapter we develop algorithms to detect these features and events. We present the basic structure of the vision system. We introduce first the idea on which the system is based and then give details of the implemented algorithm.

As we see in the introduction, the vision system has to perform two tasks necessary for the navigation of the robot: to detect and locate the objects in the surroundings of the robot and to give information about the movements of the robot it-self. Because the robot does not have direct access to information on its position or on its movements, the autonomous behaviours that will control the robot must come from the mix of these two sources of information. For instance, the obstacle avoidance, first level for a safe navigation, is a combination of the knowledge of where the obstacle are, with respect to the robot, and where the robot is going.

3.1 The idea

We present here the ideas that led to the creation of the vision software. The vision software is thought to work in real time while the robot moves from point to point. Its work can be ideally divided into two parts: to extract the selected features from the single frame and to use the information in the frames sequence to understand the environment. From a single frame the robot extracts the vertical edges and from the frame sequence it understands how the edges moves in the image. From the apparent motion of the edges we can draw conclusions about the topology of the surroundings. For instance, if an object occludes a second one, it must be closer to the robot than the second one. Let see how it works. The robot takes a snapshot at a certain location, Figure 3.1. While it moves to the next location, it processes the image. First, it performs an edge detection to extract the edges from the picture, generating a black and white image, Figure 3.2. Second, the black and white image, containing only the detected edges, is processed with a Hough transform to identify the radial lines, see Figure 3.3 where the detected radial lines are marked with a black dot. Lastly, the most delicate task is performed: the edge matching. The robot has to recognise an edge from frame to frame. Therefore every edge has to be matched with its corresponding edge in the previous frame. The matching process exploit the colours in the image. The average of the colours in small areas around the detected edge is used as a signature of the edge. By comparing these signatures in consecutive frames we solve the correspondence problem. Figure 3.4 shows the output image of our program after the edge matching. The coloured dots are used to label the edges, every edge is associated to a unique colour. The black star-like dots beside each edge are the regions where we calculate the average of the colours to calculate the edge signature.

The approach used has been inspired by the paper of Yagi [Yagi *et al.* 95]

Nevertheless, the novel matching algorithm is designed by the author.

3.2 The algorithm

In this paragraph, we present the algorithm that tracks the edges throughout the image sequence. This can be divided into three main parts: an edge detector, the Hough transform and the matching algorithm. The program we developed in this project is not real time as in the design specification. This is because in the project we focused more in the problem-understanding and in the accuracy of the results than in the optimisation of the software. For this reason in the design and choice of the software the speed was not an issue.

3.2.1 The edge detector

The edge detector used is the Canny edge detector [Canny 86]. This edge detector is computationally intensive, but it works well and compared with other edge detectors, like the Sobel's one, it produces thinner lines. The Canny edge detector does not produce a shift of the edge position with respect to their position in the original image. There is a problem: the Canny edge detector works with monochromaitc images and we have RGB colour images[1] . To overcome this problem two approaches are possible. We can transform the colour image into a grey level one and use this as input for the Canny edge detector. Otherwise, we split the colour image in three images, one for each colour channel and we apply the Canny edge detector separately to each channel image. In the end, we combine the three resulting images by applying the logical OR operator. In other words, each pixel reported as an edge pixel

[1] An RGB colour image is a colour image where each pixel is represented as a triplet of values: one value for the amount of red in the pixel, one for the amount of green and one for the amount of blue

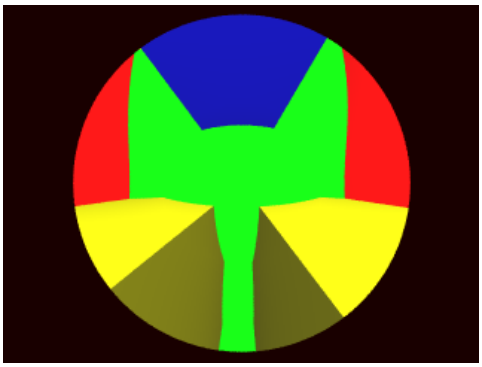


Figure 3.1: The unprocessed image as view from the robot's camera.

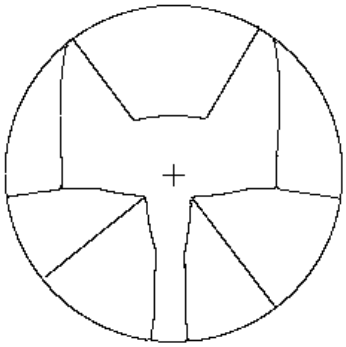


Figure 3.2: The image after the Canny edge detection.

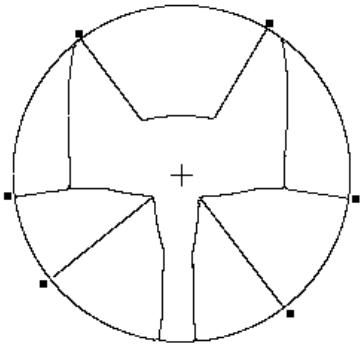


Figure 3.3: The edge image after the Hough transform.

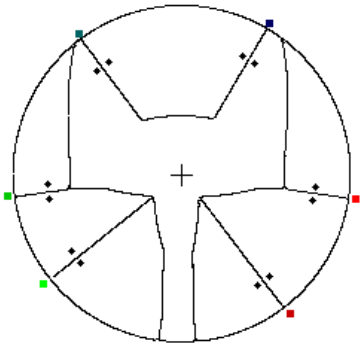


Figure 3.4: The final result showing the edge matching.

at least in one of the three edge detections is an edge pixel of the final image. The two possibilities are displayed in Figure 3.5. The latter process is computationally intensive but it is very accurate. Simulations showed that working with the grey level image obtained from the colour image can result in a loss of information. As you can see in Figure 3.5 working with the grey scale image we lost the two vertical edges on the left of the image.

3.2.2 The Hough transform

Once we have extracted the edges in the image, we have to select only the edges corresponding to the vertical edges in the scene. Therefore, we need a process to identify the radial edge in the picture. The Hough transform is such a process [Hough 62]. For this task a new parametrisation of the image was found by the author which greatly simplifies the complexity of the transform. The new parameters are the angular coordinate and the radial coordinate of the pixels in a polar coordinate system with the origin in the centre of the image. In such a polar coordinate system, a radial line is described as a sequence of pixels with the same angular coordinate and varying radial coordinate. See Figure 3.6 a). To find out where the radial lines are in the image we apply the following algorithm: for each edge pixel we calculate its angular coordinate, we round it down to the unit of degree and we increment a counter corresponding to this number. When all the edge pixels are processed, we look at the histogram of the counters values. The counters that show a value over a certain threshold correspond to the position of the radial edges in the image. See Figure 3.6 a). The threshold corresponds to the minimal length (in pixels) of the radial lines that we consider as vertical edges. The choice of the threshold to set for the minimal length of a vertical edge must be well assessed. A threshold set too low can detect as composing a radial line also pixels that have the same angular coordinate but not belong to the same line or lines that occure to be radial

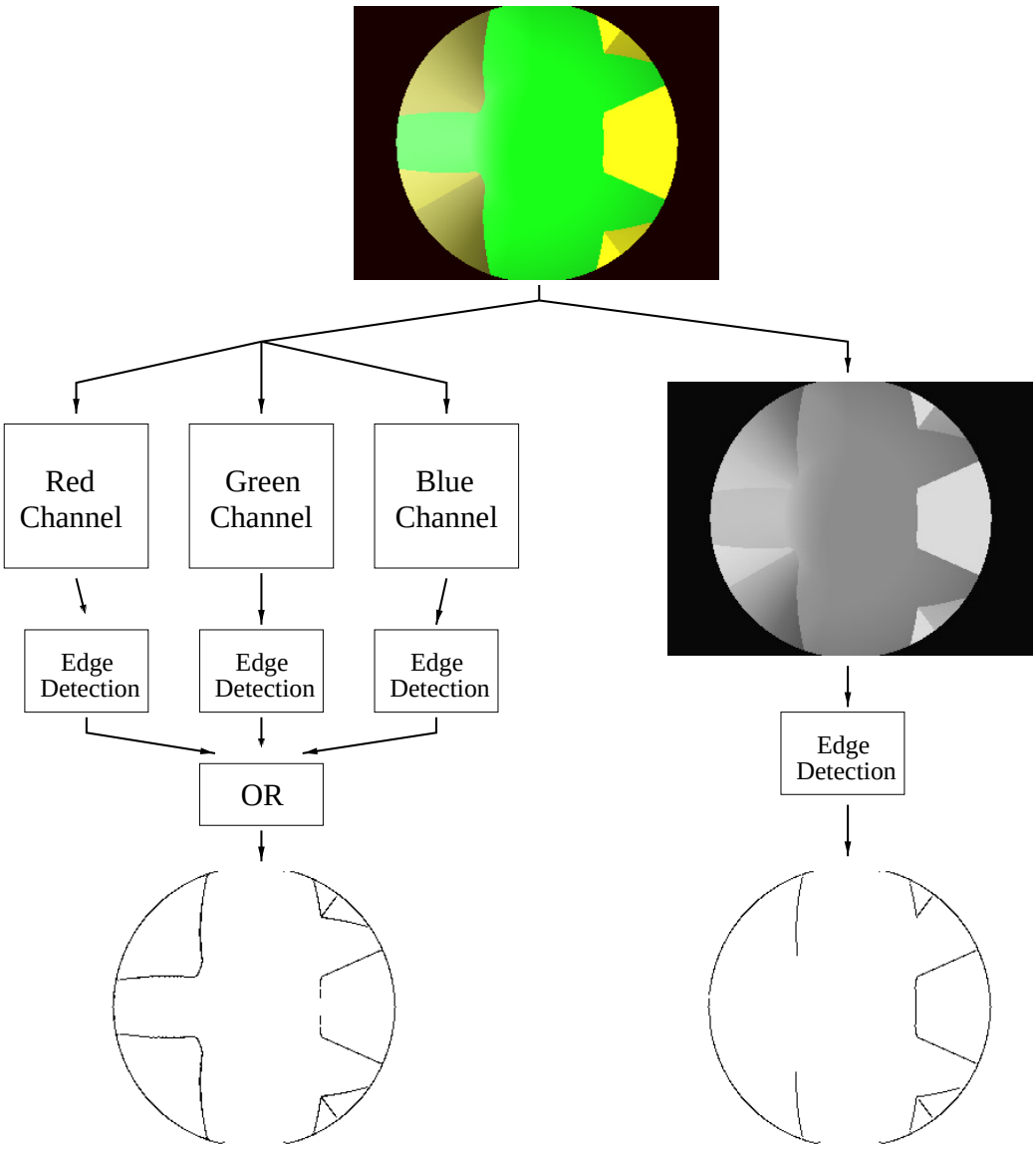
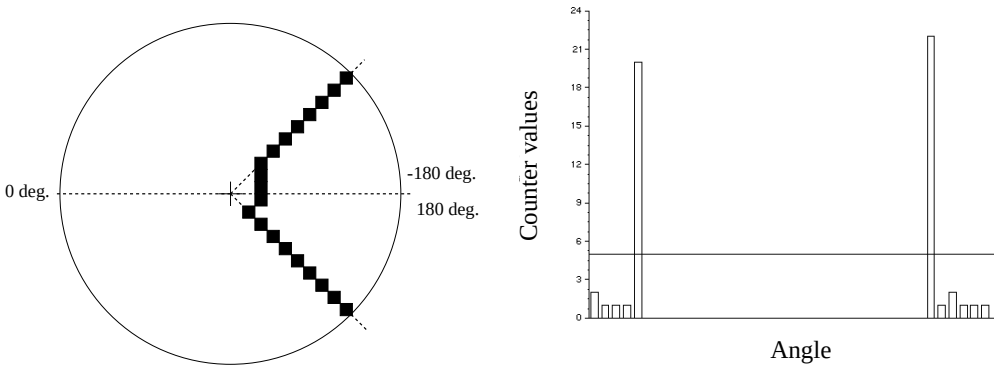


Figure 3.5: The two possible processes for a Canny colour edge detection.

just for a small bit. On the other hand, a threshold set too high does not identify some vertical edges, especially when they just appeared in the field of view and they are like small segments in the periphery of the image.



(a) (b)

Figure 3.6: A the Hough transform.
(a) The black squares are enlarged pixel. Notice they all have the same angular coordinate. (b) The histogram for the Hough transform.

3.2.3 The matching algorithm

Once we have identified the vertical edges in the pictures, we have to track them along the frame sequence, i.e. we have to be able to recognise an edge in different frames. To identify an edge, we use the colours on the left and the right side of the detected edge. To extract colour information, robust to the noise of the picture, we calculate the colour as the average colour over a window positioned across the edge. In Figure 3.7 is presented a close up of a processed image showing the averaging windows. Each window is composed of two sub-windows. These are the star-like dots on the sides of each edge. The colour of a pixel in the image is represented by a RGB triplet. The average colour calculated over the pixels of each sub-windows is represented by a RGB triplet, as well. Each component of the triplet is the average of the values of the corresponding colour channel over the pixels of the sub-windows. The windows are placed on a circumference that intersect all the significant edges, Figure 3.8. The windows are designed to follow the edge as it moves around in the circle. To understand how this is done, think of the two sub-windows connected by a rod. The rod is kept always tangential to

the ideal circle shown in red in Figure 3.8. The length of the rod is enough to keep the sub-windows at a certain distance from the edge, this permits us to avoid eventual border effects caused by the edge, but short enough not to overlap with another edge detected nearby. The shape of the sub-windows has been chosen to be approximately circular for two reasons. First, because in this way the minimum distance between the edge and the sub-windows is approximately constant and second because while the sub-windows moves around with the edge, it spans always over the same pixels. The former assures us that the sub-windows never goes too close to the edge. The latter that we calculate the average colours around the edge always on the same pixels.

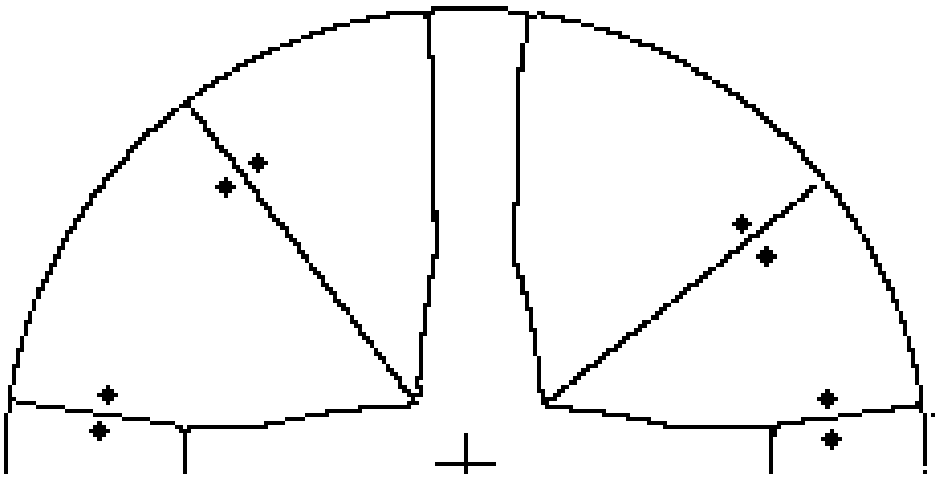


Figure 3.7: A close up of a processed image showing the averaging windows. They are the star-like dots around the radial edges.

At this stage, the program is able to assign to each edge its colour signature. To track the edges along the whole frame, the program tries to match the edges in the current frame against the edges on the previous frame and so on for the whole length of the sequence. The matching is done with the colour signature, but this could be not enough to correctly identify an edge

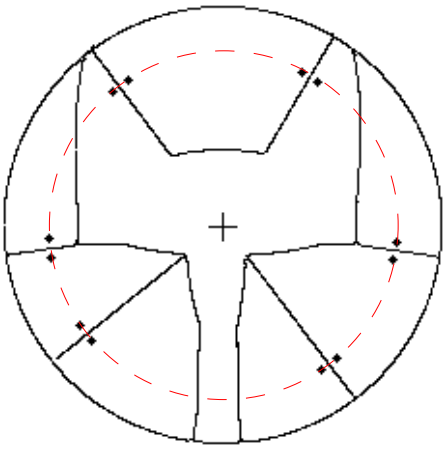


Figure 3.8: The disposition of the averaging windows in the 360°.

in the previous frame. In fact, there could be other edges in the image that have similar colours signatures. Using the two sets of constraints drawn in the previous section for the edge motion, it is possible to avoid mismatches by shrinking the area where to look for the corresponding edge. In the case of linear motion, the first constraint is that the absolute value of the azimuthal coordinate of the edge can only increase from one frame to the next, the second is that this increment is bounded to a maximum step. The size of this step depends on the maximum speed of the robot. In the case of rotation, the displacement of the single edge cannot be bounded: we do not know a priori what the turn will be and there is not a maximum turn the robot can take[2] . There is only a loosy bound given by the maximum rotational velocity, but this is not such a constraint. Therefore, an edge can match anyone of the edges in the image with a similar signature. The only constraint we can use is that every edge has to experience the same azimuthal shift. Therefore, we implemented a backtracking algorithm that finds a matching for every edge checking that all edges experienced the same azimuthal shift.

[2]Remember the robot does not have any direct access to informations about its movements

The backtracking is necessary to search completely the space of all possible edge matching.

There is a problem: we have these two different sets of constraints for the two different type of motion and the robot does not have access to any information to decide which set should be used. To overcome this, the matching algorithm tries to match the edges assuming the robot performed a rotation. If this does not results in a *good matching*, it assume a translation and applies the appropriate rules. A **good matching** is defined as a matching where more than half of the edges present in the image are correctly identified. If neither of the two set of rules is able to match more than half of the edges, the match that correctly labelled more edges is returned.

We defined a *good match* as requiring the match of more than half of the edges present in the image, because we cannot require that every edge in the current frame is matched with an edge of the previous frame. In fact, if a new edge exited from occlusion in this frame it cannot be matched with anything. On the other hand, it is very unlikely that a wrong matching could identify more than half of the edges.

In all the tests performed by the matching algorithm we introduced a certain amount of tolerance. We have tolerance on the colour signature, on the azimuthal shift and on the azimuthal increase. Let us consider them in detail. The tolerance on the colour signature states that two colour signatures are considered the same if the values of each colour component differs by less than a certain threshold. The tolerance on the azimuth shift states that two azimuthal shifts are equal if the shift angle differ less than a certain threshold. The tolerance on the azimuthal increase allows an edge to pass this check even if its actual azimuthal coordinate is slightly smaller than in the previous frame. The tolerances reported have been tuned during the experiments on simulated and real sequences in order to have a matching algorithm flexible enough to cope with the noise but not so loose to produce mismatches.

After the matching process is completed, the program writes in a log file the information about the edges in the current frame: the unique label of the edge, its azimuthal position and the values of its left and right averaging windows. This file is useful for debugging purpose and can be used off-line for the reconstruction of the topology of the environment. So far, only the first two of the types of topological transition we saw in Section 2.4 are detected on the fly. In fact, the software is able to detect the topological event if the number of edges changes from one frame to the next.

To test the software we wrote, we performed first an experiment with a simulated sequence of frames and then we implemented the system on the actual robot.

Chapter 4

The Simulated Experiment

To perform the simulated experiments, we generated a sequence of images captured by the camera under the mirror while the robot moves in the maze. This sequence of images were used as input for the vision system. Unfortunately, in POV-Ray it is possible to create only off-line simulations. It is only possible to generate a sequence of images, it is not possible to introduce an interaction between the vision system software and the rendering program. In other words, it is not possible to create an interaction between the robot and the virtual environment.

4.1 The experimental set-up

In this simulation the robot goes through the whole maze along the pre-planned path showed in Figure 4.1. It goes straight on until it finds the first corner. There it turns on the spot of 90°anti-clockwise. It moves again straightforward until the door of the corridor. It turns again on the spot of 90°, clockwise this time. Eventually, it goes straight up to the blue box.

This path permits us to test the edge tracking algorithm while the robot moves in a straight line, while it turns clockwise and anti-clockwise and when it passes from translation to rotation.

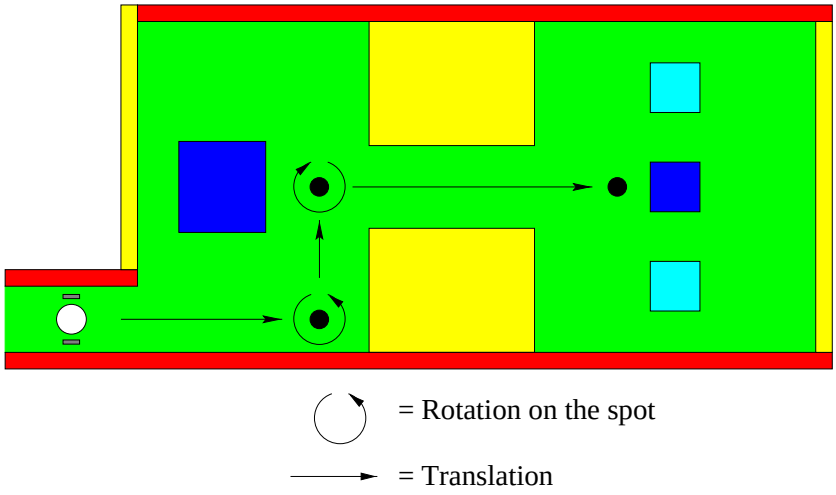


Figure 4.1: The path of the robot through the virtual environment.

The maze used is brightly coloured. Each box and wall in the environment has a different colour and the colours chosen are bright and vivid. Moreover, the colours stand out one against the other. This makes it easy for the software to recognise the edge signatures and for the reader to locate the robot in the maze, just from the colours present in the image.

4.2 The results

Running these experiments, we encountered some problems. Some of them were already previewed by the experience we gained with the simulator visualisation, others were unexpected.

For instance, we discovered that the pixels belonging to a radial lines do not actually have a unique angular coordinate. The case presented in Figure 3.6 is an ideal one. In the real case, an edges spreads over more than one angular coordinate. See Figure 4.2 for an hypothetical example. In the image the black squares represent the edge pixels of the picture enlarged for sake of clarity. The spread of the pixels is caused by their finite dimension and by the round off error we introduce in the calculation of the angular coordinate.

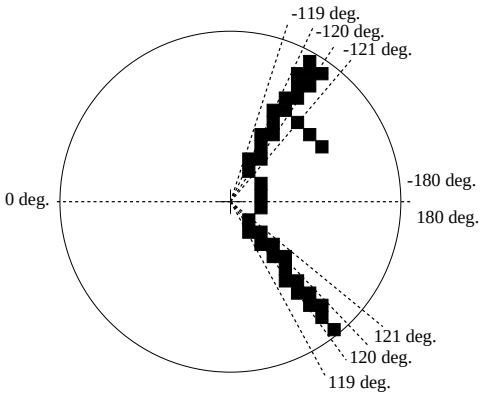


Figure 4.2: The spread of the pixel in the Hough transform.

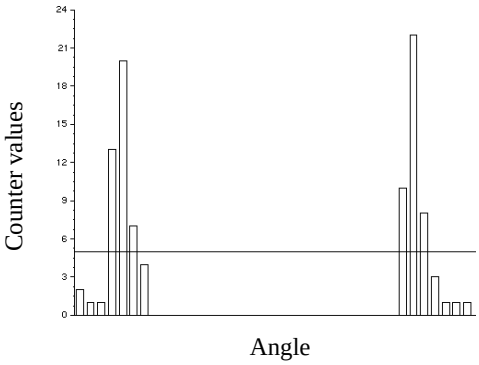
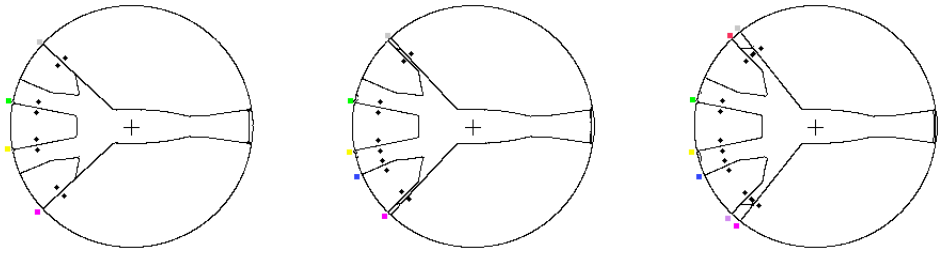


Figure 4.3: The histogram for the Hough transform.

The spread depends on the ratio between the size of the single pixel and the angular resolution we are interested in.

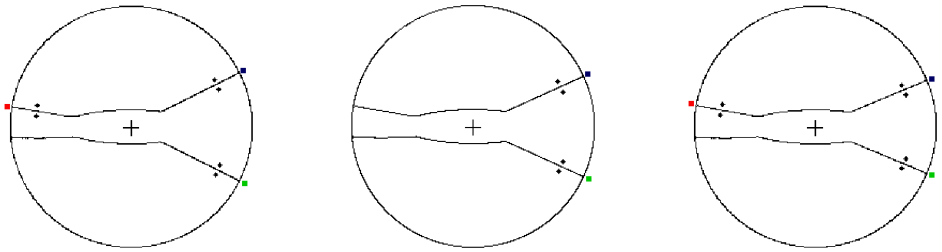
We solved the problem with a software trick, every time we find a peak over the threshold in the counters histogram, we seek in its neighbourhood the counter with the maximum value and then we pick this one as the azimuthal position of the vertical edge. We found that a good width for the neighbourhood is 3°. The drawback of this technique is that, when one edge exits from occlusion behind a second edge, we recognise it as a vertical edge only when the two are more than 3° apart. Nonetheless, this is accurate enough for our purposes. See Figure 4.4, behind the left edges of the corridor two new edges appear. These are not immediatly detected. As you can see, they are spotted only in the next frame, when they are more than 3° apart.

The foreseen problems were essentially problems in the setting of the different thresholds and tolerances in the edge tracking program. As we said in Section 3.2.2, the choice of the threshold on the azimuthal counter in the Hough transform affects what is recognised as a radial edge, and then as a vertical one. Two problems sensitive to the choice of the threshold are the *ephemeral edges* and the *false edges*. These cannot be eliminated just tuning



(1) (2) (3)

Figure 4.4: A simulated example of edges exiting from occlusion and not immediately detected. In the first frame the edges of the corridor occludes the ones of the two lateral boxes. In the second frame, the edges of the boxes are visible but not detected. In the fird frame, they are detected.

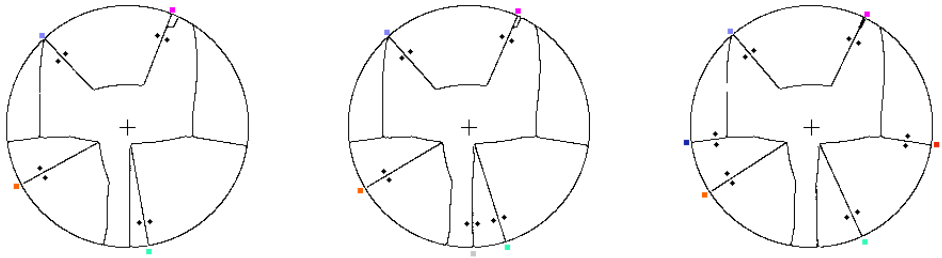


(1) (2) (3)

Figure 4.5: The simulation of an *ephemeral* edge. Notice that the edge on the left marked with the red dot is not present in the second frame.

the threshold but their effects can be lowered.

We called **ephemeral edges**, those edges whose presence in the frame sequence is not detected for some frames and after a few frames is detected again. An example is the left edge in Figure 4.5. This happens because the value of the edge counters are under the threshold just for few frames. The reason can be either that the pixels of the edge spread in the neighbourhood in such a way that none of their counter is over the threshold or that there is a variation on the number of the edge pixels introduced by the Canny edge detector. So far, the program is not able to handle correctly the ephemeral



(1) (2) (3)

Figure 4.6: The simulation of a *false* edge. Notice that in the second frame the right baseline of the corridor is detected as a radial line. This does not happen neither in the previous frame or in the next.

edges. It does not realize that the new edge that appeared is actually the same that disappeared a few frames before. This is because the matching process uses only two frames: the current one and the previous one. This problem suggests the need of implementing a memory that spans over more frames causing a persistence of the edges for a certain number of frames after they disappeared. In this way, when an ephemeral edge reappears, it can be matched with this persistent image of the edge.

The **false edges** are the accidentally radial lines described at the end of Chapter 2. The vision software erroneously detects them as radial lines. In Figure 4.6 is presented the same sequence of Figure 2.11. The right base line of the corridor is detected just for one frame as a new vertical edge. This suggests the need of a confidence measure on the confirmation of an edge. An edge should be confirmed only if it is present in a minimum number of frames.

4.3 Conclusion

Despite these problems, the vision system software showed to be able to correctly track the edges all along the path followed by the robot in these

simulations. See sequence in Figure 4.7 and Figure 4.8

The problems of the ephemeral edge and of the false edges could probably be solved off-line from the software that exploits the information on the edge to build the topological map. In other words, the memory of the system could be implemented at a higher level than the sensory level. Probably, it would be easy for this program to analyse the log file of the vision system and spot the presence of an ephemeral or false edge, easier than for the vision system to spot them on the fly.

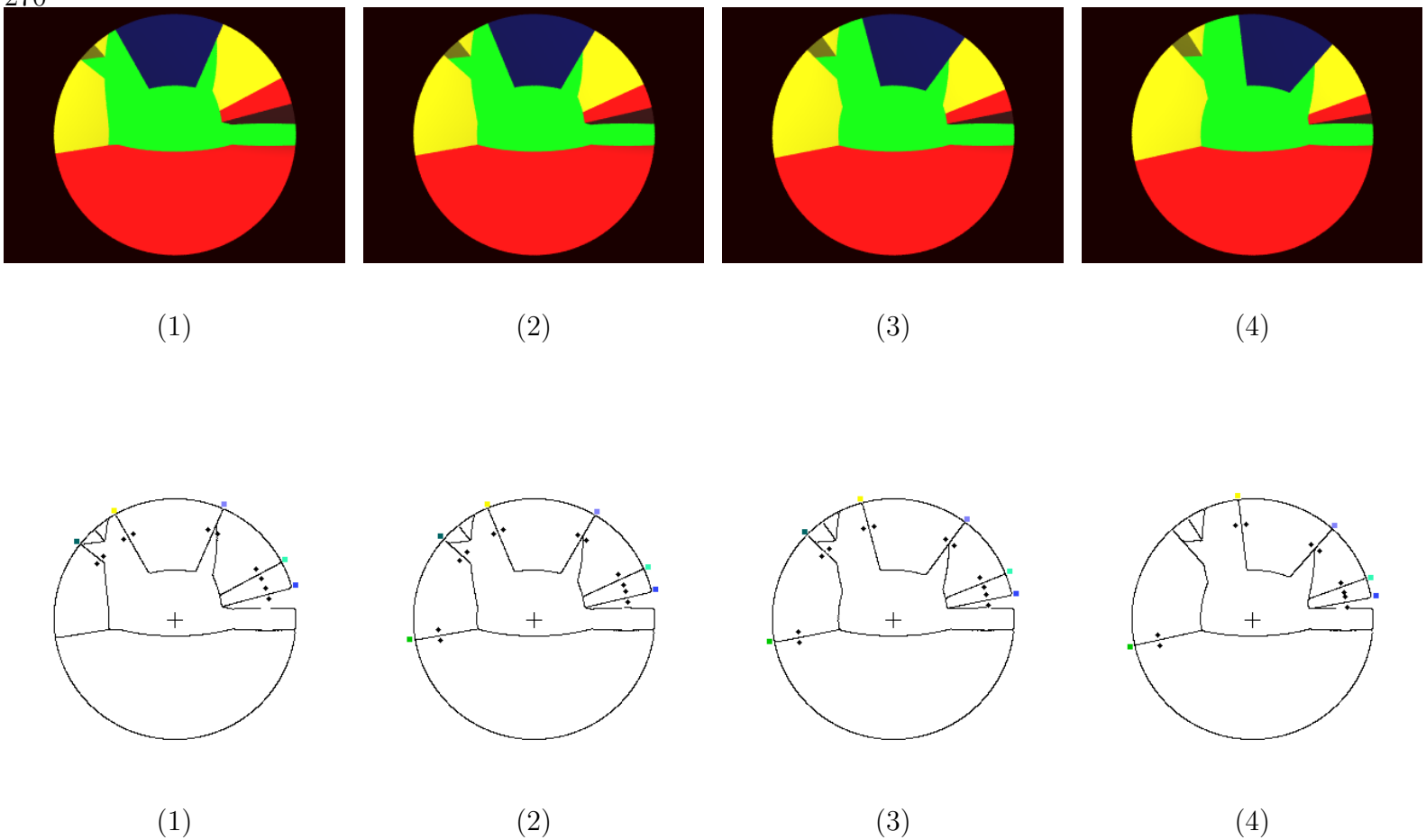
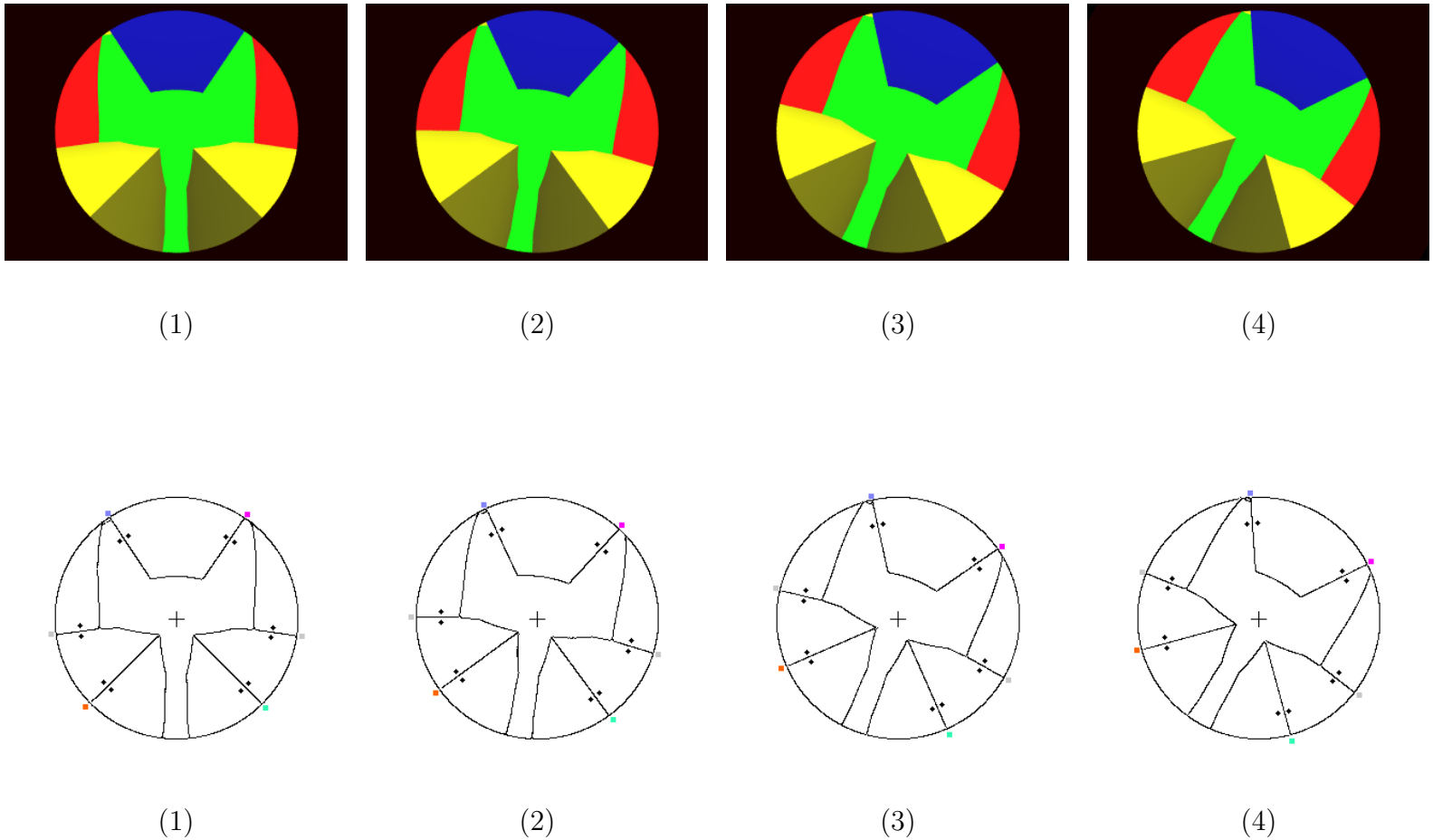


Figure 4.7: (Top) The camera view of a simulated image sequence for a translation of the robot. (Bottom) The tracking of the edges in this sequence.



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

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

Figure 4.8: (Top) The camera view of a simulated image sequence for a rotation of the robot. (Bottom) The tracking of the edges in this sequence.

Chapter 5

The Real Experiment

The final test for our program was to work with a real robot in a real environment. The robot is Lisa, the goal-keeper of the Azzurra Robot Team (ART). ART is the Italian national team for the Robocup World Championship, the robot football competition. The environment chosen for the experiment is the robot football playground at the Intelligent Autonomous Systems Laboratory at the University of Padova. In this chapter we will describe the robot itself and the environment in which we carried out our tests. We will describe the problems encountered working with a real robot and how we solved them.

5.1 The body of the Robot

Our robot has the same hardware of Lisa, but its task and its software are totally different, so it deserves the different name: **Caboto**.

5.1.1 The chassis

Caboto has a custom built chassis. It is a black box of approximately 50 cm x 40 cm x 40 cm. It has a steel frame. The frame structure is closed by tin

sheets. There are two driven ribbon wheels and two small spheres under the base boundary to balance the robot. Around the chassis there is a kicker, a tool of its goalie life. On the top, there is the omni-directional vision system.

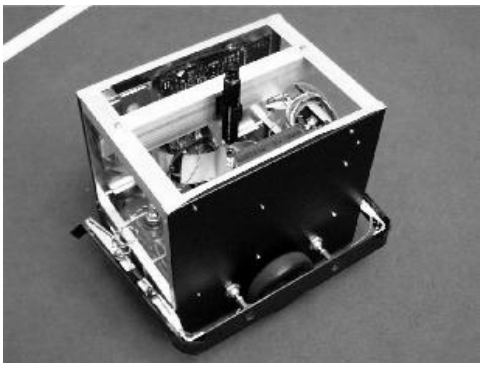


Figure 5.1: The chassis and the inside of Caboto.

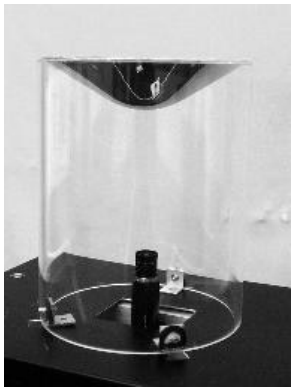


Figure 5.2: The Omni-directional Vision System of Caboto.

5.1.2 The Driving System

The driving system is that of a commercial Pioneer 2-AT. The two ribbon wheels are driven independently by two motors of 15W each. The motors are fixed to the steel frame. The maximum speed of the robot is 1.6 m/s and it can turn on the spot.

5.1.3 The Computational Power on Board

Caboto, as Lisa, is totally autonomous. It has a complete computer inside and it can carry its batteries. The communication link, with other robots and computers, is assured by a wireless network link realized with a Lucent Wavelan PCMCIA card. This allows us to monitor the robot behaviour remotely. Table 5.1 presents a detailed description of the Computational Power of the robot.

CPU	AMD K6 266MHz
Mother Board	Asus TX97E
RAM	64 Mb DIMM
HD	4Gb
Frame Grabber	BT848 Intel
Graphic Card	2Mb RAM

Table 5.1: The Computational Power on board of Caboto

This hardware runs the Linux operative system. Kernel version 2.2.15; distribution Mandrake 7.0.

5.2 The Vision System Hardware

The vision system hardware of Caboto is the same as for Lisa. It is composed of three elements: the mirror that collects the light from all directions, the CCD camera that converts the light into an analogue electronic signal and the frame grabber that converts the analogue signal into a digital signal.

5.2.1 The mirror

The mirror is the element that characterises this vision system as an omni-directional vision system. The mirror is mounted vertically over the camera with its geometrical axis coaxial at the optical axis of the camera. It is supported by a transparent Perspex cylinder. The height of the cylinder is approximately 20 cm. The mirror has a hybrid shape with respect to the ones reviewed in Section A.3. It is composed of a cone that intersect a sphere. The dimension of the cone, the radius of the sphere and the position of the point of intersection are calculated such that the cone and the sphere are tangential in the intersection point. If they were not tangential, a discontinuity would

be present in the image. For convenience of the reader in Figure 5.3 we present again the sketch of the mirror profile with its dimensions shown in Figure 2.2. Figure 5.4 shows the distances of the mirror from the camera and the floor.

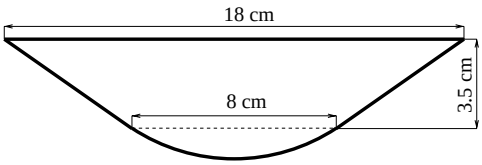


Figure 5.3: The Mirror Profile.

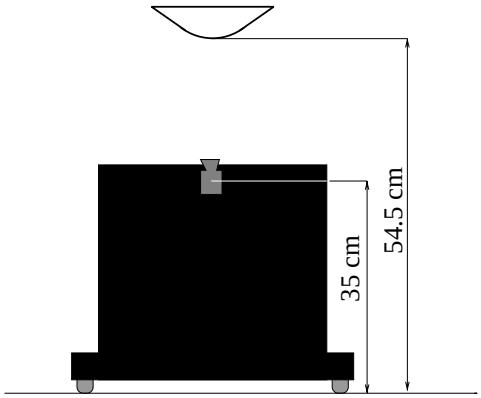


Figure 5.4: The Vision System.

This mirror has been realized from a sheet of steel pressed between two dies. This kind of working is not very accurate, it is cheap but not accurate. That is why the mirror presents visible irregularities. For our task this is not such a problem, we are more concentrated on the topological aspect of the environment, than on the metric aspect. Therefore, the sensor is reliable enough, even if not all points are mapped in the correct position because of these irregularities.

5.2.2 The camera

The camera used is a standard perspective camera. It is a SONY XC-999 with a 6mm lens. It is a high resolution colour camera. It uses the Hyper Had technology to reduce the smear effect down to an imperceptible level.

The real sensor is a CCD with horizontal resolution of 470 TV lines 752 x 582 in the PAL format. The dimensions are 22x22x120mm and it weights approximately 200 g. This is an analogue camera, therefore it needs a frame grabber to convert the analogue signal into a digital signal.

5.2.3 The frame grabber

The duty of the frame grabber is to convert the analogue signal that exit from the camera into a digital signal and to store it in the computer memory. The frame grabber is an Intel BT848. It is controlled by the Video4Linux driver.

5.3 The experiment set-up

As we said in the introduction, the environment chosen for carrying out the experiments was the robot football playground at the Intelligent Autonomous Systems Laboratory at the University of Padova. The environment is very simple and with few standard colours present in it. It is built following the specifications of the Robocup[1] . The room has white walls and a floor covering of green carpet. In this room we built a simple corridor with a turn. In the corridor were two boxes to reproduce a door-like view. See Picture 5.5. All the vertical surfaces were painted with uniform colours, exept the two boxes. This was done to avoid textures that could fool the edge detector. During the experiments, also the boxes where covered with a uniform colour fabric, because of the noise they introduced. Like for the virtual environment, the colours chosen for the surfaces are vivid and they stand out one against the other to facilitate the recognition of the edge signatures.

As we said before, the vision software is slow. It does not permit to work in real time in the experiments with the real robot. It takes about

[1]For a detailed description see the URL:
smart.informatik.uni-ulm.de/ROBOCUP/rules00/FIFA-laws00.html

Figure 5.5: The experimental set-up. This is the *corridor* where the experiments were conducted. The two box represent a door. On the right, behind the wood box there is the turn of the corridor.

two seconds to process every single image. We had no time to rewrite the software in order to optimise the code for the real time, therefore instead of speeding-up the software we slowed down the robot. We made the robot moving step by step. The single step is about 15 cm long. After each step the robot stops, it takes a snapshot, it processes the image and when he has finished, it takes another step.

When we described the omni-directional vision system we said that the optical axis of the camera and the geometrical axis of the mirror are aligned. On the real robot this calibration is not an issue. It can be done roughly by hand and then we can find in the image where the tip of the mirror is mapped. We need this information, because this is the point from where

all the radial edges sprout and where we have to centre the polar reference system. The performance of the Hough transform depends on the accuracy of this estimation. In Figure 5.6 is represented how we found the real centre of the image. We prolonged the radial edges present in the picture, the crossing point of all the edges is the image centre. This estimations is simple and accurate enough for our purposes.

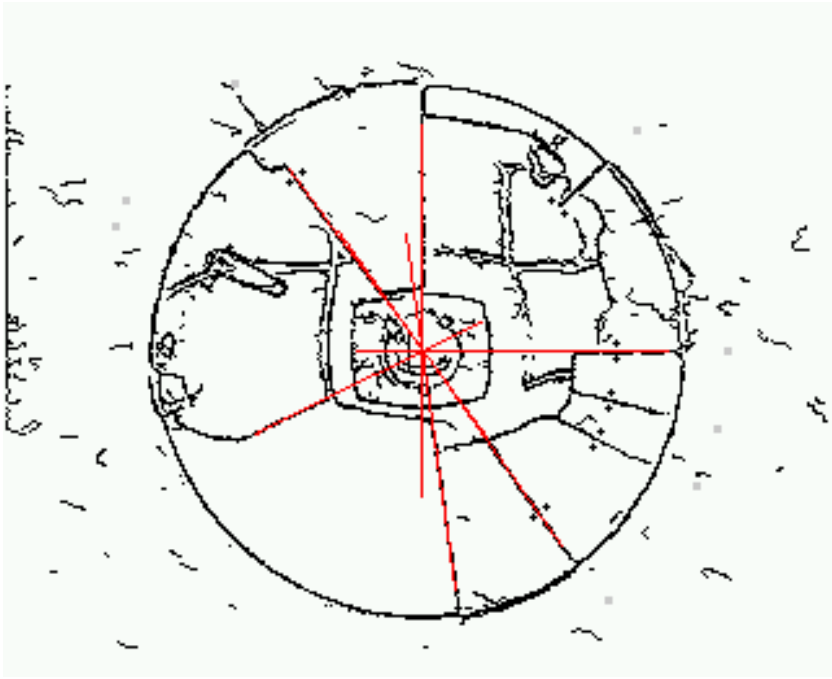
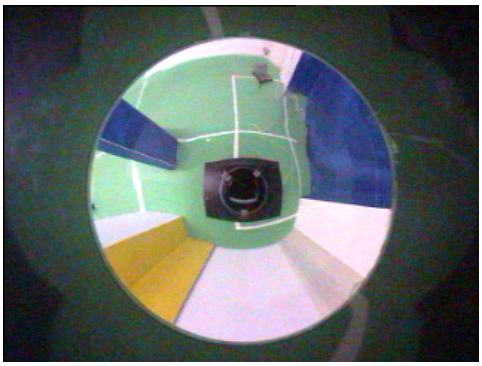


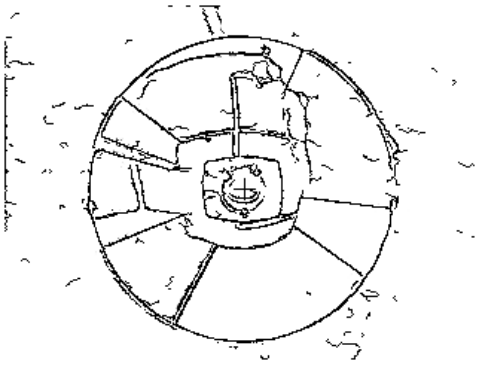
Figure 5.6: The centre of a real image. In red the prolongations of the vertical edges to find out where is mapped the vertex of the mirror.

5.4 The results

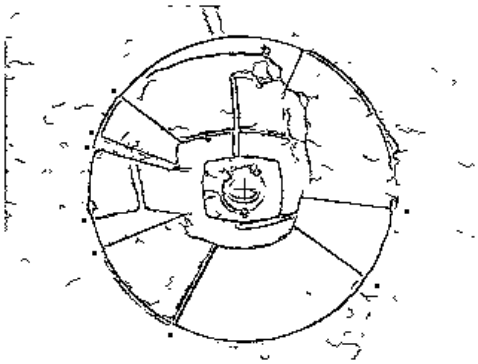
All the thresholds set in the previous chapter had to be reset when working with the real images. The images are much noisier now. In Figure 5.7 an example of the image process stages for a real image. The noise in the picture propagates down to all stages of the image process. The edge detection result is noisy, Figure 5.7 a). Several noise edgelets are detected, some edges



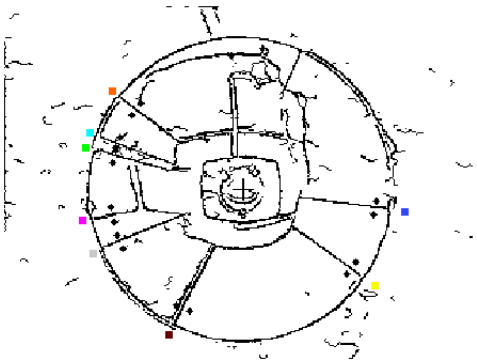
(a) The unprocessed image as view from the robot's camera.



(b) The image after the Canny edge detection.



(c) The edge image after the Hough transform.



(d) The final result showing the edge matching.

Figure 5.7: An example of the image process stages for a real image.

are broken and noise spots are present. This makes it more difficult to reliably detect the radial lines and the radial lines only. In fact, in the Hough transform the pixels of the noise sums over the edge pixels and sometimes the noisy pixels can trigger a false detection of a radial line. An example is shown in Figure 5.8. The arrow shows a marker of a detected edge but the edge does not exists in the picture. It is just the effect of several noisy pixels accidentally having the same angular coordinate. This cannot be avoided setting the threshold higher, because we would have to set it so high to loose significant edges. These false edges do not last for more than two frames and then they could be easily filtered out by the software in charge of reconstructing the topology of the environment, if this is fitted with a temporal memory.

The problem of the ephemeral edges exists also in the real image sequences. The noise of the images worsens the problem compared to the simulated images. It happens more often, that an edge disappear and then reappear after some frame. Again this could be solved by introducing a temporal memory of the edges.

The undermining problem in the real image is the noise introduced by the CCD sensor of the camera. This noise affects mostly the colour of the pixels. The noise is so high that even if the robot is steady in the same position, the edge signatures of the same edge in two consecutive frame are different. This problem has been solved with two changes in the vision system software: the tolerance within two colours are considered the same has been increased and the windows over which the mean colour is calculated have been doubled. See Figure 5.9. Doubling the size of the averaging window, we use more pixels to calculate the mean colour and then the edge signature values are more stable.

When the robot moves, the colours around the edges change slightly because of different shadows and different reflections on the surfaces of the

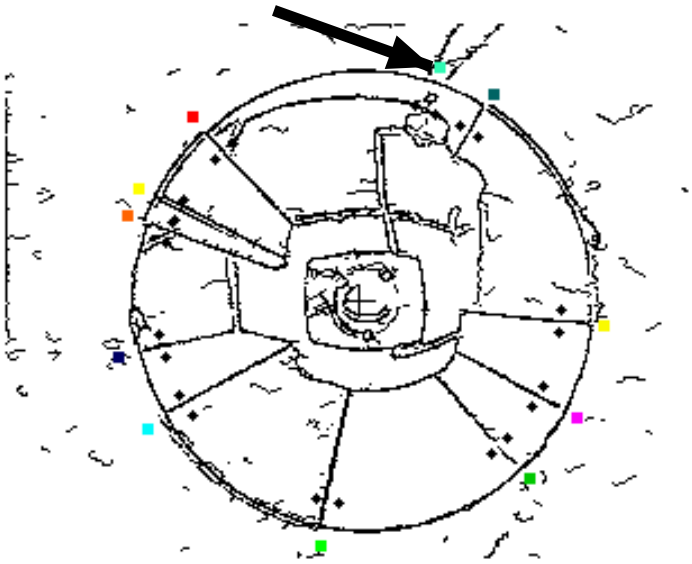


Figure 5.8: An example of false edge produced by the noise in the image. The arrow points the wrong marker.

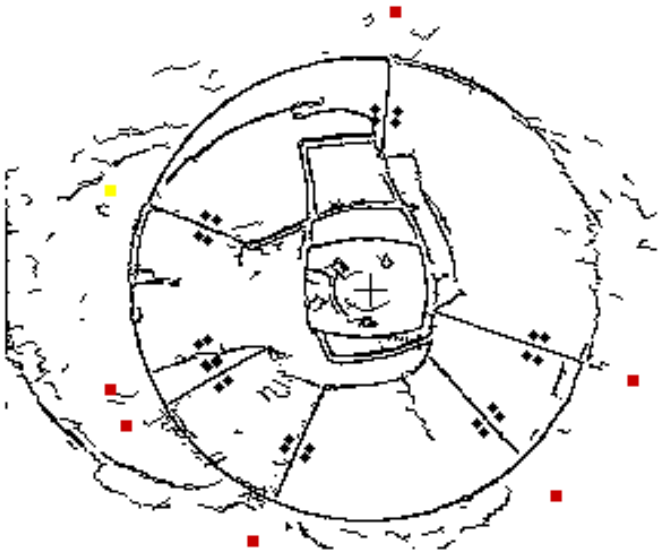


Figure 5.9: The averaging windows used in the real experiments. Notice that the averaging windows are doubled.

objects. To cope with this change we had to relax further the colour tolerance. Remember that the colour of every pixels of the image and the edge signature are represented as a RGB triplet. The new criterion we used is: two edge signatures are considered the same when at least two elements of their triplets are equal within a certain interval. This is enough to cope with the noise present when the robot moves in straight lines, but does not work when the robot turns.

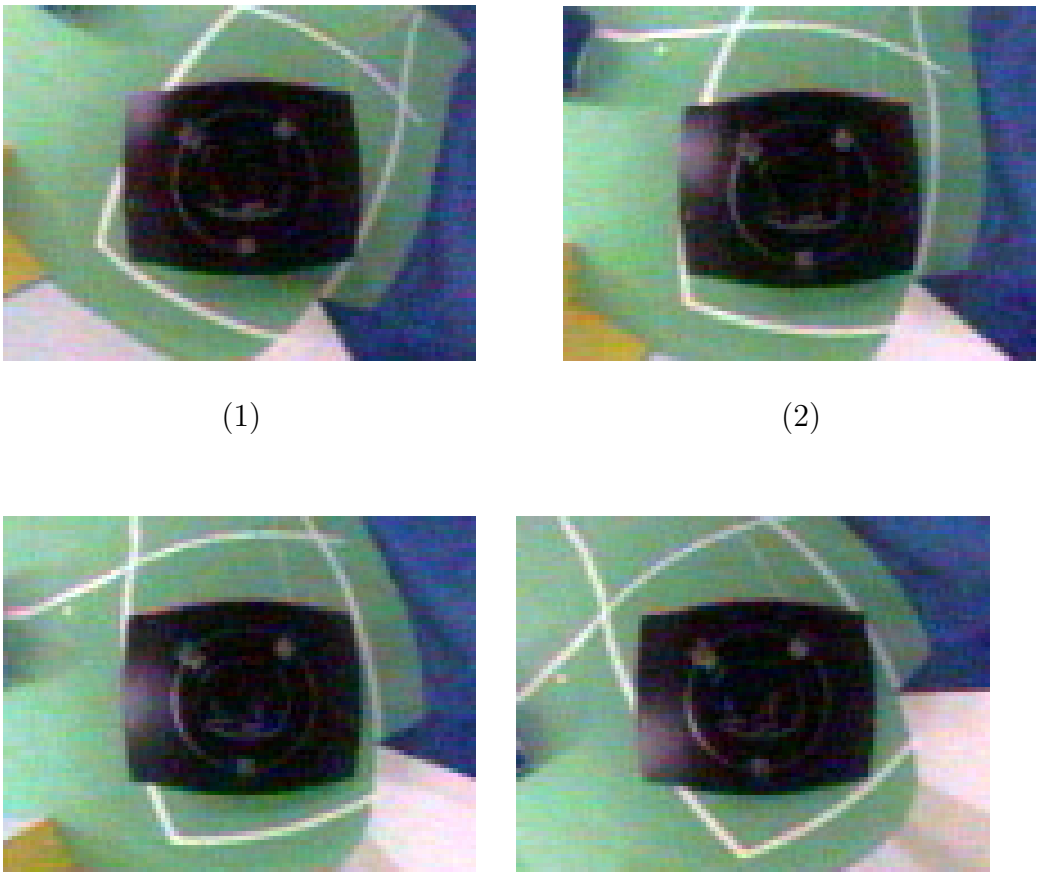
Analysing the images taken from the robot when it rotates, we discovered that the robot failed to recognise the edges because the colours on the sides of the edges in different images are actually different. Consider that the view of the scene does not change and even the shadows and the reflections on the objects' surfaces are the same, this should not happen, and the image should be exactly the same, only rotated by the angle the robot rotated. If the explanation of this change of the colours is not in the scene, it must be in the vision system. The software of the vision system cannot be held responsible for that: it works fine in the simulation. The only candidate is the hardware of the vision system. Remember the camera sees the world through the Perspex cylinder that support the mirror. The explanation of the change of colours can be found in the changes of the light reflection on the Perspex cylinder. These reflections change because the robot's body does not have a cylindrical symmetry. As you can see in Figure 1.3 not only the body is rectangular but also the cover on the top of the mirror has a rectangular shape. The purpose of this cover is to avoid that the camera is blinded by the lights on the ceiling, but it also shadows the Perspex cylinder preventing most of the reflection on it. Because the cover is rectangular, it shields better some region of the Perspex cylinder than others. When the robot rotates it sees the same points in the world through regions of the cylinders with different shadowing from the cover and so different reflections occur at the cylinder surface. The final effect is that the colours appear to

change. This effect is made even worst by the reflections occurring on the top cover of the robot body. See Figure 5.10 for an example in which these reflections are particularly strong. Notice the bright spot that seems to move on the left part of the robots body.

The explanation provided for the failure of the tracking algorithm in the rotations is consistent with the fact that this effect does not occur when the robot moves along a straight line. In fact, in this case the change in the reflection is not so sharp. See Picture 5.11 and notice how, in the case of a translation, the area shielded from the cover does not change much while in the case of rotation the shielding change totally.

5.5 The conclusion

The tracking of the edges on the image sequence acquired by the real robot works fine when the robot translates. It does not work when the robot rotates on the spot. Because of this failure it was not possible to build an autonomous behaviour for the robot. Remember that the only source of information for the robot is the vision system. If the vision system is not able to match the edges after a rotation, the robot does not know how much it has turned. This causes the impossibility to take a controlled change of direction or to check the heading of the robot. The check of the heading direction is necessary in an autonomous behaviour, because when the robot starts moving, it jerks and twists. Because of this, after some steps, the heading direction is changed. Therefore, without the edge tracking process on rotation, the robot cannot recover from cumulative errors in the heading direction and is not possible to build an autonomous behaviour.



(1) (2)

(3) (4)

Figure 5.10: A close up of a sequence showing an example of strong reflection on the body of the robot. Notice the bright spot on the left of the robot's body that appears to move from one frame to the next.

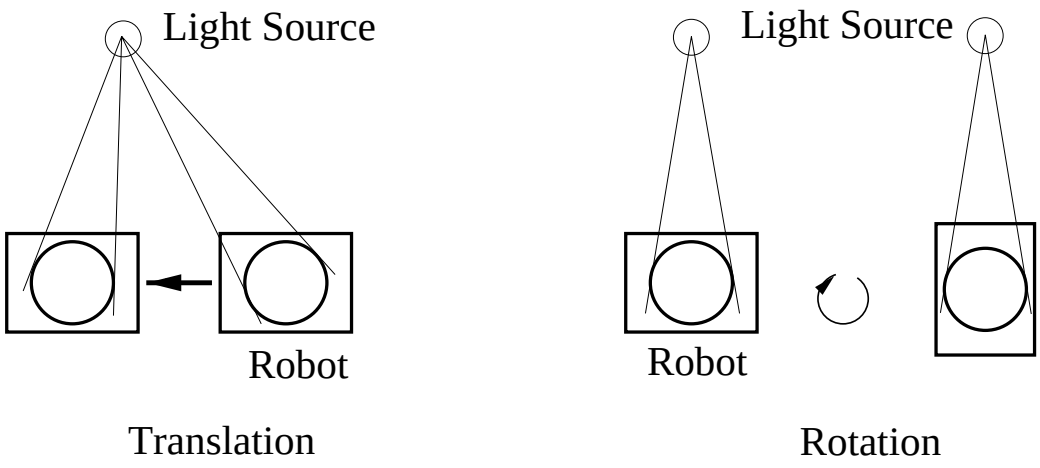


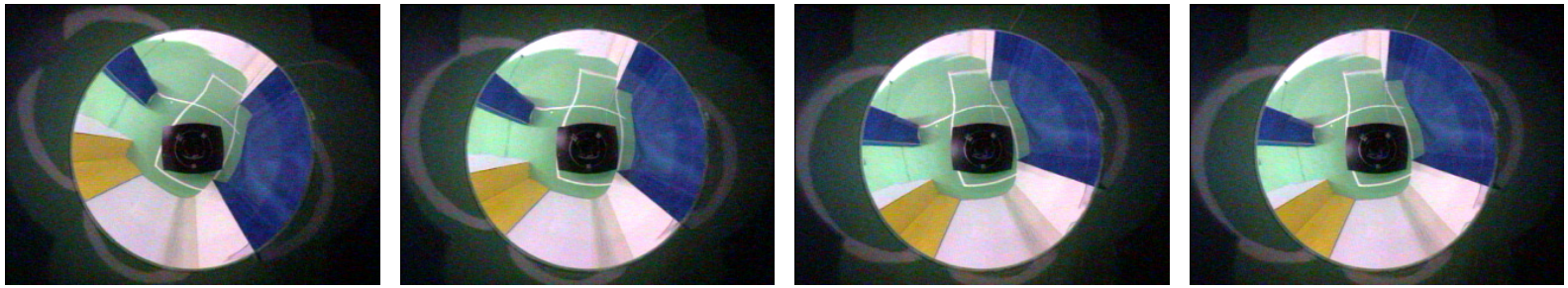
Figure 5.11: Difference in shielding between the translation and the rotation of the robot.

270

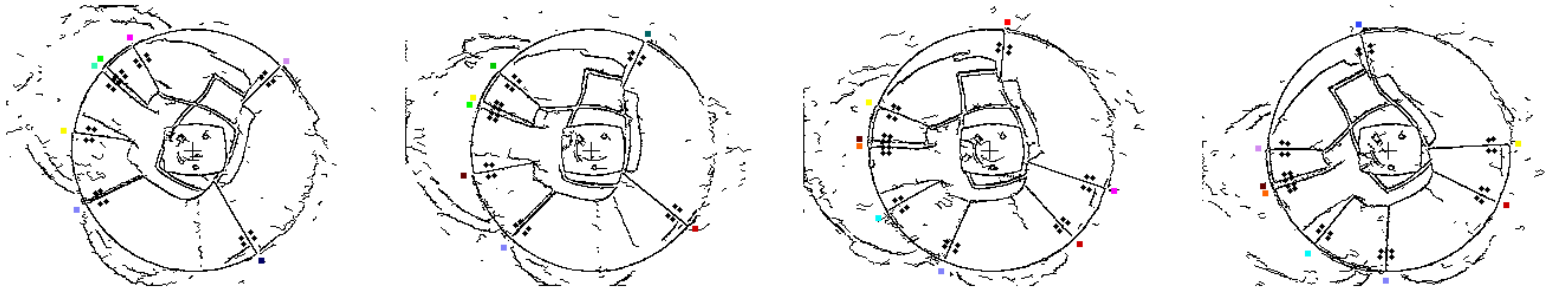
(1) (2) (3) (4)

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

Figure 5.12: (Top) The camera view of a real image sequence for a translation of the robot. (Bottom) The tracking of the edges in this sequence.



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



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

Figure 5.13: (Top) The camera view of a simulated image sequence for a rotation of the robot. (Bottom) The tracking of the edges in this sequence.

Chapter 6

Conclusions

This project achieved the first step in the implementation of the Spatial Semantic Hierarchy: we realized the sensory level with an omni-directional vision system. This is the first step towards the construction of an agent able to draw a topological map of the environment it travels through. Experiments were performed both in simulation and with a real robot. In this project, we focused on tracking of edges and no attempts were made to infer topological information from the data collected.

In Chapter 2, we showed that an omni-directional vision system is a good sensor for the SSH. We pointed out which of the features present in a omni-directional image can be used to detect the *transitions of state* needed by the control level of the SSH. We showed the existence of a strict link between the *views* of the SSH and the image taken by an omni-directional viewer. In Chapter 3, we built a vision system able to extract these features from the images. In Chapter 4 we showed that the software runs properly in a virtual environment and finally, in Chapter 5, we tested the same software on a real robot discovering that what worked in the simulations does not work in the real world. This confirms what Ronald Arkin said, which might seem self-evident but is often forgotten:

To conduct robotics research, robots are needed
—Ronald Arkin 'Behaviour-Based Robotics'

Unfortunately, we had very little time to work on the real robot, as this was available only at the University of Padua and, by the time the software was ready, the robot had to be taken apart to be brought to the **Robocup 2000 Competition** in Australia.

6.1 Future work

There is still much work to do on this project. The vision system should be improved: the sensory level should be made less sensitive to the noise, be it colour noise or noise in the edge detection.

It has to be investigated how to solve The problem of the ephemeral edges and of the false edges has to be investigated further to find an appropriate solution. We have suggested two solutions. The first is to have a vision system software with a degree of temporal memory, able to filter out a detected edge that does not correspond to a vertical edge in the scene. The second is to leave this filtering process to the software which reconstructs the topology of the environment from the raw data coming from the vision system software. It is not self-evident which of the two approaches would be best. The first would further increase the complexity of the vision software introducing a basic understanding of the environment not provided for the sensory system by the SSH. The second would allow noisy data to reach the topological level with the possibility of misinterpretations.

The problem of how to reliably detect the colour of the edges has not been fully solved. A solution might be to find a better metric to define the similarity of two colours within the RGB colour space. A better solution is probably to change the colour space. In fact, we discovered that the RGB space is not perceptually uniform. As such, the proximity of colours in RGB

colour space does not indicate colour similarity. It would probably be a good idea to transfer the image to the HSI colour space. In this space, colour pixels are defined as triplets representing hue (H), saturation (S) and intensity (I). This colour space has the advantage of being approximately perceptually uniform. Therefore, colour similarity implies proximity in this colour space [Smith 97].

Considering the great variance in the edge signatures, these would be better calculated over a wider surface than the averaging windows we used. It would be nice to use some colour segmentation technique for retrieving the whole surface between two consecutive vertical edges. This would have the double beneficial effect of maximising the surface over which to calculate the average colour and of assuring that we are sensing only pixels of the correct surface.

After the sensory level has been strengthened, all the other levels of the SSH have to be implemented. The distinctiveness measure needed by the control level has to be formalised profiting from the hints given in Chapter 2. The control laws have to be formulated and implemented. Finally, the topological level has to be built.

Finally, a real time algorithm should be produced at least for the vision system. In this way the robot could quickly collect the data and these could be analysed off-line to reconstruct the topology of the environment. The actual code is not real time, but there is space for huge improvements, a factor of twenty is not unrealistic.

Bibliography

[Bianco & Zelinsky 99] Giovanni Bianco and Alexander Zelinsky. Biologically-inspired visual landmark learning and navigation for mobile robots. *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1999.

[Bonarini *et al.* 99] Andrea Bonarini, Paolo Aliverti, and Michele Lucioni. An omnidirectional vision sensor for fast tracking for mobile robot. *Proceedings of the IEEE IMTC99*, 1999. Available at www.elet.polimi.it/Users/DEI/Sections/Compeng/Andrea.Bona

[Browitt 97] Goeff Browitt. Virtual reality telepresence using a mobile robot. Unpublished M.Sc. thesis, University of Edinburgh, Department of Artificial Intelligence, 1997.

[Canny 86] J. Canny. A computational approach to edge detection. In *Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 679–697. IEEE, 1986.

[Franz *et al.*] Matthias O. Franz, Bernhard Scholkopf, Hanspeter A. Mallot, and Heinrich H. Bulthoff.

Learning view graph for robot navigation. *Autonomous Robots*, XX:1–11.

[Greguss 96] P. Greguss. Pal-optic based instruments for for space research and robotics. *Laser and Optoelektronik*, 28:43–49, 1996.

[Hough 62] P. Hough. Method for recognizing complex patterns. Technical report, US Patent 3069654, 1962.

[Ishiguro *et al.* 92] Hiroshi Ishiguro, Masashi Yamamoto, and Saburo Tsuji. Omni-directional stereo. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, VOL. 14(NO. 2):pp. 257–262, February 1992.

[Ishiguro *et al.* 95] Hiroshi Ishiguro, Takesi Maeda, Takahiro Miyashita, and Saburo Tsuji. Building environmental models of man-made environments by panoramic sensing. *Advanced Robotics*, VOL. 9(NO. 4):pp. 399–416, 1995.

[Kortenkamp *et al.* 98] David Kortenkamp, Peter Bonasso, and Robin Murphy, editors. *Artificial Intelligence and Mobile Robotics*. AAAI Press/ MIT Press, 1998.

[Kröse *et al.* 00] B.J.A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. Feature selection for appareance-based robot localization. *Proceedings 2000 RWC Symposium*, 2000.

[Kuipers 00] Benjamin Kuipers. The spatial semantic hierarchy. *AIJ to Appear*, February 2000. Available at http://www.cs.utexas.edu/users/qr/papers/Kuipers-aij-00.html.

[Levitt & Lawton 90] Tod S. Levitt and Daryl T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3):305–361, 1990.

[Marchese & Sorrenti] Fabio Marchese and Domenico G. Sorrenti. Omni-directional vision with a multi-part mirror.

[Pierce & Kuipers 97] K. Pierce and B.J. Kuipers. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence*, 92:169–227, 1997.

[Smith 97] John R. Smith. Integrated spatial and feature image systems: Retrieval, analysis and compression. Technical report, Columbia University, 1997. Available at the URL:disney.ctr.columbia.edu/jrsthesis/thesissmall.html.

[Svoboda *et al.* 98] Tomas Svoboda, Tomas Pajdla, and Vaclav Hlavac. Motion estimation using central panoramic cameras. *IEEE Conference on Intelligent Vehicles, Stuttgart, Germany*, October 1998. Available at ftp://cmp.felk.cvut.cz/pub/cvl/articles/svoboda/svobIV98.ps.gz

[Winters & Santos-Victor] Niall Winters and Jose Santos-Victor. Mobile robot navigation using omni-directional vision.

[Wright & Deacon 00] M. Wright and G. Deacon. A catastrophe theory of planar orientation. *Int. Journal of Robotics Research*, 19(6):531–565, June 2000.

[Yagi 99] Yasushi Yagi. Omni directional sensing and its applications. *IEICE TRANS. INF. & SYST.*, VOL. E82-D(NO. 3):pp. 568–579, MARCH 1999. Available at search.ieice.or.jp/1999/pdf/e82-d_3_568.pdf.

[Yagi *et al.* 95] Yasushi Yagi, Yoshimitsu Nishizawa, and Masahiko Yachida. Map-based navigation for a mobile robot with omnidirectionalimage sensor copis. *IEEE Transaction on Robotics and automation*, VOL. 11(NO. 5):pp. 634–648, October 1995.

Appendix A

The different realizations of omni-directional vision

There are three basic classes of sensors that can see in all directions. They take advantage of different techniques:

- use of multiple images;
- use of special lens;
- use of convex mirrors;

A.1 Use of Multiple Images

This technique produces panoramic images by stitching multiple images together. The images can be taken in a sequence by panning a camera around the vertical axis passing through the focal point of the camera. This approach requires very fine calibration and synchronisation of the movements of the camera, but it provides high definition panoramic images. If the camera does not rotate about its vertical axis but around a vertical axis at a certain distance from its focal point, it is possible to obtain rough range

panoramic images by matching the views of an object from different positions [Browitt 97] [Ishiguro *et al.* 92]. See Picture A.1 . This technique is very slow. Some researchers use multiple cameras looking in different directions that shot at the same time. They merge the pictures from the different cameras to produce a panoramic cylinder [Ishiguro *et al.* 92].

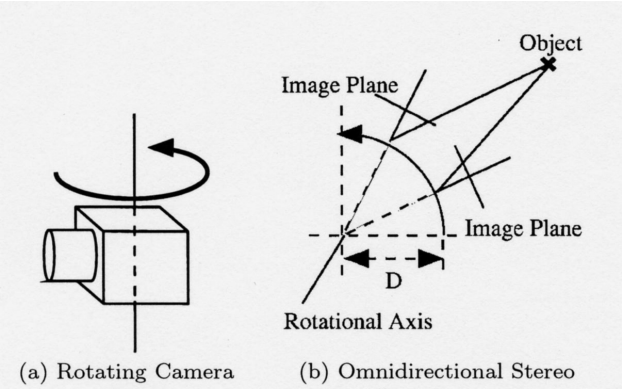


Figure A.1: The panning camera (left) and the range technique (right).(Courtesy of Prof. Y. Yagi at Osaka University)

A.2 Use of Special Lens

Cameras with the use of fish-eye lens can acquire almost a hemispherical view. The drawback is that the resolution of the images is good at the centre but very low at the periphery. This means that the resolution is very good looking at the ceiling but poor at the horizon. This is not good for robot navigation, where the objects to locate lie on the floor and they appear at the horizon or below. Greguss proposed an optical lens called Panoramic Annular Lens (PAL) composed of three reflective and two refractive planes [Greguss 96]. See Figure A.2 and Figure A.3. This does not need alignment and can be easily miniaturised.

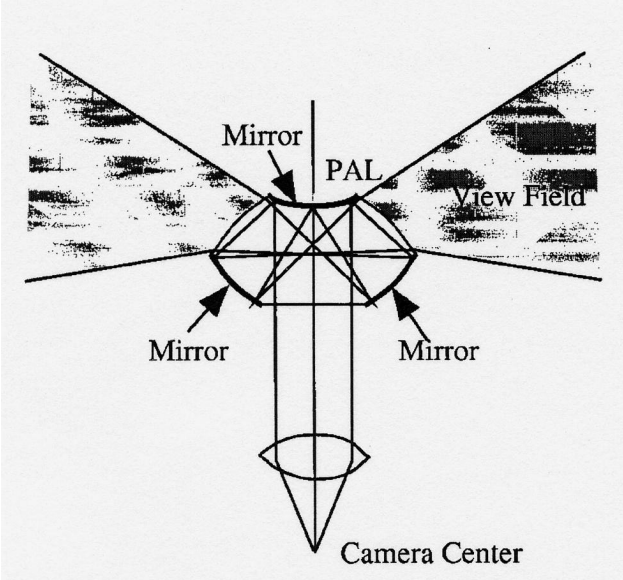


Figure A.2: The optical relation of PAL. (Courtesy of Prof. Y. Yagi at Osaka University)

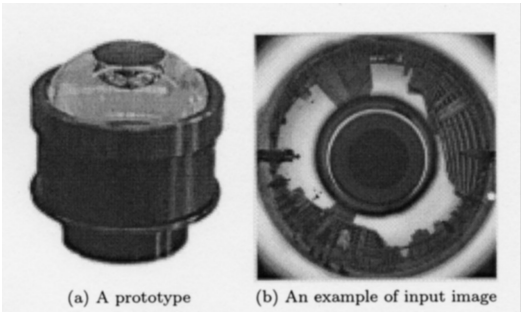


Figure A.3: A prototype of PAL (left) and an example input image (right). (Courtesy of Prof. Y. Yagi at Osaka University)

A.3 Use of Convex Mirror

This is the technique most widely used in robotics to obtain omni-directional images. The sensor is composed by a perspective camera pointed upward to the vertex of a convex mirror. The optical axis of the camera and the geometrical axis of the mirror are aligned. This system is usually fixed on top of a mobile robot like in Figure 1.3. Different shapes of the mirror have

been used. The most common are conical mirrors, hemispherical mirrors, hyperboloidal mirrors and paraboloidal mirrors; see Figure A.4. Every shape presents different properties that one has to take in account when choosing the mirror for a particular task. For instance, if the task requires the reconstruction of distortion free prospective images, it is mandatory to use a mirror with a sole focal point[1] [Svoboda *et al.* 98]. This assures the existence of a geometrical transformation from the omni-directional image to a perspective image. Let's consider more in detail the properties of the different mirrors:

- **conical mirrors** have good resolution in the peripheral, but they produce a singularity at the cone tip. They do not have a focal point.
- **hemispherical mirrors** they have good resolution in the centre area but poor resolution at the peripheral region. They do not have a focal point. On the other hand, these mirrors present the widest view angle among convex mirrors.
- **hyperboloidal mirrors** have good resolution both in the centre and in periphery. The view angle is wide almost as that of the hemispherical mirror, but the most important advantage is the presence of a focal point. This permits the reconstruction of distortion free images. As it is depicted in Figure A.5, from the image of a hyperboloidal mirror it is possible to reconstruct a panoramic cylinder or a perspective images at the desired angles.
- **paraboloidal mirrors** have the same resolution as hyperboloidal mirrors even if with smaller angle of view. They present a focal point when used with a orthographic projection camera.

For a more detailed survey on the different types of omni-directional sensor, see the survey paper of Yagi [Yagi 99].

[1]The focal point is the point where the prolongations of the reflected rays intersect.

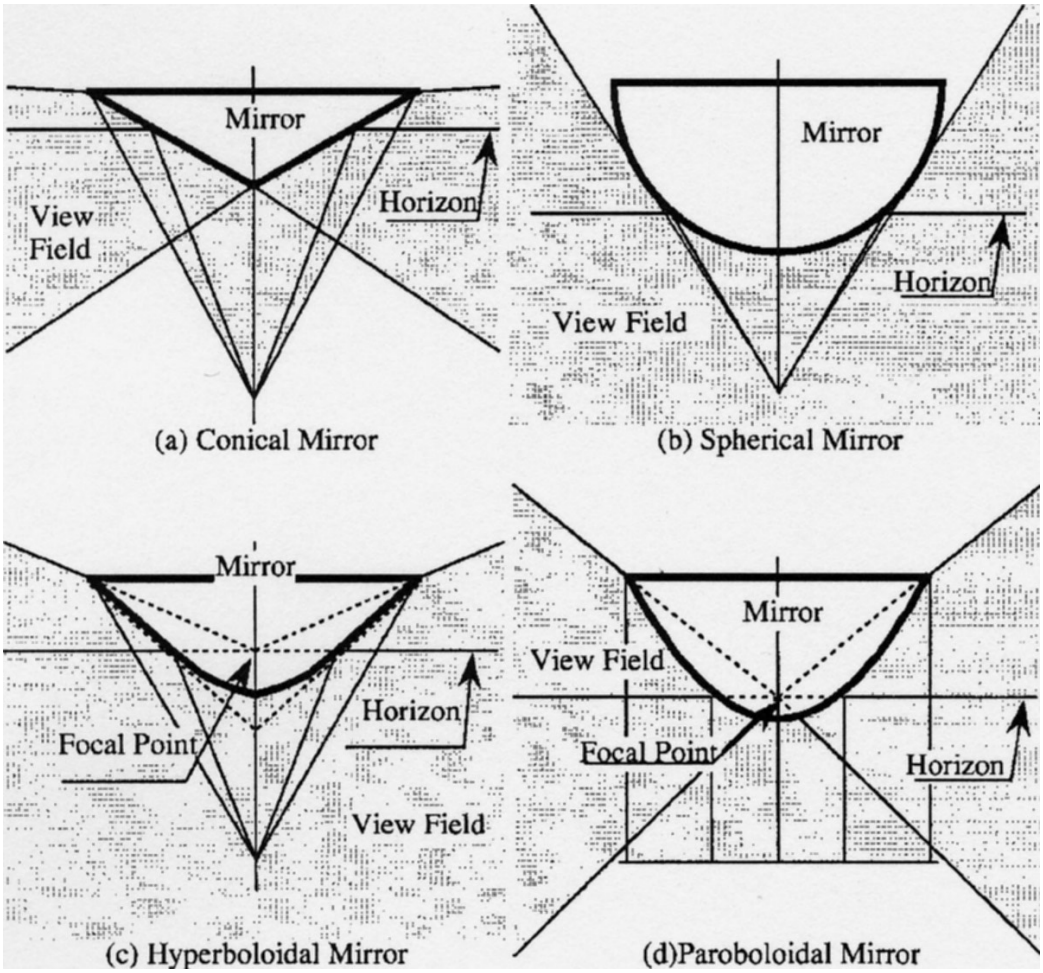


Figure A.4: The different mirror profiles. (Courtesy of Prof. Y. Yagi at Osaka University)

These typologies can be mixed to exploit the benefits of the different classes. For instance, Bonarini used a mirror composed by a sphere intersecting a reverse cone in order to avoid the excessive distortion introduced by the cone tip and have the good resolution of the hemispherical mirror in the centre of the image [Bonarini *et al.* 99]. Another example is the mirror of Marchese, he proposed a multi-part mirror for the specific task of the Robocup competitions (www.robocup.org). In this mirror, each part is devoted to the

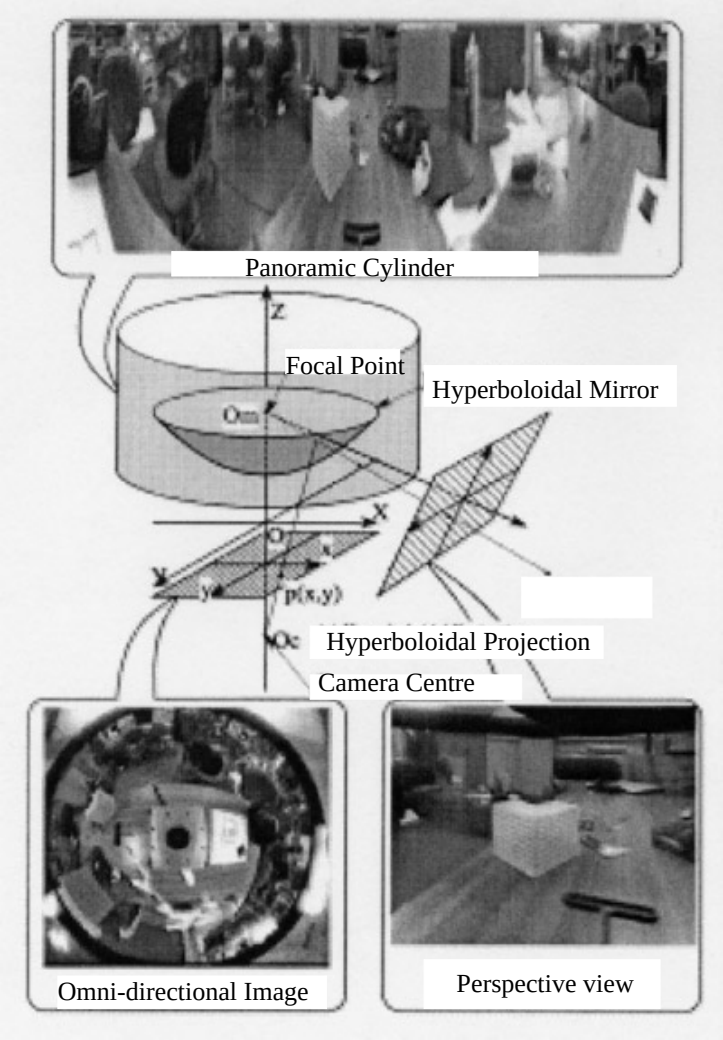


Figure A.5: Hyperboloidal projection and examples of transformed images. (Courtesy of Prof. Y. Yagi at Osaka University)

observation of a different area of the play ground [Marchese & Sorrenti].

As we see, the mirrors can be designed with arbitrary shape tailored for the application. To produce these mirrors out of glass would be very expensive and for some particular shape would be impossible. Fortunately, they can be realized from a cylinder of stainless steel shaped with a numeric control lathe. The surface of these mirror is smooth enough and reflective enough for the purposes of omni-directional vision.