Modelisation Visuelle d'un Objet Inconnu par un Robot Humanoide Autonome

Torea Foissotte

UNIVERSITE MONTPELLIER 2
SCIENCES ET TECHNIQUES DU LANGUEDOC

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE MONTPELLIER 2

Discipline: Génie informatique, automatique et traitement du signal

Formation Doctorale: Systèmes Automatiques et Microélectroniques

Ecole Doctorale: Information, Structures, Systèmes

présentée et soutenue publiquement

par

Torea FOISSOTTE

le 03 Décembre 2010

Titre:

Modélisation Visuelle d'un Objet Inconnu par un Robot Humanoïde Autonome

JURY

M. Philippe MARTINET	-	Président
M. Abderrahmane KHEDDAR	-	Directeur de Thèse
M. Ales UDE	-	Rapporteurs
M. Michel DEVY	-	
M. André CROSNIER	-	Examinateurs
M. Olivier STASSE	-	

Acknowledgments

And there go three years of study, research and discussions about humanoid robots, computer vision, optimization, etc. Along the way, of course, many works, events and people influenced the direction of this work.

My first thanks go to the members of the jury for their review and comments on this work. Particularly, I would like to thank Philippe Martinet for accepting to preside over the jury, Ales Ude and Michel Devy for their detailed and helpful comments on this manuscript, and André Crosnier for his help with various administrative tasks.

I am especially grateful to my director, Abderrahmane Kheddar, for offering me the opportunity to go back to the research world. He grabbed me out from the software development field in order to concentrate a little bit more about innovation.

A big thank you, of course, to my supervisor, Olivier Stasse, who guided my first hesitating steps on the path towards research. He motivated me to read a lot, shared a great deal of knowledge on various topics in robotics, and supported me on many technical issues which are inevitable with a system as complex as HRP-2. All this within a serious but friendly atmosphere.

My gratitude goes also to many collegues for their help and their work directly or indirectly useful for my own research: Adrien Escande for parts of his PhD work on top of which I've built upon, for the related technical support and various discussions, Pierre-Brice Wieber who introduced me to interesting algorithms I could rely upon, and Claire Dune, Nosan Kwak, Clement Petit and Thomas Moulard who worked on different vision software modules.

Many thanks to other friends and collegues in the JRL laboratory for the various discussions, their help, and the very friendly atmosphere they have helped to maintain: Sylvain Miossec, Nicolas Mansard, Jean-Remy Chardonnet, Paul Evrard, Hitoshi Arisumi, Mehdi Benallegue, and more... Wael Suleiman and Karim Bouyarmane deserve a special mention for their time and useful comments on early drafts of my manuscript. Thanks also for the rice! Special cheers to Francois Keith who shares the same interest for the Canadian culture. Thanks especially for the moral support at the dreadful time of thesis manuscript writing!

All of my work would have been seriously compromised without the constant support from my family members. My parents and sister for their constant encouragment and help, my sister (again) for her particular support during my thesis defense, Michi for supporting my choice of moving to a different town with reduced salary and reduced free time for 3 years, and Manatea for being such a lovely kid allowing me to concentrate more easily on my thesis.

My final thanks go to you, the reader, for showing some interest in my PhD work! As I have spent a lot of time and energy to work on this subject and write this

document, I surely hope it will be useful for someone somewhere sometimes! Do not hesitate to send me comments, remarks, criticisms...

This work is partially supported by grants from the Robot@CWE EU CEC project, Contract No. 34002 under the 6th Research program (www.robot-at-cwe.eu), and from the R-Blink ANR French national project.

Some of the 3D objects used in this work comes from the repository AIM@SHAPE maintained by INRIA.

The picture used in the introduction comes from wikimedia: “Two Piece Reclining Figure No.5” Henry Moore (1963-64) bronze, sited in the grounds of Kenwood House, London. Photograph: Andrew Dunn, 18 July 2005.

Several free softwares were used to help conduct my research: OpenHRP, Amelif, NLopt, blender...

Contents

List of Figures

Introduction

The beginning of knowledge is the discovery of something we do not understand.

Frank Herbert

For many years, the general interest for robotic systems has been expanding and the role of robots in society is predicted to continuously increase [Gates 2007]. Robots have been used in the industry sector for decades and they are emerging at a fast pace in the public market as toys, medical aid, specialized mobile robots, etc.

In parallel to the development of highly specialized robotic systems, several works are carried on with the purpose of developing advanced humanoid robots that are able to execute a great variety of tasks [Okada 2006]. The humanoid shape of such systems is supposed to facilitate their interaction with humans and to allow actions in human-centered design settings. The main potential applications include human assistance and autonomous handling of dangerous tasks. Some are designed specifically for the autonomous exploration of hazardous places such as contaminated areas or outer space [Bluethmann 2003].

Robots evolving in non-structured environments face particularly challenging working conditions with unpredictable modifications of their sensory information on a regular basis. This is particularly true when coming upon new places, people and objects. Nevertheless, for the proper realization of several autonomous tasks, e.g. exploration or inspection, it is necessary to adapt the robot actions automatically, based on new information. The perceived events are compared with the current knowledge in order to detect and interpret discrepancies, then the results are used to modify the future actions and perceptions of the robot. In the context of mobile robots and computer vision, this approach leads to the Active Vision field. It aims at generating the motions of a robot from the contents of the images obtained from an embedded camera. This is a particularly challenging field which includes a great variety of difficult problems that need to be solved in order to enhance the usability of robots in society.

The presented works are related to the Active Vision field and address the problem of generating autonomous behaviors for a humanoid robot based on vision. More specifically, we consider the autonomous three dimensional reconstruction of an unknown object by relying on the functionalities and specificities of humanoid robots. The task that the robot must achieve is the modeling of an object in a known en-

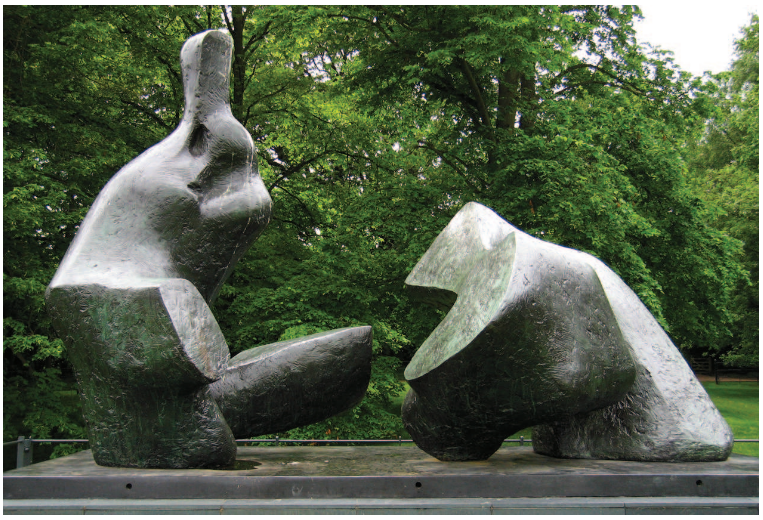


Figure 1: “Two Piece Reclining Figure No. 5” by Henry Moore

vironment that is possibly cluttered. The modeling process is autonomously guided based on the visual feedback from the cameras that are embedded in the robot head. The efficient completion of this task is complicated by the use of a humanoid robot. Indeed several constraints need to be dealt with: stability, collision, joint limits, etc.

Our goal is to obtain a model of the object that includes a 3D approximation of its shape and a set of 2D visual features detected on its surface from different viewpoints. Such a model could be used in complementary works, tackling different purposes, such as: fast detection and recognition, manipulation, or collision detection.

The selection of viewpoints of interest is the main scientific problem that we focus on. The proposed works give solutions to the Next-Best-Posture problem that we define as:

> “Given the model of the environment, the model of the robot, and the available visual information, what is the next configuration the robot should take in order to increase the information about the object?”

In order to have a better grasp of the problem, let us consider Figure 1 for a moment. A single image of the object does not hold enough information to create a reliable model. Based on a life-long learning experience, a human could infer some characteristics such as the 3D shape of the visible surface using an estimation of the material lighting properties. With familiar enough objects, a human could even guess the textured 3D model from a single image. But unless you are a connoisseur of the works of Henry Moore, you will need to move around this object in order to grasp its essence. You have several options to do so but also some constraints: the guard rail prevents you from getting too close, the objects in the scene such as the trees or the pedestal may occlude some parts, and you are probably not tall enough to see the object from 3 meters above. Given these constraints, you still have a large set of options from which to decide your next viewpoint. Your selection will most likely depends on the direct results from the processing of the object appearance based on the complex analysis of the texture, edges and concavities.

This work proposes to follow a similar approach for a humanoid. We select a perception pose that satisfy the constraints on the robot and the constraints from the environment while considering the visual information. Our methods are based on a specific posture generator, previously developed in our laboratory, where a constrained whole-body posture is computed by solving an optimization problem.

A first proposed solution is a local approach to the problem where an original rendering algorithm is designed in order to be directly included inside the posture generator. The rendering algorithm can display complex 3D shapes, represented by occupancy grids, while taking into account self-occlusions.

The second proposed method seeks global solutions by decoupling the problem in two steps: (i) find the best sensor pose while satisfying a reduced set of constraints on the humanoid, and (ii) generate a whole-body posture with the posture generator. The first step relies on global sampling and recent derivative-free optimization methods in order to converge toward good viewpoints, even in configuration spaces with many local minima. The second step is then used to generate a posture that satisfies all the constraints on the robot body while setting the robot head according to the sensor pose computed.

The processing of visual information in experimental conditions relies on various complementary vision algorithms. We need to acquire the 3D information from the robot stereo rig, detect visual features on the object surface, and update the model. The achievement of these steps is based on the investigation of different available algorithms. Using a humanoid which embeds cameras in its head, the visual processing of the images must be adapted to the perturbations of the camera motions that can occur due to the bipedal walking.

This thesis is driven toward the practical realization of a fully autonomous modeling experiment using a humanoid robot. The entire process requires the tight integration of a number of different but complementary software components, many of them being at the cutting edge of robotics research. In order to generate walking paths in a cluttered environment, we rely on a motion planner that can output collision-free trajectories that are feasible by a humanoid robot. The motion execution must ensure the continuous robust stability of the robot and the simultaneous realization of different control tasks.

We discuss the results of our approach in dynamic simulation and in real conditions with the HRP-2 humanoid robot located in the CNRS/AIST Joint Robotics Laboratory in the AIST Tsukuba research institute.

Chapter 1

Problem statement and background

Contents

1.1 Problem and context

What happens when someone, evolving in some familiar environment, find at some point an intriguing unknown piece of furniture? Let us assume that the object appearance is original enough that an approximate complete model cannot be inferred from the initial impression alone. For an average human, it is a quick and straightforward process to move around, and/or move the object, in order to create an approximate model good enough for further visual recognition and manipulation. Though it is an easy task to realize for humans, it requires the successful completion of various lower level tasks, most of them done unconsciously:

1. visual discrimination between the object and the environment

    (a) visual recognition of the environment

   (b) detection and recognition of visual features on the object

2. construct a model
   (a) recognize the 3D surface of the object
   (b) add perceived visual features on the model
   (c) check properties such as size, texture, weight, etc.
3. motion around the object
   (a) decide the necessary views
   (b) decide a destination position in order to reach the viewpoint
   (c) plan your walking motion
   (d) walk
   (e) stay stable
   (f) avoid (violent) collisions with the environment
   (g) avoid (violent) self-collisions
   (h) check your position relatively to the object
4. manipulation of the object
   (a) detect if the object is desolidarized from the environment
   (b) recognize grabbing zones on the object
   (c) decide grab zones to use
   (d) plan the motions of the upper-body
   (e) move the upper-body
5. ...

All of these tasks have been independently successfully achieved by various robotic systems in the scientific world. Nevertheless existing systems usually focus on the realization of very few of these tasks under specific conditions and relying on different assumptions.

The work presented in this thesis represents a significant step toward the realization of the autonomous visual modeling of unknown object using a humanoid robot HRP-2. A successful application relies on the coherent integration of most of the presented tasks. We believe such an ability is of interest as it can enhance the usability of multi-purpose robots in collaboration conditions with their human partners. For instance, visual models of new objects can be build and stored inside a knowledge database in an autonomous manner.

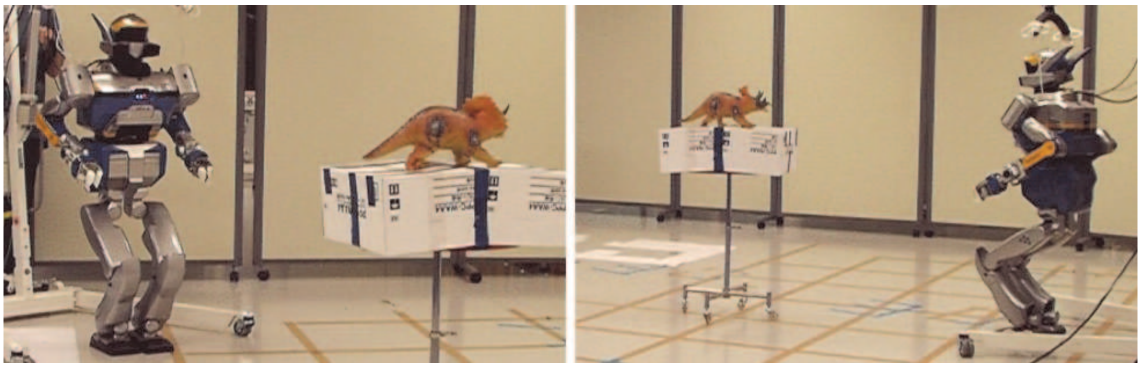


Figure 1.1: Next-Best-View for a humanoid robot: how to find the next valid posture to model an unknown object?

Though the designed framework aims at being as general as possible, We impose some limitations. Some assumptions are made about the operating conditions. Moreover, the manipulation of the object is not considered as it relies on specific research topics worthy of their own thesis: distinguish whether the object is movable or not, planning the grasp procedure, etc. We can note that some other works tackle parts of these problems using humanoids [Ude 2008] [Diankov 2009] [Vahrenkamp 2009] [Tsuji 2010] [Welke 2010].

Nevertheless, we intend to achieve a high-level behavior by integrating various software components developed by other researchers, and by developing original algorithms related to the planning of visual tasks. Specifically, our work considers the problem of generating and reaching new postures for the robot in order to complete a visual model of the object. This is known in the scientific literature as the Next-Best-View (NBV) problem.

The scientific problem tackled during this thesis relates to finding a Next-Best-View (NBV) solution for a humanoid robot. More precisely, the robot tries autonomously to create a relatively rough visual model of an unknown object by moving around it and take appropriate postures, as illustrated in Fig. 5.10.

In our case, the viewpoint search itself is not enough to ensure a valid acquisition: we must consider the limitations of the robot where the visual sensors are placed. Thus, at each iteration, our NBV algorithm finds a specific robot posture which ensures an efficient modeling process given the conditions and, more importantly, the gathered information. Indeed the efficiency of the modeling process and the optimal computation of the Next-Best-View depend on the assumptions made, the operating conditions, the caracteristics that need to be retrieved, etc. The conditions influencing which algorithm to choose are related to:

1. the sensor used: laser scanner, stereo vision, IR range sensor, active vision, etc.

2. the possible sensor motion: end effector of a robotic arm, mobile robot, humanoid robot, flying drone, etc.

3. the kind of object considered: textured, size, relatively homogeneous shape, etc.

4. the environment: known/unknown, presence of obstacles, possible occlusions, dynamic, lighting conditions, etc.

In this work, we consider the specific case of a humanoid robot HRP-2 in a known environment which can include some static obstacles. A stereo rig is embedded in the humanoid head and the available stereo vision algorithm can create range maps for textured objects with low reflectance. We do not put constraints on the shape and dimensions of the objects but an approximation of the dimensions is needed.

We note that our work does not consider the minimization of the sensor motion. Though adding the motion cost to the evaluation of a viewpoint could be use to minimize the trajectory for each iteration, nothing ensures that the total trajectory would be optimal as the number of required poses to complete the model may increase. Both criteria could be efficiently optimized if we would have access to a good estimation of the occluded parts of the object based on the known features. But, although it is an interesting research topic, the present work does not tackle the approach of learning and/or making predictions about the occluded object parts using the perceived parts.

The next section review some of the main works in related fields.

1.2 State of the art

The work realized during this thesis is part of the ”treasure hunting” on-going project [Stasse 2007] in the JRL laboratory which goal is the construction of the model of an unknown object and then its autonomous retrieval in a different environment that is considered unknown. We can notice a recent regain of interest [Meger 2008] [Forssén 2008] regarding this kind of problem with the Semantic Robot Vision Challenge [Rybski 2007].

The scientific problems addressed are at the crossing of computer vision, sensory planning and robotics; As such, various significant works in related specific fields have been studied in order to evaluate existing solutions, remaining limits and unexplored areas. In the computer vision field, the problem of visual object recognition has been thoroughly addressed; There are fast and robust methods which can be used in embedded systems [Lowe 2001]. Nevertheless, the modeling part usually relies on a supervised method where different views of an object are taken manually by a human operator and are then put in a database available to the recognition algorithm.

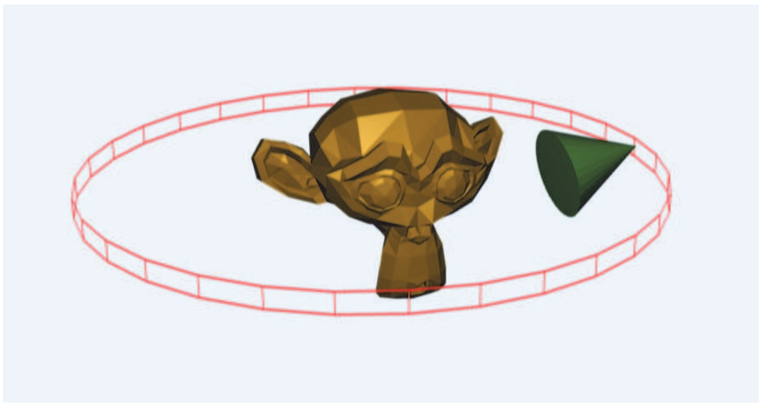


Figure 1.2: NBV problem for 1 dof system. The sensor position is constrained on the red circle. The green cone represents the camera which orientation is set automatically toward the red circle center.

The first recognized work on NBV is the one by Connolly [Connolly 1985] but works by Bajcsy [Bajcsy 1988] are considered instrumental to bring the fields of planning, control and vision together in order to develop the discipline of *Active Perception*.

Solutions to find a Next-Best-View depending on the visual information, come generally in two flavors, both introduced by Connolly: (i) create an evaluation function for the sensor pose and test a set of poses distributed in the configuration space in order to select the best one, or (ii) analyze the current model to deduce a reduced set of poses and select the best one as the next viewpoint. In both cases, as the evaluation of a pose is usually expansive, a dense sampling of the configuration space is avoided in order to retrieve a solution in an acceptable amount of time. Other shortcuts, such as the reduction of the configuration space dimension, or the non-consideration of some constraints, can also be used to speed up the evaluation. The second approach aims at having a much faster decision process by selecting very few poses, sometimes only one, based on some hypotheses on the link between parts of the visible surface and parts of the volume occluded.

This section presents an overview of selected papers on the NBV subject in order to review the main techniques. More extensive analyses of the works in the NBV field oriented toward object reconstruction are detailed in different surveys by Tarabanis *et al.* [Tarabanis 1995], Scott *et al.* [Scott 2003], and Chen *et al.* [Chen 2008]. Our review also extends to some selected papers representatives of the field of Active Vision.

1.2.1 Next-Best-View for 1 dof sensor

Few works [Maver 1993] [Pito 1999] consider the problem of object reconstruction by placing the object on a turntable while the sensor is maintained at a fixed distance, oriented toward the object center, as illustrated in Fig. 1.2. The search space of such solutions to the NBV problem is thus limited to a single dof. The algorithms mix a surface visibility criterion with an occluded surface visibility criterion to select the NBV position. Both methods rely on polygonal surfaces to describe the 3D structures of interest.

Maver and Bajcsy [Maver 1995] [Maver 1991] [Maver 1993] use an active range scanner and a turntable to model an object based on occlusions. The range scanner consists of a beam laser and a CCD camera that are located on the same frame which is translated in one direction to obtain a depth map of the entire scene. Thus the system is considered to have a single dof that is the rotation angle of the turntable. The occluded regions are approximated by polygons, and the viewing angles are computed for all of these polygons. A NBV is determined by relying on the maxima of histograms created by summing the viewing arcs for each polygon. The main goal of their NBV formulation is to scan into these polygons from directions which are not occluded. The algorithm is developed for the particular scanner setup.

In his PhD work, supervised by Ruzena Bajcsy, Pito [Pito 1996] [Pito 1999] improves the complexity of the search by reducing the number of tested sensor location. This is done by introducing a novel representation, the Positional Space, that represents the available and necessary parts to scan, in a single data structure. Pito also considers the need for surface overlap to register new surface information with the current model. The object is modelized using polygons and the occluded parts are represented with void patches, which are rectangular patches attached to the edges of the object and oriented according to the ranging rays of the sensor in order to lie at the limit of the occluded space. The size of these void patches is relatively small so that the overlapping constraint is satisfied and that no assumption is made on the object shape farther away from the edges. The algorithm then relies on the evaluation of discrete poses to find the NBV. The evaluation of a pose is done based on the number of void patches visible and the amount of already perceived surface visible. In addition, the performance of the range scanner is evaluated in order to incorporate the sensor particular sampling physics in the algorithm. Results are demonstrated both in simulation and with real objects.

1.2.2 Next-Best-View for 2 dof sensor

Now let us have a look at some works that consider a slightly bigger configuration space. Most NBV works for object reconstruction limit the sensor position to the

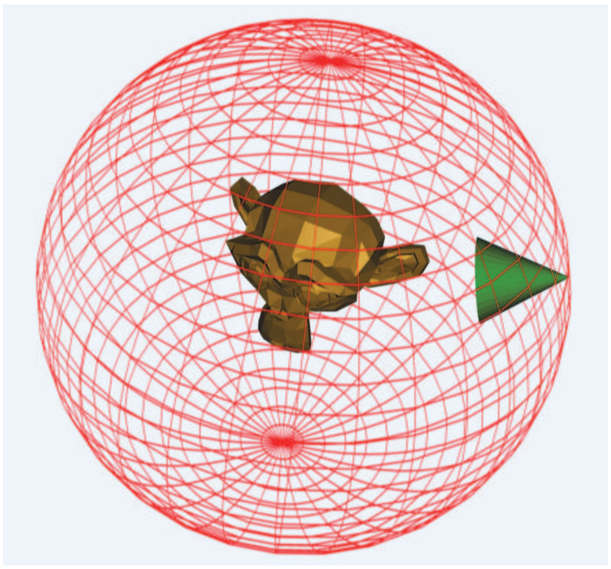


Figure 1.3: NBV problem for 2 dof system. The sensor position is constrained to the red sphere surface. The green cone represents the camera which is always oriented toward the sphere center.

surface of a sphere centered on the object to model, as illustrated in Fig. 1.3. The orientation of the sensor is set automatically toward the sphere center and thus its entire pose is decided using latitude and longitude coordinates alone.

The first paper in the NBV field is from Connolly [Connolly 1985] who presents two algorithms to model an object with an octree structure [Chien 1986] where voxels have one of three states: empty, occupied or unseen.

The first algorithm, labeled as the *Planetarium Algorithm* uniformly samples the sphere surface representing the sensor configuration space, and the visibility of the unseen area is tested for all samples. The viewpoint displaying the greater surface of unseen voxels is then selected.

The second method, labeled as the *Normal Algorithm*, is designed specifically to reduce the search time. The total area of planes between unseen and empty nodes is computed for the viewpoint in front of each of the six faces of the cube used as the root for the octree structure. The camera view direction is set toward the center of the object and its position on the sphere surface is deduced from the six areas computed. As stated in the paper conclusion, the second method is much faster than the first one but the self-occlusions are not dealt with correctly as nothing ensures that the area computed will be effectively perceived in the selected viewpoint.

In both methods, the exact quantification of the unknown area is not specified directly but we suppose it is a ray tracing method.

The approach of restricting the sensor position to a sphere surface and the sensor orientation toward the sphere center has been used a number of times in works following Connolly. Each work considers different sensor pose evaluation functions and/or rely on different methods to select a limited number of points on the sphere surface to evaluate the corresponding viewpoints.

In a first work by Banta *et al.* [Banta 1995], the next viewpoint is computed by analyzing the points in the range image of the object obtained from the current viewpoint. The three points in the range image with the highest curvature are selected. For each of these three points, the camera is set in the same direction as the normal vector and the visible unknown area is computed by ray tracing. The view with the largest visible unknown area is selected. The reconstruction process, based on an occupancy grid, terminates when no more unknown voxels remain or when a maximum number of iteration is reached.

Though the method work in simulation for the presented objects, there is a possibility that the reconstruction stay stuck in some local space. Indeed, when dealing with some object shapes which display a specific arrangement of curves, the algorithm may be attracted toward the same viewpoints.

In a more recent work by Banta *et al.* [Banta 2000], three simple algorithms are combined. The first one, labeled as the *Edge-Based Sensor Placement*, is executed only once to select the second viewpoint and aims at targeting large areas of occlusions. The method is similar to the algorithm in the previous paper discussed: it is assumed that the greatest curvature on the edges of the range map conceals greater quantities of unknown information. The next viewpoint direction is then set using the surface normal vector of the corresponding region. Theoretically, the assumption does not hold as the shape of an object is not directly linked to the shape of some local areas. Practically, this may be the case for a range of usual objects but no analysis is available.

The second method, labeled as *Targeting the Occluded Surface Centroid*, is executed twice to select the third and fourth viewpoints. The viewpoint is set toward the centroid of the occluded region. As precised in the paper, this is efficient only when there is a single continuous occluded region, and thus the method is used only amongst the first steps of the reconstruction process.

The last method, *Clustering the Occluded Surface Data*, is executed for the later view acquisitions in the modeling process. The remaining unknown voxels are classified into one or more clusters of joint voxels. The viewpoint then targets the mean point of the current largest cluster, at each iteration.

Additional methods are implemented to modify the viewpoint selection in the cases where self-occlusions appear or where the computed viewpoints are concentrated in a local area.

In a paper by Massios and Fisher [Massios 1998], also based on an occupancy grid representation, a quality criterion is inserted to judge a viewpoint in addition to the evaluation of the visibility of unknown parts. This criterion depends on the viewpoint direction and the voxels normal vector. The quality for a specific voxel is considered high when its normal vector is aligned with the viewing vector. Sampled positions on the sphere around the object are tested with the visibility and quality criteria to find the next viewpoint. Some samples on the sphere surface can be ignored in order to cope with constraints such as joint limits, collisions, etc. The algorithm is tested with a laser range scanner and a mobile base where lies the object to model. The mobile base has two degrees-of-freedom (dof): one in translation and one in rotation. The quality criterion is used mainly to refine the shape of the object to enhance the precision of the model once only small parts of unknown remains. The tests detailed do not show if the addition of the quality criterion can be useful for the reduction of the needed poses to model the object.

Wong *et al.* [Wong 1999] demonstrate another kind of NBV method in simulation which relies on an occupancy grid representation. Two methods are detailed. The first one is similar to Connolly's planetarium algorithm [Connolly 1985] but uses voxels instead of the octree structure. The second relies on unknown voxels visibility as well as their normal vector.

Chen and Li [Chen 2005] [Chen 2004] [Li 2005] use a structured light system, which is fixed on a robotic arm, to get a dense disparity map from different viewpoints. In order to compute a single NBV, the algorithm analyzes the curvature of small areas at the edges of the perceived surface. In contrast with the works of Banta *et al.* [Banta 1995], they consider that the prediction of the unseen parts of the object is more accurate when the curvature of the surface at an edge is small. But, as noted earlier, there is no practical analysis on the reliability of such hypotheses.

Bottino and Laurentini [Bottino 2006] present an interactive approach to model a convex polyhedra using silhouettes. They introduce a criterion and a formulation for testing if the reconstruction is finished. It uses a characterization of points in the visual hull and the real object silhouettes. Their algorithm starts with two random views and then gives some guidelines to help the user select the next view in order to examine a selected edge. Their method is designed explicitly for polyhedra and thus cannot be used for curved objects. Furthermore, similarly to all shape-from-silhouette traditional methods, concavities cannot be modeled.

1.2.2.1 Sensor motion

Few works consider the variation of the object visual features depending on the sensor motion. In this respect, such works are closer to the field of Visual Servoing then NBV.

Kutulakos *et al.* [Kutulakos 1992] [Kutulakos 1994] introduce a NBV algorithm for a shape-from-contour method based on the analysis of occlusion boundaries of objects. Their exploration scheme is formulated as the task of forcing the occlusion edges to slide over the points occluded from previous poses.

Another work, relatively original in the NBV field, by Arbel et al. [Arbel 2001] use entropy maps to guide an active observer along an optimal trajectory. They formulate their NBV search as the minimization of entropy. The object model is constructed using optical flow and offline PCA analysis, thus they do not have an explicit 3D shape of the object. Their system can use the model constructed to recognize objects through a minimal set of motions that aim at reducing the ambiguities.

1.2.2.2 Parametric model

The NBV system designed by Whaite and Ferrie [Whaite 1997b] [Whaite 1997a] relies on the parametric modeling of the 3D objects in a scene by using super quadrics. Their system uses a laser range finder located on a robotic arm in order to acquire a depth map of the scene. The sensor motion is constrained to a view sphere at a fixed distance from the center of the scene. The objects in the scene are described using nonlinear superellipsoid models. Their strategy for NBV selection is based on the uncertainty of the internal model, already investigated in a previous work [Whaite 1991]. More specifically, the best sensor locations are those where the ability to predict is considered worst, based on the variance in the fit of the data to the current model. The search is conservative with motions limited in their magnitude, and thus, their is an overlap between the acquired surfaces. The algorithm is tested with experiments that show the attraction of the method toward regions of high object curvature.

Parts of the work by Dune *et al.* [Dune 2009] [Dune 2008] related to NBV are inspired by Whaite formulation [Whaite 1997a]. The view selection is also done based on the uncertainty in the parameter space, the main difference is that the models are approximated by quadrics instead of superquadrics.

Such parametric approaches are well adapted to NBV solution based on Visual Servoing methods. Nevertheless, by using quadrics or superquadrics, the constructed model offers a limited descriptive value.

1.2.2.3 Polygonal model

Zha *et al.* [Zha 1997] use a laser scanner on a robotic arm. The point clouds obtained from the range sensor are converted into a polygonal surface composed of triangles. The configuration space is sampled and each position is tested using a function evaluating three criteria: (i) select a viewpoint far away from the previous ones, (ii) keep some parts of the perceived surface visible to ensure overlapping, and (iii) ensure that overlapping areas are relatively smooth.

In the works by Reed *et al.* [Reed 1997a] [Reed 1997b], the method is based on simulation and relies on a polygonal representation of the object model. The viewpoint planner aims the occluded parts of the object based on the detection of occlusion edges. The visibility of the targeted occluded parts is represented by a polygonal volume which can be used in conjunction with the sensor constraints to find the NBV. Their approach though does not address constraints such as limited field-of-view, or overlap.

1.2.2.4 NBV for object reconstruction summary

The aim of most algorithms, apart from the methods relying on parametric models, is to get an accurate 3D reconstruction of an object, using voxels or polygons, while reducing the number of viewpoints required. The most usual assumptions about the sensor are that the depth range image is dense and accurate by using laser scanners or structured lighting, and that the camera position and orientation is correctly set and measured relatively to the object position and orientation. But such assumptions cannot be made when using an embedded stereo rig on a humanoid robot controlled in an autonomous manner. Although there are many works dedicated to the planning of sensor positions in order to create the 3D model of an unknown object, the specific topic of autonomous object modeling by a humanoid robot could not be found.

In previous works, the object to model is considered to be inside a sphere or on a turntable, i.e. the sensor positioning space complexity to evaluate is reduced since its distance from the object center is fixed, and its orientation is set toward the object center. This restricts the variety of objects that can be modeled, e.g. objects with complex concavities may require the sensor to get closer to some specific parts and/or require having a viewing vector not targeting the object center. Previous methods also consider that the object is always completely included in the sensor field of view, but this may not be the case if the object size is large compared to the boundaries of the sensor admissible configuration space.

Thus hypotheses used previously impose some constraints on the sensor characteristics as well as on the object size and complexity which may not be adequate with a humanoid carrying tasks in a shared environment with humans. To get rid of such limitations while taking into account the constraints related to a humanoid robot, an original NBV algorithm need to be designed. Works addressing a broader range of problems have thus been reviewed.

1.2.3 Next-Best-View for exploration

Though the configuration space of the robotic system where the sensor is embedded is rarely considered in NBV works, it is more prominent in works dealing with environment exploration tasks.

The works by Wang, Huang and Gupta [Wang 2003] [Huang 2005] [Wang 2006] consider explicitly the problem of the configuration space of the sensor for a NBV problem but they restrict their analyses to simulations in a 2D plane. They use a eye-in-hand system composed of a planar 2-link robotic arm with a triangle FOV range sensor. The algorithm is designed to balance two goals: (i) explore the local environment of the robot in order to detect obstacles and increase the area of the configuration space of the robot, and (ii) maximise the amount of unknown visible by the sensor at each perception.

Sanchiz and Fisher [Sanchiz 1999] consider a NBV method for environment modeling, using a mobile robot equipped with a range sensor in simulation. The environment is represented with a voxel map which is updated using ray tracing. Each voxel has one of five states amongst: unknown, empty, occupied, occluded, and occlusion plane, which is an occluded voxel adjacent to an empty voxel. A normal vector is assigned to each occupied voxel using the normals of the points in the range maps obtained by the range sensor. Their NBV method relies on three criteria mixed in a polynomial function used to evaluate a sensor pose: (i) overlapping with known views, (ii) elimination of occlusion planes areas, and (iii) observation of unseen areas. The search of a 5 dof sensor pose is done locally to the current pose using two search methods: (i) the simplex method for selecting the best 3D position and (ii) an exhaustive sampling that test a finite number of 2D orientations (pitch and yaw) at each tested position. Although their approach demonstrates exploitable results in simulation, we note that the simplex methods is known to have convergence problems even for convex functions [McKinnon 1998] and thus other optimization methods are preferred.

Klein and Sequeira [Klein 2000] consider a mobile robot with a range sensor composing an acquisition system that provides 8 dof: 3D position, 3D orientation, field of view and resolution. They propose a NBV method relying on occlusion analysis and on a criterion related to the quality of the sensing. Their method uses hardware acceleration to decrease the computation time for visibility evaluations. They use environment maps [Greene 1986] to compute an approximation of the visibility of the scene for one 3D position of the sensor. These maps are then used for fast evaluations of their criterion. Global optimization of their NBV criterion is ensured through a simple global sampling. The use of environment maps is an interesting technique in the case of the environment exploration problem as large areas can be perceived simply by changing only the sensor orientation. This cannot be applied efficiently to the problem of object reconstruction, though, as, in this case, modifications of the sensor orientation alone are usually less pertinent than the modification of the sensor complete pose for the perception of unknown areas.

Martinez-Cantin *et al.* [Martinez-Cantin 2009] address the problem of online path planning for optimal sensing using a low-cost camera on a mobile robot with three dof: 2D location and orientation. The mobile robot starts with a rough probabilistic estimate of its pose and the location of some landmarks in the environment, and then tries to reduce the uncertainty about these. The planning problem is modeled using a POMDP [Kaelbling 1998] [Thrun 2005] where the reward is a function of the belief state. The expected cost function is approximated by a gaussian process and is used to balance two strategies: exploration (minimize uncertainty in the unknown parts of the policy space), and exploitation (reinforce current estimation). The application of such an approach to the NBV problem for object modeling is not straightforward as we need to recover and update a complex 3D shape. Because of the novelty of this work, we could not further investigate its proper exploitation for our problem, but probabilistic approaches definitely need consideration.

Lopez-Damian *et al.* [Lopez-Damian 2009] present a NBV method for indoor environment modeling, relying on an occupancy grid. They simulate a range sensor on a mobile robot that has 5 dof: 3 for the position and 2 for pitch and yaw orientation. The NBV search is similar to the work by Sanchiz and Fisher [Sanchiz 1999]: they use the simplex method to test positions and a tessellated sphere for the orientations. Their evaluation function is formulated as a probability density function that incorporates the percentages of voxels visible depending on their type, and a quality factor based on the view angle relatively to the voxels normal vector.

1.2.4 Visual Search

In the context of visual search, only one work was found which explicitly considers a sensor embedded in a humanoid robot though the sensor configuration space is limited to 4 dof: 2D position and pitch and yaw orientation. This work was done in our laboratory by Saidi *et al.* [Saidi 2007b] [Saidi 2007a] with HRP-2. The proposed algorithm aims at maximizing a target detection probability while minimizing criteria such as time, trajectory length and energy. The state of the environment is modeled with an occupancy grid. In order to decrease the computation time of a NBV, they use the concept of visibility map that is relatively close to the concept of Positional Space introduced by Pito [Pito 1999].

The work by Chung and Burdick [Chung 2007] presents a theoretical analysis of a decision-making framework adapted to probabilistic search. The environment is modeled as an occupancy grid and each voxel has a probability of containing the target. They design and compare five simple search strategies that are used to compute the successive sensing actions: the first two do not consider the current knowledge and just serve as a basis for comparison, the third one is an exhaustive search of a local part of the configuration space, and the two last consider the best candidate in the local area.

Forssen *et al.* [Forssén 2008] use cameras and a planar laser range sensor on a mobile robot to detect and recognize some target objects in an unknown environment. Their system has five dof: 2D position, pan, tilt and zoom. Potential objects are detected using depth from stereo cameras and saliency maps [Hou 2007]. The environment is modeled as a 2D grid map. Their visual search is based on three behaviors: (i) exploration phase to discover new areas with the range sensor, (ii) visual coverage to look for potential objects, and (iii) image acquisition of promising objects from new perspectives. The decision making strategy is based on several task-dependent heuristics that are not developed in the article.

Shubina and Tsotsos [Shubina 2010] consider the visual search of an object using a range sensor on a mobile robot which has three dof: 2D position and orientation. The environment is modeled as a 3D occupancy grid where each voxel is assigned a probability of containing the target. Their approach consider the orientation search and the position search separately and thus several different orientations are tested for each different position of the mobile robot.

1.2.5 Viewpoint selection related works

A little outside of the NBV field but still in the domain of looking for viewpoints to model an object, Ahuja and Veenstra [Ahuja 1989] aim at constructing an octree model using orthographic silhouettes of an object. Their paper argues for the use of 13 predefined viewpoints in order to get a rough model in a reduced computation time. They obtain good results for relatively complex shapes but their method is tested only in simulation and rely on a sensor with orthographic projection.

Works by Triggs and Laugier [Triggs 1995] consider the problem of visual inspection of a 3D model by a robot mounted CCD camera, which can be related to the "art gallery" problem [O'Rourke 1987]. The method relies on an original global function optimization technique, based on an octree structure, to examine specific parts of a known model while avoiding occlusions and collisions. As the object model as well as the environment are known, the method first computes an optimal set of unordered viewpoints and then optimize the trajectory of the sensor, similarly to the traveling salesman problem [Cormen 2001]. This method shares some similarities with our approach and is discussed in more details in chapter 3.

In the context of mobile robot control for automatic 3D modeling of an object relying on vision, we can mention the works by Yamazaki *et al.* [Yamazaki 2004a] [Yamazaki 2004b]. In these works, a small mobile robot makes a circle motion around an object in order to build its model. The algorithm uses visual features detection and structure from motion in order to recover a rough 3D model. The motion around the object is planned off line: it begins with a straight line in order to reach a fixed distance to the object, and then the robot executes a circular motion which radius depends on the given size of the object.

Works by Ude *et al.* [Ude 2008] create an image-based model of an object by manipulating it. They consider a humanoid whose cameras embedded in the head are used for stereo perception, and whose arm allow grasping and manipulation tasks. Their algorithm brings the object to a fixed distance from the camera and then rotates it to obtain several images. The possibility to change the grasping is not considered and thus some parts of the object cannot be modeled.

The works developed in the field of sensory planning aim at generating and controlling vision tasks during the robot motion. In this context, the environment model is available and the robot motion trajectory is an input of the algorithm. The problem is to compute the orientation of the robot sensor depending on its location in the trajectory in order to keep some landmarks or known object visible.

Works by Deng *et al.* [Deng 1996] consider the sensory planning problem in simulation for a 2D map. The algorithm relies on methods from graph theory to track limited sets of point landmarks along the trajectory and takes into account the cost of transition between the perception of different sets.

Seara *et al.* [Seara 2001] [Seara 2003] [Seara 2004] consider the practical problem of locating a humanoid robot while it is following a planned trajectory. The visual attention of the robot is shared between two perception tasks: (i) obstacle avoidance, and (ii) landmark detection and tracking for online localization. A multi-agent approach is used to compute the humanoid head orientation for a discrete set of positions on the trajectory.

Marin-Hernandez *et al.* [Marin-Hernandez 2005] present a method for a camera embedded on a mobile robot, demonstrated in simulation. The algorithm considers planar landmarks in the environment. It relies on a discretization of the configuration space and dynamic programming to compute the sensor orientation relatively to its location.

A work by Michel *et al.* [Michel 2008] tackles a slightly different problem where a humanoid trajectory is planned according to the feasibility of the visual task to realize. The goal is to visually track a known object while moving to a desired position with a HRP-2 humanoid robot. The algorithm considers the target object visibility and possible occlusions by 3D obstacles in the environment in order to generate a possible trajectory.

1.2.6 Visual servoing

The technique of Visual Servoing uses information extracted from a camera image, or another visual sensor, in order to control a robot motion. Works in this field date back to 1979 [Agin 1979] [Hill 1979] and are still actively addressed. Related control schemes aim at minimizing an error between the desired values of some visual features and their current measurement.

In the initial works related to Visual Servoing, two basic approaches have been developed [Corke 1994]: image-based (IBVS) and position-based (PBVS). Both approaches need to recover the camera intrinsic parameters. In IBVS, some visual features in the image plane coordinates need to be set to some specific location. The PBVS approach is related to the 3D localization problem in computer vision and relies, first, on the 3D model of the target object to track and, second, on the estimation of the camera pose in a reference coordinate frame. For more information about the state of art in Visual Servoing, Chaumette and Hutchinson made a tutorial [Chaumette 2006] and a presentation of recent advanced techniques [Chaumette 2007].

Visual Servoing methods for the purpose of autonomous object reconstruction have been exploited by some researchers, including Marchand [Marchand 1996] and Berry *et al.* [Berry 2000], both using an eye-in-hand robotic arm. Marchand considers the problem of scene modeling based on edge detection and recognition. The algorithm searches for the best trajectory based on the volume swept by the sensor (computed by ray-tracing), the joint limits and the motion cost. The camera 6 dof pose is restricted to the volume of a half-sphere around the scene. A pose solution is found by relying on the multi-scale ICM algorithm [Besag 1986].

The algorithm of Berry first moves the camera in order to center the object to model in the image. The camera then moves at a fixed distance from the object in the lateral and vertical planes, based on the object principal axes.

In a recent work, close to the subject of this thesis, Chesi [Chesi 2009] presents a method to compute a local trajectory for a 6 dof camera located at the end-effector of a robotic arm. That work presents a general parametrization of the trajectory based on homogeneous forms. Linear Matrix Inequalities optimization is used to solve the path planning problem while satisfying several constraints: target points visibility, limits on the camera configuration space, robot joint limitations, and collision and occlusion avoidance.

We can note an inherent limitation of the Visual Servoing approach: as the methods rely on the features on the camera image to generate or update the control law, we must ensure that enough features are visible along the camera trajectory until the completion of the task. Though some systems can cope with partial occlusions, a higher level of control is required to overcome total occlusions. In this regard, a visual servoing approach to the problem of NBV is possible but requires two additional constraints: (i) set some limitations on the type of visual occlusions, and (ii) ensure the motion control of the robot can allow the continuous, and robust, tracking of the target. Though some works are being conducted to address the second constraint, the related problem is still not completely solved at the time of writing.

1.2.7 Structure from motion

It is worth mentioning a particular class of methods related to Visual Servoing: Structure from Motion (SfM) [Aggarwal 1988] [Beardsley 1996] [Beardsley 1997]. The algorithms developed in SfM aim at constructing 3D models of object by analyzing a stream of images from a moving camera, or by moving the object in front of a fixed camera. The methods thus aim at computing simultaneously the projection matrices and the 3D coordinates of some features using only the corresponding points in each view. This approach has the advantage of being cheaper than stereo systems for 3D reconstruction, as a single camera is enough. To achieve a good model though, it is necessary to detect and track reliably many features locally on the target object. There are also larger uncertainties concerning the Euclidean distance between 3D points as, contrary to a stereo rig, the pose modification between different views is not known.

We can mention a number of recent works that demonstrate impressive results. Pan *et al.* [Pan 2009] present a system to modelize a 3D object online by moving it in front of a camera. The system tracks features on the moving object to recover their geometrical relationship and create a polygonal approximation of the object. The algorithm uses a probabilistic approach to update constantly the model according to the new viewpoints that are reached.

Newcombe and Davison [Newcombe 2010] present a method for dense reconstruction of an entire scene using a single camera moved by hand. The method relies on the robust estimation of the camera 3D pose and a sparse point cloud both available through the PTAM framework [Klein 2007]. The dense 3D surface is obtained from the point cloud using Multi-Scale Compactly Supported Radial Basis Function (MSCSRBF) technique [Ohtake 2003]. The depth maps are then updated using the TV-L1 optical flow [Wedel 2008] computed from consecutive images.

A recent work by Stühmer *et al.* [Stühmer 2010] implements the same application as Newcombe and Davison but claims a faster processing time. Their method processes the live video stream by computing the depth maps directly with a variational approach.

While these recent works are of particular interest for the problem of object 3D modeling, some limitations can appear in some particular cases. The main problems to consider are (i) the occlusions of the object by obstacles in the environment, and (ii) the robustness of the method when dealing with a moving camera whose motion is perturbed. As these works were available relatively late in the course of this thesis, a more thorough analysis of the possibilities in the context of our work could not be conducted.

1.3 Contribution

Despite a number of interesting works in varied fields in the last 20 years, we can remark that the problem of the autonomous acquisition of an object 3D model by a mobile robot has not been solved yet. Most previous works in the NBV field set some strong constraints on the sensor configuration space in order to reduce the complexity of the search. But such approaches can lead to poor results if we consider

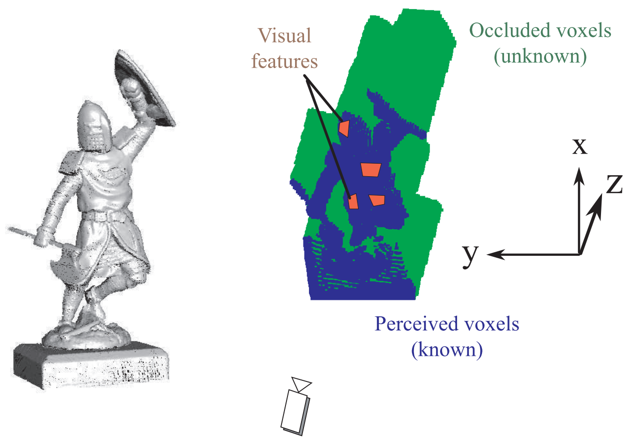


Figure 1.4: Virtual model construction. (left) Original 3D model. (right) First model constructed using stereo vision and visual features detection.

the presence of obstacles around the object. A limited configuration space may also lead to many violations of constraints related to the sensor motions.

Methods from Visual Servoing and Structure from Motion fields can be applied to estimate an object shape in specific conditions, e.g. when no obstacle is present. But the robustness of the available algorithms needs to be assessed in the context of object modeling using a sensor whose motion can be perturbed. We can also note that some related works in Visual Servoing can generate sensor motions relatively to the estimation of the target object principal axes, but none have addressed the problem of deducing the motions based on the 3D surface of the object, for example, in order to consider concavities.

1.3.1 Goal

In contrast, our approach to the object modeling problem is designed specifically based on the HRP-2 humanoid specificities in terms of constraints, embedded vision sensor and redundant motion capabilities. Assumptions about the environment and the object to model are kept minimal in order to allow a greater flexibility of the designed algorithm.

Regarding the model itself, we aim to build a rough 3D estimation of the target object surface, and, at the same time, gather a set of 2D visual features on its surface from different viewpoints, as illustrated in Fig. 5.11. The obtained model is supposed to serve different purposes in future works:

- *Collision avoidance.* Especially if the object is relatively large and/or have a complex shape where a simple bounding box approximation would include an unnecessary large amount of empty space.

- *Manipulation task.* A model providing a precise enough 3D surface, even if not completely accurate, can allow the efficient planning of various grasp-related tasks such as the grasp position on the object, the gripper pose, etc.

- *Fast detection of the object.* Given a set of 2D visual features which have particular robustness properties, e.g. SURF [Bay 2008], we can quickly process images of some unfamiliar settings in order to detect the presence and position of the model.

- *Object pose estimation.* Given an image and the visible 3D surface of an object, two methods can be used to recover the object pose based on its known model. We can compare the visible 3D surface with parts of the model surface using methods such as the Iterative Closest Point (ICP) algorithm [Chen 1991] [Besl 1992]. Or we can detect some visual features on the target object, match them with features in the model, and check the 3D coordinates of the matched features in the model.

Following most works in the NBV field for object reconstruction and environment exploration, the model 3D surface is modeled as an occupancy grid, also referred as *voxel grid* in some works.

The visual features should be robust to various modifications of the perception modalities while allowing relatively fast detection and recognition. Throughout this work, we do not set more specific conditions on the type of region detectors and feature descriptors that can be used. Thus the implementation can rely on Shi-Tomasi features [Shi 1994], SIFT [Lowe 2001], SURF [Bay 2008], MSER [Matas 2002], etc.

Regarding the modeling process, this thesis is driven toward the practical realization of a fully autonomous modeling experiment using a HRP-2 humanoid robot. The entire process requires the tight integration of a number of different but complementary software components, many of them being at the cutting edge of robotics research. A detailed presentation of the experiment is presented in section 1.3.3, after the review of our working hypotheses.

1.3.2 Hypotheses

Due to the hardware and software components which we rely on, some restrictions on the working conditions need to be set. These depends on the specificities of the HRP-2 robot as well as the current state of functionalities of some key software components developed by other researchers. The assumptions made for the efficient realization of the modeling experiment are:

1. *A wide-angle camera is available in order to perform vision-based localization.* This is necessary to have an overview of the environment in order to facilitate

the localization of the robot. When the object model is updated using images from new viewpoints, the robot pose relatively to the object pose must be as precise as possible. The wide-angle camera can also be used to detect the target object position and detect features on its surface.

2. *A depth map can be computed.* In order to build the object 3D model, we need to use a specific sensor which can provide the visible 3D surface. The sensor can be a laser scanner, IR range sensor, etc. In the case of HRP-2, we use the narrow-angle stereo cameras embedded on the side of the head. A stereo rig has the advantage that we also have the color information that can be used for visual features detection and recognition.

3. *Robot model is known.* The geometrical model of the robot is known as well as various information such as joint and torque limits, kinematics, characteristics of the embedded cameras, etc. This is necessary to compute valid and useful postures.

4. *Known environment.* We consider that a geometrical model of the environment is available so that we can compute collision-free postures and motion trajectories for the robot. Some known landmarks in the environment can also be present in order to help the robot localization process. Finally, the assumption that the environment is known is necessary to discriminate it autonomously from the target object to model.

5. *Static environment.* We do not yet consider the presence of dynamic objects in the environment. Nevertheless, in some specific conditions, it is possible to allow some changes. For example, the environment model can be modified if no visual occlusions and no obstacles on the robot trajectory appear between the moments when the robot compute a new posture and when it reaches this posture.

6. *Flat floor.* Though a number of software component can deal with uneven floor, the current pattern generator used in our experiment restricts the step sequences to a flat floor.

7. *The object to model has a known position and an approximate size.* We assume that the unknown object has been detected in some preliminary phase that is not addressed in this thesis. This detection phase outputs the location of the object in the environment model and also gives a first approximation of the object size. These assumptions are used to initialize the parameters of the occupancy grid used to model the 3D surface of the object. We note that these assumptions are not critical for our algorithm and thus it is possible to start with a specific grid size and position, and include additional occupancy grids in places where parts of the object are getting larger than what expected.

8. *The object is textured and its surface is lambertian.* Due to the characteristics of the vision system used, we rely on the object texture to model its 3D surface and detect some visual features. To reduce the influence of the lighting conditions on the appearance of the object, we require the object surface material to have a low light reflection coefficient.

1.3.3 Modeling process

The practical application that we aim in this work begins with the humanoid robot facing the unknown object to model. During our modeling process, the algorithm loops through a sequence of successive steps:

1. Get the depth map from the actual viewpoint.

2. The filtered depth map is processed with a Sobel operator to create a normal map, following a common method used in computer graphics to perform bump mapping [Hertzmann 1999].

3. Detect visual features on the perceived envelop of the object and compute their normal vector.

4. The voxels of the occupancy grid are compared with the depth map in order to realize a space carving operation and thus update the 3D model. Each voxel is set to one of the 3 states: empty, known (i.e. perceived from the camera) or unknown (i.e. occluded by known voxels or out of the field of vision), similarly to approaches such as Sanchiz and Fisher [Sanchiz 1999]. Actually, our algorithm only needs to take into account the voxels on the model surface being build, i.e. known or unknown voxels which have at least one empty neighbor voxel. Each visual features detected previously is attributed a normal vector using the normal map.

5. The resulting occupancy grid is given as an input to our NBV algorithm. This one will search for a target robot head pose considering the current model as well as some constraints related to the use of a humanoid robot.

6. The modeling task stops in two cases: the model is considered finished when (i) the prediction of unknown to be perceived falls under a desired threshold value, or (ii) this amount cannot be reduced after a pre-defined number of successive poses. In the second case, we acknowledge the fact that the model is incomplete but good viewpoints are considered to be out of reach from the robot.

7. A whole body robot pose is then generated using a Posture Generator detailed in chapter 2, section 2.1.1.

8. A collision-free step sequence is computed to move from the current position to the generated one.

9. The robot moves from the current posture to the generated one. The robot motions are executed through a sequence of tasks controlled by generalized inverse kinematics.

10. Evaluate the robot pose relatively to the object based on visual cues in the environment and the object.

11. If necessary, modify the walking motion to reach the computed pose.

12. go back to (1).

1.3.4 Scientific interest

The main originality of this work comes from the solution given to generate viewing postures for a humanoid robot considering its incremental knowledge of the environment and the object of interest.

Chapter 2 presents a first approach which formulates this visually-guided posture generation in a coherent manner by implementing an original evaluation function of the visibility of a 3D model. This specific approach relies on an existing software component in our laboratory which allows the fast generation of constrained postures for HRP-2. The algorithm details a local NBV search based on the 3D surface visibility of the object while satisfying the robot constraints.

Chapter 3 introduces another approach to the NBV problem adapted to a humanoid robot. This original algorithm is inspired by works from Sanchiz and Fisher [Sanchiz 1999], as well as Triggs and Laugier [Triggs 1995], and implements a global approach more robust to local minima in the object visibility estimation function. In this approach, our visual model is built through the repetition of two main processes: considering the current knowledge of the object, a preferred viewpoint is deduced in order to reveal occluded volumes of the object while taking into account the constraints related to the embodiment of the vision sensors in the humanoid head. Then a whole robot posture is generated using the desired head pose while satisfying several constraints: static stability, collision avoidance, etc.

Chapter 4 presents our investigation over various vision algorithms which are required for the experimental realization of the autonomous modeling application. More specifically, we detail our technical approach to model the object, as well as a number of methods to localize the robot relatively to the object.

Finally, chapter 5 details the complete practical realization of the autonomous reconstruction of an object. In particular, we present the tight integration of our NBV solution with other recent works relative to path planning and control in order to execute the modeling task using an HRP-2 robot.

Chapter 2

$\mathcal{C}^1$ formulation of an object visibility for Next-Best-View selection

Contents

We describe here a first algorithm aiming at generating a realistic pose for HRP-2 which maximizes a vision-based criterion. In this chapter, we focus on a local resolution scheme. The solver that we use works efficiently with $\mathcal{C}^1$ functions. Therefore we propose stability constraints and a visual criterion for which an analytic gradient can be derived.

2.1 Modeling constraints

This section introduces the posture generation scheme developed upon the work of Adrien Escande. The constraints taken into account in our problem are presented.

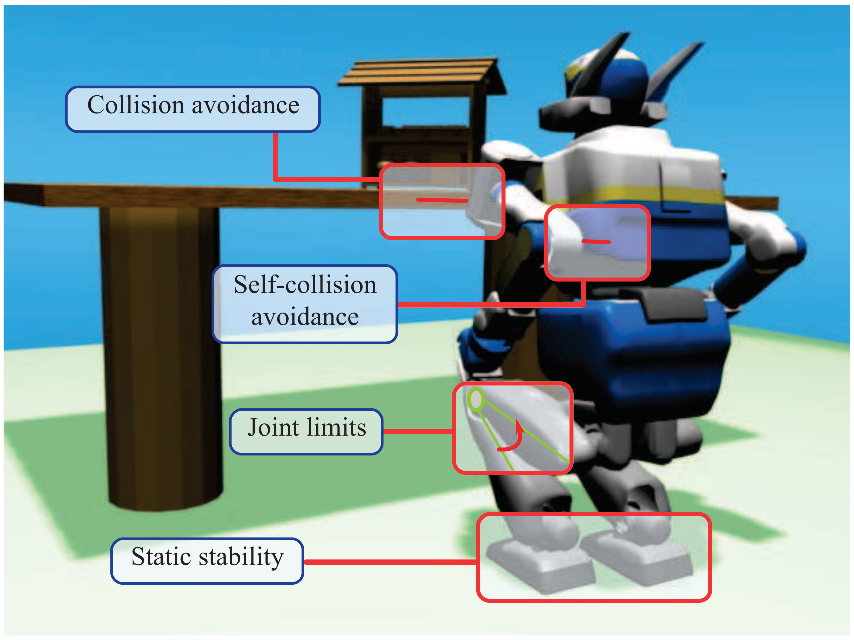


Figure 2.1: Usual constraints to obtain a posture for a humanoid robot: avoid collisions with the environment, avoid self-collisions, respect joints limits and ensure stability.

Finally, we review the possible ways to deal with the evaluation of the visibility of unknown parts of the object. This evaluation is particularly critical for the appropriate selection of viewpoints.

2.1.1 Posture generation

Our Next-Best-View (NBV) selection procedure specificity consists in taking into account the constraints on the positionning of the humanoid where the vision sensors are embedded. The 6 degrees of freedom of the cameras are limited due to the capabilities of the body in many aspects, even if we do not consider the vision constraints. For example, the camera cannot be placed at a higher height than the robot size and it cannot be placed at the same position as an obstacle in the environment. The constraints we must take into account when looking for a valid static posture for the humanoid robot are illustrated in Fig. 5.12. They consist in: staying within joints limits, avoiding collisions and ensuring stability. As such constraints have a great impact on possible solutions for a viewpoint selection, we are interested in designing a NBV solution tightly coupled with the posture generation process. Our NBV problem can then be expressed as the minimization of a criterion related to the visual data while satisfying some constraints on the robot body.

In this work, the computation of a posture is realized by taking advantage of the Posture Generator (PG) proposed in [Escande 2006] and [Stasse 2007]. The PG addresses the problem of generating a posture by solving an optimization problem:

$$\min_{\mathbf{q} \in X} f(\mathbf{q}) \tag{2.1}$$

where f is the objective function to be minimized which will be described in paragraph 2.2.1. X represents the set of constraints on the robot body illustrated in Fig. 5.12. $\mathbf{q}$ is a set of parameters defining the posture:

$$\mathbf{q} = [r \ w \ \Theta]^\top \tag{2.2}$$

with r the position of the free-floating body, w its orientation, and Θ a vector containing all the robot joints angle.

In order to solve this optimization problem, the PG uses the library FSQP [Lawrence 1997], standing for Feasible Sequential Quadratic Programming. This library provides a gradient-based optimization method. The PG is also composed of various functions to facilitate the optimization of postures for an HRP-2 humanoid robot. FSQP can cope with linear and nonlinear constraints formulated as equalities or inequalities. First it tries to satisfy the linear constraints and the non-linear inequalities constraints, and then tries to solve the non-linear equalities constraints while minimizing the given objective function. The proof of convergence of the algorithm relies on the assumption that the objective function and the constraints are formulated as $\mathcal{C}^\infty$ functions [Panier 1987]. However, in practice, $\mathcal{C}^1$ formulations are enough for our applications [Escande 2008]. The optimization process is initialized by using a starting posture and an initial free-flyer position and orientation. In this work, the default posture is set as a squatting one as it is easier to reach more challenging joint configurations from this particular posture.

The constraints to take into account in the set X are formulated as:

$$\begin{cases} \Theta_{\min} \leq \Theta \leq \Theta_{\max} & (2.3) \\ d\mathcal{C}_{min} \leq d(B_i(\mathbf{q}), B_j(\mathbf{q})) \ \forall (i,j) \in \mathcal{C} & (2.4) \\ F_l^z(\mathbf{q}) = F_r^z(\mathbf{q}) = 0 & (2.5) \\ d(\mathbf{pc}, F_{seg}) \leq \xi_s & (2.6) \end{cases}$$

In the first constraint (2.3), $\Theta_{\min}$ and $\Theta_{\max}$ are two vectors representing each joint limits.

The second constraint (2.4) ensures the avoidance of auto-collision as well as collisions with objects in the environment. $d(B_i(\mathbf{q}), B_j(\mathbf{q}))$ is the $\mathcal{C}^1$ distance between two bodies approximated by a specific type of bounding volume introduced by Escande *et al.* [Escande 2007]. This distance must be greater than a precision value $d\mathcal{C}_{min}$. $\mathcal{C}$ is the set of collision pairs which are tracked to avoid non desirable collisions and auto-collisions. These bounding volumes were designed specifically in order to be included in the PG as it allows a $\mathcal{C}^1$ formulation of the collision avoidance constraint.

The constraint (2.5) imposes to the feet to be on the ground as a first requiremeent to be stable. This is done by ensuring that the height values of the left foot F_l and right foot F_r are set to 0.

The static stability of the posture is further enforced by the last constraint (2.6) where $\mathbf{pc} = [c_x \; c_y]\top$ is the projection of the CoM on the floor, F_{seg} is the segment between the feet centers and ξ_s is a value small enough so that the CoM cannot be projected outside of the support polygon of the robot. We can remark that, with this formulation, the equilibrium is ensured only if the humanoid evolves on top of a horizontal plane. This assumption is valid in this work as the generated postures are intermediate points between walking periods, and the pattern generator used for walking deals only with motions on a horizontal floor.

The constraint (2.6) has been developed for the purpose of this work and is presented in more details in the following section.

2.1.2 Equilibrium constraint

The robot is required to be statically stable while taking pictures of the object. It could be possible to build the object while walking. However, when walking, induced motion might result in a blurred image. This happens especially during the landing of the foot; resulting impact propagation creates oscillations at terminal points such as the head, where the cameras are located.

In some previous works [Stasse 2007], the requirement for stability is ensured by three constraints and a criterion:

- one constraint sets the feet height to 0, similarly to the constraint (2.5) seen earlier.
- another constraint sets the relative position of the robot feet.
- the third constraint forces the projection of the CoM to be in the robot feet support polygon. This polygon has a fixed shape due to the previous constraint.
- the criterion minimizes the distance between the CoM projection and the barycenter of the support polygon.

However this approach has two limitations: (i) the poses of the feet relatively to each other cannot be modified, and (ii) a margin between the CoM projection and the edges of the support polygon is necessary in the constraint implementation. In fact, practically, if $\mathbf{pc}$ is close to the limits of the convex hull of the feet, the robot can be in an unstable position due to the flexibility in its ankles.

The new approach developed is to set the robot stability as a constraint where the distance from $\mathbf{pc}$ to the segment between both feet must be less than a fixed value ξ_s, typically 2cm for HRP-2, as illustrated in Fig. 2.2. This can appear more

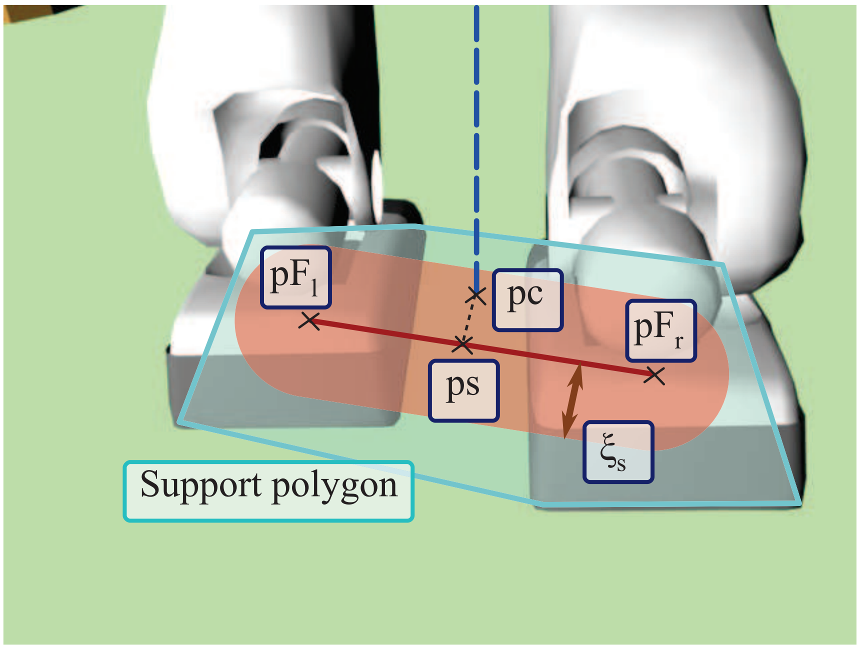


Figure 2.2: New stability criterion: set a maximum distance between the CoM projection on the floor and the segment between the two feet.

restrictive than the previous approach as **pc** is constrained to a space much smaller than what is practically safe, considering a specific feet configuration. But actually, as the feet relative pose can be freely modified by the solver in the new approach, a greater number of solutions can be achieved. There are also four more advantages: (i) we can still express the constraint as a $\mathcal{C}^1$ function in order to use the PG, (ii) a dedicated criterion is not required, (iii) we are sure that the posture generated is stable practically, and (iv) we do not rely on manual inputs from the user to set the feet relative position.

2.1.2.1 Mathematical formulation

The formulation of the equilibrium constraint can be expressed in a 2D coordinate system as we consider only standing postures on a horizontal floor.

First, the distance between the CoM projection, **pc** and the segment between the robot left foot center $\mathbf{pF}_l = [F_l^x \; F_l^y]^\top$ and right foot center $\mathbf{pF}_r = [F_r^x \; F_r^y]^\top$ needs to be computed. This distance is noted $g(\mathbf{q})$ and thus the constraint to comply with is: $g(\mathbf{q}) \leq \xi_s$, with ξ_s a pre-defined threshold. When a specific robot pose results in a $g(\mathbf{q})$ distance small enough so that the CoM projection stays inside the support polygon, including a margin of error, then the robot stability constraint is solved.

The computation of $g(\mathbf{q})$ depends on the relative position of the three points **pc**, $\mathbf{pF}_l$ and $\mathbf{pF}_r$. Three distinct cases are possible: the closest geometric object to **pc** is (i) $\mathbf{pF}_l$, (ii) $\mathbf{pF}_r$ or (iii) the segment between $\mathbf{pF}_l$ and $\mathbf{pF}_r$. For a given posture, the case encountered, i.e. the formula to use for the distance computation, can be found by analyzing the point $\mathbf{ps}$, the projection of $\mathbf{pc}$ on the segment F_{seg}. The value of $\mathbf{ps}$ can be computed by relying on α_p, the length of the segment between $\mathbf{pF}_l$ and $\mathbf{ps}$, and thus using the following two formulas:

$$\mathbf{ps} = \mathbf{pF}_l + \alpha_p \left(\mathbf{pF}_r - \mathbf{pF}_l\right) \tag{2.7}$$

$$\left(\mathbf{pc} - \mathbf{ps}\right) \cdot \left(\mathbf{pF}_r - \mathbf{pF}_l\right) = 0 \tag{2.8}$$

By replacing $\mathbf{ps}$ in (2.8) by its expression (2.7) we can find the value of α_p

$$\alpha_p = \frac{\left(\mathbf{pc} - \mathbf{pF}_l\right) \cdot \left(\mathbf{pF}_r - \mathbf{pF}_l\right)}{\parallel pF_r - pF_l \parallel} \tag{2.9}$$

The value of α_p defines the closest geometric object cg to $\mathbf{pc}$ in the three possible cases:

1. if $\alpha_p \leq 0 \Rightarrow cg(\mathbf{q}) = \mathbf{pF}_l(\mathbf{q})$
2. if $\alpha_p \geq 1 \Rightarrow cg(\mathbf{q}) = \mathbf{pF}_r(\mathbf{q})$
3. if $0 \leq \alpha_p \leq 1 \Rightarrow cg(\mathbf{q}) = \mathbf{ps}(\mathbf{q})$

In our work, the stability constraint is expressed as the square distance between cg and $\mathbf{pc}$:

$$g(\mathbf{q}) = \left(\mathbf{pc}(\mathbf{q}) - cg(\mathbf{q})\right)^\top \left(\mathbf{pc}(\mathbf{q}) - cg(\mathbf{q})\right) \tag{2.10}$$

The Euclidean distance is not computed in order to avoid a costly computation of a square root which is not needed; it is only necessary to check that $g(\mathbf{q})$ is below a defined threshold slightly smaller than the square of half the width of the robot foot to ensure stability.

2.1.2.2 Gradient formulation

In order to generate a pose which satisfies our stability constraint, FSQP relies on the partial derivatives formulation of the constraint. Three formulations are possible, depending on the value of α_p. For simplicity, let us write $\dot{g}(\mathbf{q}) = \partial g(\mathbf{q})/\partial \mathbf{q}$.

From (2.10), the gradient is formulated as:

$$\dot{g}(\mathbf{q}) = 2\mathbf{o}(\mathbf{q})\dot{\mathbf{o}}(\mathbf{q}) \tag{2.11}$$

with

$$\mathbf{o}(\mathbf{q}) = \mathbf{pc}(\mathbf{q}) - cg(\mathbf{q}) \tag{2.12}$$

and

$$\begin{cases} \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{pF}}_l(\mathbf{q}) \quad \text{if } \alpha_p \leq 0 & (2.13) \\ \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{pF}}_r(\mathbf{q}) \quad \text{if } \alpha_p \geq 1 & (2.14) \\ \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{ps}}(\mathbf{q}) \quad \text{if } 1 \geq \alpha_p \geq 0 & (2.15) \end{cases}$$

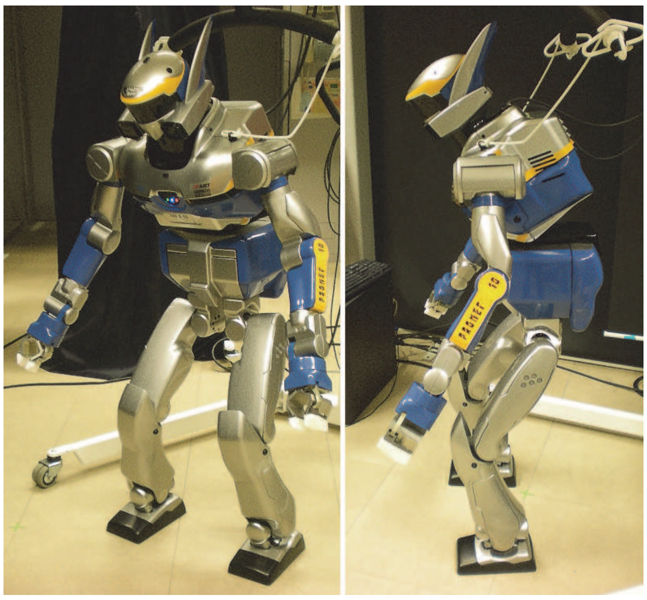


Figure 2.3: Example of stable poses generated with the new stability constraint.

By solving these three formulas, we can verify the gradient continuity by ensuring that (2.13) and (2.15) are equivalent to the same expression when $\alpha_p = 0$, and that the same is true for (2.14) and (2.15) when $\alpha_p = 1$.

2.1.2.3 Tests

The stability function was tested using a separate simplified problem. A robot pose is generated taking into account the following constraints: collisions, joint limits, feet on the ground, and the constraint that the robot camera must be looking at a specified point using a specified view direction vector. By modifying the target point and view direction vector, we checked that the different poses obtained have the projection of the CoM on the floor lying close enough to the segment between the feet contact point.

Simulations under various conditions validated the new constraint formulation and some postures were verified with a real HRP-2 robot. Examples of poses generated with the real robot are shown in Fig. 2.3.

2.1.3 Unknown representation

Many Next-Best-View algorithms rely on an approximation of position and quantity of the object unknown parts to compute a solution. For the objective function in the PG, we want to maximize the area of unknown voxels that will be visible from the next robot pose in order to reduce the number of required viewpoints to complete the object model. Such a function formulation depends directly on the representation of the object.

In some works ([Pito 1999], [Li 2005]), the envelop of the object is represented as an aggregation of planar surfaces, usually triangles. This is the usual rendering method for displaying 3D objects which is well supported by available libraries and hardware. The problem though is that it is not trivial to get a $\mathcal{C}^1$ formulation using planar surfaces. Thus such a representation cannot be integrated in the PG directly.

Others works ([Whaite 1997b], [Dune 2008]) have used a parameterized surface, such as quadrics and superquadrics, to caracterize the object surface. A proper formulation for the PG is then possible but we are limited to objects with simple shapes in order to have a meaningful approximation. Indeed a problem arises if we want to deal with complex objects which present concavities; we must then use many parameterized surfaces and have to deal with many possible self-occlusions. In such cases, the quantification of the visible area using a $\mathcal{C}^1$ formulation might not be possible.

Further analysis of these two kinds of representation may lead to exploitable solutions that can be integrated in the PG. Nevertheless, in this thesis, we focus on representations directly related to occupancy grids. This choice is motivated by a concern to ensure continuity with previous works in the laboratory [Saidi 2007b] [Saidi 2007a] [Stasse 2007] dealing with occupancy grids in the contexts of environment modeling and visual search. Three rendering methods have been considered: 3D cubes, 2D polygonal projections of cubes on the camera plane and 2D Gaussians on the camera plane.

2.1.3.1 Cubes

The first considered representation is the occupancy grid which has already been used by previous authors ([Connolly 1985], [Banta 1995], [Banta 2000]). The occupancy grid, as illustrated in Fig. 2.4, is a discretization of the part of the 3D space where the object of interest is contained. The object is then approximated by a set of cubes which can have different properties such as color, normal vector, empty/filled status, etc. Depending on the object shape, the number of voxels needed to represent it can be reduced by using octree structures [Connolly 1985]. We can note that this approach is adapted to commonly available libraries such as OpenGL in order to take advantage of graphic hardware to speed up rendering computation.

However the main problem with such a representation is that the object surface is represented by planar surfaces and thus formulating a $\mathcal{C}^1$ evaluation of the visibility of a specific area is difficult; the area of unknown voxels visible corresponds to a sum of related pixels in the framebuffer of the rendered object, which is a non-continuous quantity.

Though cubes cannot be used directly in the PG, we can still rely on such a representation to compute an auxiliary Next-Best-View solution which relies on a different

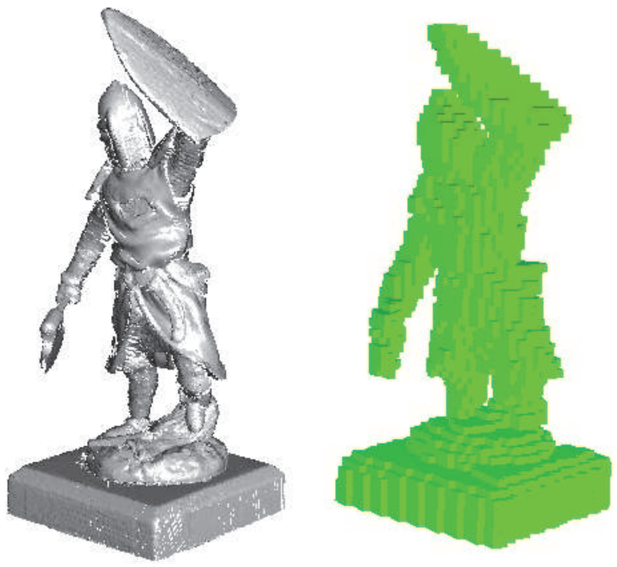


Figure 2.4: Using an occupancy grid to represent a 3D object.

optimization process; then the result can be integrated with other constraints in the PG in order to get the final posture. This approach has been developed during this thesis and is described in chapter 3.

2.1.3.2 Polygons

In a different approach, also relying on the occupancy grid but trying to tackle the continuity problem, voxels are still represented by cubes but we consider mainly the projections of the voxels faces on the camera image plane. Each voxel projection can then be represented by a polygon whose area can be computed analytically from the voxel vertices 3D coordinates. It is also possible to use an octree structure to represent the original object in order to speed up the computation. Using such a formulation, the amount of unknown visible is equal to the sum of visible polygons surface of unknown voxels. To compute the total area, we rely on an algorithmic process which runs as detailed in Alg. 1. The intersection of polygons can be computed using the General Polygon Clipper (GPC) library [Vatti 1992]. The area of a non-intersecting polygon, defined by N 2D points (X, Y), can be computed using the following formula:

$$A_j = \sum_{k=0}^{N-1} X_k Y_{k+1} - X_{k+1} Y_k \quad \text{with} \quad (X_N, Y_N) = (X_0, Y_0) \tag{2.16}$$

Though the implementation of the visibility evaluation is straightforward, a formulation that relies on polygons rendering has 3 disadvantages:

Algorithm 1 Unknown visible area computation based on polygons

```
1: $area \leftarrow 0$
2: for each voxel do
3:     project voxel corners on camera plane
4:     consider polygon formed by convex hull of these projections
5:     if polygon is outside of camera image then
6:         remove polygon
7:     if polygon intersects with image boundaries then
8:         truncate polygon
9: Sort polygons from nearest to farthest from camera
10: for each polygon $j$ do
11:     if polygon is related to known voxel then
12:         check next polygon
13:     find polygons which occlude $j$
14:     update $j$ by removing occluded parts
15:     compute $A_j$, area of $j$
16:     $area \leftarrow area + A_j$
17: return $area$
```

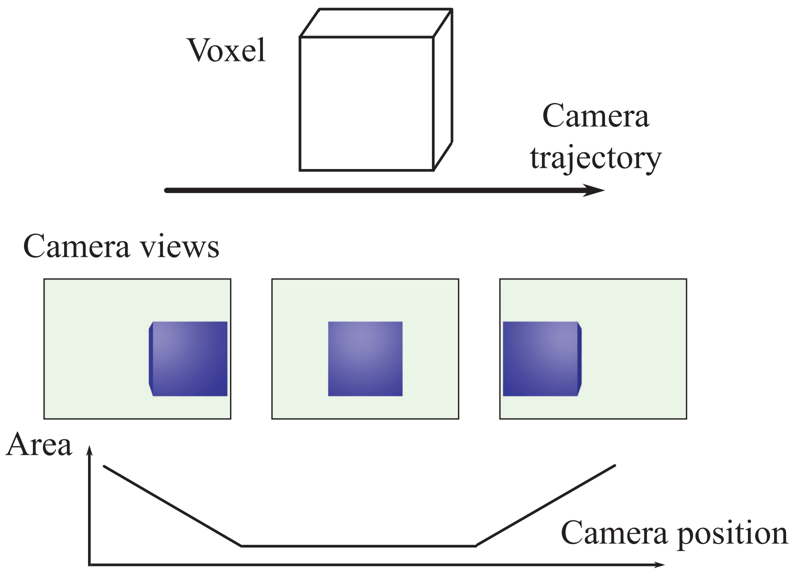


Figure 2.5: Area of a voxel visible depending on camera trajectory.

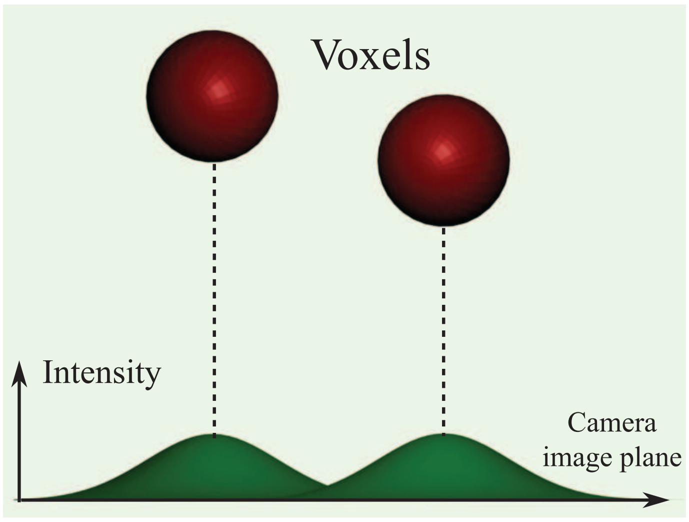


Figure 2.6: Principle of voxel rendering using Gaussians.

- The computation time is much bigger than rendering pixels using dedicated hardware as it is necessary to compute the polygonal projection of each cube face and then test for occlusions.
- The final formulation of the unknown area surface is changed depending on events such as the appearance or disappearance of self-occlusions. Thus an approximation of its gradient is difficult to formulate.
- As illustrated in the simple example in Fig. 2.5, the gradient is not continuous everywhere. The visible area varies linearly but the linear equation of the variation changes abruptly each time a cube face appears or disappears.

For these reasons, the quantification of unknown using polygons has not been further investigated during this thesis.

2.1.3.3 Gaussians

In the Gaussian approach, we consider the projection of voxels on the image plane as 2D Gaussians. Our new function, developed during this thesis, is inspired by the splatting algorithm [Westover 1989] where a voxel projection on the image plane is represented by a pre-defined kernel. Westover's algorithm aims at rendering 3D scenes by aggregating on the framebuffer depth-sorted 2D functions representing the projection of voxels.

In our case, the voxel is considered as a fuzzy sphere and thus the kernel is a 2D Gaussian function, as illustrated in Fig. 2.6 for the 1D case. With such an assumption, it is possible to obtain a function which is at least of class $\mathcal{C}^1$ and which can be used as a criterion to minimize in the PG. The next section introduces our original formulation based on Gaussians.

2.2 Next-Best-View local selection relying on the gradient

This section introduces a first solution to the NBV problem which can be used inside the PG in order to ensure that the resulting viewpoint satisfies the constraints on the humanoid body. This solution relies on a Gaussian representation of the visual data in order to get the C^1 formulation required by the PG.

2.2.1 C^1 objective function for the Posture Generator

In the present formulation, a voxel is considered as a sphere and its influence on any pixel (x, y) in the resulting image is expressed as a 2D Gaussian function:

$$G_i(\mathbf{q}) = \exp\left(-0.5\left(\frac{(x - X_i(\mathbf{q}))^2}{\sigma_i(\mathbf{q})^2} + \frac{(y - Y_i(\mathbf{q}))^2}{\sigma_i(\mathbf{q})^2}\right)\right) \tag{2.17}$$

$(X_i(\mathbf{q}), Y_i(\mathbf{q}))$ are the coordinates of the perspective projection of the voxel i center $\mathbf{V}_i$ on the camera image plane with respect to the current state vector of the robot $\mathbf{q}$. They are computed relatively to the camera focal length f, its position $\mathbf{C}(\mathbf{q})$ and its orthonormal basis vectors ($\mathbf{e_i}$, $\mathbf{e_j}$, $\mathbf{e_k}$):

$$Z_i(\mathbf{q}) = (\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_k} \tag{2.18}$$

$$X_i(\mathbf{q}) = f\,\frac{(\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_i}}{Z_i(\mathbf{q})} \tag{2.19}$$

$$Y_i(\mathbf{q}) = f\,\frac{(\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_j}}{Z_i(\mathbf{q})} \tag{2.20}$$

$\sigma_i(\mathbf{q})$ defines the Gaussian dimension and is directly related to the fixed size of the voxels, noted σ:

$$\sigma_i(\mathbf{q}) = f\,\frac{\sigma}{Z_i(\mathbf{q})} \tag{2.21}$$

The σ constant is defined proportionally to the size of the world subspace discretized by the occupancy grid and the grid resolution.

In order to measure the visibility of unknown voxels, we need to distinguish them from known ones in our formulation and occlusions must be taken into account. The first issue is simply solved by setting a parameter S_i to each voxel based on their status:

1. if $\mathbf{V}_i$ is unknown $\Rightarrow S_i = 1$

2. if $\mathbf{V}_i$ is known $\Rightarrow S_i = -1$
3. if $\mathbf{V}_i$ is empty $\Rightarrow S_i = 0$

To help dealing with occlusions, a weight is defined for each voxel depending on their distance to the robot camera. This weight increases when the voxel is closer to the camera:

$$D_i(\mathbf{q}) = \exp\left(-\sigma_d \left(\frac{Z_i(\mathbf{q}) - Z_{min}}{Z_{max} - Z_{min}}\right)^2\right) \tag{2.22}$$

The weight value depends on three values arbitrarily set: σ_d, $Z_{\min}$ and $Z_{\max}$.

The σ_d parameter influences the discrimination based on distance. A small value results in a poor discrimination while a large value excessively reduces the influence of voxels relatively far away. Thus we typically set $\sigma_d = 20$.

$Z_{\min}$ and $Z_{\max}$ are parameters delimiting the maximum and minimum distance between the camera and the object voxels. These parameters help to influence further the distance-based discrimination but must be coherent with the allowed robot movement space and the relative object position and size.

We can remark that some theoretical limits are set depending on the sensor specificities. In the case of a stereo rig, the limits depend on the intersection of the space that is visible by the two cameras. Practically, though, it is preferable to have a reduced interval by setting a smaller value of $Z_{\max}$ in order to have a better discrimination.

Finally another coefficient is added to enhance the voxel occlusions handling by using the voxel normal vector $\mathbf{n}_i$. We consider that if the normal vector of a voxel is pointing away from the camera, then it belongs to a part of the object that is not visible. The dedicated coefficient is then formulated as follows:

$$N_i(\mathbf{q}) = \exp\left(-0.5 \left(\frac{\mathbf{n}_i \cdot -\mathbf{e_k} - 1}{\sigma_n}\right)^2\right) \tag{2.23}$$

The σ_n parameter is chosen so that angles greater than 90 degrees between $\mathbf{n}_i$ and $-\mathbf{e_k}$ are close to 0, e.g. 0.4.

For each pixel in the camera image, we set together these coefficients:

$$P_{x,y}(\mathbf{q}) = \sum_{i=0}^{N} S_i\, G_i(\mathbf{q})\, D_i(\mathbf{q})\, N_i(\mathbf{q}) \tag{2.24}$$

The sign of $P_{x,y}$ is supposed to depend on the status of the closest visible voxel projecting at x, y. In some cases where one voxel with a specific status occludes many neighboring voxels of the other status, the sign of $P_{x,y}$ may not reflect the real occlusion. To minimize this problem, only voxels on the perceived envelope of the object are considered. This also has the advantage of speeding up the computation.

By thresholding $P_{x,y}$, we can find the pixel contribution on the total area of unknown that is currently visible. The continuous threshold function used is a sigmoid defined as:

$$T(\tau) = (1 + \exp(-\alpha\,\tau))^{-1} \tag{2.25}$$

The α parameter influences the slope of the sigmoid. In our case, a large value is required to be discriminant enough but not too much so that discontinuities introduced by number coding precision can be avoided. Typically we use $\alpha = 10^7$. Using this function, negative values of $P_{x,y}$ are set close to 0, i.e. voxels occluded by a known one are not counted in the total area sum.

The total area is then expressed as:

$$A_{tot}(\mathbf{q}) = \sum_{x=0}^{W}\sum_{y=0}^{H} T\left(P_{x,y}(\mathbf{q}) - \varepsilon\right) \tag{2.26}$$

W is the image width and H its height. Due to the use of Gaussian functions, $P_{x,y}(\mathbf{q})$ can result in a small positive value in the image parts where no voxels are projected, thus an arbitrarily defined ε term is used to set such values close to 0 through the threshold function. Typically we set $\varepsilon = 10^{-4}$.

The projection process expressed by our formulation is illustrated in Fig. 2.7 for a simple 2D case.

2.2.2 Gradient formulation

In order to find the robot pose which will lead to the best visibility of the unknown parts of the object considering all other constraints, an objective function to minimize is required. The PG seeks an optimal value by using the objective function gradient. In our case, $A_{tot}(\mathbf{q})$ is used with a negative sign so that the minimum values relate to the biggest amounts of unknown area visible. The gradient is then expressed as:

$$-\dot{A}_{tot}(\mathbf{q}) = -\frac{\partial}{\partial \mathbf{q}} \sum_{x=0}^{W}\sum_{y=0}^{H} T\left(P_{x,y}(\mathbf{q}) - \varepsilon\right) \tag{2.27}$$

A common multiplier to all partial derivatives can be found by developing the equation above:

$$-\dot{A}_{tot}(\mathbf{q}) = -\sum_{x=0}^{W}\sum_{y=0}^{H} \frac{e^{-\alpha\,(P_{x,y}(\mathbf{q})-\varepsilon)}}{\left(1 + e^{-\alpha\,(P_{x,y}(\mathbf{q})-\varepsilon)}\right)^2}\,\alpha\,\dot{P}_{x,y}(\mathbf{q}) \tag{2.28}$$

By developing the derivative of $P_{x,y}(\mathbf{q})$, we obtain:

$$\dot{P}_{x,y}(\mathbf{q}) = -\sum_{i=0}^{N} S_i\, G_i(\mathbf{q})\, D_i(\mathbf{q})\, N_i(\mathbf{q})\, \dot{\Psi}(\mathbf{q}) \tag{2.29}$$

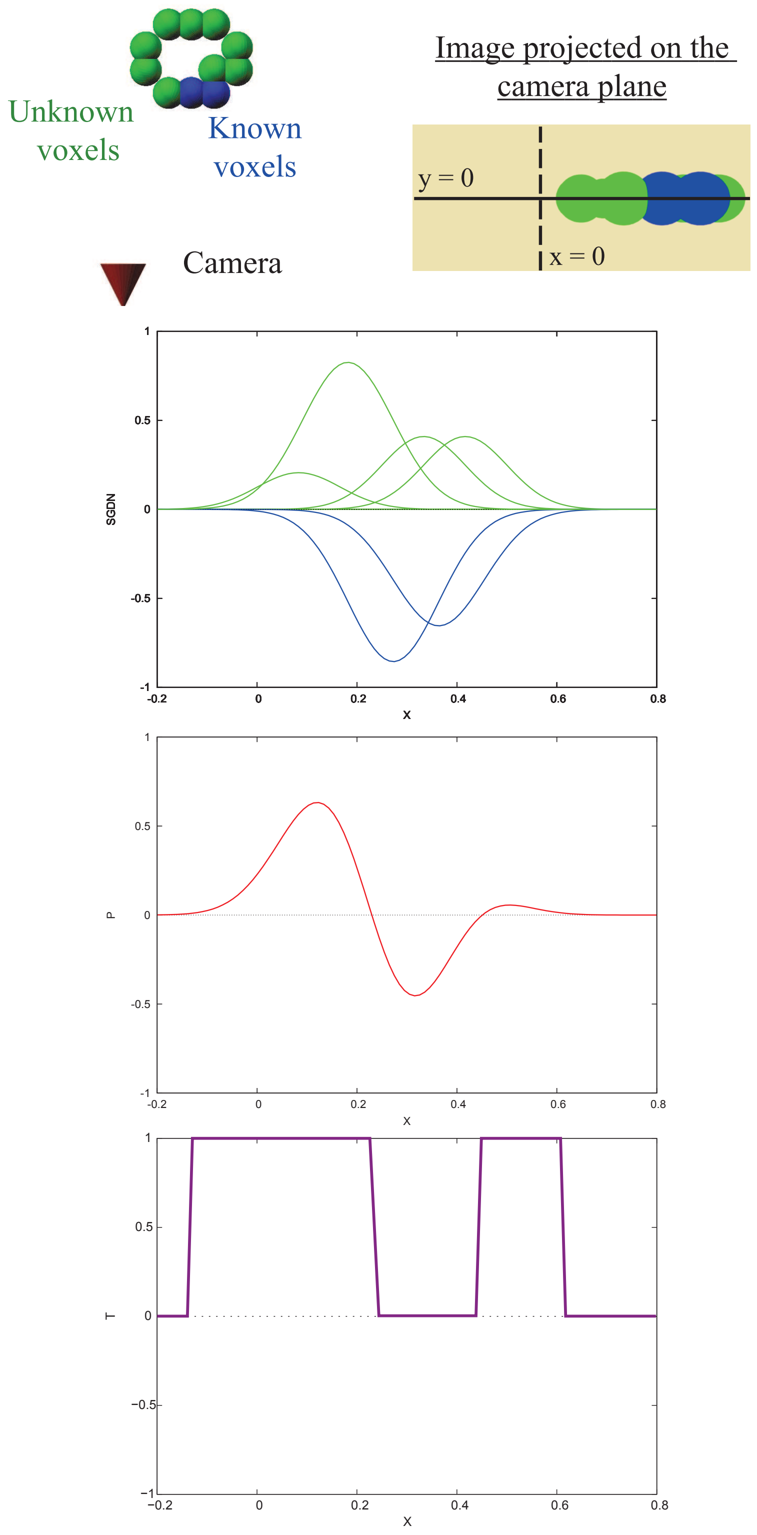


Figure 2.7: Illustration of the voxels projection process when using a Gaussian representation. The curves are computed using the setup shown in the upper part and considering a constant y=0.

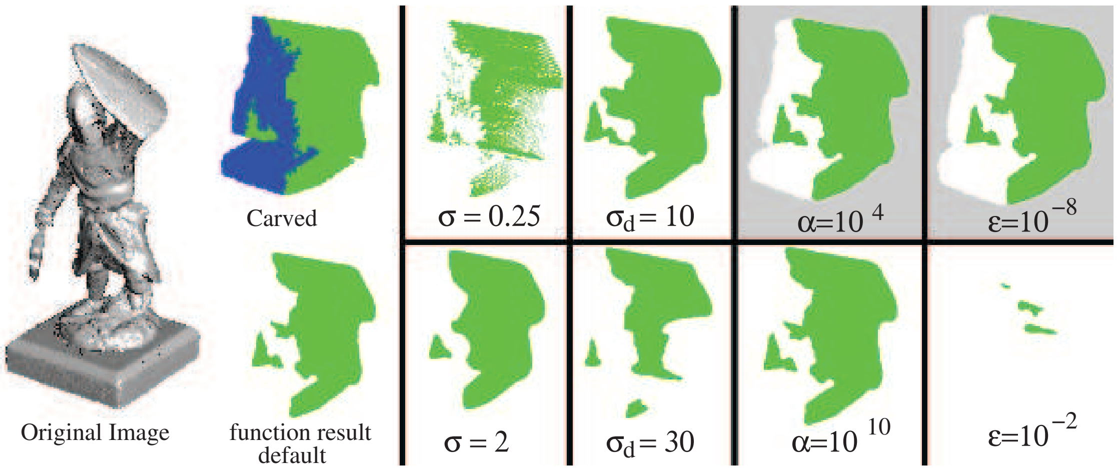


Figure 2.8: Influence of the $\mathcal{C}^1$ function parameters on the unknown area visibility estimation.

with

$$\begin{aligned}\Psi(\mathbf{q}) =& \frac{(x - X_i(\mathbf{q}))^2 + (y - Y_i(\mathbf{q}))^2}{2\,\sigma_i(\mathbf{q})^2} \\ &+ \frac{\sigma_d\,(Z_i(\mathbf{q}) - Z_{\text{min}})^2}{(Z_{\text{max}} - Z_{\text{min}})^2} + \frac{(\mathbf{n}_i \cdot -\mathbf{e_k} - 1)^2}{2\,\sigma_n^2}\end{aligned} \tag{2.30}$$

As the function depends only on the robot head position and orientation, we can first compute the partial derivatives of the function relatively to the robot head pose $\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{C}}$, $\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_i}}$, $\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_j}}$ and $\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_k}}$. The partial derivative relatively to the robot pose can then be expressed as:

$$\dot{\Psi}(\mathbf{q}) = \left[\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{C}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_i}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_j}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_k}}\right]\left[\dot{\mathbf{C}}\ \dot{\mathbf{e_i}}\ \dot{\mathbf{e_j}}\ \dot{\mathbf{e_k}}\right]^T \tag{2.31}$$

The gradient continuity was verified by developing all partial derivatives.

2.3 Tests of the objective function

This section presents various analyses of our formulation $A_{tot}(\mathbf{q})$ to assess its validity, verify its properties and test its limits. This is done by conducting different experiments in simulation using various objects, mixing simple and complex shapes.

2.3.1 Function parameters

The unknown quantification function contains various parameters that need to be set manually. As the result of the function is a 2D image directly related to the object image perceived by the camera, these parameters can be tuned experimentally from the obtained images. Fig. 2.8 presents some samples of images computed depending

	3916 voxels	15268 voxels	65726 voxels
200×200 pixels	11s	38s	142s
300×300 pixels	25s	86s	314s

Table 2.1: Computation time table for the function depending on the sampling size and the number of voxels.

on different parameters values. On the left of the figure are the original image, the carved image rendered using OpenGL with known voxels in blue and unknown voxels in green, and the result of our function using the following default parameters: $\sigma = 1.0$, $\sigma_d = 20$, $\alpha = 10^7$ and $\varepsilon = 10^{-4}$. Quantitative results for $\sigma_d < 15$ show that occlusions by known voxels are not correctly rendered when a relatively large amount of unknown voxels is behind. On the other hand, distant, but not occluded, voxels may not appear with a large σ_d value. Small values for α and ε create a background noise, rendered in light gray in the figure, as the values of the function $P_{x,y}(\mathbf{q})$ get close to 0, resulting then in $T(P_{x,y}(\mathbf{q})-\varepsilon) = 0.5$. Large values for ε make the function too restrictive. Setting α to a large value gives a correct rendering but the function reacts as a discrete function with higher variations of the gradient.

2.3.2 Computation time

The complexity of our formulation depends on the sampling of the camera image and the number of voxels. It can thus be expressed as $\Theta(n^3)$. Practically, for an accurate result, the process involves a high number of samples and voxels. Some examples of computation time to evaluate the unknown area are presented in table 2.1 for 3 objects with different number of voxels and 2 different image sizes. Tests were performed with a C implementation of the algorithm on an Intel Xeon 3.2GHz processor with 1GB of RAM. Some basic techniques were applied to reduce the computation time: only pixels of the image which are close to the projection of the voxel center are considered, and a parallel implementation of the algorithm was realized with 2 threads.

2.3.3 Comparison with OpenGL rendering

To ensure that the function optima are linked to the same camera poses as those of a traditional rendering method, we implemented a point-based rendering with OpenGL where voxels are displayed on the screen as points with a fixed size. Though it is not an accurate method, as the size and shape of voxels does not change depending on the distance and orientation of the camera, the approximation is still good enough to result in a satisfying rendering of an object. An example of

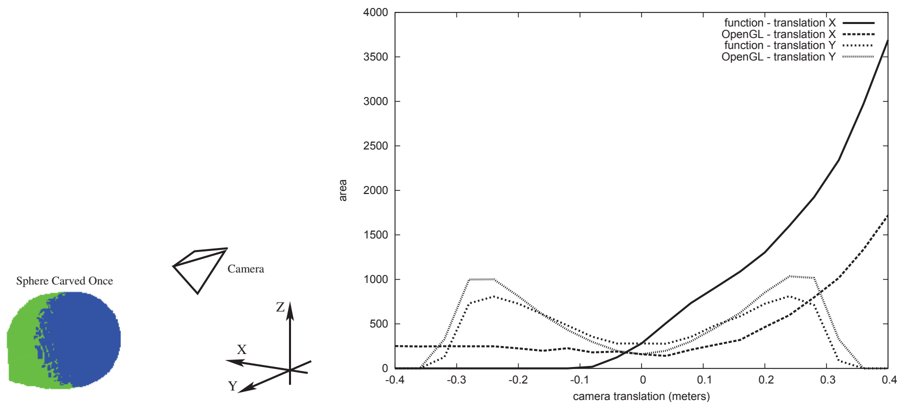


Figure 2.9: *Left*: setup to test the function variations relatively to the camera movement around a sphere. Known voxels are represented in blue and unknown ones in green. *Right*: comparison of the amount of unknown area visible depending on camera position for our evaluation method and a basic voxel rendering method.

test setup is illustrated in the left part of Fig. 2.9 where the camera is translated around a sphere which has been carved once. Known voxels are represented in blue whereas unknown voxels are in green. The camera location and orientation is represented as the black coordinate system. The camera is then moved on the X or Y axis. The corresponding results, obtained with the default parameters for the function, are shown in the right part of Fig. 2.9. Though the resulting values are not equal between the 2 methods, the overall variations of the 2 curves match and both methods detect the optima in the same positions. This confirms that our function gives a consistent approximation of the unknown visible area.

2.3.4 Unknown representation comparison

We compared the rendering results for different representations of the 3D data.

The polygon representation is compared with the $\mathcal{C}^1$ function and a simple point-based rendering of the voxels using OpenGL. We also included a sampled version of the polygon approach, where polygons are displayed on the camera image and the unknown area that is visible is quantified as the number of corresponding pixels displayed. Evaluation results with the 4 methods are displayed for a small translation of the camera in front of a single unknown voxel in Fig. 2.10.

For our $\mathcal{C}^1$ function and the OpenGL method, the evaluation should be constant, as the distance between the camera plane and the voxel does not change. Nevertheless oscillations appear mostly due to appearance and disappearance of pixels, highlighting the problem with pixel-based rendering approaches.

The polygon approach gives results consistent with the expectations. As this formulation does not rely on a threshold function nor any sampling, problems due

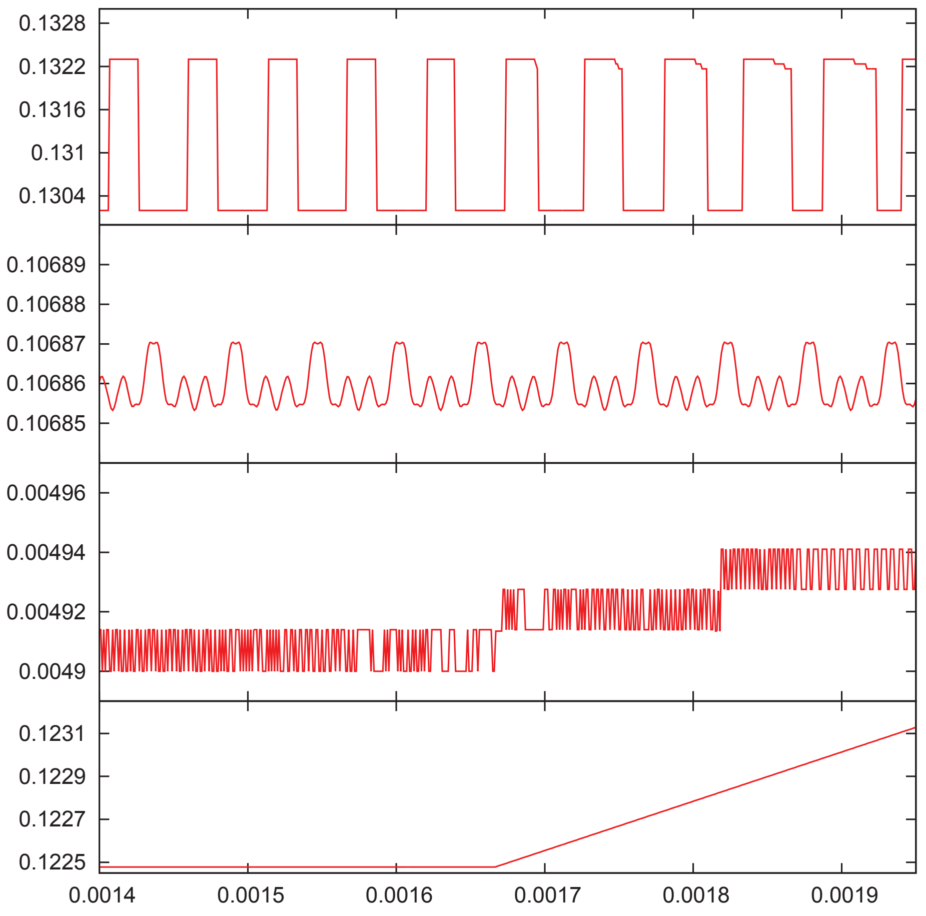


Figure 2.10: Comparison of rendering methods for the evaluation of an unknown voxel visibility relatively to the camera position. A single voxel is displayed and the camera is translated in the Y axis. *top*: OpenGL. 2nd *row*: $\mathcal{C}^1$ function. 3rd *row*: Discretized polygon. *bottom*: Polygon.

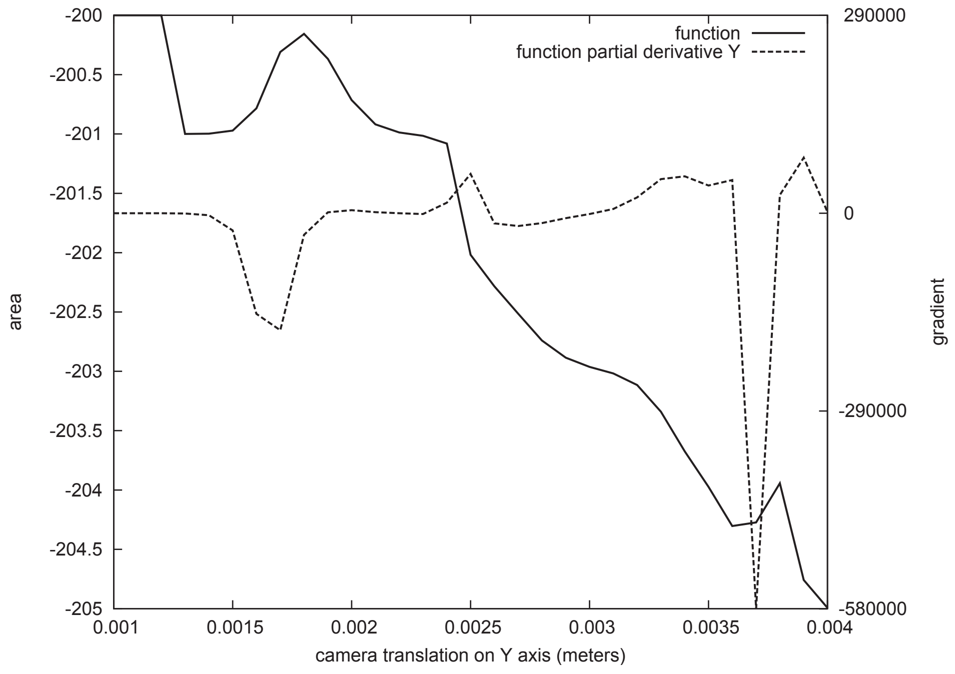


Figure 2.11: Close-up of area and gradient depending on the camera position on the Y axis.

to discretization are not present. Indeed the evaluation is constant when only one face of the cube occludes all the others, then it increases linearly when a side face becomes also visible.

The discretized polygon method would ideally stay constant then increase in a series of stair steps but the sampling introduces oscillations.

2.3.5 Gradient evaluation

Though the function has an overall evolution matching our expectations, the gradient has a higher variability than expected. In fact, at smaller movement scale than presented in the previous section, the function shows abrupt variations of low amplitude. A typical example of this problem is illustrated in Fig. 2.11 where the conditions are the same as in the previous section. It appears that the cause of such variations comes from our formulation which relies on a sampling of the data by using the result image pixels. In fact, the values of some pixels can change drastically during small movements of the camera around the object. Small variations in camera pose can result in some parts of voxels becoming visible whereas other parts become occluded or out of the visual field. The camera pose relatively to the object and the object specificities can influence the rate at which the value of different pixels in the resulting image can increase or decrease, as it is clearly shown in

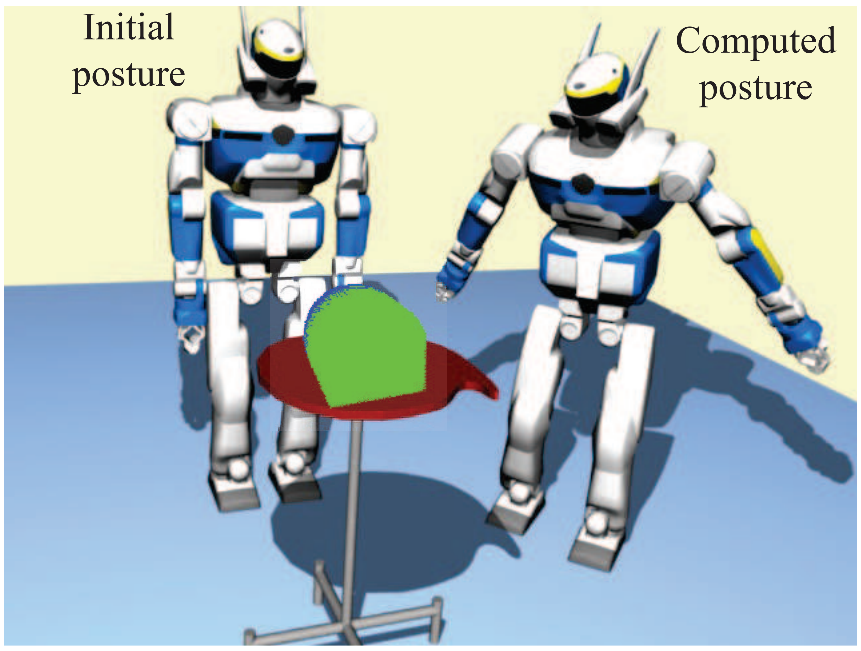


Figure 2.12: Example of a generated pose after one iteration of our NBV algorithm. The object to model is a sphere which has been perceived once from the initial pose.

Fig. 2.10. Unfortunately this affects badly the optimization process which therefore cannot converge properly, and reflects in the computation time needed to find a solution.

2.3.6 Pose generation

We tested our posture generation solution in simulation using various virtual objects, including the sphere shown in Fig. 2.9 and the soldier shown in Fig. 2.8. Two main problems with our criterion (2.26) need to be faced: (i) the computation time, as it takes from a few seconds to a few minutes to compute an area or a gradient depending on the number of voxels to process, and (ii) the presence of many possible local optima. When seeking a global optimal solution, these problems lead to a processing time, in order to generate a pose, between several minutes to a few hours. In such experiments, the optimization algorithm can solve all constraints efficiently but gets stuck in one of the objective function local minima, relatively far from any obvious better solution. Thus a complete modeling process cannot be achieved in an acceptable amount of time using this criterion alone. Fig. 2.12 gives the result of a typical pose generated from the initial robot posture in front of the object. The humanoid moved about 1m from its starting position and correctly oriented itself toward the object. As it can be noticed, though, further movements on the side would lead to a much higher amount of unknown visible area.

2.4 Conclusion

In this chapter, we presented a first solution to the Next-Best-View problem which tackles the constraints of the humanoid robot body. The described solution relies on the specificities of the Posture Generator which uses formulations which are at least $\mathcal{C}^1$ for the constraints and the objective function.

The proposed algorithm can generate statically stable postures without constraining the feet relative position thanks to a new equilibrium constraint.

An original formulation of the visibility of 3D data, adapted to the PG, has been introduced which results in rendering results that are consistent with traditional methods. Though it aims at solving the NBV problem while taking into account the constraints on the robot body in one coherent step, it has a relatively high computation cost and it also presents high variations in the gradient which result in convergence problems to generate a posture. Moreover it is difficult to put additional vision constraints in such a formulation.

Though the main cause of limitations for our $\mathcal{C}^1$ function is now well understood, we could not find a way to modify the formulation in order to decrease the amount of local optima without increasing the computation time, for example, by increasing the sampling frequency.

For these reasons, we designed and tested another approach, presented in the next chapter, to the formulation of the NBV selection which does not rely on FSQP.

Chapter 3

Local quadratic approximation of an object visibility for Next-Best-View selection

Contents

This chapter introduces another original approach to the NBV problem for autonomous object modeling which also tackles the constraints on the humanoid body.

The new algorithm aims at solving the problems induced by our $\mathcal{C}^1$ formulation of unknown visual information visible detailed in the previous chapter: (i) the slow convergence due to the presence of local minima and (ii) the high computation time for each viewpoint evaluation. To avoid these problems, we remove the reliance on a $\mathcal{C}^1$ formulation for the evaluation of visual data by moving this evaluation process out of the Posture Generator. The generation of the robot posture is then done in two steps:

1. We find a camera position and orientation that maximizes the amount of unknown visible while solving specific constraints related to the robot head. Various visual constraints can also be introduced to enhance the robustness of the algorithm. The search involves the evaluation of a great number of viewpoints, typically few thousands, in order to cope with a possibly great number of local minima depending on the object shape complexity.

2. We generate a whole-body posture for the robot by considering the desired viewpoint found in the first step as well as additional constraints related to the humanoid body. In the cases where a posture cannot be generated in the second step, we modify some conditions in the first step and launch the process again.

Though these two steps could be more tightly coupled, e.g whole-body postures are generated for each viewpoint tested, this would result in a great increase in the convergence time as both steps' computation time would be multiplied instead of being added.

Though most works in the NBV field have offered different solutions for the first step, we designed an original algorithm in order to cope with restrictive common assumptions, and in order to deal with the second step constraints. For example, as seen in chapter 1, various works in the litterature reduce the problem dimensionality by constraining the sensor position and orientation, and sample the configuration space in order to retrieve a solution in an acceptable amount of time. By using a Graphical Processing Unit (GPU) and recent optimization algorithms, though, it is possible to relax these assumptions while keeping a reasonable computation time.

We introduce this two-steps NBV algorithm, illustrated in Fig. 5.13, in order to broaden the types of object to be modeled while taking into account the constraints related to the use of a humanoid robot. Moreover the algorithm can be easily adapted to a broader range of problems such as the exploration of an unknown environment.

We propose to solve the first step by using a global sampling coupled with NEWUOA [Powell 2004], an optimization method adequate for noisy functions. The properties of NEWUOA allow us to broaden our choice for the practical quantification of visual information as detailed in section 3.1. The objective function given to NEWUOA is detailed in section 3.2 and the viewpoint serach process is presented in 3.3.

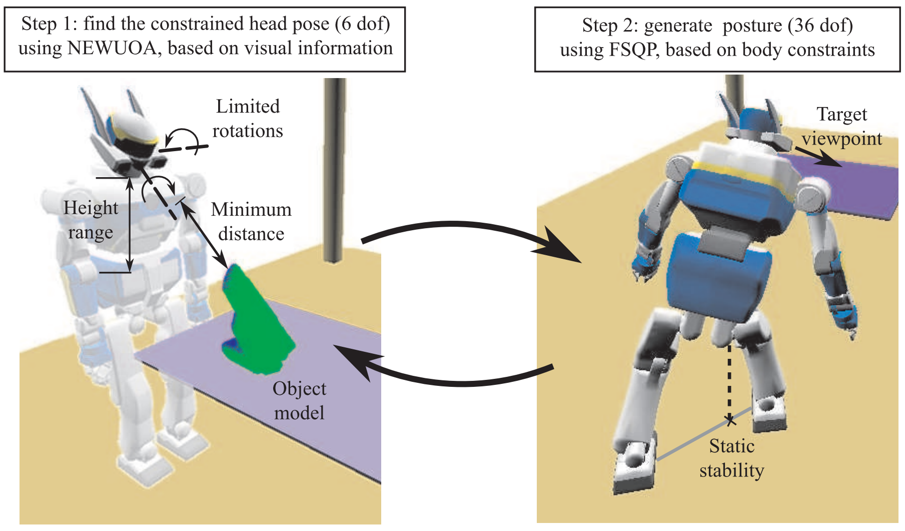

Step 1: find the constrained head pose (6 dof) using NEWUOA, based on visual information
Limited rotations
Height range
Minimum distance
Object model
Step 2: generate posture (36 dof) using FSQP, based on body constraints
Target viewpoint
Static stability

Figure 3.1: Two steps to generate the next posture to update the object model.

The second step is solved using the Posture Generator presented in the previous chapter with the same constraints. The objective function is replaced with a function that sets the robot toward a desired posture and thus acts as an esthetic criterion. We then add a constraint to set the robot head in the configuration found in the first step. The second step is detailed in the section 3.3.3.

3.1 Basis for viewpoint selection

Before the presentation of the details for the first step realization, we need to introduce the basic foundations to find an adequate viewpoint in the first step. First we examine briefly the specificities of the NEWUOA algorithm which is used to evaluate an original objective function in order to find a viewpoint. Then we review possible quantification methods to evaluate the amount of visual data of interest depending on the camera viewpoint. We focus on methods that are adapted to NEWUOA and efficient in terms of computation time.

3.1.1 NEWUOA optimization method

NEWUOA (NEW Unconstrained Optimization Method) [Powell 2004] is a local derivative-free optimization method. The search for an optimum is done by constructing and updating a quadratic approximation of the objective function through a deterministic iterative sampling. The process to find a local minimum is illustrated in Fig. 3.2. The sampled vectors at each step in the NEWUOA search process are selected according to the previous sampling results and the actual state

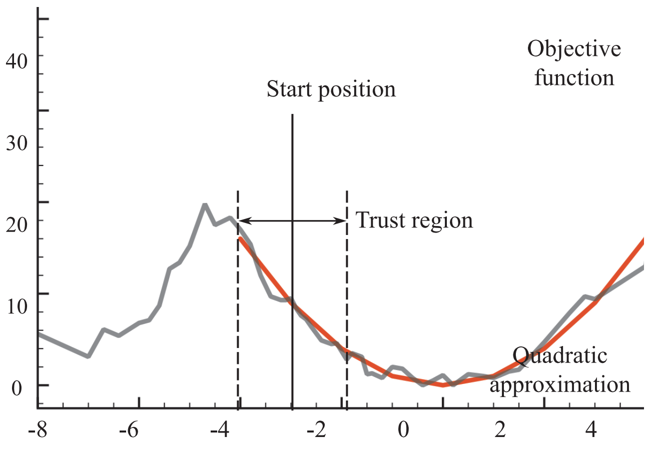


Figure 3.2: NEWUOA method to find the minimum of a non-differentiable function.

of the quadratic approximation. A spherical trust region must be defined using two radius parameters: ρ_{beg} and ρ_{end}, and a given starting vector which will be the camera pose in our case. The parameter ρ_{beg} sets the maximum variation that can be taken by the objective function variable for the initial quadratic approximation. The parameter ρ_{end} sets the desired accuracy of the optimum search. We can remark that the trust region influences the sampling process but it does not limit it. Indeed, depending on the quadratic approximation found, vectors outside of this region can be tested and selected as the optimal found.

NEWUOA has the advantages of being fast and relatively robust to noise while allowing us to keep the 6 degrees of freedom required for the viewpoint selection.

3.1.2 Evaluation of the unknown visible

As we do not rely anymore on a function inside the PG to evaluate the visibility of the object, we can use a quantification of visual information which uses traditional rendering methods. In this approach, the estimation of unknown data visible from a specific viewpoint is based on the visualization of the current occupancy grid. It can be computed quickly by taking advantage of current hardware acceleration through the OpenGL library. As we have seen in the section 2.3.4 of chapter 2, pixel based rendering methods introduce oscillations of small amplitude and high frequency in the measured quantity. But such oscillations have only a negligible influence on the convergence of NEWUOA when the variations of the measurements inside the trust region are greater than the oscillations. We thus consider such approximate representation to be a useful indicator to find a good viewpoint.

Two practical quantification of the visible area are possible with a hardware-accelerated rendering of an occupancy grid, as illustrated in Fig. 3.3:

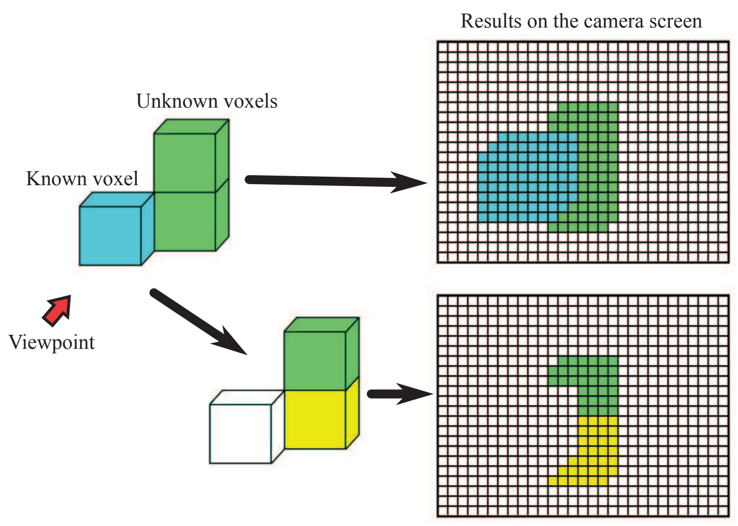


Figure 3.3: Practical quantification of unknown data visible for an occupancy grid. We can set all unknown voxels to the same color and count the number of corresponding pixels (top), or set a different color to each unknown voxel and count the number of different colors (bottom).

1. pixel-based: the number of pixels visible which are related to the unknown voxels are counted. This is easily implemented by setting all unknown voxels to the same color, e.g green, make sure all other displayed objects have a different color, and counting the number of green pixels in the result image taken from the current viewpoint. The problem is that it tends to attract the camera unnecessary close to the object; The camera image can be filled by the pixels from the desired type though only very few voxels are perceived.

2. voxel-based: the number of unknown voxels visible are counted. An efficient implementation, which relies on the GPU, assigns a different color for each unknown voxel while all other objects and known voxels share the same color as the background. The area of interest is then equivalent to the number of different colors displayed. This limits the number of unknown voxels to the number of colors that can be used, e.g $256^3 - 1$ for RGB 24 bits rendering, but this limitation is enough in practice; Even with large occupancy grids of dimension 512^3, only voxels corresponding to the object surface or the occupancy grid surface are displayed.

   This quantification method though has the opposite problem compared to the pixel-based method: the camera tends to be too far away from the object. Indeed, due to the perspective projection, more voxels can be visible from farther position but then most of them appear unnecessary small.

We choose to mix the two quantifications in order to increase the number of voxels visible while having a minimum visibility for each voxel. The amount of unknown visible, noted N_u, is set as the number of *unknown* voxels visible, but a voxel is considered visible only if the number of corresponding visible pixels is above a defined threshold, typically 5. This ensures the robot is neither too close nor too far from the object.

Our viewpoint selection problem thus relies on the maximization of N_u (or more precisely, the minimization of $-N_u$) while satisfying other constraints.

We can note that a more accurate shape of the object, based on triangles instead of voxels, can be used also for a hardware-accelerated evaluation of visible areas. In our case, though, we rely on an occupancy grid to update the shape of the object through the space carving algorithm. We thus choose to work directly with the occupancy grid in order to reduce the computation cost as we avoid the conversion of the object model from voxels to triangles each time the model is updated.

3.1.3 Related works

Let us first review various works in the literature which have considered occupancy grid representation as a basis for their NBV algorithm. They have been introduced in previous chapters but we analyze some of them in more details in this section in order to clarify the main specificities of our approach.

3.1.3.1 Next-Best-View for object modeling

As we have seen in chapter 1, the main common assumptions for the NBV algorithms dedicated to object modeling are that (i) the viewpoint position is constrained to the surface of a sphere around the occupancy grid, (ii) the viewpoint orientation is set toward the grid center, and (iii) the field of view is large enough to perceive the entire object. Such constraints can be problematic with concave or elongated objects where the camera would be too close to some parts while being too far from other. Moreover, especially when considering objects with complex concavities, some optimal viewpoints may not be oriented toward the object center.

3.1.3.2 Next-Best-View for scene exploration

Though the problem of scene exploration is slightly different then object modeling, the dedicated algorithms can address some similar points to our problem.

In [Sanchiz 1999], the presented algorithm manipulates the five degrees of freedom of a mobile robot to build the model of an unknown environment. The paper proposes different criteria to optimize depending on the viewpoint:

- the amount of unknown area visible.

- a minimum amount of known area visible. This is set in order to correct eventual positioning errors of the mobile robot when registering new data with the current model.
- the quality of the viewpoint. The criterion, similar to [Massios 1998], is based on the voxel normal vector and the sensor view vector when the range map is acquired.
- the navigation cost from the current position to the computed viewpoint.

Actually only one criterion is tested on the paper which is a function based on the unknown area visible and the quality of known voxels visible.

The search of a viewpoint is done locally to the current pose using two search methods: (i) the simplex method for selecting the best 3D position and (ii) an exhaustive sampling that test a finite number of 2D orientations at each tested position.

Our approach, though roughly similar in some aspects to this work, has many particularities:

- The simplex method is replaced with the more recent NEWUOA method which improves the convergence speed, especially in a high dimension space.
- We need to include in the objective function some constraints that reflect some of the constraints on the humanoid body. This is necessary for the proper execution of the humanoid robot posture generation.
- The correction of eventual positionning errors is done using visual landmarks that can be located on the object or in the environment.

Triggs and Laugier [Triggs 1995] used a relatively similar approach than our work but oriented toward the problem of visual inspection. They use a robot mounted CCD camera to analyze parts of a known model. The evaluation of a sensor pose is formulated as a weighted function that includes many constraints related to the robot: kinematics, visibility of the model, score of the pose for the specific inspection task, mobility of the robot, robot/environment collisions. The method relies on an original global function optimization technique to examine specific parts of a known model while avoiding occlusions and collisions. As the object model as well as the environment are known, the method first computes an optimal set of unordered viewpoints and then optimize the trajectory of the sensor. To reduce the complexity, they restrict the search to the 3D position of the camera and the orientation is set automatically to point toward the target of the visual task to realize. The 3D positions that are evaluated during the search are chosen by relying on an octree structure. Based on the evaluations at the corners of the cubes, parts of the octree are subdivided to allow the tests of new positions. As noted in the paper, though, the model is strictly local as the function samples at each region vertex act as the controlling vertices of the region. This may be problematic when the function is noisy and thus we prefer to rely on optimization techniques such as NEWUOA.

3.2 Formulation of the constraints on the camera pose

Though NEWUOA is supposed to be used for unconstrained optimization, some constraints on the camera pose need to be solved in order to generate a posture with the PG in the second step from the computed viewpoint in the first step. This is done by using classically the method of Lagrange multipliers.

The constraints on the camera position $\mathbf{C}$ and orientation $\psi_{\mathbf{c}}$ included in the evaluation function of the first step given to NEWUOA are formulated as:

$$\begin{cases} C_{zmin} < \mathbf{C}_z < C_{zmax} & (3.1) \\ \forall V_i, d_{min} < d(\mathbf{C}, \mathbf{Vc}_i) & (3.2) \\ \psi_{\mathbf{c}xmin} < \psi_{\mathbf{c}x} < \psi_{\mathbf{c}xmax} & (3.3) \\ \psi_{\mathbf{c}ymin} < \psi_{\mathbf{c}y} < \psi_{\mathbf{c}ymax} & (3.4) \\ N_l > N_{lmin} & (3.5) \\ \forall i, \mathbf{C} \neq F_i \vee \psi_{\mathbf{c}} \neq Fr_i & (3.6) \end{cases}$$

The range of the camera height is limited by (3.1) to what is accessible by the humanoid size and joints possible configurations. Practically for an HRP-2 robot, using a squatting posture, it is possible to set the camera at a height of about 1 meter. The standing posture sets the camera at about 1.3 meter.

The constraint expressed in (3.2) sets a minimum limit distance d_{min} which is needed between the robot head and $\mathbf{Vc}_i$ the center of each voxels of the object. This is required to obtain depth information using the stereo rig. We can also remark that as the camera is embedded in a robot, it needs a minimum distance to avoid physical collisions with the object. This constraint help to ensure that, even if the camera does not face some parts of the object, the robot will respect a safety distance from them. The value of d_{min} is then deduced from the specificities of the stereo rig and the robot size.

The rotations on X and Y axes are limited by (3.3) and (3.4) to ranges set according to the robot particularities, mostly the joint limits of the neck.

The constraint (3.5) keeps a minimum number of landmarks, i.e. visual features that are known or were detected in previous views, visible from the resulting viewpoint. This constraint is used to help recover precisely the pose of the newly perceived 3D points relatively to the previously acquired data. Though most works in the NBV field consider the sensor motion to be precise enough so that the registration of new points in the model is not problematic, motion errors cannot be ignored with a humanoid. The occurrences of drifts is usual when the robot moves from a posture to another and thus the final pose of the camera is not exactly the one computed. Therefore we want to consider visual landmarks to correct the estimation of the reached pose. We can remark that, for this NBV method, we do not restrict the landmarks to be on the object; they can be positionned in the environment around the object as well.

Finally, the particular constraint (3.6) ensures that the resulting pose will not be near previously found poses, with position F_i and orientation Fr_i, which could not be reached by following steps in the modeling process. Another purpose of this is to offer a method to avoid positions in the environment where known obstacles are located. We can remark that this constraint is necessary to ensure that the algorithm can converge toward a valid posture even though some constraints are not expressed in the viewpoint or posture generation search. For example, some obstacles in the environment may limit the possible motion trajectories while not visually occluding the view of the camera and thus, the motion planner may fail to find a way between the current posture and the target one.

3.2.1 Evaluation function formulation

In order to include the constraints into the function that NEWUOA evaluates, we designed continuous mathematical formulations to express them.

3.2.1.1 Interval

The interval constraints (3.1), (3.3) and (3.4), are expressed as:

$$K_v = (\alpha\, v - \mu)^p \tag{3.7}$$

where parameters α and μ are set according to the limits possible for the variable N_u. These are used to modulate, respectively, the interval center and width depending on the parameter v to constrain and are thus directly deduced from the robot body specificities. v can correspond to the viewpoint height $\mathbf{C}_z$, and orientation angles $\psi_{\mathbf{c}x}$ and $\psi_{\mathbf{c}y}$. p can be set to a large value, typically 4, so that the result is close to 0 inside the interval and increases quickly outside of it. The parameters are computed in order to reach the maximum possible value of N_u when we reach the point where the constraint is not satisfied.

3.2.1.2 Minimum distance

The inequality constraint (3.2) related to the minimum distance between the camera and the object is formulated as:

$$K_d = \exp^r\left(\gamma\ (d_{min} - d(\mathbf{C}, V_{near}))\right) \tag{3.8}$$

where γ and r parameters are also set according to N_u. V_{near} designates the closest voxel relatively to the current camera position. The minimum distance d_{min} is computed using the stereo rig parameters and the robot size.

3.2.1.3 Landmark visibility

For the landmark visibility constraint (3.5), the formulation relies both on the visible surface of landmarks and their normal vector. The surface visibility for each landmark i is computed relatively to its amount of pixels visible from the current viewpoint pv_i using a sigmoid function:

$$ls_i = \frac{1}{1 + exp\left(pmin_i - pv_i\right)} \tag{3.9}$$

The parameter $pmin_i$ is the minimum amount of pixels required to consider the landmark i visible, and its value depends on the original landmark size.

The visibility of each landmark relatively to their normal vector $\mathbf{Nl}_i$ and the current camera view direction vector $\mathbf{C}_{view}$ is expressed using another sigmoid:

$$ln_i = \frac{1}{1 + exp\left(\beta\left(\left(\mathbf{C}_{view}.\mathbf{Nl}_i\right) + \phi\right)\right)} \tag{3.10}$$

where ϕ is related to the angle range allowed, and β determine the slope of the sigmoid function.

The final visibility coefficient for each landmark is computed by multiplying ls_i with ln_i. We set an arbitrary defined minimum number of visible landmark Nlm_{min} which is compared to the obtained coefficients by :

$$lv = \left(\sum_{i=0}^{N} ls_i.ln_i\right) - Nlm_{min} \tag{3.11}$$

The constraint for the evaluation function is defined in one of two ways depending on the sign of lv. Configurations maximizing lv are slightly encouraged when it is positive:

$$K_l = -\eta\, lv \tag{3.12}$$

The η parameter can be small so that the minimization of other constraints and the maximization of unknown visible both have a greater priority than the increase of number of visible landmarks beyond the defined threshold. In the other case, where $lv \leq 0$, the configurations are greatly penalized:

$$K_l = 2\, I_p \left(\frac{lv}{Nlm_{min}}\right)^2 \tag{3.13}$$

The penalty is expressed in relation to the total number of pixel I_p in the camera image.

3.2.1.4 Forbidden poses

The constraint to avoid unreachable postures (3.6) is simply formulated as a distance between the viewpoint considered and each of them:

$$K_f = \sum_{fp} exp\left(-\delta.D_{fp}\right) \tag{3.14}$$

where D_{fp} represents the sum of absolute differences between the values of the actual viewpoint and the unreachable pose fp. The δ parameter corresponds to the sensitivity of the constraint.

3.2.1.5 Viewpoint evaluation function

The evaluation function, used as input to the NEWUOA algorithm, includes all previously defined constraints formulations as well as the evaluation of visual data:

$$f_e = \lambda_z K_{\mathbf{C}_z} + \lambda_x K_{\psi_{\mathbf{c}_x}} + \lambda_y K_{\psi_{\mathbf{c}_y}} + \lambda_d K_d + \lambda_l K_l + \lambda_f K_f - N_u \tag{3.15}$$

where the λ parameters are computed to modify the scale of each constraint in order to match the range of values that can be taken by the variable N_u. More specifically, when a constraint is violated, the constraint formulation multiplied by the corresponding λ parameter should be equal to the largest value that can be taken by N_u.

3.2.2 Evaluation function behavior

Due to the constraints used and the specificities of objects to model, many different cases can result in local minima in our evaluation function that are quite disjoint as can be seen in the example shown in Fig. 3.4. This figure illustrates the behavior of some constraints in the evaluation function for different camera position around an object carved twice. For each position, the camera orientation is set toward the object center. The simulated object is 2 meters high, the camera is placed at a height of 1.3 meters and its X and Y positions are in the interval [-10,10] meters. There is a distance of 5 millimeters between each position tested.

The visual occlusions, the landmarks visibility and the obstacles in the environment introduces many local maxima. We can note in the final evaluation (center row-left column) that the landmark constraint do not allow the algorithm to select a camera pose where the visibility of unknown is the best: near the opposite side of the perceived object face.

A particularity of our algorithm is highlighted by the constraint on the minimum distance (bottom row-right column) where the camera is allowed to be placed inside the model space. For objects which present concavities, when space carving is applied, some parts inside the occupancy grid appear to be empty. This means that

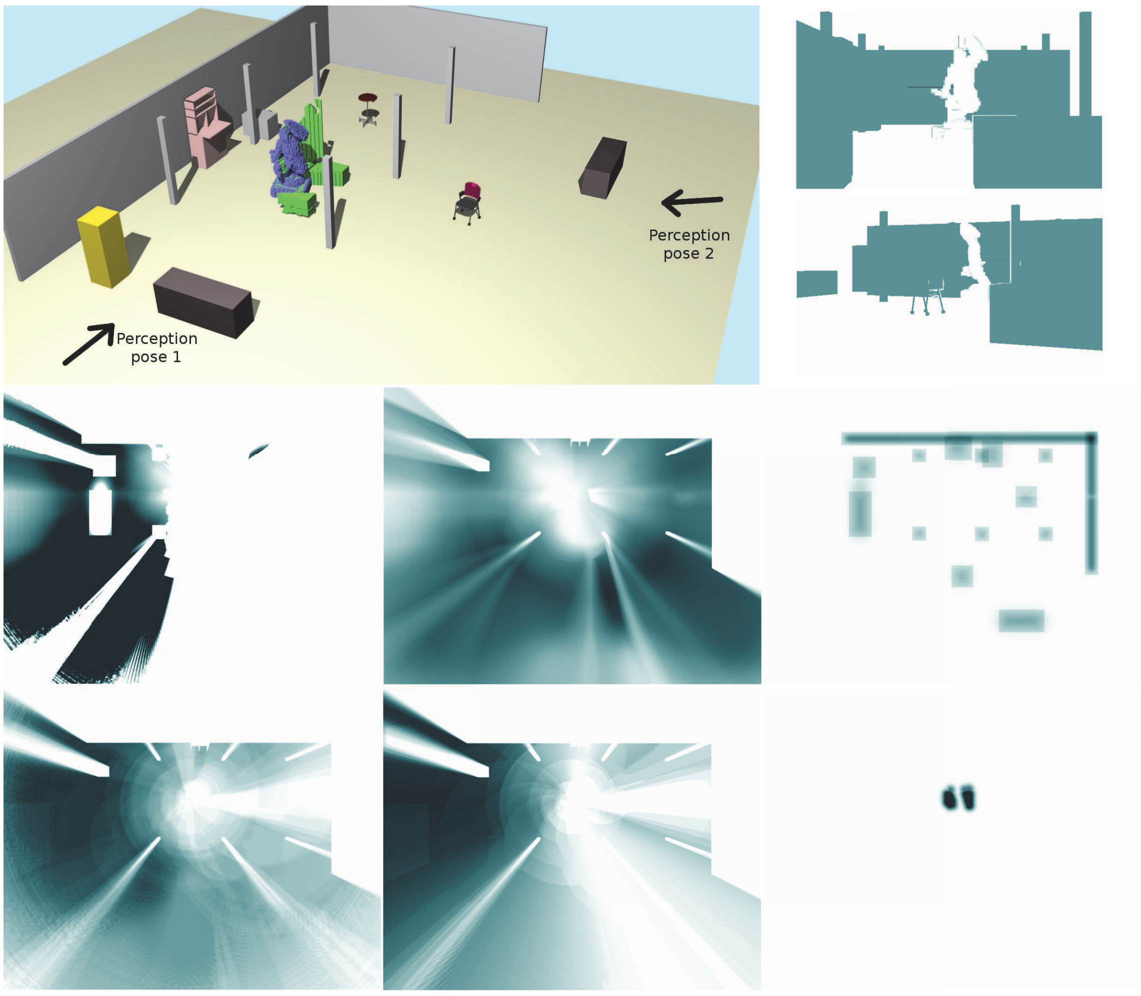


Figure 3.4: Example of the evaluation function components values depending on the viewpoint position. The components are computed with a fixed camera height and depending on the camera 2D position on the plane XY around an object which has been perceived twice. *top-left*: object model and environment used. *top-right*: the two views used to update the 3D model: perception pose 1 (top) and perception pose 2 (bottom). *center row, from left to right*: evaluation function result f_e, amount of unknown that is visible N_u, and forbidden poses constraint K_f. *bottom row, from left to right*: landmark visibility constraint with $\sum_{i=0}^{N} ls_i$ and $\sum_{i=0}^{N} ln_i$, and minimum distance constraint K_d.

the algorithm has the possibility to set a robot pose inside the object if it is large enough, for example a house. Thus, though it is not its initial goal, the algorithm can also be used for exploration tasks by adding the consideration of the motion trajectory between the computed poses.

3.3 Viewpoint search process

As the function to evaluate in order to compute a viewpoint can have local minima, we are completing the algorithm through three additional process: (i) limit the trust region in NEWUOA, (ii) launch iterative searches, and (iii) test different starting camera poses.

3.3.1 NEWUOA configuration

In this work, NEWUOA is used to seek locally the minimum of f_e by approximating it with a quadratic model. As seen earlier in the paragraph 3.1.1, three parameters are used as input to this optimization algorithm: an initial vector from where the search is started, a value which delimits the trust region around the initial vector in order to build the initial quadratic approximation, and a desired accuracy value used as a stopping criterion.

Due to the constraints in the evaluation of viewpoints, the trust region used as an input to NEWUOA needs to be limited to some local space around the initial vector in order to be pertinent enough. The limits of the trust region are set according to the voxels size and the distance between the camera and the object. More specifically, it is computed as the size of a voxel multiplied by the initial distance. We need to take into account these parameters for computing the size of the trust region in order to get significant visual changes when NEWUOA tests some vectors inside the trust region.

3.3.2 Iterative search

As a local method, the quality of the results found by NEWUOA can depend greatly on the starting poses given. Thus two additional techniques are implemented which are illustrated in Fig. 3.5. First, we run NEWUOA in an iterative way, i.e. it is run once using a defined starting pose and run again by using its result configuration as a new starting pose. This is done until a chosen maximum number of iterations has been reached, or until the result pose is not better than the last starting one. A step of this Iterative NEWUOA (*ItN*) search is formulated as:

$$pose_k = Newuoa_k\left(pose_{k-1}\right) \tag{3.16}$$

with k the iteration number of the NEWUOA algorithm from 1 to n, and $pose_{k-1}$ and $pose_k$ respectively the starting and found camera poses.

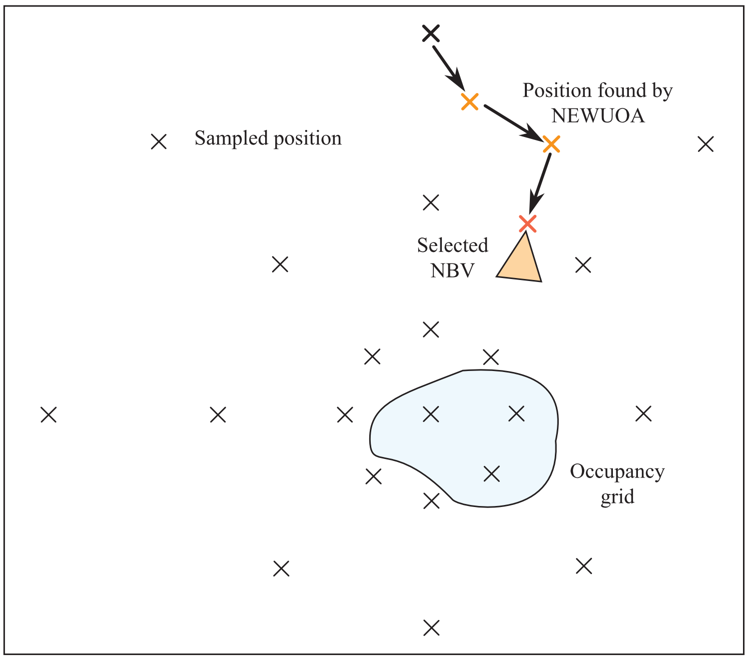


Figure 3.5: Illustration of the NBV selection for the robot head.

Second, we sample the 3D space around the object to obtain a set of starting camera viewpoints and we launch the *ItN* process for each of them. Results of all optimizations are then compared to select the best camera pose.

To reduce the computation time, the search can be executed in two stages. First, one *ItN* process is launched for each starting pose with a limited maximum number of iterations, typically one or two. Then we compare the results and launch one more *ItN* search from the best found pose, allowing a larger maximum number of iterations.

We can note that the sampled positions can be generated inside the object to handle cases where it has large empty spaces. For example, the algorithm can be applied to model both the inside and outside of a house. This also gives us the ability to bias the search toward some specific area by modifying the distribution of the starting poses.

In this work, the positions are distributed in the space relatively to their distance to the object: the density gets smaller when getting far away from the object as greater motions are required to get significant visual changes.

3.3.3 Posture Generator configuration

Once a viewpoint is computed in the first step of our algorithm, we use it as a constraint on the robot head to generate a complete posture for the humanoid 36 degrees of freedom. The Posture Generator (PG), presented in the paragraph 2.1.1, provides us with the constrained whole-body posture.

Our previous algorithm, presented in chapter 2, includes directly the evaluation of viewpoints as a $\mathcal{C}^1$ function which is set as the objective function to minimize. Other constraints are added to generate a posture: static stability, self-collisions avoidance, collisions avoidance with the environment, keeping the feet flat on the ground, and joints limits.

In the algorithm described in this chapter, we keep the same constraints but remove the objective function. We then add constraints to put the robot head in the desired position $\mathbf{C}_{s1}$ and orientation $\psi_{\mathbf{c}s1}$, formulated as:

$$\mathbf{C} = \mathbf{C}_{s1} \textit{ and } \psi_{\mathbf{c}} = \psi_{\mathbf{c}s1} \tag{3.17}$$

For this algorithm, the objective function for the PG is not necessary. Nevertheless it is possible to use it as an esthetic criterion to place the robot posture close to a reference posture.

The posture generation starts by using an initial posture and an initial free-flyer position and orientation. In this work, the default initial posture is set as a squatting posture which has proved to be well suited for reaching various difficult joint configurations after several tests. The free flyer position and rotation on the Z axis are set accordingly to the desired viewpoint pose.

In cases where the PG cannot converge, the goal camera pose is put inside the list of forbidden poses which is used in the constraint 3.6 and the first step is launched again to find another viewpoint.

3.4 Simulation results

Let us now present additional tests on the performance of the first step to select a pertinent viewpoint. We investigate further the quantification of unknown, the influence of the parameters for NEWUOA, the efficiency of a NEWUOA search relatively to a uniform sampling, and the average computation time to generate a whole-body posture for the humanoid robot. Finally some results of object modeling processes in simulation are presented.

3.4.1 Unknown quantification comparison

We compare the improvement of our unknown quantification method against a simple quantification of pixels corresponding to unknown voxels.

The results are roughly similar when modeling small objects. The robot cannot get too close to the object due to the minimum distance constraint and thus the camera viewport encompasses the entire object.

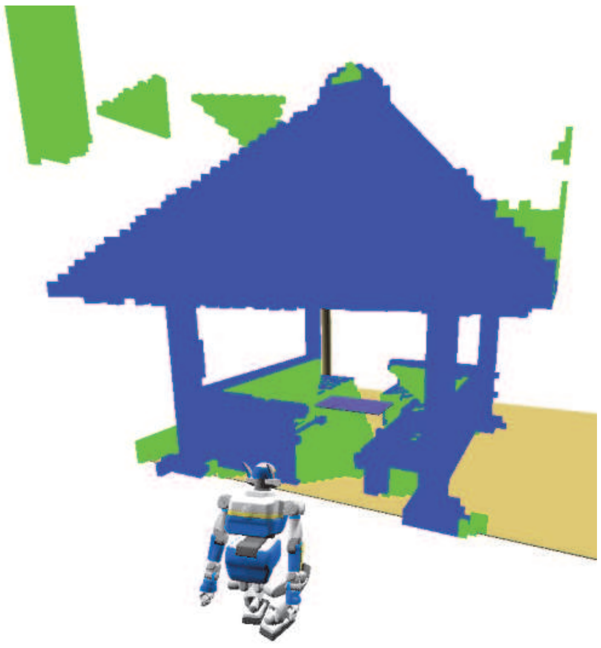


Figure 3.6: Reconstruction of a large object by quantifying the unknown using pixels quantification only.

If we consider pixels only, a problem arises when we build a model of a large object, as can be seen in Fig. 3.6. Patches of unknown voxels are separated from the rest of the occupancy grid as the robot is set too close to the object when trying to maximize the number of unkown pixels visible. Parts of the object are then not included in the camera view space.

During our tests, we could verify that our estimation of unknown based on voxels and pixels results in a significant reduction, which can go up to 40 percent, of the number of poses necessary to model objects with a height of about 4 meters.

3.4.2 NEWUOA tests for camera pose evaluation

Using our objective function f_e, described in 3.2.1, the initial conditions for a NEWUOA search influences the viewpoint found. We thus tested the influence of the trust region parameters (Fig. 3.7) and the starting position (Fig. 3.8). Fig. 3.7 shows the average results for the viewpoint obtained depending on the ρ parameters. ρ_{beg} sets the maximum variation that can be taken by the camera pose parameters for the initial quadratic approximation, and the parameter ρ_{end} sets the desired accuracy of the optimum search. The tests were conducted by selecting a camera pose around an object model and by launching the optimization with different values for ρ_{beg} and ρ_{end}. This was repeated for 14 different objects with 3 different starting poses for each. Overall, better evaluated poses are obtained when

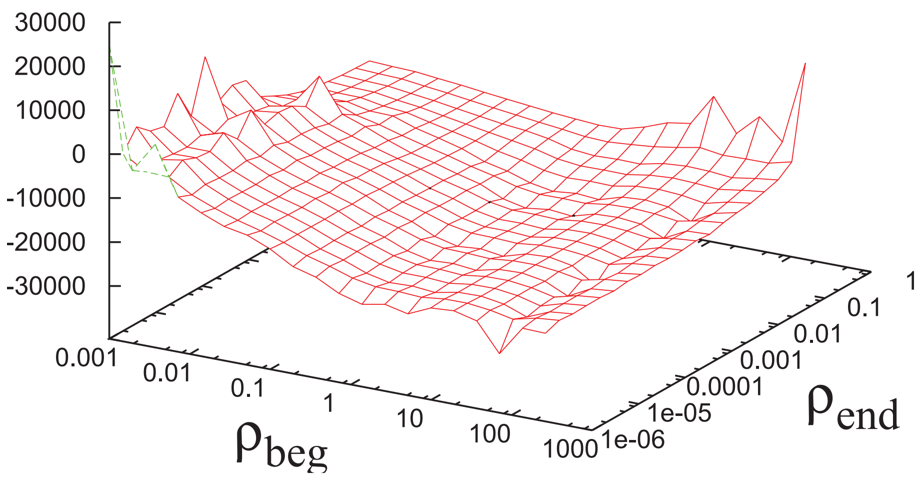


Figure 3.7: Influence of NEWUOA trust region parameters on the search results. The ρ_{beg} and ρ_{end} parameters are multiplied by the object maximum size.

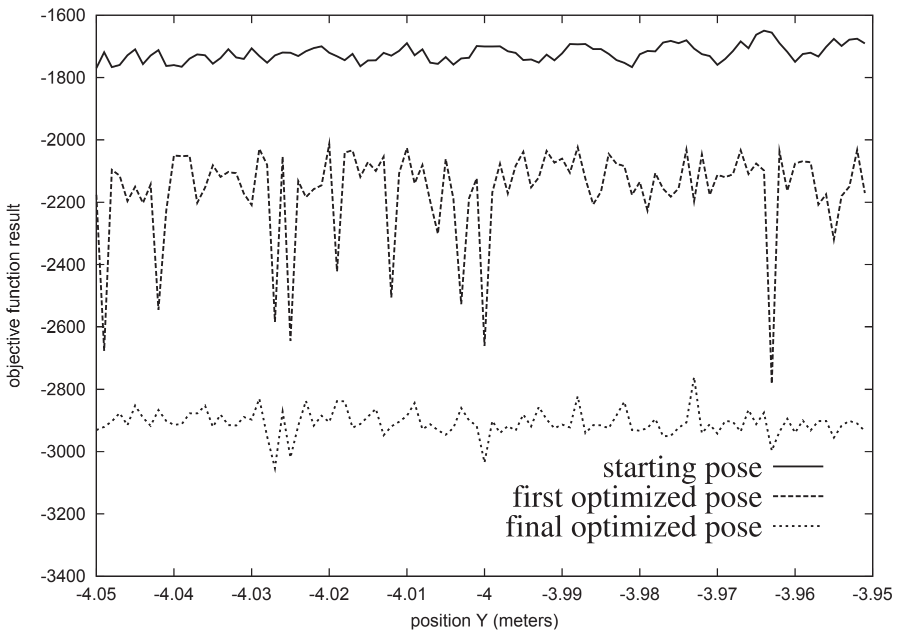


Figure 3.8: Influence of the starting position on $f_e(pose_s)$ and the viewpoints found by our *ItN* process $Newuoa_1\,(pose_s)$ and $Newuoa_n\,(pose_{n-1})$.

ρ_{beg} is equal or superior to the object maximum size, and when ρ_{end} is smaller than one hundredth of this size.

Fig. 3.8 shows the influence of the starting pose on the viewpoint found. This was tested by launching a NEWUOA search with different initial configurations, i.e the camera is translated on the Y axis in front of an object model. These tests were done with $\rho_{beg} = 0.4$ and $\rho_{end} = 10^{-5}$.

First, we can note that the evaluation of the unknown function, i.e the 'starting pose' curve, can change abruptly even with small variations of the pose. This highlights the complexity of our evaluation function, already discussed in 3.3.1, which has a lot of local minima. Depending on the starting position, NEWUOA can thus generate relatively different quadratic approximations which will lead to different samples selected.

This graph also highlights that, though a single iteration of NEWUOA results in an improved pose, it is often stuck inside a local minima. Nevertheless, by using successive iterations, much better viewpoints are reached. In some tests, the camera can get moved up to 0.7 meters and rotated up to about 50 degrees in many final optimized poses around a small object, e.g. 0.4 meters long. In order to find a good pose, a large number of iterations is not necessary. In this test, the average number of iterations was 5 and the maximum number allowed, which was set to 10, was reached for only 2 percent of the tested initial poses.

3.4.3 NEWUOA VS fixed sampling

We compared the results obtained with a simple NEWUOA search against a pre-computed fixed sampling of the 6D viewpoint configuration space. This sampling is done around the last position where a space carving operation has been done. The number of samples as well as the limits of the area to test are defined manually for each of the 6 dimensions.

Not surprisingly, the fixed sampling can result in viewpoints with similar or better results using roughly the same number of sampled vectors. As noted earlier, depending on various parameters such as the object complexity or the landmarks distribution, the NEWUOA search may find itself restricted to local minima close to the starting pose. Nevertheless, such local minima can be reached by NEWUOA using less samples than a fixed sampling of the local space. Thus our search for a viewpoint, presented in paragraph 3.3.2, includes the two methods: first, have a rough sampling of positions in the areas of interest, then use NEWUOA to refine the search for the closest local minima.

3.4.4 BOBYQA

In the meantime of the development of this NBV approach, Powell released a new algorithm called Bound Optimization BY Quadratic Approximation (BOBYQA)

[Powell 2009], which is an upgrade of his NEWUOA algorithm. As the name implies, the main particularity of BOBYQA, compared to NEWUOA, is that it can use bound constraints on the set of parameters to optimize.

By using BOBYQA, we can remove our formulation of the interval constraints on the camera pose $(K_{\mathbf{C}_z}, K_{\psi_{\mathbf{c}_x}}, K_{\psi_{\mathbf{c}_y}})$ from our evaluation function f_e in Eq. (3.15). The other constraints (K_d, K_l, K_f) have complex bounds that cannot be expressed directly in BOBYQA, therefore they are left inside f_e.

Though we discover the existence of BOBYQA relatively late, we could perform some preliminary tests using the implementation of the algorithm available in the NLopt open-source library [Johnson 2009]. The performance of BOBYQA, in terms of resulting pose score and number of iterations, was compared to NEWUOA on a number of virtual objects. Overall BOBYQA could reach viewpoints with a score in an order of magnitude similar to those reached by NEWUOA. But the great advantage of BOBYQA over NEWUOA is the numer of iterations needed which is about 10 times less. One reason is that the current formulation of the interval constraints in the evaluation function for NEWUOA drive the pose search away from the limits and thus restricts the optimization algorithm search to a smaller space that the real one. But BOBYQA can reach poses with high scores that are close to the limits of the interval constraints satisfaction as these are not expressed in the evaluation function.

Though more tests are needed to assess the performances of BOBYQA in our approach, simulation tests show it as a good alternative to NEWUOA. It has thus been included in our NBV software.

3.4.5 BOBYQA VS Nelder-Mead Simplex

The Simplex method is another derivative-free optimization algorithm, which was introduced by Nelder and Mead [Nelder 1965]. Though some convergence problems have been demonstrated for some convex functions [McKinnon 1998], it has been widely used, including in some works related to the NBV problem [Sanchiz 1999] [Lopez-Damian 2009].

We tested the implementation of this method, provided by the NLopt library [Johnson 2009], to evaluate its performance with our evaluation function and to compare the results with BOBYQA. The tests were conducted by testing different models and searching for the Next-Best-View from different starting poses. We then compare the score of the found view, according to our evaluation function f_e (Eq.3.15), and the number of evaluations needed.

Overall, BOBYQA results in better scores although, in a small number of specific configurations, the Simplex can give better results. Moreover the Simplex has a much slower convergence, testing between 2 to 10 times more evaluations before terminating. As BOBYQA converges faster toward local minima, we prefer to use it inside our viewpoint search process, presented in section 3.3. In most cases, a Simplex search makes more function evaluations than a global sampling with a BOBYQA search. By relying on a global sampling method, the algorithm can test specific viewpoints that cannot be accessed by a single local search. This is particularly useful in two specific cases: (i) when visual occlusions by obstacles in the environment create local minima, and (ii) when only sparse sets of unknown voxels remain in the object concavities in the final stages of the modeling.

3.4.6 Computation time

Each evaluation of a viewpoint relies on the OpenGL visualization of the occupancy grid which is loaded inside the graphic card memory. The evaluation time is thus relatively small and stays in the order of 10^{-2} seconds though, of course, it can vary depending on the number of voxels in the model, and the performances of the graphics card and the dedicated driver.

The search for the best viewpoint in the first step of the algorithm typically requires few thousands of evaluation. This depends on the number of sampled positions for the preliminary search, the input parameters for NEWUOA, and the number of iterations. During our tests, the first step could give a solution between 10 seconds and one minute.

The second step can generate a posture in the order of 10^{-1} seconds if the starting conditions are relatively close to the solution and if the space in the final location is not too constrained by obstacles. In others cases, it can take up to few seconds to get a solution or to abort the search.

3.4.7 Modeling process simulation

The experimental setting is simulated by having a 3D object perceived by a virtual camera. One example of generated postures to complete the modeling is presented in Fig. 3.9. The posture 0 is set manually and the six following are generated using our NBV algorithm with BOBYQA as the local optimization method. The trust region parameters, ρ_{beg} and ρ_{end}, were set respectively to 0.4 and 10e-5. Other parameters settings are: $p = 6$, $\gamma = 20$, $d_{min} = 0.6$, $Nlm_{min} = 5$, $\eta = 1$, $\delta = 1$, $\lambda_d = 100$, $\lambda_l = 1$ and $\lambda_f = 1000$.

The simulated modeling process loops through the following steps:

1. The disparity map is constructed using the object 3D informations and is used to perform a space carving operation on the occupancy grid. Some known voxels are randomly selected to be considered as landmarks.

2. The viewpoint search process, as described in the section 3.3, is launched. We sample the space around the object and launch iterative NEWUOA searches.

3. When an optimal camera pose is found, it is sent to the PG in order to generate a whole-body posture. If the PG does not converge, we add this camera pose

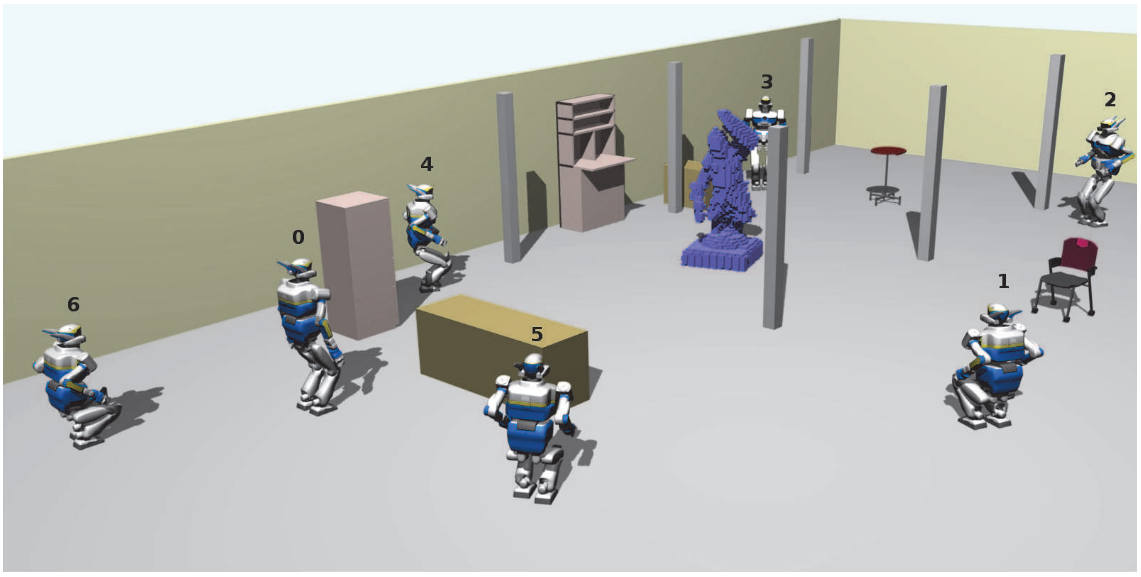


Figure 3.9: Postures generated for the reconstruction of a 2 meters high object in a cluttered environment.

in the list of poses to avoid, using the constraint (3.6), and run the first step again.

Another example of postures generated during the successful modeling process of a 50 centimeters ship on a table is illustrated in Fig. 3.10. The modeling was completed after 7 poses.

At the end of the process, we can note that some voxels in the upper deck on the back of the ship could not be perceived. This the result of the constraints on vision distance and limited height of the camera pose. The robot cannot be set close enough to the object as the parts of the object perceived by the two cameras in the stereo rig would be different and thus the 3D surface cannot be retrieve. Moreover, due to the limited size of the humanoid robot and the position of the stereo rig on the robot, the cameras height cannot be set high enough to observe the unknown surface.

There are also some patchs of unknown voxels in other various concavities of the object. This can be explained by the distribution density of the starting poses for the NBV search. The patches remaining between the ship sails are only visible from a restricted subspace of the configuration space of the camera. Depending on the concavity, this subspace is too small compared to the sampling size of the poses tested. A possible solution is to increase the number and ranges of sampled starting poses for the NEWUOA routine at the cost of increasing the computation time.

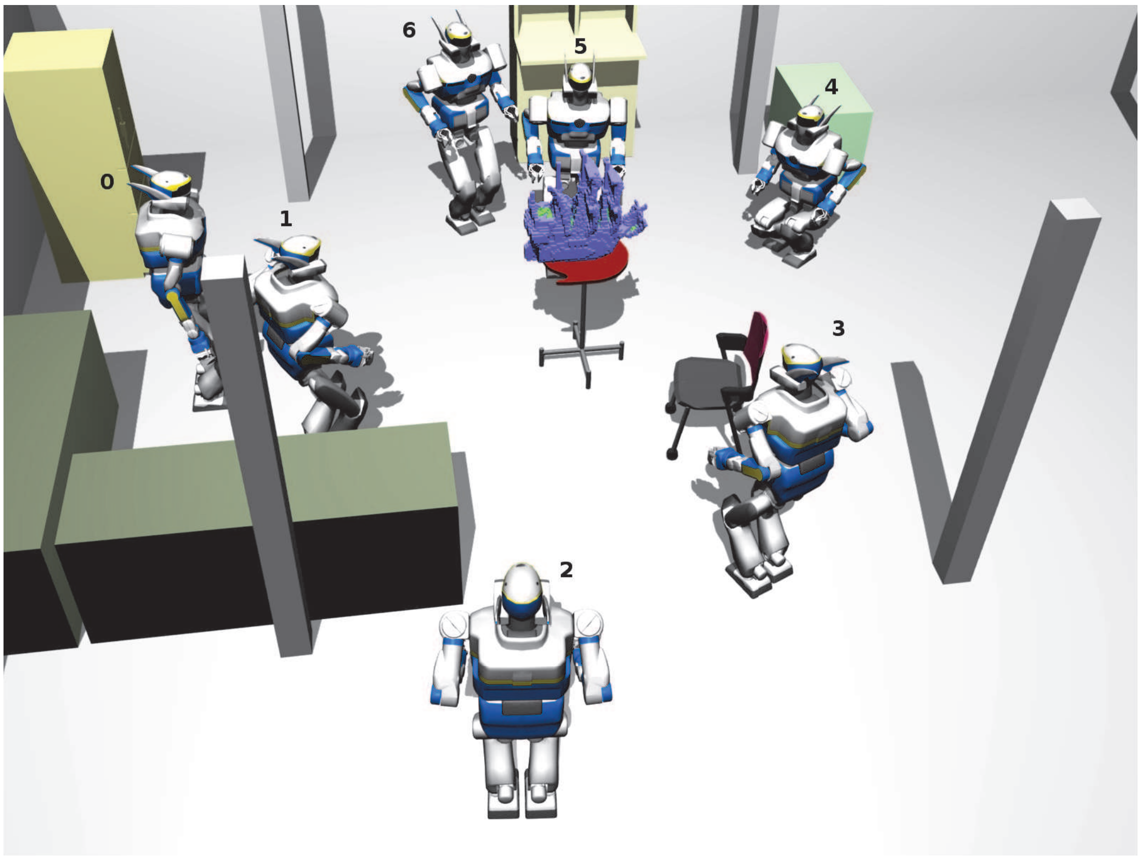


Figure 3.10: Postures generated for the reconstruction of a 50 centimeters high object in a cluttered environment. Remaining unknown parts on the object are displayed in green.

3.5 Conclusion

This chapter introduced a new method to generate automatically postures for a humanoid robot depending on visual cues in order to model unknown objects. The algorithm presented is general enough to efficiently find a pertinent viewpoint in order to complete models of objects which can have complex shapes, various sizes, and which are located in cluttered environments with possible visual occlusions. The postures are selected amongst the possible configurations allowed by stability, collisions, joint limits and visual constraints, so as to complete the modeling of an unknown object using a minimum number of postures.

Three complementary optimization methods: global sampling, NEWUOA and FSQP, are used in this two-steps algorithm in order to generate each Next-Best-Posture. The NEWUOA search, coupled with a fixed sampling of the robot head configuration space, can deal efficiently with the noise and discontinuities of the viewpoint evaluation function to minimize while keeping the six degrees of freedom (dof) of the robot head. We also tested the possibility of replacing the NEWUOA search with the newer optimization method BOBYQA. The Posture Generator can then quickly find a posture addressing all necessary constraints on the humanoid body to compute a solution in the 36 dof configuration space.

This approach was validated in simulation by building successfully models of various objects with complex shapes using a limited number of postures and a limited computation time. Though the separation in two steps of the posture generation process induces the possibility that the result of the first step leads to a second step that cannot converge, a solution has been implemented which launch again the first step by modifying the constraints in order to generate a new solution.

The current approach can be enhanced in various directions:

1. The motion planning between generated postures is not considered for the moment when evaluating possible viewpoints. This can result in cases where the computed postures is not reacheable as there are obstacles on the way from the current posture. A temporary solution for such cases is currently implemented by setting the unreacheable posture in the list of forbidden poses and launch the viewpoint selection process again. Another solution would be to integrate a path planning algorithm inside the viewpoint selection process at the cost of increased computation time. This would allow to consider the motion cost when selecting a viewpoint and would be required to adapt the current algorithm to the task of explorating an unknown environment.

2. A solution may exist that merge the two steps in one coherent step: searching the high dimensional configuration space of the humanoid body based on visual cues. The approach presented in the previous chapter tackles this problem but the results are not exploitable practically. Another approach could incorporate the constraints on the humanoid body in the BOBYQA search.

In order to achieve the autonomous modeling application with a real HRP-2 robot, we focused our efforts on solving various technical constraints and could not invest more time on these interesting problems.

Chapter 4
Vision tools

Contents

This chapter introduces the vision based methods that play an important role in the successful realization of the modeling experiment.

First, we present briefly the vision related hardware at our disposal to carry out the experiment.

We then discuss the processing of the visible object surface that allows to compute the sequence of robot poses necessary to the completion of the object model. As seen in the previous chapters, we rely on the perceived 3D surface of the object and the detected visual features in order to generate the Next-Best-View and the related Next-Best-Pose for the humanoid.

Finally we discuss the central role of vision for the autonomous localization of the robot. The precise position and orientation of the robot with respect to the object is required in order to register correctly newly obtained visual information with the current model. Practically though, when the robot walks, some drift usually occur [Stasse 2006a]. We thus rely on the environment model and the detected features to recover the robot pose using vision.

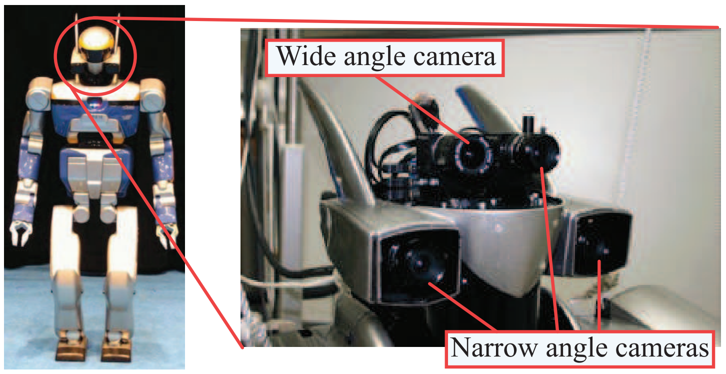


Figure 4.1: *left*: HRP-2 No.10. *right*: close up of the head with the location of the embedded cameras. In this work, we rely on the wide angle camera and the two narrow angle cameras located on the side of the head.

4.1 Vision hardware

Our experimental setup relies on the available robotic platform at the Joint Robotics Laboratory in AIST Tsukuba. Our work takes advantage of the functionalities offered by the robotic platform HRP-2 [Kaneko 2004], presented in Fig. 5.14.

4.1.1 Camera specifications

The head of HRP-2 number 10 is dedicated to vision tasks and includes four Point grey Flea 1.0 IEEE1394a cameras. All cameras can output 640x480 Bayer images at 60fps. Let us notice that only three cameras can be used simultaneously due to the limitation related to the bus controller and the device drivers used.

The two cameras on the side of the head and the left camera inside the head have a narrow angle with a vertical field-of-view of 25 degrees. The side cameras constitutes the robot stereo rig and are aligned horizontally with a baseline of 144 millimeters which makes them adapted for stereo vision on objects located between 50 centimeters and 4 meters [Stasse 2006b]. The right camera inside the robot head has a wide-angle with a field-of-view of 90 degrees and can be used for scene analysis.

We can remark that the cameras embedded inside the head are located behind a rounded dark plastic shield which results in visual artifacts in the images such as additional distortions and lens flares. This is a particular drawback for many visual processes. Nevertheless the plastic shield is left during experiments in order to answer concerns about the interaction between the humanoid robot and humans. To facilitate the acceptance of robots in environments shared with humans, the importance of the robot design is generally recognized.

4.1.2 Pose limits

In addition to the visual constraints depending on the vision system, we must consider the constraints on the vision sensors pose that are based on the robot characteristics. The height of the stereo rig can vary between $\sim$ 1.m and 1.39m, when the robot stands on its two feet. The lower limit is set according to the lowest squatting posture the robot can achieve with regards to its joint limits and the constraint on self-collision avoidance. Of course, lower heights can be reached if the robot is set in specific configurations such as quadrupedal postures. The upper limit on the stereo rig height is reached when all joints are set to 0, i.e. the robot is in its standing posture.

Regarding the orientation, all four cameras have a pitch of 10 degrees relatively to the head joint. The limits of the camera pitch, using the head and chest joints, are $[-25, 115]$ degrees but the lower limit may be reduced by relying on hip and legs joints.

The cameras roll depends only on the configuration of the hip and legs joints and is thus directly linked to the roll of the waist. Unstable configurations are quickly reached when the roll increases and thus the rotation on this axis is severely restricted. To ensure the robot stability, the pitch and roll limits will vary depending on the robot configuration.

The yaw is limited to $[-90, 90]$ degrees if we rely only on the chest and head joints, but is, of course, unrestricted if the robot is allowed to perform walking motions.

4.1.3 Computing power

For processing purposes, HRP-2 integrates two computers in its chest: hrp2010c and hrp2010v.

hrp2010c is dedicated to the control of the robot, managing most of the sensors and all actuators. It runs Ubuntu 8.04 on an Intel Pentium M processor at 1.80GHz, with 1 GB of RAM.

hrp2010v is dedicated to the vision tasks, accessing the cameras and processing the images. It runs Ubuntu 8.04 on an Intel Core 2 CPU T7400 at 2.16GHz, with 1GB of RAM. It also includes an Intel Mobile 945GME Express graphics controller.

As the robot is equipped with a wireless connection IEEE 11a, it is possible to distribute an application between the robot and other external computers in order to share the computation load.

4.2 Object modeling

As we have seen previously in chapter 2 and illustrated in Fig. 4.2, the model to build in our experiment consists of an occupancy grid and a set of visual features

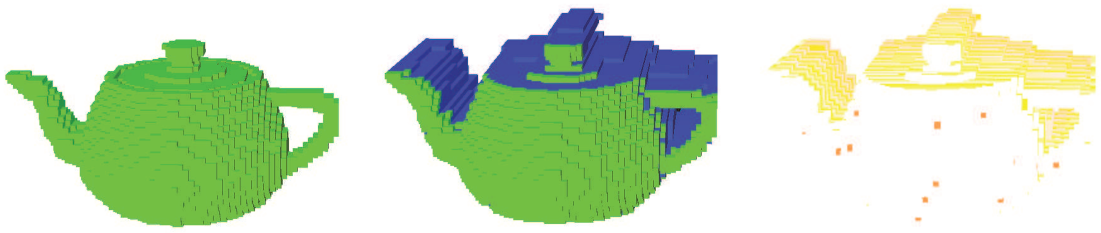


Figure 4.2: Model of a teapot using an occupancy grid. *left*: original 3D object. *center*: model obtained after one perception. perceived voxels are displayed in green and unknown space is displayed in blue. *right*: model constructed for fast visibility estimation. Simulated visual features on the object surface are displayed in orange and the unknown voxels are displayed in yellowish colors.

whose 3D coordinates in the object coordinate system are available. The choice of the model was based on previous works in the laboratory on the treasure hunting project [O. 2008] which makes use of both visual feature processing and occupancy grid. As this PhD work is part of this project, our model is meant to be integrated in a coherent manner with the existing works.

Similarly to some existing approaches to the problem of environment exploration [Saidi 2007a], an occupancy grid is used to quantify the amount of space that needs to be perceived in order to build the object model. This quantification is made possible by giving a specific status value to each voxel amongst *Known*, *Unknown* and *Empty*. At the end of the reconstruction process, the grid provides a rough 3D model of the object shape.

The detection of visual features on the object surface serves two purposes: (i) to have an additional way to retrieve the robot pose relatively to the object pose, and (ii) to provide a quick way to detect and recognize the object. Indeed, once the object model is completed and we have a set of features distributed on the object surface, this set can be used afterwards in order to find the object when it is placed at some unknown location in different environments [Stasse 2007].

4.2.1 Occupancy grid

4.2.1.1 Initialization

An occupancy grid has three main characteristics: size, position in the world, and the resolution. In this work, we consider that all of these characteristics are given as an input of the object modeling process. We rely on the hypothesis that, as our goal is to model an unknown object in a known environment, the object has already been detected and isolated from the rest of the environment so that an approximation of its size and position is available.

The resolution of the grid can be directly deduced from the object size and the accuracy of the model we want to obtain. We must keep in mind that the maximum accuracy we can get depends also on the precision of our stereo matching as well as on the robustness of the registration of new 3D surfaces of the object with the current model.

4.2.1.2 Depth map acquisition process

Each time the robot reaches a new posture in front of the object, the grid is updated using a depth map obtained from the stereo cameras embedded in the robot head. The update process relies on a specific implementation of the space carving algorithm that is presented in section 4.2.1.4. This algorithm requires, as an input, a conservative depth map of the object, i.e. each pixel of the depth map is closer or at the same distance to the camera viewpoint than the corresponding points on the surface of the real object.

In order to obtain this depth map, several successive steps, illustrated in Fig. 4.3, are required:

1. Acquisition of left and right images.
2. Rectification of both images.
3. Creation of the disparity map.
4. Filtering of the disparity map.
5. Creation of the depth map.

For vision processing, we use a pinhole camera model. To perform the rectification step, we need the matrix $\mathbf{I}_c$ of intrinsic parameters of the camera as well as the matrix $\mathbf{E}_c$ of extrinsic parameters of the stereo rig:

$$\mathbf{I}_c = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{E}_c = (\mathbf{R}|\mathbf{t})$$

f_x and f_y are the focal lengths expressed in pixel. c_x and c_y are the coordinates of the principal point in pixel, which, in practice, is slightly off the image center, depending on the quality of the camera. The joint rotation-translation matrix $\mathbf{E}_c$ represents the pose difference between the first and the second camera in the stereo rig. Additional parameters are computed to modelize the radial and tangential distortions of the camera lens.

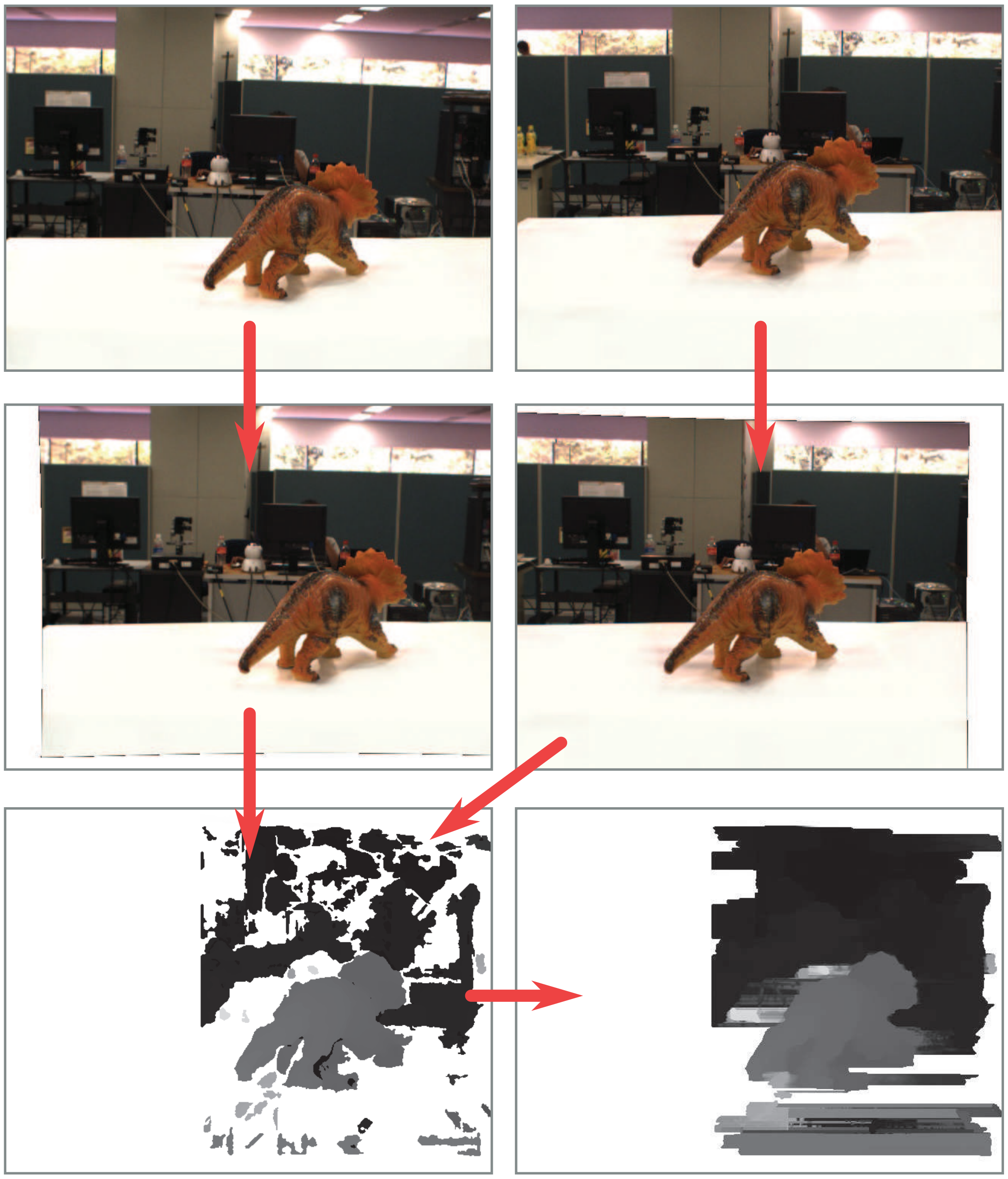


Figure 4.3: Steps to get a disparity map for space carving. *top row*: original images. *center row*: rectified images. *bottom-left*: disparity map obtained with OpenCV Semi-Global Block Matching algorithm. *bottom-right*: filtered disparity map. the white parts have unknown values.

All these parameters can be computed using a calibration process where several images of a known pattern are taken while moving the pattern in different configurations [Bouguet 2004]. The calibration process generates the transformation matrices $\mathbf{I}_c$ and $\mathbf{E}_c$ required to correct the image distortions of the cameras and to align the epipolar lines of the stereo images.

The rectified images are then used in the stereo matching algorithm in order to generate a disparity map. This map encodes the detected motion, in pixels, between parts of the left image and the corresponding parts in the right image. The disparity map is generated using the Semi-Global Block Matching (SGBM) algorithm available in OpenCV, which is a modified version of Hirschmüller algorithm [Hirschmüller 2008].

As some parts of the images cannot be correctly matched, mainly because of occlusions or lack of texture, the corresponding parts of the disparity map have no values. We thus use the post-processing function presented in 4.2.1.3 in order to recover a disparity value for empty parts of the disparity map.

Finally, the depth map is obtained by using the transformation matrices in order to recover the 3D position of the image pixels in the world coordinate system.

4.2.1.3 Disparity map filtering

In practice, most algorithms for stereo matching produce a disparity map where many pixels do not have any value. In cases where the texture on an object is insufficient and there is no model of the object available, the matching of the corresponding surfaces between the stereo images becomes too ambiguous to assign a value. This problem occurs also in cases where some parts of a scene are visible in one view but not in the other. For example, there are self-occlusions in the scene, or the invisible parts belong to subspaces of one of the cameras view frustum that do not overlap with the other camera view frustum.

The lack of values in the disparity map can become problematic when there is a great number of such unknown parts which are sparsely distributed in the scene, especially when they are close to the surface of the object to model. In such cases, the occupancy grid, updated through the space carving algorithm, displays a lot of concavities which hinders the ability of our NBV algorithm to find a pertinent viewpoint. An example of resulting grids is shown in Fig. 4.4.

We thus implemented a simple filtering of the disparity map based on the comparison of image blocks. As the post processing of a disparity map is still an open problem in computer vision, we considered only a simple interpolation method which provides relatively good results in practice. The corresponding algorithm is detailled in Alg. 2. Although this filtering process has its flaws, e.g. we cannot find a value for elements when either a left or right valued element is not present, it gives relatively good results in practice with the settings tested, as is illustrated in Fig. 4.4.

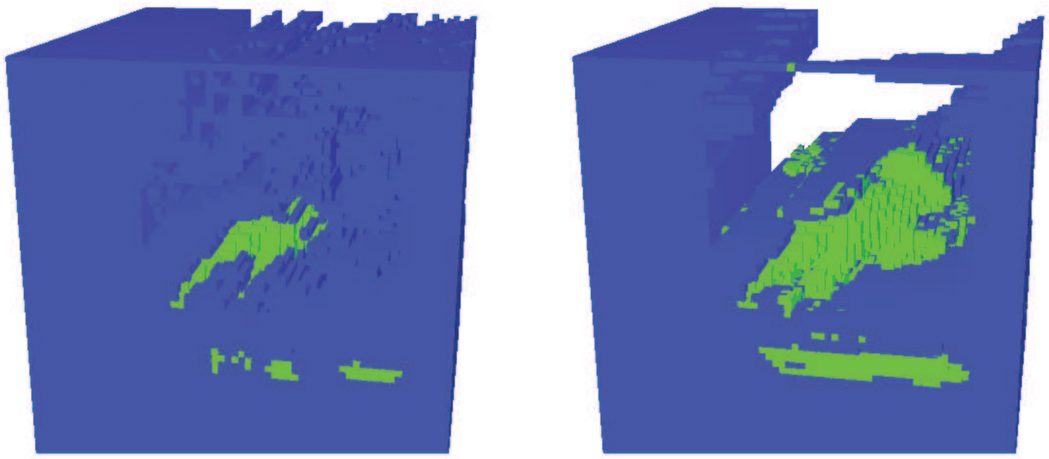


Figure 4.4: Examples of occupancy grid obtained after a space carving process using the disparity maps from Fig. 4.3. *left*: using the unfiltered dispary map. *right*: using the filtered disparity map.

Algorithm 2 Filtering of disparity map

```
 1: for each element $i$ of the disparity map do
 2:     if value is valid then
 3:         check next element $i$
 4:     get related 3x3 image block $I_b$ in left image
 5:     find disparity element $d_l$ on the left which has a value
 6:     if $d_l$ does not exist then
 7:         check next element $i$
 8:     find disparity element $d_r$ on the right which has a value
 9:     if $d_r$ does not exist then
10:         check next element $i$
11:     get 3x3 image block $I_{bl}$ related to $d_l$
12:     get 3x3 image block $I_{br}$ related to $d_r$
13:     compute entrywise 1-norm: $\| I_b - I_{bl} \|_1$ and $\| I_b - I_{br} \|_1$
14:     assign disparity value by interpolating $d_l$ and $d_r$ values, weighted by norms
```

4.2.1.4 Space carving

Space carving is a method to construct 3D models of objects using a set of images and/or depth maps. The process relies on the segmentation of the regions of the space corresponding to the object and the rest of the scene.

A first method [Martin 1983], known as shape-from-silhouette, considers only the images of an object taken from different viewpoints in order to update an occupancy grid. As there is no depth information, for each viewpoint, the contour of the object is segmented apart from the background scene, and all voxels that appear outside of the contour are removed. Though it is an efficient method for mostly convex objects, it may require a lot of viewpoints in order to obtain a relatively accurate shape. Indeed, this method does not consider the texture or shadow information that can be used to decrease the number of views needed to complete the object model. There is also a constraint on the object which should be separable from the background in order to find the contour.

Voxel coloring [Seitz 1999] [Kutulakos 2000], an improvement of the original shape-from-silhouette algorithm, uses the color information on the object surface in order to deal with concavities. Some recent works demonstrate remarquably accurate results by using graph-cut methods [Sinha 2005] [Furukawa 2009]. To be efficient though, the objects to model are assumed to be Lambertian, i.e. the color of the points on the surface of the object does not change depending on the viewpoint.

Another attempt to improve the shape-from-silhouette method by dealing with concavities is the shadow carving method [Savarese 2007]. The algorithm adds a moving light source in the reconstruction setting that creates shadows in the concavities of the object surface. By analyzing the modifications of the shadow depending on the light source position, it is possible to make a conservative estimate of the object surface which is more accurate than the shape-from-silhouette method alone.

Using depth maps to augment the 3D information available for space carving has been exploited in a number of works [Kanehiro 2005] [Yemez 2007]. This approach helps to further reduce the number of viewpoints required to complete the model of an object. In our case, this means we can decrease the number of robot motions necessary for the reconstruction process. Of course, the quality of the results depends directly on the approximation of the depth map.

Our implementation of the depth map-based space carving algorithm is detailed in Alg. 3. This algorithm is conservative as the voxels are considered empty only if most of their volume is closer to the camera position than the values found in the depth map. It is necessary to be cautious when removing voxels as we do not create new voxels even when the information in the depth map is not consistent with the current state of the occupancy grid.

A better way to handle such kind of errors is to use probabilistic space carving methods [Elfes 1989] [Broadhurst 2001] [Nakhaei 2008]. Such approaches though increase the computation time needed to update the grid and to evaluate the view-

Algorithm 3 Depth map based space carving

```
1: sort voxels in the current model from nearest to farthest from camera
2: for each voxel $i$ do
3:     compute Euclidean distance $d_i$ to the camera
4:     project $i$ on camera image plane
5:     for each pixel $j$ in the projection do
6:         compare $d_i$ with depth map value $d_j$ according to voxel size $v_s$
7:         if $d_m - d_i \leq \frac{v_s}{2}$ then
8:             mark voxel as filled
9:             check next voxel $i$
10:    mark voxel as empty
```

points. We thus rely on a fast and simple implementation of the space carving algorithm to update the model in order to allow the realization of preliminary experiments.

4.2.2 Visual features

The detection of visual features on the object surface has been shown to be an efficient way to detect the location of the object in some unknown settings and to compute an estimation of its pose [Forssén 2008]. Most successful object recognition systems rely on robust feature descriptors and invariant region detectors, such as SIFT [Lowe 2004], SURF [Tongphu 2009], or MSER [Obdrzalek 2005] [Forssén 2007].

As mentioned earlier in this chapter, we use the features on the surface of the object to retrieve the camera pose relatively to the object pose. This is possible when enough features are matched between images obtained from different viewpoints. Of course this relies on the practical robustness of the feature detection and matching processes in our particular setting. During our experiments, the robot modifies its distance to the object as well as its relative position and orientation. Thus the feature chosen should be robust to modifications of scale, translation and rotation. Moreover, in practice, as the objects to model are not Lambertian, the features recognition should also be robust to minor illumination changes in order to cope with the modifications of appearance induced by the specular component of the light.

We discarded the MSER features as they are known to be less robust than SIFT or SURF to change of distance. We can mention the ASIFT [Morel 2009a] that were designed to enhance the robustness of feature recognition to the modification of the latitude and longitude angles. The main drawback, though, is the detection and matching computation time of the currently available implementation [Morel 2009b] which is about 20 times longer than for SURF features.

In our current work, the processing of visual features rely on OpenCV implementation of SURF features [Bay 2008] as they are faster to compute than SIFT and keep similar robustness qualities.

4.2.2.1 Features selection

Once the rectified stereo images as well as the corresponding depth map are available, we want to find a set of features on the object surface which can be used for the object detection and recognition. We also want to recover the surface pose relatively to the robot pose and thus we aim at selecting features which are relatively robust to viewpoint change. We tested two methods for feature selection, both based on the stereo images of the object and both relying on a number of assumptions:

- *The size of the features needs to be tested.* Features that have a small size relatively to the object size in the image have low discriminative values. On the other hand, large features include significant parts of the background.

- *The feature must be on the object surface.* Using the stereo rig, we can recover the 3D location of the feature and compare it with the approximative position of the object.

- *Matching features are on the same epipolar line.* Taking advantage of the stereo rectification of the images, each couple of corresponding features in both images should lie on the same line. Moreover the column of the right feature can be deduced using the disparity value related to the feature in the left image.

- *Matching features have similar scale.* As the distance between the cameras in the stereo rig is relatively small compared to their distance to the object, the scale of matching features should be the same.

The first method, presented in Alg. 4 filters the detected features based on their geometrical properties and then matches the remaining candidates based on their descriptor. While this filtering is relatively basic, it is able to remove a great number of false and weak matches. An example of result is illustrated in Fig. 4.5. One concern, though, is that, as the descriptor matching is done late in the process, most features are only tested against a single feature based on their descriptor. Thus the discriminative value of the feature descriptor is not correctly tested.

To answer this concern, we tested a second method, detailed in Alg. 5, that first finds the best match for all the features based on their descriptor, and then filters the matches based on the properties of the features. This second method results in a slightly reduced number of matched features for each viewpoint, as illustrated in Fig. 4.6. Nevertheless, the ratio of correct matches on the total number of matches when comparing images taken from different viewpoints increases substantially.

Algorithm 4 First method for feature selection based on stereo images

```
1: detect features in left and right rectified images.
2: for each feature $i$ in the left image do
3:     compare feature size $s$ with limits $s_{min}$ and $s_{max}$
4:     if $s \leq s_{min} \cup s \geq s_{max}$ then
5:         check next feature $i$
6:     if descriptor response is low, or feature not on the object then
7:         check next feature $i$
8:     matching feature $m_i \leftarrow dummy$
9:     best match score $m_{score} \leftarrow dummy$
10:    second best match score $m_{2score} \leftarrow dummy$
11:    for each feature $j$ in the right image do
12:        if not on the same epipolar line as $i$ then
13:            check next feature $j$
14:        if disparity value of $i$ is different than the distance between the location of
           $i$ and $j$ in the images then
15:            check next feature $j$
16:        if scale difference between $i$ and $j$ is large then
17:            check next feature $j$
18:        compute Euclidean distance $j_{score}$ between $i$ and $j$ descriptors
19:        if $j_{score} < m_{score}$ then
20:            $m_i \leftarrow j$
21:            $m_{2score} \leftarrow m_{score}$
22:            $m_{score} \leftarrow j_{score}$
23:        else if $j_{score} < m_{2score}$ then
24:            $m_{2score} \leftarrow j_{score}$
25:    if $m_{score}$ is below a threshold then
26:        check next feature $i$
27:    if $\frac{m_{score}}{m_{2score}} \geq 0.5$ then
28:        check next feature $i$
29:    save $i$
```

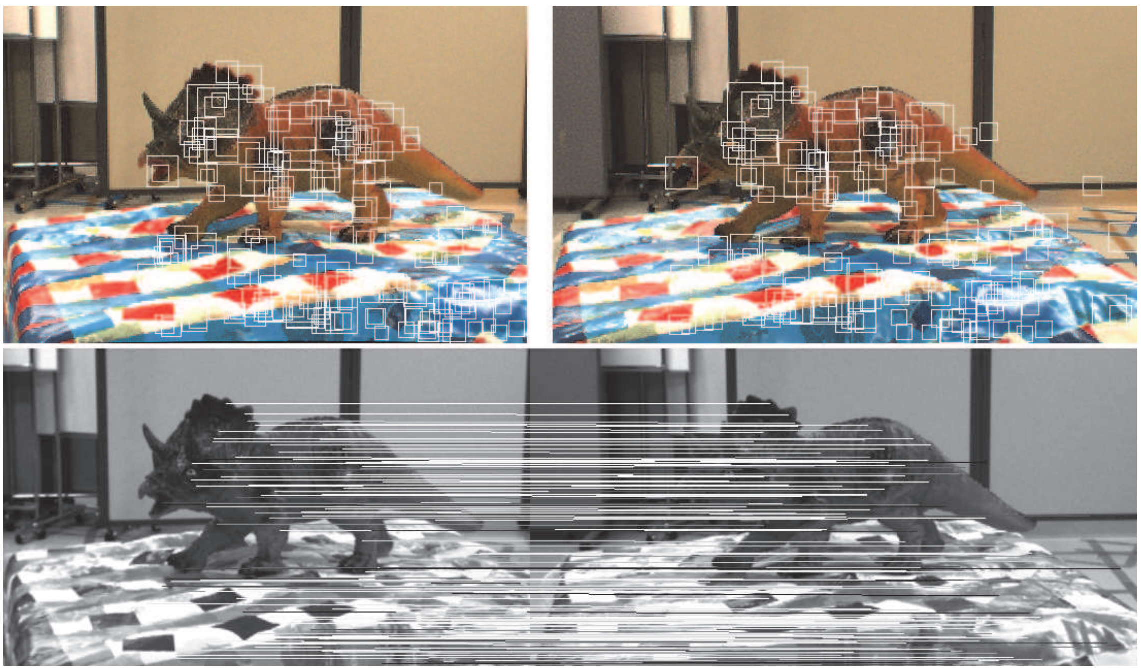


Figure 4.5: Example of features matching on rectified stereo images using SURF. 334 features are matched.

Algorithm 5 Second method for feature selection based on stereo images

```
1: detect features in left and right rectified images.
2: for each feature $i$ in the left image do
3:     find feature $m_i$ which matches $i$ the best, based on descriptors Euclidean distance
4:     if $i$ descriptor response is low then
5:         check next feature $i$
6:     if feature $i$ distance from camera is different than object distance then
7:         check next feature $i$
8:     if $i$ and $m_i$ not on the same epipolar line then
9:         check next feature $i$
10:    if disparity value of $i$ is different than the distance between the location of $i$ and $m_i$ in the images then
11:        check next feature $i$
12:    if scale difference between $i$ and $m_i$ is large then
13:        check next feature $i$
14:    save $i$
```

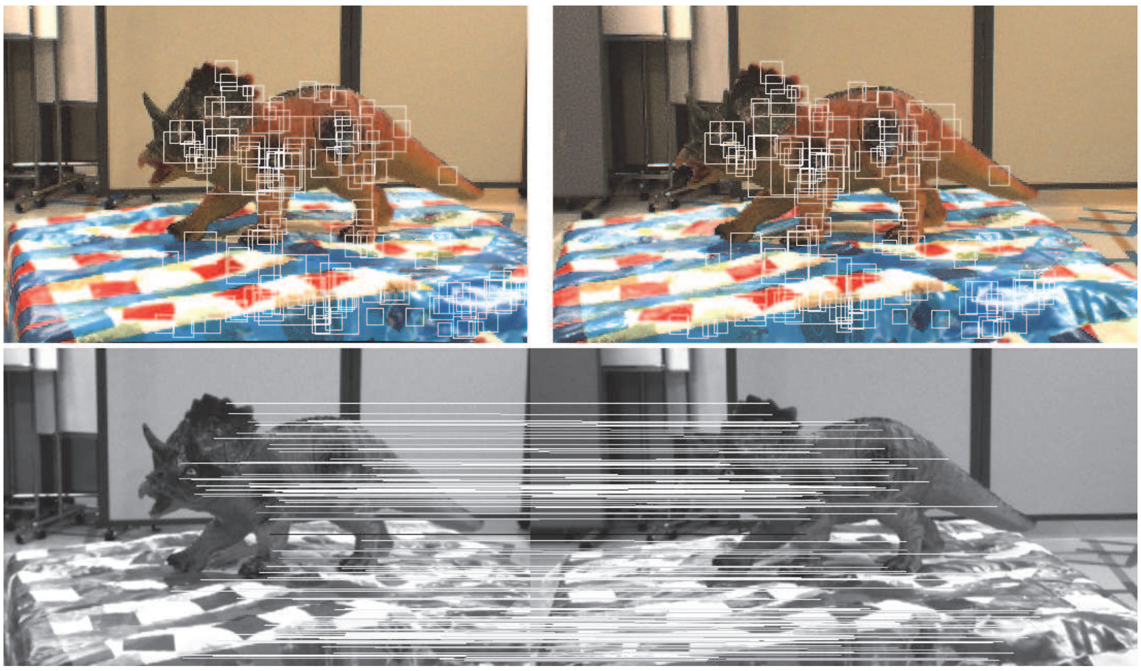


Figure 4.6: Example of features matching on rectified stereo images using the second method. 286 features are matched.

4.2.2.2 Matching features from different viewpoints

In order to update the occupancy grid correctly when a new depth map is obtained, we need to ensure that our estimation of the camera pose relatively to the object is correct. The first two methods presented in section 4.3 aim at providing a rough estimation of the robot pose by using a known model or a feature map of the environment. We then want to use the features detected on the object surface from previous viewpoints in order to refine the pose estimation. To achieve such a result, we need to detect, amongst all features that are visible from the current viewpoint, which features correspond to already perceived ones from previous viewpoints. Once we have a set of matched features, we can recover the pose modification between the two viewpoints using a bundle adjustment method [Brown 1976] [Triggs 2000].

The matching of features between different viewpoints is based on the features descriptor comparison as well as on their relative geometric relations, as detailed in Alg. 6. The algorithm above can give exploitable results if the number of false matches is limited compared to the number of correct matches. Some false matches may remain but, by using the bundle adjustment method presented in section 4.3.3, the recovered pose computation can cope with them. An example of viewpoint matching is presented in Fig. 4.7.

For this matching process to be reliable, we note that the difference of orientation between the current viewpoint and the previous ones should be limited. The limits

Algorithm 6 Matching features from different viewpoints

```
 1: detect features in left images of the two viewpoints.
 2: for each feature $i$ from viewpoint 1 do
 3:    find feature $j$ which matches $i$ the best, based on the descriptors Euclidean
       distance
 4:    find feature which is the second best match to $i$
 5:    if match score below a threshold then
 6:       check next feature $i$
 7:    if scores of first two best matches are close then
 8:       check next feature $i$
 9:    store $i$ and $j$ in list of matches $m_{12}$
10: perform similar matching tests for each feature from viewpoint 2 and construct
    list of matches $m_{21}$
11: for each match $(i, j)$ in $m_{12}$ do
12:    find $j$ in list $m_{21}$ and check best match $k$
13:    if $i \neq k$ then
14:       remove match from $m_{12}$ and $m_{21}$
15: for each viewpoint left image do
16:    compute Euclidean distance between each pair of remaining features
17: for each match $(i, j)$ do
18:    check Euclidean distances between $i$ and all remaining features in the image
       is coherent with the Euclidean distances between $j$ and all remaining features
       in the other image.
19: save a fixed number of the first best matches
```

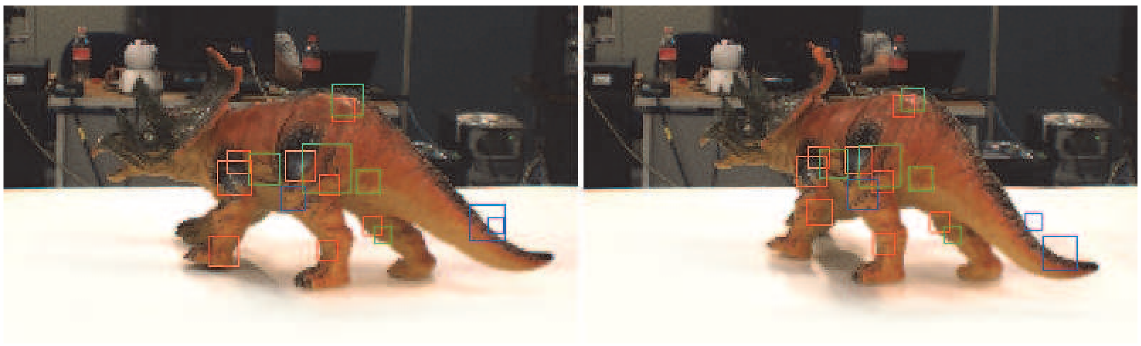


Figure 4.7: Example of features matching on different viewpoint images. The features have different color in order to display more clearly the matches. Couples of matched features have the same color: either red, green or blue. The object has been rotated by approximately 20 degrees and the distance to the camera is the same. In this case, there are 4 false matches on the 16 couples in total.

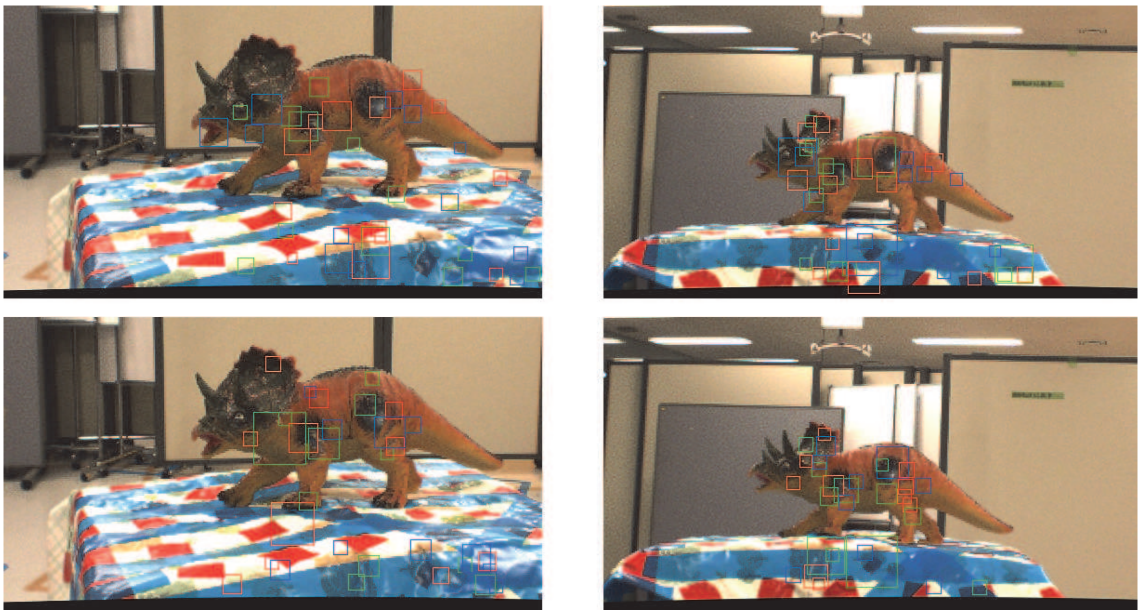


Figure 4.8: Example of features matching on different viewpoint images. *top row*: matching using the first method presented in section 4.2.2.1. There are 8 correct matches on a total of 37. *bottom row*: matching using the second method. There are 14 correct matches on a total of 32.

depend on a number of parameters such as the object material reflective properties, the type of feature used, or the shape of the object. The variation of the distance between the camera and the object also has an influence on the matching results. An example of difficult case that we need to handle in our experiments is illustrated in Fig. 4.8, where the camera orientation and position are both modified. Though the second feature selection method presented in section 4.2.2.1 can improve slightly the robustness of the matching process, the detected matches are mostly outliers.

We conducted some experiments to measure the response of our viewpoint matching process depending on the rotation angle between viewpoints. Using SURF features on the triceratops object, we obtain a satisfactory matching of features between two viewpoints when these have an orientation difference between 5 and 30 degrees. The results depend greatly on the shape of the perceived object. When the camera is facing a mostly planar surface, e.g. the side of the dinosaur, the features can be correctly matched through a larger pose modification of the viewpoint. On the opposite, the viewpoint modification that can produce a good matching score is severly limited when the camera faces the head or the tail of the triceratops.

4.3 Robot pose estimation based on vision

It is critical for our application to have a precise pose estimation of the robot cameras relatively to the object in order to update our object model efficiently. Small discrepancies between the real and estimated camera pose can lead to great variations in the results of the space carving algorithm that may hinders the usability of our model to find a pertinent robot pose.

The problem we must face, though, is that when the robot moves toward a computed position in the environment based on odometry alone, there is a difference both in position and orientation between the computed and the effectively reached pose. The magnitude of these differences depends on the length and complexity of the motion. The errors are partly due to the flexibility of the ankle and the friction forces between the robot foot sole and the floor. Our reconstruction application requires the robot to perform several circular motions around the object to model, and such motions are known to result in significant drifts. The robot pose errors are particularly important in our case as we rely on the narrow-angle cameras with a vertical field of view of 25 degrees to analyze the object. Minor errors in the robot orientation can result in the partial or even complete disappearance of the object from the camera images. Thus this section presents our investigation of three different methods to retrieve autonomously the robot position and orientation using visual features in the environment and on the object to model.

By using a model of the environment and some visual features on the object surface, we can track the features perceived when the robot tries to follow a computed trajectory. These features are used to recognize the robot surroundings and localize the robot position with respect to the environment. This allows us to correct the sequence of steps before its completion if the robot moves out from the computed trajectory. As the object to model is supposed to be fixed in the environment, we can compute the object pose relatively to the robot pose when the walking motion is finished. The robot pose, i.e. head and chest orientation, can then be modified to ensure that the object is included in both cameras field of view.

The main difficulty, in our case, is the online estimation of the robot localization while walking. Even when following a straight trajectory, lateral motions are induced as the robot alternates the foot used for each single support contact phase. Moreover, the contact of a foot on the floor also induces jerk movement at the head which results in blurred images. These particularities represent critical constraints on the kind of feature tracking system that can output exploitable results.

4.3.1 Monocular SLAM

A first method relies on the preliminary detection, in the environment, of a set of visual features that can be relatively robustly recognized and tracked in real time. The goal is to build a feature map of the known environment which can be used

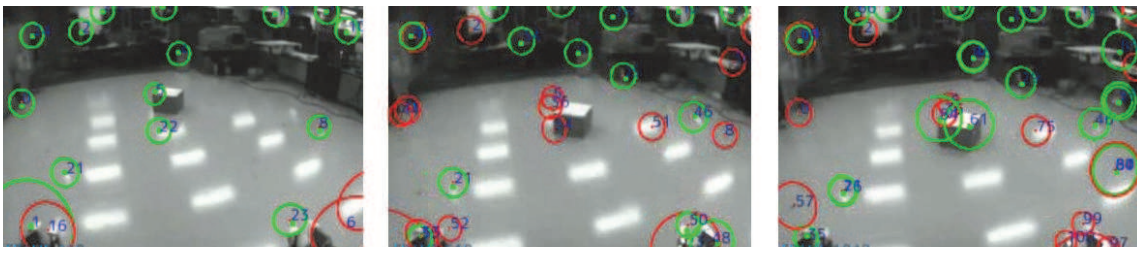


Figure 4.9: Example of monocular SLAM results. Features robustly tracked are displayed as green circles. Red circles displays the features which tracking is lost. The images are part of a sequence taken as the robot walks.

to localize the robot. A similar topic is still actively investigated in the SLAM (Simultaneous Localization And Mapping) [Davison 1998] research field. As the name suggests, a SLAM system aims at building a map of an unknown environment while simultaneously localizing the position of the perception sensor. Vision-based monocular SLAM considers the special case where a single camera is used to execute the mapping.

Though, in SLAM, the mapping and localization are updated in parallel, we use the method in a slightly different way. Before the modeling experiment, we first execute a number of trajectories in the environment until we get (i) a relatively good distribution of features covering most of the space, and (ii) a robust localization of the robot. During the modeling experiment, we then use the constructed feature map to mainly localize the robot. When the robot is placed in some random pose, its position and orientation can be retrieved based on the features visible from there.

In this work, we take advantage of the software component for monocular SLAM [Chekhlov 2006] [Chekhlov 2007] developed by the Computer Vision Group of the University of Bristol [Calway 2007]. This component relies on a novel SIFT-like feature descriptor and a scale prediction scheme in order to overcome two limitations of usual real-time monocular SLAM algorithms which are (i) the restricted range of views over which they can operate, and (ii) the lack of robustness against erratic camera motion or visual occlusions. These properties are particularly adapted for our case where the camera is embedded in a humanoid robot which induces sway motions when moving. Thus we choose the Bristol method over our previously used monocular SLAM [Davison 2007] which relied on Shi-Tomasi features [Shi 1994], less robusts to viewpoint changes. A sample of feature tracking results is illustrated in Fig. 4.9.

The integration and preliminary tests of this method were executed mainly by Clement Petit, a master student at INSA Lyon. The first tests gave results which were not reliable enough to localize the robot in the environment. One issue that is yet to be solved is the robust recognition and tracking of landmarks in our experimental environment when the viewpoint changes. For example, the reflection of the lights on the floor appears as valid candidates for tracking but their position changes depending on the camera viewpoint. This is apparent in Fig. 4.9 with the features 8, 21, 22, 51 and 75. Another issue, inherent to many methods of monocular SLAM, is the problem of scale where the distance between the camera and the features cannot be reliably recovered without using some information about the camera motion. There is also a precision problem on the location of the landmarks where the uncertainty grows depending on the distance between the camera and the landmarks. We note that a recent work by Stradat *et al.*[Strasdat 2010] aims at solving such problems.

Possible solutions to enhance the robustness of the algorithm include the integration of (i) the robot odometry to have an estimation of the camera motion, and (ii), for known environments, the 3D coordinates of some of the landmarks in the environment. The implementation of these solutions relies on EKF-based methods to correctly identify the noise parameters for the measurements and the process. Practically it is quite difficult to tune correctly those parameters when dealing with several measurements source. The behavior of the system can be quite sensitive to the choice of those values and thus it is still an on-going work.

4.3.1.1 Parallel Tracking And Mapping

Another SLAM approach, which is being investigated in our laboratory, relies on the Parallel Tracking And Mapping (PTAM) framework [Klein 2007]. The PTAM system makes use of multi threading to split the tracking and mapping processes into two separate tasks. The first task estimates the camera motion by tracking a set of known 3D points. The second task produces a 3D map of features from the video stream using bundle adjustment. Preliminary tests of this framework are performed by Sebastien Druon, a visiting researcher from LIRMM. As PTAM has been specifically designed for Augmented Reality tasks in a small space, further tests are needed to ensure its usability in our setting for our requirements.

4.3.2 Tracking and virtual visual servoing

In parallel to the Monocular SLAM approach, we tested a second method to localize the robot in its environment that relies on the use of a geometrical model of the environment. This approach has been already used by other researchers with HRP-2 [Michel 2008]. In this work, we take advantage of the ViSP (Visual Servoing Platform) software [Lagadic 2000] developed by the Lagadic group in INRIA Rennes [Marchand 1999] [Marchand 2005].

In an initialization phase, an image is grabbed when the robot is standing in a static pose. The model initial pose relatively to the robot is manually set by matching points in the image with points in the 3D model. The image stream of the camera is then analyzed to match the visual features, such as line segments, with

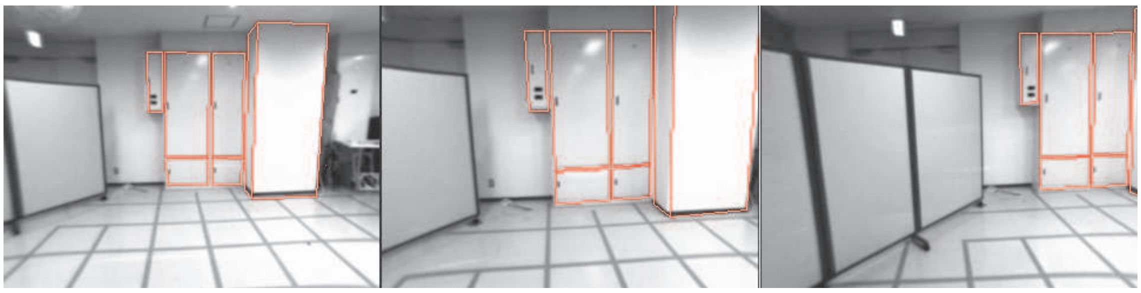


Figure 4.10: Example of visual tracking of a geometric model. The model tracked is composed of the electric panel and the part of the wall highlighted in red. The images are part of a sequence taken as the robot walks.

the features of the virtual model and recover the modification of the model pose.

4.3.2.1 Tracking tests

To track the environment, two approaches are being tested: (i) track a single virtual model of the entire environment, or (ii) track different smaller models of objects in the environment depending on the robot location and orientation in the environment. Preliminary tests and comparison of the two approaches were conducted by Claire Dune, a JSPS postdoc researcher in our laboratory, Thomas Moulard, a PhD student at LAAS, and Stephane Embarki, a master student at IFMA. They used the video sequence from which images are presented in Fig. 4.10. They checked the tracking results for two models: (i) the electric panel and wall part displayed in the figure, and (ii) a model that includes the previous one plus the panels displayed on the left part of the image and the lines on the floor. The bigger model resulted in the loss of the tracker in the middle of the motion while the smaller model could be robustly tracked during the entire sequence. The tracking problem with the bigger model can have different causes that need to be further investigated. We need to consider (i) the precision errors between the constructed model and the real environment, (ii) the precision of the camera calibration, and (iii) the proper parameterization of the ViSP program to handle models including objects at different depths.

4.3.2.2 Future works

Though the second approach appears to be more robust, we need to handle the problem of automatic initialization of models which are not visible at the initialization phase. In this work, we consider only static objects in the scene as models to be tracked and, thus, their relative position is known beforehand. The problem is then to have a tracking of the models which is robust and precise enough to enable the automatic initialization of a new tracker when another modeled object enters the visual field of the camera.

Once we have a reliable tracking of the visible model of the environment in real time, we can use this information to recover the current location of the robot in the environment. The robot position is compared to the odometry information in order to check if the motion of the robot follows precisely the computed trajectory. In the case where the robot is deviating, we can use a reactive pattern generation method to modify the robot future steps and recover from the detected errors [Morisawa 2007].

An alternative approach to online trajectory correction based on vision is to take advantage of the works done by Claire Dune and Andrei Herdt, a PhD student at INRIA Grenoble. Andrei Herdt is developing a reactive pattern generator which can generate steps automatically to react to perturbations of the CoM [Herdt 2010]. Claire Dune worked on the realization of a visual task which generates artificial perturbations of the CoM in order to guide the robot toward a desired pose. The trajectory of the robot in a known environment can then be controlled by the successive execution of visual tasks that are defined depending on the features visible throughout the trajectory.

4.3.3 Object pose estimation

The last method investigated aims at refining the pose estimation of the robot relatively to the object by using the images of the object itself. When we reach a new viewpoint of the object, we execute a SURF detection and look for some of the features that were detected on the object surface in previous viewpoints, as presented in section 4.2.2.2. In order to recover the pose parameters of the camera with respect to the object, we need at least four correctly matched 3D points. In practice, as there are false candidates amongst the matched features, we use the entire set and rely on the hypothesis that we have a small number of outliers compared to inliers.

4.3.3.1 Estimation based on feature matching

A first basic method performs the camera pose recovery in two steps, using the results of the feature matching process presented in section 4.2.2.2.

1. First we modify the camera position in order to put the two centers of gravity of the matched features at the same 3D position.

2. Then we formulate the recovery of the camera orientation as an optimization problem where the objective is to minimize the distance between the matched features. As the detection of the features location, as well as the matching process, are prone to errors, we cannot compute the result directly. The minimum search is done by using a Levenberg-Marquardt optimization.

This method gives exploitable results when most of the matches are correct and when the 3D coordinates of the features have enough precision. Typically, the estimation process fails when the rotation on the yaw axis of the camera is too large depending on the object surface curvature, resulting in an increasing amount of mismatches of features. The feature detector for SURF appears to be sensitive to viewpoint changes and thus it fails to detect similar patches of the object in such conditions. The estimation process can also fail when the distance between the camera and the object is too large compared to the object size, resulting in a poor precision in the depth estimation of the object surface.

4.3.3.2 Estimation based on point cloud matching

In collaboration with Sebastien Druon, we tested a second method based on the matching of two 3D point clouds corresponding to the object visible surface from two distinct viewpoints. One of the dominant registration methods in the literature for aligning pairs of range images is the Iterative Closest Point (ICP) algorithm [Chen 1991] [Besl 1992]. We used a slightly modified version that also considers the point colors to match the surfaces.

The problem that we encounter with this method is that its results depend greatly on the percentage of overlap between the two surfaces. For small objects like the dinosaur, the algorithm can cope only with small modifications of the camera pose in order to produce exploitable estimations. Practically, such severe restrictions on the camera motion result in a significant increase in the number of required poses to build the object model.

Another inconvenient of this method is the great amount of computation time needed to compute a solution. Typically for two clouds of about 60 000 3D points, the current implementaiton of the algorithm takes about 5 minutes to recover the camera pose.

Though there are a number of other efficient methods for point cloud matching that have been implemented [Rusinkiewicz 2001], we did not have time to further test the results of these algorithms on our problem settings.

4.4 Conclusion

This chapter presented the main vision tools that are at our disposition for the practical realization of the autonomous object modeling experiment. We rely on the stereo rig and the wide-angle camera embedded in the HRP-2 head in order to build a model of the object and localize the robot in its environment.

The stereo rig is used to capture the visible 3D surface of the object at different viewpoints in order to obtain a rough 3D model by relying on algorithms such as stereo matching and space carving. Images from the stereo rig are also analyzed to detect SURF features on the object surface. The aim is to obtain a set of 2D features, covering the object entire surface, which can be used to detect the object and recognize its pose relatively to the camera pose.

Our experiments highlighted the difficulty of the robot pose estimation based on visual cues. Monocular SLAM methods have to deal with the scale uncertainty which depends on the magnitude of the camera motion relatively to the distance and size of the landmarks. Though the algorithm developed by the Computer Vision group of the University of Bristol is designed to increase the robustness of the SLAM against such limitations, in our first tests, we could not achieve a precise enough localization for our purpose in our particular setting.

We could obtain better results through the visual tracking approach which estimates in real-time the pose of an object based on the matching of its geometrical features with its virtual 3D model. Experiments validated the robust tracking of one object in the environment but some points need to be addressed in order to use the tracker in our modeling application. As the object tracked may not be visible during the entire experiment, we need to consider either the tracking of the environment entire model, or the successive tracking of smaller models. Some works are still required to test the usability of these two approaches. We note that, in order to be used efficiently, the visual tracking should consider the possible occlusions of the tracked objects by some obstacles. Visual tasks should then be determined in order to avoid such occlusions depending on the robot trajectory and the environment settings.

Relying on the features detected on the object surface, we tested the object pose estimation based on the matching of features between the current view and the previous viewpoints. It appears that the matching is efficient only when there are relatively small modifications of the viewpoint distance and yaw or pitch orientations. More robust types of feature would be required.

Overall the main problem that still needs to be addressed for our modeling experiments is the precise estimation of the pose of the object relatively to the robot pose. This is critical for the valid registration of new information in the current object model, and the validity of the model is critical for the pertinence of the viewpoint and postures generated in order to complete a useful model. The methods that have been tested so far are not precise and/or robust enough in order to be used on the humanoid robot. Thus more tests and research are required to validate our global NBV approach experimentally.

Chapter 5

Practical realization of the modeling experiment with HRP-2

Contents

5.1 Autonomous modeling application

The work developed in this thesis aims at allowing a robot to move autonomously around an object in order to modelize it using the stereo rig embedded in its head. So far, such application has not been demonstrated with a humanoid robot. A similar functionality has been implemented using mobile robots with much less degrees of freedom in the NBV research field but previous works consider controlled environments without obstacles and limitations on the complexity of the shape of the object to model. As noted in the first chapter, different software modules are used to make this application possible on a HRP-2 robot. This chapter details how these modules are integrated together with the original algorithms introduced in this thesis to realize an autonomous object modeling experiment.

5.1.1 Scenario assumptions

Let us recall the main assumptions that are considered in this work.

Known environment The environment model is considered to be available. This hypothesis can be used to compute collision-free motion trajectories. Moreover, it facilitates the discrimination of the object and the environment. For example, it is possible to filter the range map to keep only the parts related to the object and then analyze the corresponding parts of the color images.

The environment model should also include a set of known visual features as we rely on them in order to correct the robot positioning errors induced by its walking motions.

Known object size and position The second main assumption is that an unknown object has been detected previously and the detection phase can output an approximation of the bounding box position and size for the object. Though this detection step is not addressed in this thesis, it is not considered a critical step of the application; By using the model of the environment, it is possible to compare the perceptions of the environment with the available model in order to analyze discrepancies.

We can remark that the assumption that the object should be included inside a known occupancy grid can be easily removed. It is straightforward to detect parts of the object going out of the grid and add other occupancy grids where necessary. By keeping in memory all perceived range maps with all corresponding view poses, we can launch the space carving algorithms on the aggregation of occupancy grids to obtain an updated model.

Textured object Finally, as we use a stereo rig and aim at exploiting visual features from the object surface, the application requires the object to be well-textured. Moreover, the vision algorithms rely on the assumption that the object surface exhibits Lambertian reflectance.

5.1.2 Modeling process overview

The algorithm begins with the humanoid robot placed initially in front of the detected unknown object. By using a given approximation of the object size and position, we can create a virtual occupancy grid at the proper location in the environment model. The grid is used as the object model that needs to be updated.

The updated object and environment models are then analyzed in order to compute the sequence of postures needed to complete the object reconstruction.

Practically, the algorithm loops through the following states:

1. Observe the object.
    - Place the robot head toward the object.
    - Grab the images of the object using the stereo rig and the wide-angle camera.
2. Put the robot in a standing posture. This is needed as some generated postures can require relative high torques in some joints. For example, for the knee and hip joints when the robot is in a squatting posture. We thus need to minimize the time spent in such configuration.
3. Process the grabbed images (see chapter 4).
    - Use OpenCV implementation of the Semi-Global Block Matching algorithm to get a disparity map of the object surface from the actual viewpoint [Hirschmüller 2008].
    - Filter the disparity map to decrease the noise. The method implemented is detailed in section 4.2.1.3.
    - Compute the depth map from the filtered disparity map.
    - Create a normal vector map using the filtered depth map. The normal map can be used to assign a normal vector to voxels, as required in our first NBV method (chapter 2), or to the features on the object surface, as required in the second NBV method (chapter 3).
    - Detect visual features on the object surface. In the current implementation, we use the SURF descriptors [Bay 2008] as they are considered relatively robust to changes of illumination, scale and rotation. The features are characterized by their descriptor, size and orientation. By relying on the computed depth map, we also add the 3D position and normal vector to the characterization of a feature.
    - Match the features that are currently visible with previously perceived features in order to recover the object pose relatively to the robot pose. As presented in section 4.3.3, we must point out that our feature matching process is not robust enough to handle significant variations of the object appearance induced by the modification of the camera pose. Therefore, further investigations are needed to compute a reliable estimation of the robot pose in the environment.
4. Update the object internal model (see section 4.2).

- Match the visible features with the already known features in order to enhance the precision of the depth map registration with the current internal model. This is a critical step as a wrong estimation of the camera pose when updating the model may result in the deletion of valid voxels which cannot be recovered afterwards. This pose estimation using vision is discussed in more details in section 4.3.
- Space carving operation: remove voxels in the occupancy grid perceived as empty using the filtered depth map. The algorithm is presented in section 4.2.1.4.
- Set labels for newly discovered voxels: perceived voxels as known and occluded voxels as unknown.
- Attach detected features to corresponding voxels. This is done by checking the 3D distance between the voxel and the feature, and compare it with the size of the feature.

5. Generate the next posture (see chapter 3).
    - Use the current grid to generate the next viewpoint for the camera in our Next-Best-View algorithm detailed in chapter 3.
    - If the score of the found viewpoint is below a defined threshold, the algorithm stops and the model is considered complete. We also use another termination criteria which considers the currently found viewpoint score with the score of the last two generated viewpoint. If the variation stays below a defined threshold, i.e. the algorithm has a slow rate of convergence, then the reconstruction is terminated.
    - Use the found viewpoint to generate a whole-body posture with the Posture Generator, as described in 3.3.3.
    - If the posture generator cannot produce a posture using the desired viewpoint, the first step of the Next-Best-View algorithm is launched again with this unreacheable viewpoint included in the list of forbidden poses.
6. Plan the step sequence to get from the current position to the final posture.
7. If the planning fails to compute a step sequence to the desired position, the Next-Best-View algorithm is launched again with this unreachable final pose included in the list of forbidden poses. This is done to ensure that the new found posture will not result in the same walk planning failure.
8. Move to the final posture.
    - Launch the walking motion.

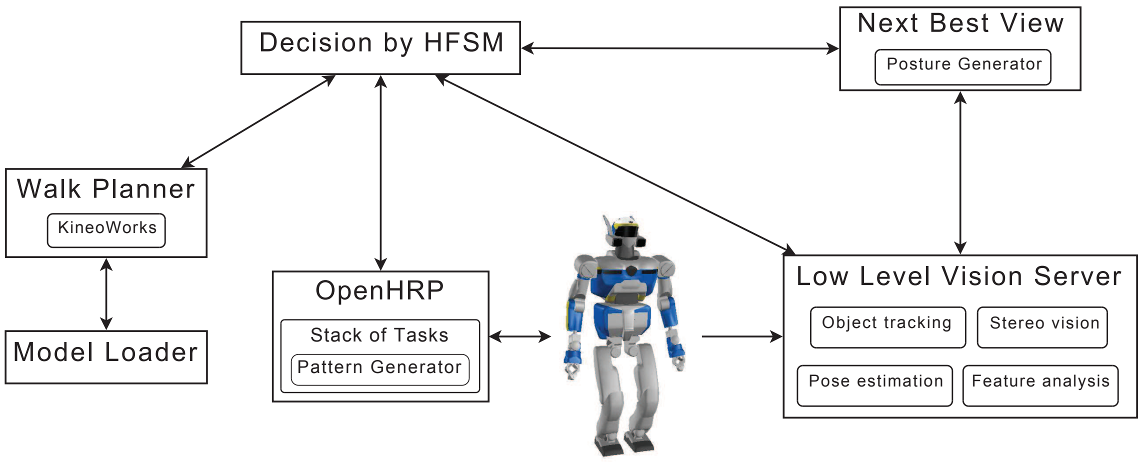


Figure 5.1: Main components involved in the Modeling experiment. The communication between the components is ensured by CORBA.

- *Modify online walking planning using visual cues to correct possible drift.* Although there are on-going works on this subject in our laboratory, they are not mature enough to be included in the application at the time of writing.
- Stop the walking motion.
- Evaluate the robot position and orientation using visual cues.
- Set the posture to the desired one that is generated by our NBV algorithm.

9. Go back to 1.

The key software components used to realize the complete application are presented in more details in the next section.

5.2 Architecture of HRP-2 control for the modeling experiment

The application is controlled by several components which interact with each other, as illustrated by Fig. 5.15. The main high-level operator is the entity labeled as *Decision By HFSM (Hierarchical Finite State Machine)* which synchronizes the execution of some specific components depending on the current sub-task to achieve. The communication between the components is ensured by omniORB, an implementation of the CORBA standard. The components can then be dispatched to different computers and thus the computation load is shared between many CPU and/or GPU.

Though we limit our description to the architecture involved in the object modeling task, we can note that the *Decision By HFSM* component can handle different high level tasks by connecting with other components as well.

In our application, the main interlocutor of *Decision By HFSM* is the *Stack of Tasks* which is used to control the robot motion through OpenHRP. The *Stack of Tasks* monitors the *Pattern Generator* which is in charge of creating the walking motion. The sequence of steps needed to realize the motion comes from the *Walk Planner* component which generates a collision-free trajectory based on the robot approximative pose and the environment model given by the *Model Loader*. The pose to reach is computed by the *Next-Best-View* component which uses the visual information in the range images and a set of features provided by the *Low Level Vision Server* component.

Each component is described in more details in the following sections.

5.2.1 High level control: *Decision By HFSM*

The *Decision by HFSM* is a library developed by Olivier Stasse in the context of the Robot@CWE European project [Stasse 2008]. It aims at providing a simplified interface to create Hierarchical Finite State Machines (HFSM) which act as high level controllers for specific applications.

The original HFSM used in the final demonstrator for the Robot@CWE project synchronizes several components in charge of executing various tasks. For the object modeling application, we re-use the Walk Planner, Stack of Tasks and Low Level Vision Server, and ignore two components related to web connection and teleoperation. Some functionalities are added in the *Decision By HFSM* communication module with the *Stack of Tasks* in order to control the modification of the robot posture. Finally we add the *Next-Best-View* component connected through CORBA to *Decision By HFSM*.

In our application, the components are activated through the different processes that are presented in the statechart in Fig. 5.2 and detailed below. These processes can communicate by putting and retrieving specific data structures from a white board which can be accessed by all actors of the modeling application.

Grab images. Initially, we assume that the robot is facing the unknown object, thus the application starts by grabbing the images from the robot stereo rig. In the current implementation, the images are directly grabbed and stored by the *Next-Best-View* component. The *Decision By HFSM* sends a command to the *Next-Best-View* component in order to initiate the grabbing and then it waits for the completion.

In case of failure, e.g. communication or image retrieval problems, the robot is put to the last half-sitting pose recorded in the white board and the application

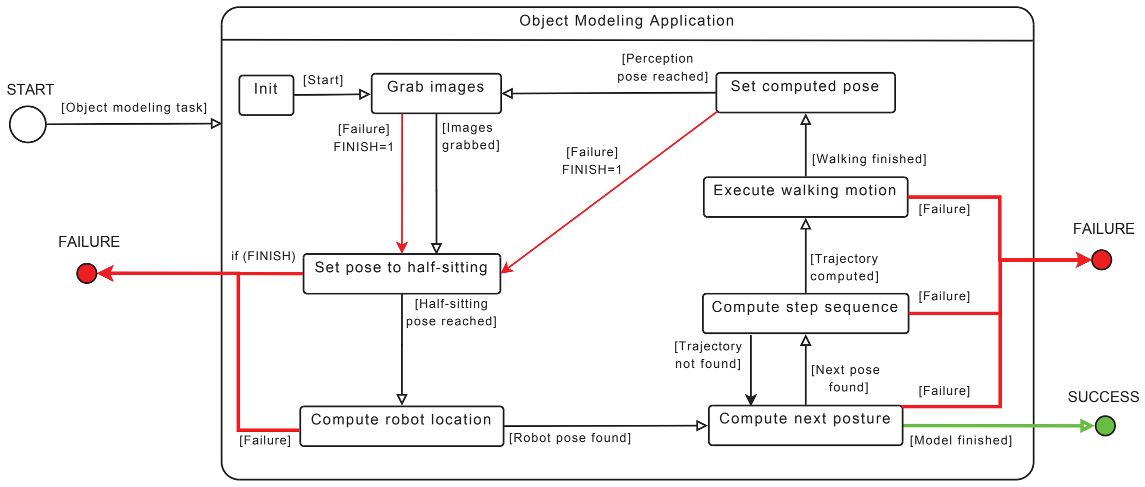


Figure 5.2: Composition of the HFSM for the object modeling task.

exits.

Set pose to half-sitting. The second process puts the robot back to the last half-sitting pose that is recorded in the white board. Initially, as no poses are recorded, nothing is done and the next process is called.

If some poses are present, the *Stack of Tasks* component is called to set the poses of the robot waist, chest, and head, back to the configuration reached at the end of the last walking motion. This is to ensure the proper initialization of the *Walk Planner* for the next walking motion.

Compute robot location. At the time of writing, this process is not yet finalized and thus its implementation is not fixed. This is the component in charge of analyzing the images from the cameras in order to recover the robot position and orientation in the environment. Some of the works presented in section 4.3 are still being tested. Once the pose is recovered, the process sends the corrected camera pose to the *Next-Best-View* component.

Compute next posture. In the current implementation, the *Next-Best-View* component is activated when the stereo images are available and when the robot goes back to a half-sitting posture. The depth map is computed and the SURF feature detection is executed in order to update the object model. The NBV algorithm presented in chapter 3 is then launched in order to output the next robot pose. The *Next-Best-View* component returns the robot 2D position and orientation in the environment as well as the desired joints configuration. These data are stored in the white board so that the *Walk Planner* component can use them.

The *Next-Best-View* component can also return whether the modeling is finished or not. If the model is considered complete, the application exits with a Success report.

In cases of failure, e.g. communication error, the application exits.

Compute step sequence. If the model is not completed, once the computation of the next posture is finished, the *Walk Planner* component retrieves the next robot pose in the environment from the white board. The model of the environment is loaded through the *Model Loader* component. The *Walk Planner* then relies on KineoWorks to generate a sequence of steps for the robot that ensures a collision-free trajectory.

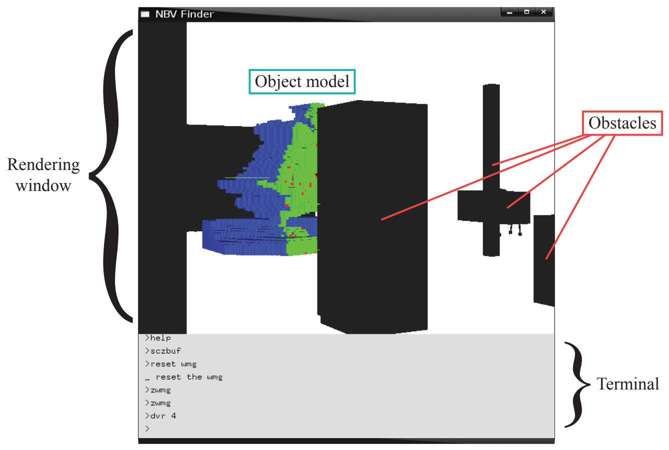
The current implementation of the component considers that the robot start and final postures are set to a default half-sitting posture. But the postures that are computed by the *Next-Best-View* component may place the robot joints, and especially the feet relative pose, in different configurations. We thus need to modify the step sequence by adding a few steps at the beginning and at the end. The steps at the beginning are used to set the feet back to the half-sitting pose. The required steps depends on the last step sequence that has been executed and thus the required motions are communicated through the white board. The steps at the end put the feet in the last configuration computed by the *Next-Best-View* component.

The complete list of steps is then stored in the white board. If the planner cannot output a feasible trajectory, the desired robot pose is set in the list of forbidden poses that is inside the *Next-Best-View* component and the previous process (**Compute next posture**) is called again.

Execute walking motion. This process retrieves the step sequence in the white board and initiates the walking motion using the *Stack of Tasks* component. The *Pattern Generator* loaded inside the *Stack of Tasks* generates the robot motion that is sent to the robot through OpenHRP. *Decision By HFSM* queries the *Stack of Tasks* at regular interval to detect the end of the walking motion.

In cases of failure, the application exits.

In a future version, this process will also include some works that aim at evaluating the robot position and location based on vision, computing the drift in real-time, and modifying the step sequence in order to correct the drift. The real time correction of the step sequence is already available, based on Morisawa *et al.* pattern generator [Morisawa 2007]. It has been implemented and tested on HRP-2 in some previous works [Stasse 2009]. As mentioned in chapter 4, some works on localization are currently investigated by other members of the laboratory. They have not yet demonstrated results robust enough and thus we cannot include them in the application for now.



```
>help
>sczbuf
>reset wmg
_ reset the wmg
>zwmg
>zwmg
>dvr 4
>
```

Figure 5.3: The OpenGL program dedicated to the modeling of a 3D object based on our Next-Best-View search.

Set computed pose. When the walking motion is terminated, this process accesses the white board in order to retrieve the desired joint configuration computed by the *Next-Best-View* component. Then it uses the *Stack of Tasks* component to query the current robot joint configuration and store it in the white board. This current configuration will be recovered by the "**Set pose to half-sitting**" process before realizing another walking motion. The *Stack of Tasks* component is then used again to move the robot joints toward the configuration. *Decision By HFSM* queries the *Stack of Tasks* at regular interval to retrieve the tasks errors and modify the activation of the tasks accordingly.

In case of failure, the robot is put to the last half-sitting pose recorded in the white board and the application exits.

5.2.2 Next posture generation according to visual cues: *Next-Best-View*

This component has been developed in the course of this thesis to help design, test, and improve the works detailed in chapters 2 and 3.

In the context of the modeling application, it takes as input a first estimate of the position and size of the object to model as well as a list of forbidden poses representing the obstacles locations in the environment. The initial model of the object is thus represented by an occupancy grid with a fixed size and position.

At each step of the modeling, when the robot reaches its perception pose, the component is activated by *Decision By HFSM*. It connects to the *Low Level Vision Server*, retrieves the images from the stereo camera and launches their processing (see section 4.2). When the visual processing is finished, the two datas required for the object modeling are available: (i) the depth map representing the visible surface of the object, and (ii) the set of visual features that were detected on the object surface. The current model is updated with the depth map by using a space carving operation in order to remove empty voxels and modify the state of the voxels on the model surface (see section 4.2.1.4).

The updated model is used by our NBV algorithm (see section 3.3), to find a feasible camera pose that can perceive an optimal amount of the object occluded parts while satisfying various constraints.

Then we generate a whole-body posture that ensures that the robot camera is set to the desired pose and that the constraints on the robot body are satisfied. The whole-body posture is currently computed by relying on the functions from the *Posture Generator* library made by Adrien Escande [Escande 2008] (see section 2.1.1). The posture is defined by the CoM position, the waist 6 dof pose and the joints angle. For the walking motion, we also compute the 3 dof pose of each feet.

We note that, Karim Bouryarmane, a PhD student in our laboratory is currently adding functionalities to the *Posture Generator*, for example, the ability to compute postures with contact points on non-planar surfaces. In future works, his implementation could be used to relax some constraints on the camera pose by considering the additional set of postures that can be achieved based on the 3D models of the objects in the environment.

In a more general context, the *Next-Best-View* component is developed as a standalone C++/OpenGL application whose GUI is presented in Fig. 5.3. This program was developed during the course of this thesis to allow the fast prototyping of evaluation functions for the purpose of the NBV search. It offers four modes of interaction:

- The virtual camera can be controlled by using mouse and keyboard actions on the rendering window of the program.
- A large set of commands can be executed through the embedded terminal in the lower part of the program.
- A CORBA server is launched at the program initialization and provides an interface to the commands used by the *Decision By HFSM* component.
- A script can be executed at the program initialization by giving the script filename as an input argument.

The program also includes a CORBA client that is in charge of the communication with the *Low Level Vision Server*. In the current implementation, the *Low Level Vision Server* is just used to retrieve the images from the robot cameras. The processing of the stereo images is done in the *Next-Best-View* component itself, by using the OpenCV library.

For tests purpose, a virtual 3D object can be loaded in the program as an occupancy grid. The object is used to simulate the 3D perception of the stereo rig. We also used this functionality to test complete sequences of motion tasks with HRP-2 while avoiding the problems of recovering the exact pose of the camera relatively to the object. Results of such a test are presented in section 5.3.1.

The program also allows to load the model of the environment as a list of 3D objects. These are considered in our NBV algorithm to compute viewpoints while taking into account collisions and visual occlusions.

5.2.3 Planning walking steps: *Walk Planner*

This component relies on the motion planning software component for the humanoid build upon KineoWorks [Laumond 2006]. It is implemented as an OpenRTM component which communicates with the *Model Loader* component in order to have access to the environment model. It receives the current and desired positions and orientations of the robot from *Decision By HFSM* and give it back a sequence of steps and the reference foot at the start of the sequence. Each step is composed of 3 variables representing the 2D position and the yaw orientation of the next foot relatively to the previous foot. The current implementation of the component generates the step sequence by considering that the starting and ending postures are set as the default half-sitting postures. As the *Posture Generator* can output postures where the feet are placed in various configurations, we need to modify the step sequence, as mentioned in section 5.2.1.

The planner uses Rapidly-exploring Random Trees to find a trajectory between the current posture and the desired position and orientation. The trajectory is composed of a set of specific motion segments which matches the motion capabilities of the humanoid robot. Self-collisions are avoided by imposing some constraints on the curvature of the possible motions. The transitions between any two states are validated by considering a simplified model of a non-holonomic mobile robot. The collision avoidance with the objects in the environment is simplified by using a bounding box around the robot when it is in the half-sitting posture. These characteristics allow the method to compute solutions typically in a few tenths of a second.

Finally, we note a limitation of the current version which is its inability to plan trajectories on non-flat terrains.

5.2.4 Motion control: *Stack of Tasks*

The control of the robot complex motions is ensured by generalized inverse kinematics [Siciliano 1991] using the *Stack of Tasks* component [Mansard 2006]. The *Stack of Tasks* executes several control tasks in parallel with different priorities. It ensures that tasks with lower priorities do not perturb the execution of higher priority tasks.

It includes a number of default tasks that can be used to control some body parts at a higher level of abstraction.

The *Stack of Tasks* is used as a plugin for OpenHRP in order to access the control of the actuators and get sensors feedback from the robot. The execution of the *Stack of Tasks* can be controlled by using scripts sent through a CORBA interface.

From the beginning until the completion of the application, we fix the arms joints of the robot with the hands at the height level of the waist. The arms are not used in this application and thus they are put in a configuration that ensures self-collision avoidance even when the robot assumes squatting stances. As seen in the HFSM presented in section 5.2.1, three main states are achieved by *Stack of Tasks* during the modeling application:

- *Setting the robot to the half-sitting posture.* Before walking, we set the robot in a default posture that ensures stability during the walking motion. Four low-level control tasks are executed simultaneously in order to reach the desired configuration while keeping the feet in their current pose. The first one sets the CoM to the computed 3D position and the second puts the waist in the desired 6D pose. The third and fourth tasks set respectively the two chest joints and the two head joints.

- *Executing the walking motion.* The walking motion is supervised inside the *Stack of Tasks* by the *Pattern Generator* library, presented in section 5.2.5.

- *Setting the robot to a computed pose.* The posture computed by the *Next-Best-View* component is set through the same four control tasks that are used to set the robot to the half-sitting posture.

5.2.5 Walking motion execution: *Pattern Generator*

The *Pattern Generator* computes the robot body reference trajectories used for the walking motion. It is coded as a library called inside the *Stack of Tasks* component. The current algorithm used is an implementation of Morisawa's algorithm [Morisawa 2007]. The component takes the sequence of steps computed by the *Walk Planner* and executes the corresponding walking motions.

The particularity of the algorithm is that it is reactive in the sense that it allows the modification of the step sequence before the walking motion finishes. This functionality can thus be used to correct the possible drifts in the motion of the robot while it is supposed to follow a specific trajectory.

5.2.6 Vision processing: *Low Level Vision Server*

The processing of the robot camera images is done by the *Low Level Vision Server* component. It is implemented as a CORBA server running on hrp2010v, the com-

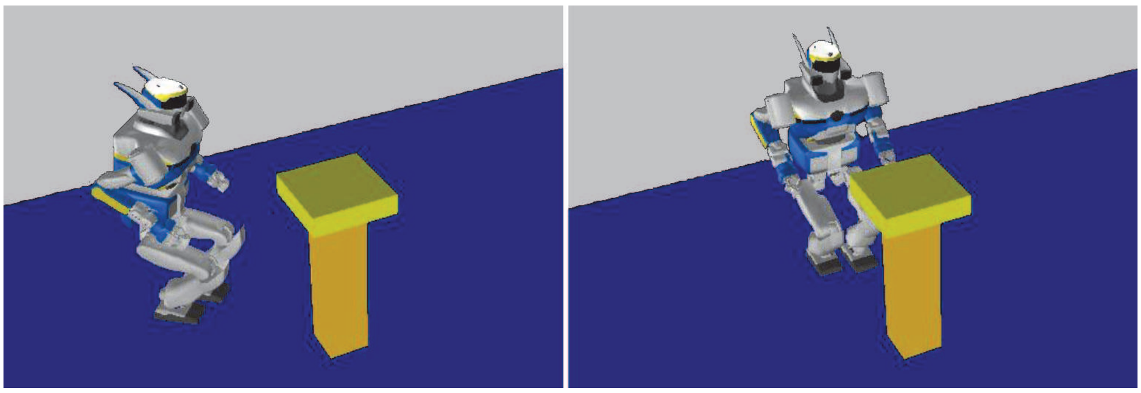


Figure 5.4: Simulation of the modeling experiment with OpenHRP. The application generates successively the robot motions to model an object that is supposed to be on top of the table.

puter dedicated to vision located inside the robot. It relies on the libdc1394 library for grabbing the images from the robot IEEE cameras.

The algorithms used to process the images comes from different open source softwares and libraries dedicated to computer vision such as ViSP and OpenCV. We specifically rely on algorithms for stereo vision, 2D features detection and recognition, SLAM, and visual tracking. The methods that are called for the purposes of our modeling application are presented in greater details in chapter 4.

5.3 Validation of the modeling application

The complete software architecture of the application has been first tested in dynamic simulation with OpenHRP. This first test phase is used to check the stability of the application as well as the safe realization of the motion tasks. A specific attention has been directed toward the test of the torques on the joints of the robot during several simulations. This is necessary as some of the postures generated in our Next-Best-View algorithm are close to the limits of what can be safely achieved by HRP-2. In particular, when the humanoid is set in a squatting posture, as illustrated in Fig. 5.4, the torques increase significantly in the knee and hip joints. Though theses torques remains under the maximum limit of the motors, we need to shorten the time spent in this configuration. Thus, as soon as the computed pose is reached, the images of the cameras are grabbed and the robot is set back to the standing posture before processing the images.

Several tests showed that the application was reliable and, more importantly, safe for the robot in terms of collisions and stress on the motors. The torques obtained remain below the maximum limits and the highest values are only reached for a very limited amount of time.

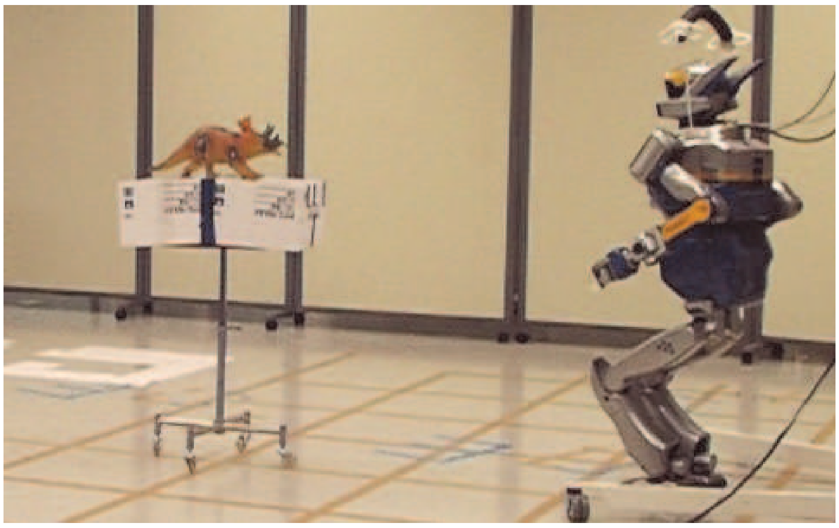


Figure 5.5: Experimental setup for the modelization of a dinosaur toy in a known environment.

5.3.1 Experiments without vision

Following the simulation tests, the application and its software architecture have been verified through experiments with a real HRP-2 robot in a basic setting, as shown in Fig. 5.5. The first experiments of the autonomous modeling application with HRP-2 consider simplified conditions:

- No obstacles in the environment around the object apart from the table.
- No perception of the real object in front of the robot. The object perception is simulated with a 3D model loaded in memory and the virtual camera pose is always considered to be exactly at the computed pose. In real experiments, if the vision processing is not reliable enough to compute the precise camera pose relatively to the object pose, the pertinence of the updated model degrades quickly and becomes unusable after 2 or 3 perceptions.
- No robot localization correction based on vision.

Such approach is used to further test the integration of the software components in real situations. We also want to estimate the magnitude of the drift in the robot trajectory with the kind of walking motions that are generated to reach the computed poses.

An example of the sequence of reached poses is presented in Fig. 5.16, with the corresponding trajectory in Fig. 5.17. Even though the walking motions between two successive poses are relatively short, a significant drift appears. There are thus important differences between the computed poses and the configurations that

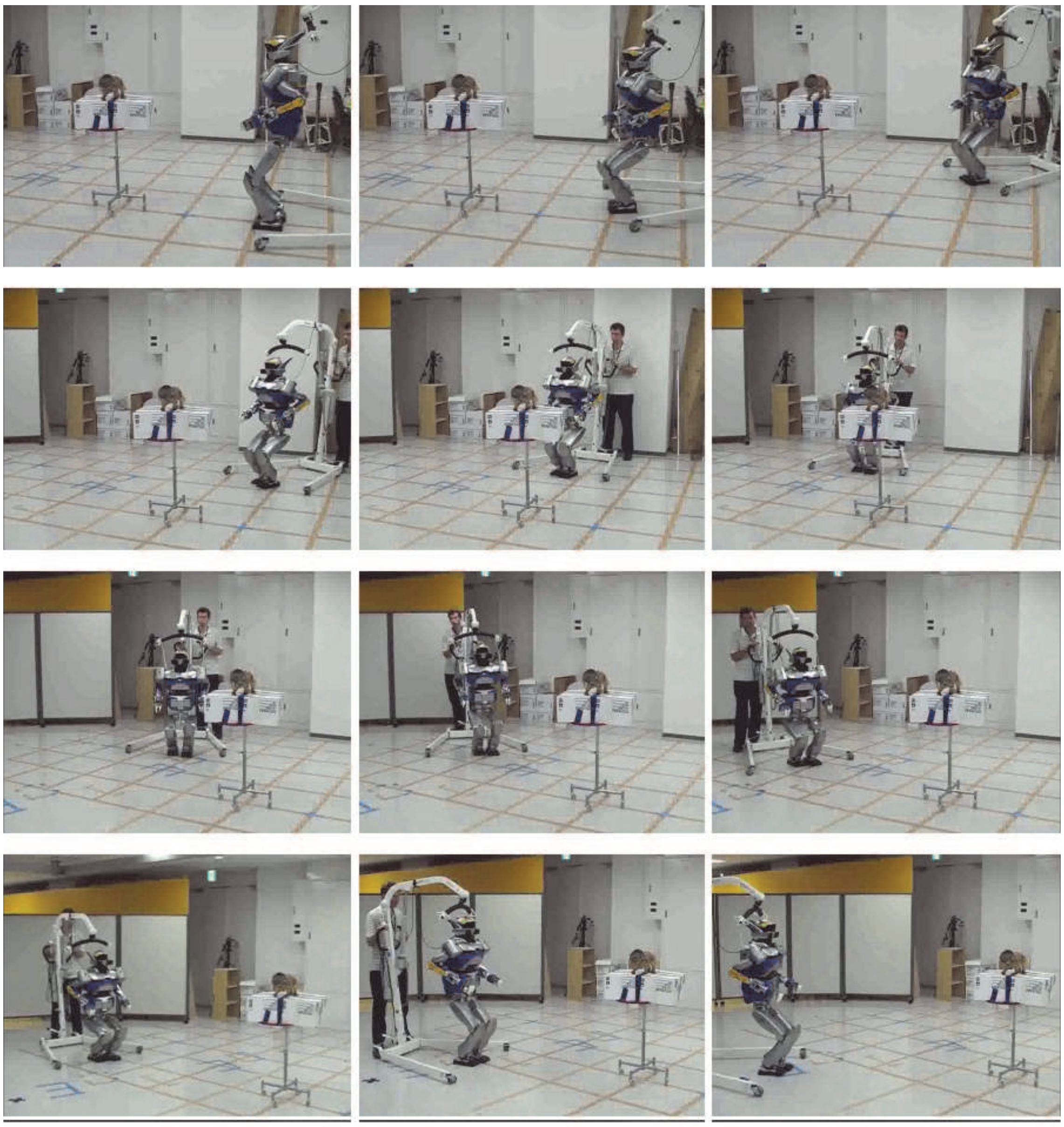


Figure 5.6: Poses reached by HRP-2 during a simulation of object modeling.

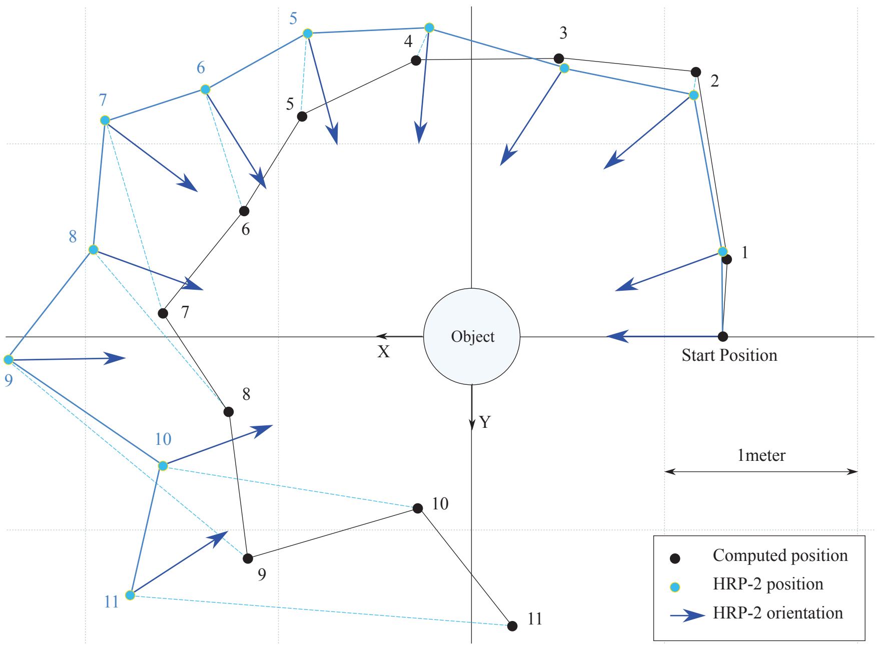


Figure 5.7: Comparison between the trajectory computed and the trajectory executed by HRP-2. The robot position and orientation are not corrected during the experiment. The object perception is simulated with a virtual 3D object and the camera pose is set to the configuration computed by the NBV algorithm.

are actually reached by HRP-2, and these differences increase with the number of motions that are realized.

The object is still visible in most, but not all, images from the stereo rig, and always visible with the wide-angle camera. It is thus possible to use specific vision processing tasks in order to analyze the object and its surroundings in order to retrieve the robot pose relatively to the object.

5.3.2 Experiments including vision

As discussed in chapter 4, a number of vision-related works are being conducted in the laboratory in order to retrieve the robot position and orientation relatively to some reference frame. Monocular SLAM and visual tracking methods have been investigated but the first works on these subjects still need improvements in order to be exploited in our application.

A modeling experiment with vision has been executed where the robot pose estimation is based first on odometry. Then we refine the estimation by using feature matching on the object surface, as presented in section 4.3.3. The problem is then to limit the magnitude of the motion between generated viewpoints sufficiently enough in order to have a robust feature matching. This can be achieved in our Next-Best-View algorithm presented in chapter 3, by using the parameters of the landmark visibility constraint (see section 3.2.1.3).

In the conditions tested, it appears that the feature matching process gives unreliable results even when the robot is moved by as little as 1m. The difficulty of the task has been already highlighted in section 4.2.2.2 and is confirmed when using the robot with our NBV algorithm. Fig. 5.8 illustrates the problem when the drift occurrence results in reached pose that are significantly different than the computed one. In such cases, the object is projected at different part of the occupancy grid. Depending on the implementation of the space carving algorithm, the model becomes unusable, e.g. if it is composed of small patches of the object surface at different locations, and even empty if we remove all voxels perceived as empty and never create any even when they are perceived as occupied.

There are a number of possible solutions to allow the estimation of the pose of the camera based on the object feature matching in real conditions. We can restrict the application to specific objects with large planar surfaces and robust artificial features on their surface, or we can further reduce the distance between two successive viewpoints. But these approaches severely limit the usability of the application. An investigation of the robustness of other visual feature detectors and descriptors than SURF could lead to a better matching process but we lack time to include such a study in this thesis. Moreover, there is a possibility that the drift of the robot can be large enough so that the object is not visible in the camera images. It is thus more advantageous to rely on the environment than on the object for the camera pose estimation.

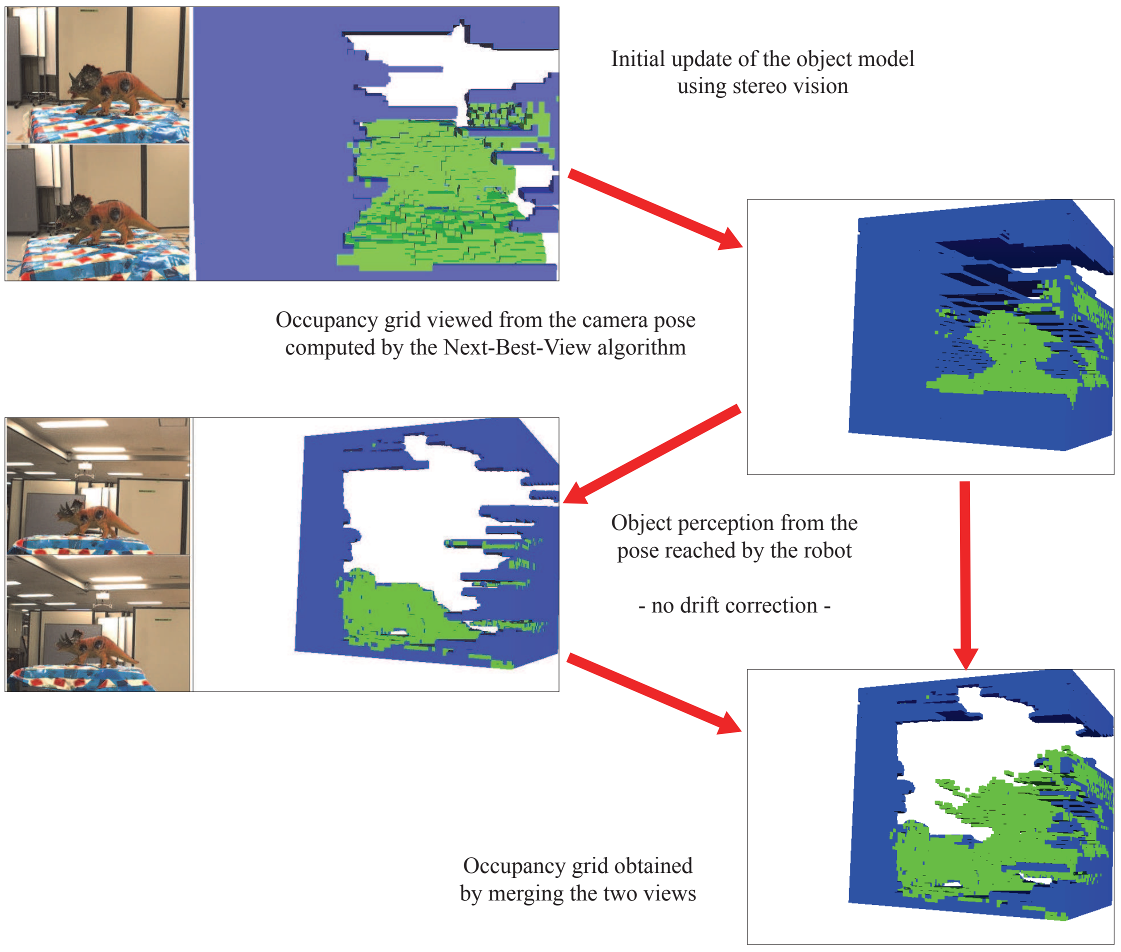

Initial update of the object model
using stereo vision
Occupancy grid viewed from the camera pose
computed by the Next-Best-View algorithm
Object perception from the
pose reached by the robot
- no drift correction -
Occupancy grid obtained
by merging the two views

Figure 5.8: Modeling experiment without drift correction. The projection of the object 3D surface in the occupancy grid is erroneous and thus the object model is unreliable. *top-left*: stereo images obtained from the initial robot posture and the corresponding updated occupancy grid. *center-right*: occupancy grid perceived from the computed Next-Best-View. *center-left*: stereo images obtained from the posture reached after walking without drift corrections. The occupancy grid obtained using the corresponding depth map shows that the object is not located at the desired position. *bottom-right*: merge of the initial and following occupancy grids. Due to the presence of drift during the walking motion, the occupancy grid is not properly positioned relatively to the robot and thus the model is badly updated.

5.4 Conclusion

This chapter presented the software architecture of our modeling application that is based on the specificities of the HRP-2 humanoid robot. The object reconstruction process is implemented as a Hierarchical Finite State Machine where each state is used to interact with one or more independent software components. The execution of most of these components is distributed among different computers on the laboratory internal network by relying on the CORBA middleware. This has the advantages of (i) distributing the computation load on different hardware units and (ii) ensuring the reusability of the components for different tasks as their implementation is not tightly coupled to the code of the current application.

The complete modeling application has been simulated with the dynamic engine of OpenHRP 3. This first test phase validated the complete software architecture by checking the smooth interactions of the components and by ensuring the reliability of the control tasks sent to the robot. It is particularly important to test the stability of the robot and the torque of the joints before experimenting with a real robot.

Real experiments with a HRP-2 were carried out and they highlighted the problem of drift. When the robot moves from its current pose to a specific computed position, the reached pose is erroneous in terms of both position and orientation. The magnitude of the error can vary between few millimeters to several centimeters or more, depending on the nature of the steps realized and the length of the trajectory.

Because of such a drift problem, it is necessary to compute a reliable estimation of the robot pose by using vision cues. The visual features on the object to model proved to be difficult to match robustly when the robot reaches significantly different viewpoints. Robot localization approaches that rely on the environment model have the potential to offer better robustness but further investigations are necessary in order to apply such methods to our problem. In particular, preliminary works on the ViSP software, which performs the visual tracking of polyhedral objects, and the PTAM framework, which allows the localization of a monocular camera by relying on a structure-from-motion method, gave promising results that need to be more thoroughly tested with HRP-2.

Conclusion and Perspectives

The only true voyage of discovery, the only fountain of Eternal Youth, would be not to visit strange lands but to possess other eyes, to behold the universe through the eyes of another, of a hundred others, to behold the hundred universes that each of them beholds, that each of them is;

Marcel Proust

This thesis has attempted to help HRP-2 on its quest to discover the model of an object by providing a way to iteratively compute, and reach, new viewpoints. The autonomous modeling of objects in a realistic environment is a challenging task, especially when it is realized by complex systems such as a humanoid robot. Like many scientific works, much has been learned, much has been done and much more remains to be investigated.

The first focus of this work has been on the proper evaluation of 3D visual information in order to select pertinent viewpoints while satisfying the constraints on the humanoid robot body. This has lead to an automatic posture generation scheme that considers the incremental knowledge about the object of interest.

A first approach formulates this visually-guided posture generation in a coherent manner by implementing an original evaluation function of the visibility of a 3D model. This method relies on the Posture Generator that allows the fast generation of specific postures for HRP-2 by solving an optimization problem based on smooth functions. A new equilibrium constraint has been added to the Posture Generator in order to generate statically stable postures without constraining the feet relative position. This approach performs a local Next-Best-View search based on the 3D surface visibility of the object while satisfying the robot constraints. Though our surface visibility evaluation function has the potential to solve the NBV problem while considering the specificities of the robot body in one coherent step, it has a relatively high computation cost and presents high variations in the gradient that lead to convergence problems. Moreover it is a local method whose convergence is sensitive to local minima created by visual occlusions.

A second approach relies on both global and local optimization methods to enhance the robustness of the NBV search to local minima. The visual model is built

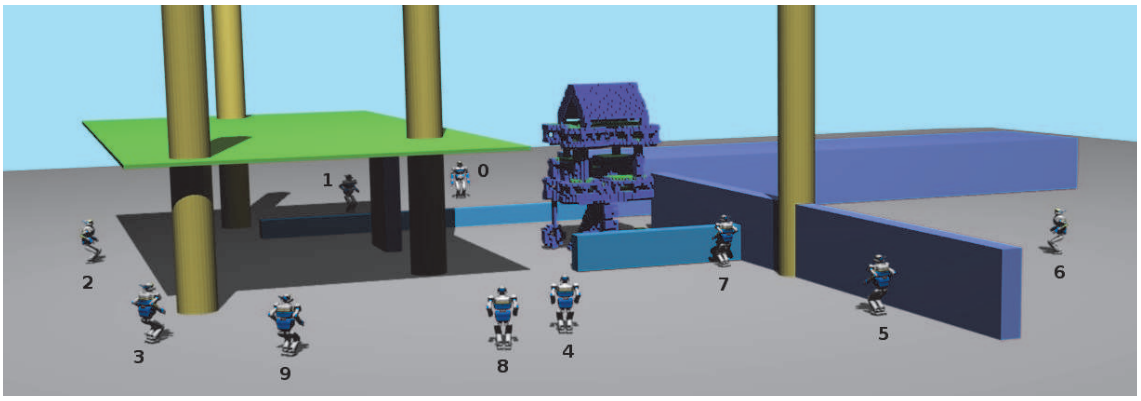


Figure 5.9: Simulation of the modeling experiment for a 3m high treehouse. The environment includes many obstacles that create visual occlusions.

through the repetition of two main processes. First, considering the current knowledge of the object, a preferred viewpoint is deduced in order to reveal occluded volumes of the object while taking into account the constraints related to the embodiment of the vision sensors in the humanoid head. Then a whole robot posture is generated using the desired head pose while satisfying several constraints: static stability, collision avoidance, etc. The first process can deal with visibility evaluations of complex objects in cluttered environments by relying on a global sampling coupled with an iterative local minimization based on BOBYQA [Powell 2009]. The second process is achieved by the Posture Generator, which outputs constrained whole-body postures. This approach was validated in simulation by successfully building models of various non-convex objects. Though the separation of the posture generation process in two steps induces possible convergence problems, a solution has been implemented that put any unreachable configuration in a list of poses to avoid. The first step is then launched again in order to generate a new solution.

This NBV approach is general enough to allow the modeling of objects in difficult conditions, as illustrated in Fig. 5.9. We can reconstruct objects of various sizes even when many obstacles and visual occlusions are present. Nevertheless, there are numerous ways to further improve the algorithm. For example, the reliability can be enhanced by including the motion cost inside the viewpoint evaluation. Our NBV method could then be also applied to the problem of environment exploration. Another interesting development is to consider probabilistic methods to represent and update the object 3D model. Similarly to a human, the ability to infer the shape of occluded parts from the visible surface can be of great value to further speed up the modeling process.

The second focus of this work has been on the evaluation of various state-of-the-art research works in order to realize practical experiments with a HRP-2 robot.

A system as complex as a humanoid robot relies on robust components to execute different tasks while ensuring stability and self-collision avoidance. These components are executed on different computers and interact with each other through the CORBA middleware. This ensures the distribution of the computation load on different hardware which helps improve the reactivity of the system for complex processes.

The object modeling application is implemented as a Hierarchical Finite State Machine. Each state communicates with the different components (i.e. CORBA servers) in order to realize specific low-level tasks for the humanoid robot such as: vision processing, Next-Best-Posture generation, motion planning, and motion control. Apart from the Next-Best-Posture generation component, the application reuses libraries that are developed by collaborating researchers, and open softwares that are freely available for academic purposes.

Images from three of the cameras embedded inside the robot head are processed for the vision tasks of the application. The stereo rig is used to recover the 3D shape of the object and to detect the visual features on its surface. The processing of the stereo images is realized by recent algorithms available through the OpenCV library. The wide angle camera is intended to help localize the robot. Works in the fields of SLAM and visual tracking are being carried on, in our laboratory, in order to compute a reliable estimation of the robot pose by using visual cues.

Although the current architecture has been validated in simulation and through experiments, a robust localization of the robot in the environment is still required in order to realize the object modeling application. Real experiments with a HRP-2 highlighted the problem of drift that occurs in most walking motions. The robot is thus unable to reach the computed postures and the model cannot be correctly updated. Some of the works on the robot localization have shown promising results but further investigations are required to ensure a robust estimation during the robot motions. Once such an estimation is possible, the robot walking motion can be modified in real-time to correct the drift and reach the computed pose with minimal pose errors. It is thus the only bottleneck of the application that, hopefully, will soon be cleared.

This thesis attempted to provide a meaningful contribution to the problem of autonomous vision-based object modeling by a humanoid robot. We introduced a general viewpoint selection scheme, detailed a coherent software architecture to realize high-level tasks, and presented the main practical problems that need to be considered. Although some challenges remain, this work is already a solid starting point for the future realization of autonomous vision-based high-level tasks.

Publications

International Journal

- Torea Foissotte, Olivier Stasse, Pierre-Brice Wieber, Adrien Escande and Abderrahmane Kheddar, *Autonomous 3D Object Modeling by a Humanoid using an Optimization driven Next-Best-View Formulation*, International Journal of Humanoid Robotics, Special issue on Cognitive Vision, to appear (end 2010)

International Conferences

- Torea Foissotte, Olivier Stasse, Pierre-Brice Wieber and Abderrahmane Kheddar, *Using NEWUOA to drive the autonomous visual modeling of an object by a Humanoid Robot*, International Conference on Information and Automation (ICIA), June 2009.

- Torea Foissotte, Olivier Stasse, Pierre-Brice Wieber, Adrien Escande and Abderrahmane Kheddar, *A Two-Steps Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot*, International Conference on Robotics and Automation (ICRA), May 2009.

- Torea Foissotte, Olivier Stasse, Adrien Escande and Abderrahmane Kheddar, *A Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot*, 8th IEEE-RAS International Conference on Humanoid Robots, December 2008.

International Workshops

- Torea Foissotte, Oliver Stasse and Abderrahmane Kheddar, *Autonomous 3D Object Modeling by a Humanoid using a Next-Best-View Algorithm*, (ICCV), Workshop on Computer Vision for Humanoid Robots in Real Environments, September 2009.

- Olivier Stasse, Torea Foissotte, Diane Larlus, Abderrahmane Kheddar and Kazuhito Yokoi, *Treasure hunting for humanoids robot*, (Humanoids), Workshop on Cognitive Vision, December 2008

Japanese National conferences

- Torea Foissotte, Oliver Stasse and Abderrahmane Kheddar, *Autonomous Modeling of an Unknown Object with HRP-2*, RSJ, Humanoids Session, September 2010.

- Torea Foissotte, Oliver Stasse and Abderrahmane Kheddar, *Autonomous 3D Object Modeling by a Humanoid*, Robomec, May 2009.
- Torea Foissotte, Oliver Stasse, Adrien Escande and Abderrahmane Kheddar, *Towards a Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot*, RSJ, Humanoids Session, September 2008.

Résumé en français

Ce travail est focalisé sur le problème de la construction autonome du modèle tridimensionnel d'un objet inconnu en utilisant un robot humanoïde. Plus particulièrement, nous considérons un HRP-2 guidé par la vision au sein d'un environnement connu qui peut éventuellement contenir des obstacles. Notre méthode considère les informations visuelles disponibles, les contraintes sur le corps du robot ainsi que le modèle de l'environnement dans le but de générer des postures adéquates et les mouvements nécessaires autour de l'objet.

Le calcul autonome d'une séquence de poses pour un capteur en vue de construire le modèle d'un objet a fait l'objet de nombreuses recherches dans le domaine du "Next-Best-View" durant les 25 dernières années. Toutefois, les contraintes à prendre en compte dûes au système robotique où se trouve le capteur ne sont que rarement prises en compte dans des expériences pratiques.

Le problème de sélection de vue ("Next-Best-View") est abordé en se basant sur un générateur de postures qui calcule une configuration par la résolution d'un problème d'optimisation. Une première solution est une approche locale où un algorithme de rendu original a été conçu afin d'être inclus directement dans le générateur de postures. Une deuxième solution augmente la robustesse aux minima locaux en décomposant le problème en 2 étapes: (i) trouver la pose du capteur tout en satisfaisant un ensemble de contraintes réduit et (ii) calculer la configuration complète du robot avec le générateur de posture. La première étape repose sur des méthodes d'optimisation globale et locale (NEWUOA, BOBYQA...) afin de converger vers des points de vue pertinents dans des espaces de configuration admissibles non convexes.

Notre approche est testée en conditions réelles par le biais d'une architecture cohérente qui inclut différents composants logiciels spécifique à l'usage d'un humanoïde. Ces expériences intègrent des travaux de recherche en cours en planification de mouvements, contrôle de mouvements et traitement d'image, afin de construire de façon autonome le modèle 3D d'un objet.

5.5 Problème et contexte

Que se passe-t-il lorsque quelqu'un, se déplaçant dans un environnement familier, tombe, à un moment donné, sur un objet inconnu? Imaginons que cet objet ait une forme et une texture suffisamment particulières qui ne permettent pas de deviner le modèle complet en s'appuyant sur l'apparence initiale perçue. La tâche est considérée comme simple pour un humain qui peut à loisir se déplacer autour de l'objet, ou le prendre et le manipuler, afin d'observer les différentes surfaces. L'humain est alors capable de créer un modèle visuel simple qu'il pourra utiliser pour la réalisation de tâches ultérieures. La réalisation de cette tâche par un humain est aisée, toutefois elle nécessite la mise en oeuvre de beaucoup de tâches bas-niveau:

1. discrimination entre l'objet et l'environnement
2. construction d'un modèle de l'objet
   (a) reconnaissance de la surface 3D de l'objet
   (b) addition des amers visuels perçus dans le modèle de l'objet
   (c) détection des propriétés telles que taille, texture, poids...
3. mouvement autour de l'objet
   (a) calcul du prochain point de vue pour percevoir l'objet
   (b) calcul de la pose du robot permettant l'acquisition du point de vue désiré
   (c) plannification du mouvement permettant d'atteindre la pose désirée
   (d) mise en oeuvre du mouvement
   (e) prise en compte de la stabilité
   (f) évitement de collisions (violentes) avec l'environnement
   (g) évitement de collisions (violentes) avec soi-même
   (h) test de la pose d'arrivée par rapport à l'objet
4. manipulation de l'objet
   (a) détection de la désolidarisation de l'objet par rapport à l'environnement
   (b) reconnaissance des zones aptes à être utilisées pour prendre l'objet
   (c) calcul de la préhension de l'objet
   (d) plannification du mouvement de manipulation
   (e) mise en oeuvre du mouvement
5. ...

Toutes ces tâches ont été abordées par différents travaux. Toutefois les systêmes existants ne se concentrent que sur un ensemble très réduit de ces tâches et sous des conditions très particulières.

Le travail présenté dans cette thèse s'attache au problème pratique de la construction d'un modèle 3D virtuel basé sur la prise de différentes vues d'un objet inconnu par un robot humanoïde HRP-2 (voir Fig. 5.10). Pour se faire, nous nous appuyons sur une intégration cohérente des résultats de plusieurs travaux de recherche tout en apportant une contribution scientifique axé sur le calcul du prochain point de vue.

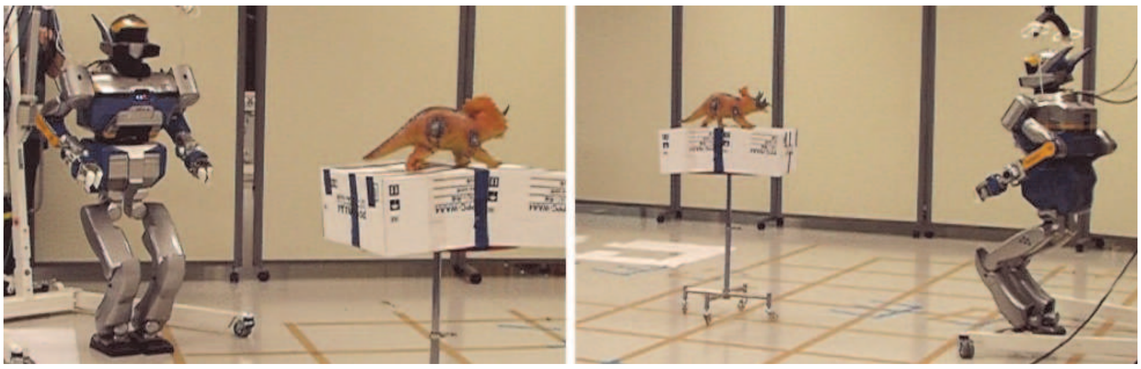


Figure 5.10: Sélection de point de vue pour un robot humanoïde: recherche d'une posture valide pour modéliser un objet inconnu.

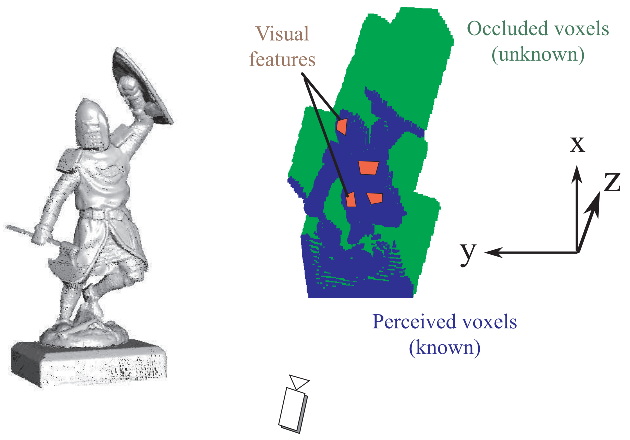


Figure 5.11: Construction du modèle virtuel. (gauche) modèle 3D original. (droite) modèle mis à jour en utilisant la vision stéréo et la détection d'amers visuels.

Il est important de noter que, dans ce travail, nous ne considérons pas les différents problèmes liés à la manipulation de l'objet.

Plus particulièrement, notre but est de générer et atteindre de nouvelles postures pour le robot tout autour de l'objet afin de créer un modèle visuel 3D. Ce problème a été abordé plusieurs fois dans le domaine du "Next-Best-View".

Le modèle visé se compose d'une grille de voxels ainsi que d'un ensemble d'amers visuels distribué sur la surface de l'objet (voir Fig. 5.11). Un tel modèle a pour but d'être utilisé dans des tâches ultérieures telles que la détection et la reconnaissance de l'objet en question, ou encore sa manipulation.

Le calcul des points de vue nécessaires à la modélisation se fait en tenant compte de la forme et de la texture de l'objet, des contraintes du robot, ainsi que des obstacles présents dans l'environnement. Deux méthodes originales sont présentées

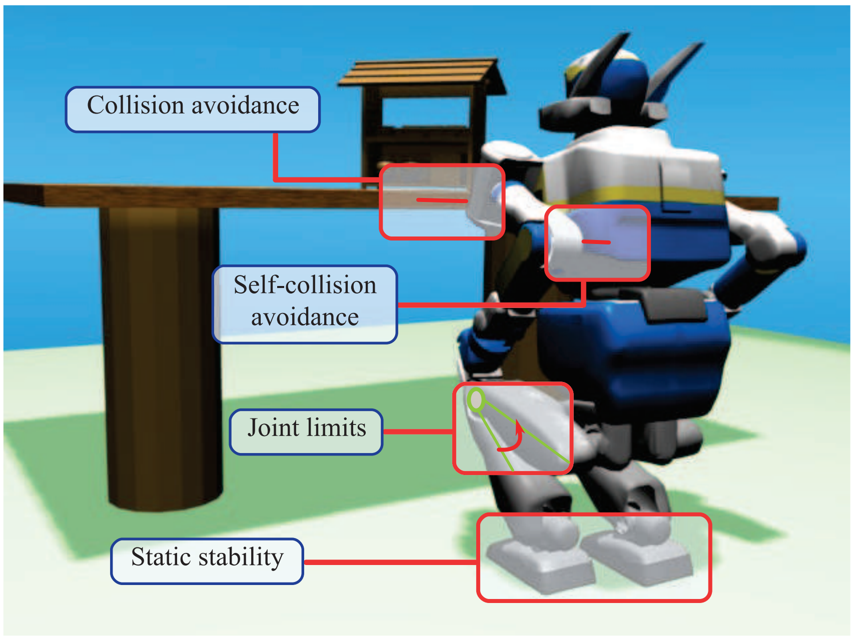


Figure 5.12: Constraintes de base pour obtenir une posture pour un robot humanoïde: évitement de collisions avec l'environnement, évitement d'auto-collisions, respect des limites de joints et la stabilité statique.

dans les chapitres 2 et 3. Le chapitre 4 présente les outils matériels et logiciels sur lesquels nous nous appuyons pour réaliser une expérience pratique complète. Le dernier chapitre présente les résultats expérimentaux.

5.6 Formulation $\mathcal{C}^1$ de la visibilité d'un objet pour la sélection de point de vue

Ce chapitre introduit un premier algorithme original servant à générer une pose réalisable pour un robot HP-2 qui permet de maximiser un critère visuel. Cet algorithme représente une solution locale au problème posé et repose sur une méthode d'optimisation adaptée à des fonctions $\mathcal{C}^1$. Nous introduisons une contrainte de stabilité ainsi qu'un critère visuel dont le gradient peut être calculé de manière analytique. L'originalité de notre procédure de sélection de point de vue est de prendre en compte les contraintes sur la configuration du corps du robot humanoïde. Par exemple, la hauteur de la caméra est limitée par la taille du robot et les points d'appui possibles. Les contraintes prises en compte pour générer une posture sont illustrées dans la Fig. 5.12 et sont: la limite des joints, l'évitement de collisions et la stabilité.

De telles contraintes ont un impact important sur les solutions possibles pour la sélection de point de vue, de ce fait, notre algorithme est conçue afin d'être intégré au coeur même de notre méthode de génération de posture. Nous cherchons ainsi à minimiser un critère visuel tout en satisfaisant des contraintes sur le corps du robot. Ce travail repose sur un générateur de posture disponible dans le laboratoire, dévelopé par Adrien Escande. Ce générateur utilise le solveur de FSQP pour résoudre différentes contraintes tout en minimisant une fonction objectif relatifs à la configuration d'HRP-2.

Un premier travail sur ce générateur de posture présente une formulation adaptée de la stabilité statique pour le robot. Ce critère assure que la distance entre la projection du centre de gravité du robot sur le sol est située à une distance limitée par rapport au segment entre le centre de chaque pied. Ceci permet d'assurer la stabilité statique du robot tout en autorisant le solveur à bouger la pose des pieds du robot librement.

Pour calculer la pose du robot de façon itérative afin de construire et compléter le modèle de l'objet inconnu, nous cherchons à observer les parties du modèle 3D de l'objet qui n'ont pas encore été perçue. Le modèle est représenté par un ensemble de voxels 3D qui prennent trois valeurs principales: vide, perçu et inconnu. La seconde contribution scientifique de cette thèse est une formulation originale de la visibilité des voxels inconnu d'un modèle 3D en fonction de la pose à 6 dimensions du capteur. Cette formulation représente la projection des voxels tri-dimensionnel sur le capteur par des gaussiennes en deux dimensions. Différents coefficients sont également présentés afin d'approximer les occlusions entre différents types de voxels. Cette formulation est de type $\mathcal{C}^1$ et peut donc être intégrée en tant que fonction objectif dans le générateur de posture utilisé.

Toutefois cette formulation à deux inconvénients pratiques: (i) le temps de calcul important et surtout (ii) la présence typique de nombreux minimums locaux dans le gradient qui nuisent a la convergence du solveur. Afin de répondre aux limites de cette formulation nous testons une autre approche, présentée dans le chapitre suivant.

5.7 Approximation locale quadratique de la visiblité d'un objet pour la sélection de point de vue

Ce chapitre introduit une autre méthode originale au problème de sélection de point de vue prenant en compte les contraintes liées à l'utilisation d'un robot humanoïde. Ce nouvel algorithme est formulé afin de résoudre les deux principaux problemes rencontrés avec notre précédente formulation: (i) la lente convergence dûe à la présence de nombreux minimums locaux, et (ii) le temps de calcul élevé pour l'évaluation de chaque point de vue. Il est possible d'éviter ces problèmes en évaluant les données visuelles en dehors du générateur de posture, retirant ainsi la nécessité d'une formulation $\mathcal{C}^1$. La génération de posture est donc faite en deux étapes, illustrées dans la

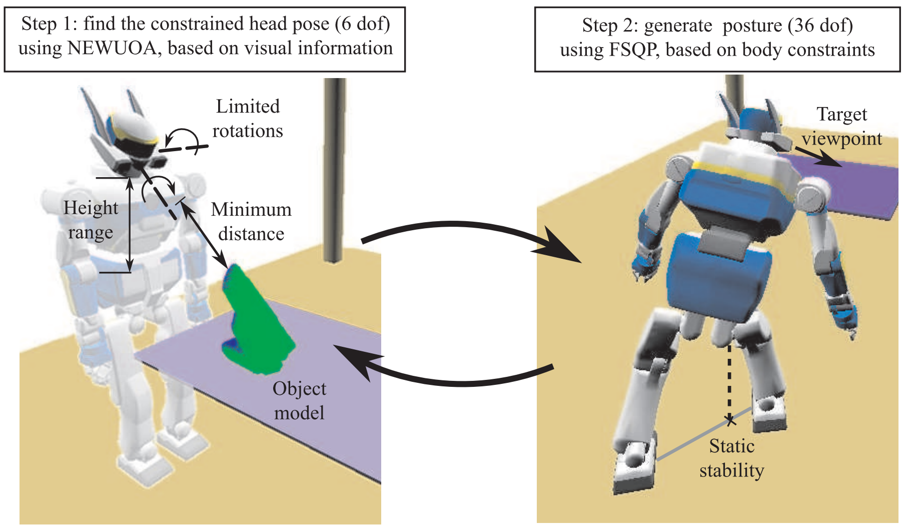

Step 1: find the constrained head pose (6 dof) using NEWUOA, based on visual information
Limited rotations
Height range
Minimum distance
Object model
Step 2: generate posture (36 dof) using FSQP, based on body constraints
Target viewpoint
Static stability

Figure 5.13: Deux étapes pour la génération de posture afin de compléter le modèle de l'objet.

Fig. 5.13:

1. Trouver une position et orientation de caméra qui maximize la quantité de voxels inconnu visible tout en satisfaisant certaines contraintes sur la tête du robot. Différentes contraintes visuelles peuvent également être introduites afin d'améliorer la robustesse de l'algorithme.

2. Générer une posture complète pour le robot en considérant le point de vue trouvé dans la première étape ainsi que des contraintes additionnelles.

La première étape est résolue en s'appuyant sur un échantillonage global de l'espace des configurations de la caméra ainsi que sur une méthode d'optimization sans dérivée telle que NEWUOA ou BOBYQA. La deuxième étape calcule la posture complète pour le robot en s'appuyant sur le générateur de posture existant.

Ce chapitre se focalise plus particulièrement sur la réalisation de la première étape, notamment la formulation de la fonction à minimiser par les méthodes NEWUOA ou BOBYQA, ainsi que le processus de recherche d'un point de vue. La fonction objectif considérée est constituée de différents critères: visibilité des voxels inconnus, visibilité des amers visuels connu, occlusions visuelles par des obstacles dans la scène, et distance à l'objet. La recherche d'un minimum global de cette fonction complexe est réalisé par l'utilisation d'un échantillonage de faible densité de l'espace des configurations, couplé à des recherches locales en s'appuyant sur NEWUOA ou BOBYQA.

L'algorithme présenté est formulé de façon a générer des postures pour le robot humanoïde afin de contruire des modeles d'objets qui ont des formes complexes, des

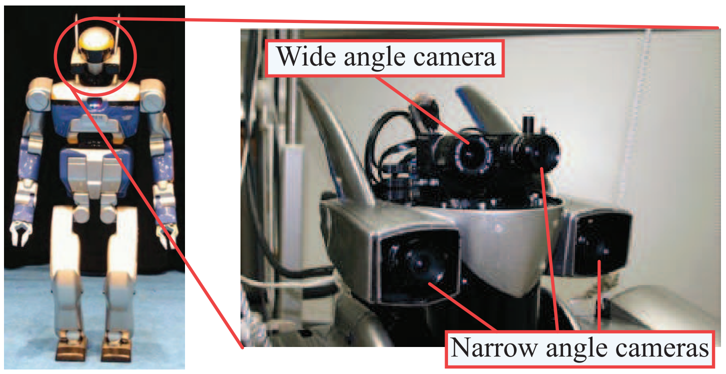


Figure 5.14: *gauche*: HRP-2 No.10. *droite*: gros plan sur la tête de l'humanoïde avec la position des caméras. Dans ce travail, nous utilisons une caméra grand angle ainsi que les deux caméras petit angle situees sur les cotes.

tailles variables et qui peuvent être dans des environnements comprenant beaucoup d'obstacles. Les postures générées sont sélectionnées parmi les configurations satisfaisant les contraintes de stabilité, d'évitement de collisions, et de respect des limites de joints. Différents critères visuels sont également évalués afin de limiter le nombre de postures nécessaires tout en prenant en compte les caractéristiques visuelles de l'environnement et de l'objet.

Cette approche est validée avec succès en simulation pour différents objets de forme complexe.

5.8 Outils dédiés à la vision

Ce chapitre présente le matériel ainsi que les outils logiciels qui sont utilisés pour réaliser l'application complète de modélisation automatique d'objet.

Notre expérience s'appuie sur la plateforme robotique disponible au Joint Robotics Laboratory à l'AIST de Tsukuba. Ce travail utilise les fonctionalites offertes par le robot humanoïde HRP-2 présente dans la Fig. 5.14. La tête comprend 4 caméras au total dont 3 sont utilisees pour notre application. Les caméras a angle de vue reduit, situees sur le cote de la tête, sont utilisees pour avoir une vue stéréo permettant l'obtention de la surface tri-dimensionnelle detaillee de l'objet a modeliser. Les images stéréo couleurs sont egalement utilisee pour la détection de points d'intérêts sur la surface de l'objet. Une caméra à grand angle, située à l'intérieur même de la tête, permet d'avoir une vue d'ensemble de la scène et ainsi peut être utilisée pour localiser le robot dans l'environnement.

L'utilisation de la vision sert donc deux intérêts principaux: (i) la modélisation même de la surface visible de l'objet et (ii) le calcul de la position et l'orientation du robot par rapport à l'objet et l'environnement.

Comme il a été abordé dans les chapitres précédents, nous nous basons sur la surface 3D perçue de l'objet ainsi que sur une détection de points d'intérêts afin de générer un point de vue pertinent et la pose complète du robot qui correspond. Les images stéréo obtenues sont analysées par un algorithme de correspondance stéréo pour obtenir une partie de la surface 3D de l'objet. Ici, nous utilisons l'algorithme "Semi-Global Block Matching" qui est disponible librement dans la librairie OpenCV. Un algorithme de "space carving" est ensuite chargé d'intégrer cette surface 3D dans la grille d'occupation représentant l'état courant du modèle 3D de l'objet. Finalement nous faisons une détection de descripteurs visuels SURF dans les images stéréo pour trouver les points d'intérêts sur la surface visible. Ces points d'intérêts sont mis en relation avec des voxels du modèle 3D.

Une autre part importante de la vision abordée dans cette thèse est la localisation autonome du robot. La position et orientation précise du robot par rapport à l'objet est nécessaire afin d'intégrer les nouvelles données dans le modèle de l'objet en construction. En pratique, lorsque le robot marche, des erreurs de positionnement s'accumulent, dûs à une prise en compte incomplète des glissements des pieds sur le sol. Trois méthodes sont testées pour résoudre ce problème: (i) SLAM monoculaire, (ii) asservissement visuel virtuel, et (iii) reconnaissance de points d'intérêts. Pour la première méthode, nous utilisons le logiciel fourni par le groupe de vision par ordinateur de l'université de Bristol. L'asservissement visuel virtuel repose sur le logiciel ViSP du groupe Lagadic à l'INRIA de Rennes. Enfin, pour la dernière méthode, nous nous appuyons sur les fonctionalités disponibles dans la librairie OpenCV.

Ces travaux de localisation automatique n'ont pas pu aboutir à des résultats exploitables en pratique avant la fin de cette thèse. Toutefois différentes pistes à explorer sont mentionnées.

5.9 Réalisation pratique de l'expérience de modélisation avec HRP-2

Le travail dévelopé dans cette thèse vise à générer de façon automatique toutes les étapes nécessaires pour la modélisation tri-dimensionnelle d'un objet inconnu au moyen d'un robot humanoïde HRP-2. Cela représente une application complexe qui doit, entre autres, planifier et atteindre des postures spécifiques, planifier et exécuter des mouvements, assurer la stabilité et l'évitement de collisions, et prendre en compte les informations visuelles accessibles. Pour mener à bien ces travaux, nous nous reposons sur divers outils logiciels dévelopés par d'autres chercheurs. Une vue d'ensemble de l'architecture logicielle de notre application est présentée dans la Fig. 5.15. La communication entre les composants est assurée par omniORB, une implémentation du standard CORBA, ce qui permet d'exécuter ces composants sur différents ordinateurs, répartissant ainsi la charge de calcul.

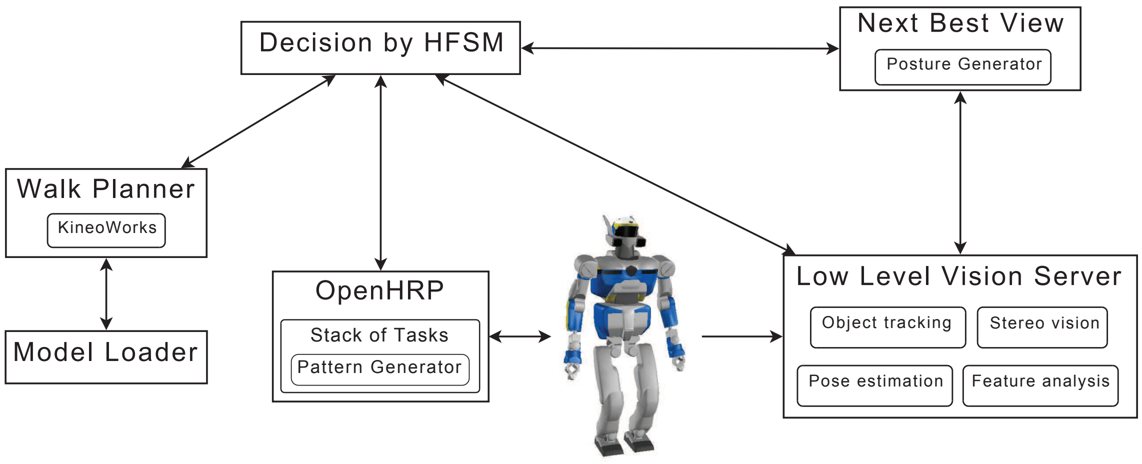


Figure 5.15: Principaux composants actifs dans l'expérience de modelisation.

Le chef d'orchestre est représenté par l'entité *Decision By HFSM (Hierarchical Finite State Machine)* qui va synchroniser l'exécution des autres composants en fonction des sous-tâches courantes à réaliser. Le principal interlocuteur à *Decision By HFSM* est *Stack of Tasks*, une implémentation de cinématique inverse généralisée réalisée au LAAS qui permet de controler les mouvements du robot en passant par le logiciel OpenHRP réalisé à l'AIST. L'entité *Stack of Tasks* commande *Pattern Generator* en charge de la création du mouvement de marche. La séquence de pas requise pour amener le robot vers une pose désirée tout en évitant les collisions avec les obstacles présents dans l'environnement, dont le modèle est chargé grace à *Model Loader*, est produite par *Walk Planner*. La pose du robot utilisée pour capturer les images de l'objet sont générées par le composant *Next-Best-View* qui implemente l'algorithme détaillé dans le chapitre 3. Finalement, la partie bas niveau des opérations de vision est réalisée par le composant *Low Level Vision Server*.

L'application complète est testée en simulation au moyen du logiciel OpenHRP et en utilisant des objets et des obstacles virtuels. La simulation confirme la plausibilité de notre approche, i.e. les postures et mouvements générés sont stables et sans collisions, et les moments de force sur les joints du robot obtenus n'atteignent pas de niveaux dangereux pour les moteurs.

L'application est ensuite testée au travers d'expériences avec un humanoïde HRP-2. Les premières expériences sont réalisées dans des conditions simplifiées:

- Pas d'obstacles dans l'environnement mis à part la table où est posé l'objet.

- La vision du robot est simulée, i.e. les images prisent par les caméras embarquées sur le robot ne sont pas utilisées. La perception de l'objet est donc simulée en utilisant un objet 3D virtuel et nous considerons que la caméra est toujours placée à l'endroit calculé.

- Pas de correction de la localisation du robot basé sur la vision.

Ces tests sont conduits afin de vérifier l'intégration de tous les composants logiciels. Il est également utile d'évaluer l'ampleur des glissements du robot durant la marche pour atteindre une position voulue. Il faut noter que la précision de la localisation du robot par rapport à l'objet est primordiale pour mettre à jour de façon correcte le modèle de l'objet. Une mauvaise localisation rend le modèle inexploitable après seulement deux ou trois perceptions. Un example de séquence de poses atteintes est présentée dans la Fig. 5.16, avec la trajectoire correspondante dans la Fig. 5.17. Nous pouvons remarquer l'amplitude des erreurs de déplacement même sur des courtes distances. Il en resulte une importante disparité entre la pose du robot obtenue et celle désirée. Ainsi il est nécessaire de localiser avec précision le robot pendant ses déplacements.

Les travaux de localisation basé vision n'ont pu aboutir à des résultats exploitables avant la fin de cette thèse et donc une expérience réelle complète de modélisation d'objet n'a pu être effectuée. Néanmoins différentes pistes sont mentionnées afin de surmonter ces difficultés dans des travaux futurs.

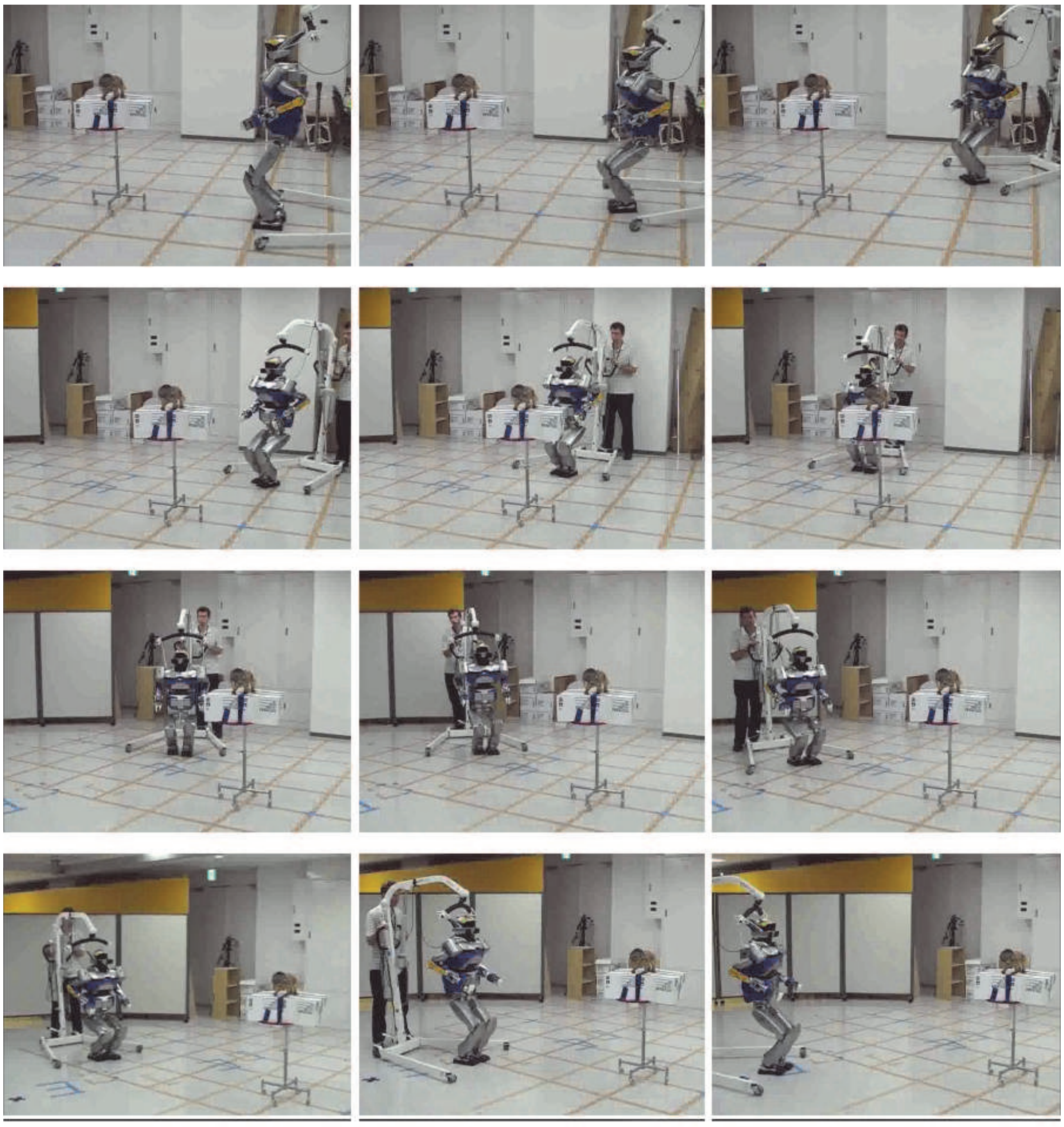


Figure 5.16: Poses atteintes par HRP-2 pendant une simulation de modélisation d'objet.

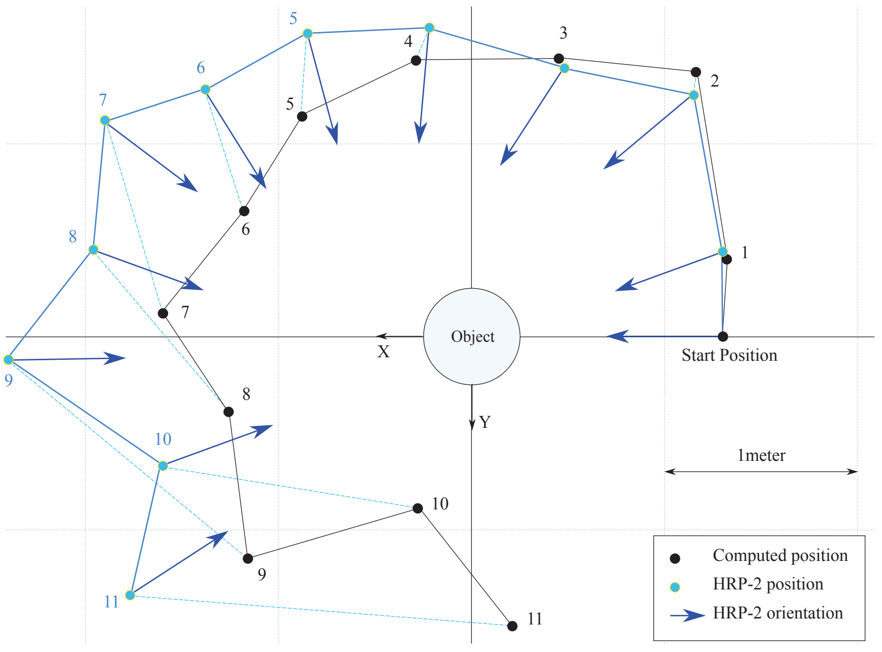


Figure 5.17: Comparaison entre la trajectoire calculée et la trajectoire exécutée par HRP-2. Les positions et orientations du robot ne sont pas corrigées. La perception de l'objet est simulée avec un objet virtuel 3D et la pose de la caméra virtuelle est placée dans la configuration calculée par l'algorithme de sélection de point de vue.

Bibliography

[Aggarwal 1988] J. K. Aggarwal and N. Nandhakumar. *On the computation of motion from sequences of images, a review*. Proceedings of the IEEE, vol. 76, pages 917–935, 1988. 26

[Agin 1979] G.J. Agin. *Real Time Control of a Robot with a Mobile Camera*. Technical report, SRI International, February 1979. 25

[Ahuja 1989] N. Ahuja and J. Veenstra. *Generating Octrees from Object Silhouettes in Orthographic Views*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, pages 137–149, 1989. 24

[Arbel 2001] T. Arbel and F. Ferrie. *Entropy-based gaze planning*. Image and Vision Computing, vol. 19, no. 11, pages 779 – 786, 2001. 20

[Bajcsy 1988] R. Bajcsy. *Active perception*. Proceedings of the IEEE, vol. 76, no. 8, pages 966–1005, August 1988. 15

[Banta 1995] J. Banta, Y. Zhien, X. Wang, G. Zhang, M. Smith and M. Abidi. *A best-next view algorithm for three-dimensional scene reconstruction using range images*. In Proceedings SPIE, volume 2588, pages 418–429, 1995. 18, 19, 40

[Banta 2000] J. Banta, L. Wong, C. Dumont and M. Abidi. *A Next-Best-View System for Autonomous 3-D Object Reconstruction*. IEEE Transactions on Systems, Man and Cybernetics, vol. 30, pages 589–598, september 2000. 18, 40

[Bay 2008] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool. *SURF: Speeded Up Robust Features*. Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008. 29, 89, 105

[Beardsley 1996] P. A. Beardsley, P. Torr and A. Zisserman. *3D model acquisition from extended image sequences*. In European Conference on Computer Vision, pages 683–695, 1996. 26

[Beardsley 1997] P. Beardsley, A. Zisserman and D. Murray. *Sequential Updating of Projective and Affine Structure from Motion*. International journal of computer vision, vol. 23, no. 3, pages 235–259, June 1997. 26

[Berry 2000] F. Berry, P. Martinet and J. Gallice. *Turning around an unknown object using visual servoing*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000. 26

[Besag 1986] J. Besag. *On the statistical analysis of dirty pictures.* Journal of the Royal Statistical Society, vol. 48, no. 3, pages 259–302, 1986. 26

[Besl 1992] P. Besl and N. McKay. *A Method for Registration of 3-D Shapes.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pages 239–256, 1992. 29, 100

[Bluethmann 2003] W. Bluethmann, R. Ambrose, M. Diftler, S. Askew, E. Huber, M. Goza, F. Rehnmark, C. Lovchik and D. Magruder. *Robonaut: A Robot Designed to Work with Humans in Space.* Autonomous Robots, vol. 14, pages 179–197, 2003. 10.1023/A:1022231703061. 7

[Bottino 2006] A. Bottino and A. Laurentini. *What's NEXT? An interactive next best view approach.* Pattern Recognition, vol. 39, no. 1, pages 126–132, 2006. 19

[Bouguet 2004] J-Y. Bouguet. *Camera Calibration Toolbox for Matlab*, 2004. 85

[Broadhurst 2001] A. Broadhurst, T.W. Drummond and R. Cipolla. *A probabilistic framework for space carving.* In Eighth IEEE International Conference on Computer Vision (ICCV), pages 388–393, Vancouver, BC , Canada, July 2001. IEEE. 87

[Brown 1976] D. C. Brown. *The bundle adjustment - progress and prospects.* International Archives of Photogrammetry, vol. 21, no. 3, pages 3–36, 1976. 92

[Calway 2007] A. Calway, W. Mayol-Cuevas, D. Chekhlov, A. Gee and J. Martinez Carranza. *MonoSLAM.* `http://www.cs.bris.ac.uk/Research/Vision/slam.jsp`, 2007. 96

[Chaumette 2006] F. Chaumette and S. Hutchinson. *Visual servo control, Part I: Basic approaches.* IEEE Robotics and Automation Magazine, vol. 13, no. 4, pages 82–90, December 2006. 26

[Chaumette 2007] F. Chaumette and S. Hutchinson. *Visual servo control, Part II: Advanced approaches.* IEEE Robotics and Automation Magazine, vol. 14, no. 1, pages 109–118, March 2007. 26

[Chekhlov 2006] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas and A. Calway. *Real-Time and Robust Monocular SLAM Using Predictive Multi-resolution Descriptors.* In 2nd International Symposium on Visual Computing, November 2006. 96

[Chekhlov 2007] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas and A. Calway. *Robust Real-Time Visual SLAM Using Scale Prediction and Exemplar Based Feature Description.* In IEEE International Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, June 2007. 96

[Chen 1991] Y. Chen and G. Medioni. *Object Modeling by Registration of Multiple Range Images.* In IEEE International Conference on Robotics and Automation, pages 2724 – 2729, 1991. 29, 100

[Chen 2004] S. Y. Chen and Y. F. Li. *Automatic Sensor Placement for Model-Based Robot Vision.* IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics, vol. 34, no. 1, pages 393–408, february 2004. 19

[Chen 2005] S. Y. Chen and Y. F. Li. *Vision sensor planning for 3-D model acquisition.* IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics, vol. 35, no. 5, pages 894–904, 2005. 19

[Chen 2008] S. Chen, Y. Li, J. Zhang and W. Wang. Active sensor planning for multiview vision tasks. Springer, 2008. 15

[Chesi 2009] G. Chesi. *Visual Servoing Path Planning via Homogeneous Forms and LMI Optimizations.* IEEE TRANSACTIONS ON ROBOTICS, vol. 25, no. 2, pages 281–291, april 2009. 26

[Chien 1986] C. H. Chien and J. K. Aggarwal. *Volume/surface octrees for the representation of three-dimensional objects.* Computer Vision Graphics and Image Processing, vol. 36, no. 1, pages 100–113, 1986. 17

[Chung 2007] T. Chung and J. Burdick. *A Decision-Making Framework for Control Strategies in Probabilistic Search.* In IEEE International Conference on Robotics and Automation, Roma, Italy, April 2007. IEEE. 23

[Connolly 1985] C. Connolly. *The determination of next best views.* In IEEE International Conference on Robotics and Automation, 1985. 15, 17, 19, 40

[Corke 1994] P. Corke. *Visual Control Of Robot Manipulators – A Review.* In Visual Servoing, pages 1–31. World Scientific, 1994. 25

[Cormen 2001] T. Cormen, C. Leiserson, R. Rivest and C. Stein. Introduction to algorithms (2nd ed.), chapter 35.2, pages 1027–1033. MIT Press and McGraw-Hill, 2001. 24

[Davison 1998] A. Davison and D. Murray. *Mobile Robot Localisation using Active Vision.* In 5th European Conference on Computer Vision, pages 809–825, 1998. 96

[Davison 2007] A. Davison, I. Reid, N. Molton and O. Stasse. *MonoSLAM: Real-Time Single Camera SLAM*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pages 1052–1067, 2007. 96

[Deng 1996] X. Deng, E. Milios and A. Mirzaian. *Landmark Selection Strategies for Path Execution*. Robotics and Autonomous Systems, vol. 17, pages 17–3, 1996. 25

[Diankov 2009] R. Diankov, T. Kanade and J. Kuffner. *Integrating Grasp Planning and Visual Feedback for Reliable Manipulation*. In IEEE/RAS International Conference on Humanoid Robots (Humanoids), pages 646–652, 2009. 13

[Dune 2008] C. Dune, E. Marchand, C. Collowet and C. Leroux. *Active rough shape estimation of unknown objects*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages p. 3622–3627, september 2008. 20, 40

[Dune 2009] C. Dune. *Localisation et caractérisation d'objets inconnus à partir d'informations visuelles*. PhD thesis, Université de Rennes 1, France, 2009. 20

[Elfes 1989] A. Elfes. *Using Occupancy Grids for Mobile Robot Perception and Navigation*. Computer, vol. 22, no. 6, pages 46–57, 1989. 87

[Escande 2006] A. Escande, A. Kheddar and S. Miossec. *Planning support contact-points for humanoid robots and experiments on HRP-2*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2974 – 2979, 2006. 35

[Escande 2007] A. Escande, S. Miossec and A. Kheddar. *Continuous gradient proximity distance for humanoids collision-free optimized postures*. In IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2007. 35

[Escande 2008] A. Escande. *Planification de points dâappuis pour la génération de mouvements acycliques : application aux humanoides*. PhD thesis, Ecole des Mines ParisTech, France, 2008. 35, 112

[Forssén 2007] P-E. Forssén and D. Lowe. *Shape Descriptors for Maximally Stable Extremal Regions*. In IEEE International Conference on Computer Vision, volume CFP07198-CDR, Rio de Janeiro, Brazil, October 2007. 88

[Forssén 2008] P-E. Forssén, D. Meger, K. Lai, S. Helmer, J. Little and D. Lowe. *Informed visual search: Combining attention and object recognition*. In IEEE International Conference on Robotics and Automation, volume CFP08RAA-CDR, pages 935–942, Pasadena, CA, USA, May 2008. IEEE, IEEE Robotics and Automation Society. 14, 23, 88

[Furukawa 2009] Y. Furukawa and J. Ponce. *Carved Visual Hulls for Image-Based Modeling*. International Journal of Computer Vision, vol. 81, pages 53–67, 2009. 87

[Gates 2007] B. Gates. *A robot in every home*. In Scientic American Magazine. Scientic American, january 2007. 7

[Greene 1986] N. Greene. *Environment mapping and other applications of world projections*. IEEE Computer Graphics and Applications, vol. 6, no. 11, pages 21–29, 1986. 22

[Herdt 2010] A. Herdt, H. Diedam, P-B. Wieber, D. Dimitrov, K. Mombaur and M. Diehl. *Online Walking Motion Generation with Automatic Foot Step Placement*. Advanced Robotics, vol. 24, pages 719–737, 2010. 99

[Hertzmann 1999] A. Hertzmann. *Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines*. In SIGGRAPH 99 Course Notes, 1999. 31

[Hill 1979] J. Hill and W. Park. *Real time control of a robot with a mobile camera*. In 9th International Symposium on Industrial Robots, pages 233–246, 1979. 25

[Hirschmüller 2008] H. Hirschmüller. *Stereo Processing by Semiglobal Matching and Mutual Information*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pages 328–341, 2008. 85, 105

[Hou 2007] X. Hou and L. Zhang. *Saliency Detection: A Spectral Residual Approach*. In IEEE Conference on Computer Vision and Pattern Recognition, June 2007. 24

[Huang 2005] Y. Huang and k. Gupta. *An adaptive configuration space and work space based criterion for view planning*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3366–3371, august 2005. 22

[Johnson 2009] S. Johnson. *The NLopt nonlinear-optimization package*. `http://ab-initio.mit.edu/nlopt`, 2009. 73

[Kaelbling 1998] L. Kaelbling, M. Littman and A. Cassandra. *Planning and acting in partially observable stochastic domains*. Artificial Intelligence, vol. 101, no. 1-2, pages 99–134, 1998. 23

[Kanehiro 2005] F. Kanehiro, T. Yoshimi, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, K. Kaneko, H. Hirukawa and F. Tomita. *Whole Body Locomotion Planning of Humanoid Robots based on a 3D Grid Map*. In IEEE International Conference on Robotics and Automation, 2005. 87

[Kaneko 2004] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi and T. Isozumi. *Humanoid robot HRP-2*. In IEEE International Conference on Robotics and Automation (ICRA), pages 1083–1090, 2004. 80

[Klein 2000] K. Klein and V. Sequeira. *View Planning for the 3D Modelling of Real World Scenes*. In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000. 22

[Klein 2007] G. Klein and D. Murray. *Parallel Tracking and Mapping for Small AR Workspaces*. In Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan, November 2007. 27, 97

[Kutulakos 1992] K. Kutulakos and C. Dyer. *Recovering Shape by Purposive Viewpoint Adjustment*. In IEEE Conference on Computer Vision and Pattern Recognition, pages 16–22, 1992. 19

[Kutulakos 1994] K. Kutulakos, C. Dyer and V. Lumelsky. *Provable Strategies for Vision-Guided Exploration in Three Dimensions*. In IEEE International Conference on Robotics and Automation, pages 1365–1372, 1994. 19

[Kutulakos 2000] K. Kutulakos and S. Seitz. *A Theory of Shape by Space Carving*. International Journal of Computer Vision, vol. 38(3), pages 199–218, 2000. Marr Prize Special Issue. 87

[Lagadic 2000] Lagadic. *ViSP*. `http://www.irisa.fr/lagadic/visp/visp.html`, 2000. 97

[Laumond 2006] J-P. Laumond. *Kineo CAM: a success story of motion planning algorithms*. IEEE Robotics and Automation Magazine, vol. 13, no. 2, pages 90–93, 2006. 113

[Lawrence 1997] C. T. Lawrence, J. L. Zhou and A. L. Tits. *User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints*. Technical report, Institute for Systems Research, University of Maryland, 1997. 35

[Li 2005] Y. F. Li, B. He, S. Y. Chen and P. Bao. *A view planning method incorporating self-termination for automated surface measurement*. Measurement Science and Technology, vol. 16, no. 9, pages 1865–1877, september 2005. 19, 40

[Lopez-Damian 2009] E. Lopez-Damian, G. Etcheverry, L. Enrique Sucar and J. Lopez-Estrada. *Probabilistic View Planner for 3D Modelling Indoor Environments.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4021–4026, october 2009. 23, 73

[Lowe 2001] D. Lowe. *Local feature view clustering for 3D object recognition.* In IEEE Conference on Computer Vision and Pattern Recognition, 2001. 14, 29

[Lowe 2004] D. Lowe. *Distinctive image features from scale-invariant keypoints.* international journal of computer vision, vol. 60, no. 2, pages 91–110, january 2004. 88

[Mansard 2006] N. Mansard. *Enchainement de taches robotiques.* PhD thesis, Université de Rennes 1, France, 2006. 113

[Marchand 1996] E. Marchand. *Strategies de perception par vision active pour la reconstruction et l'exploration de scenes statiques.* PhD thesis, Université de Rennes I, France, 1996. 26

[Marchand 1999] E. Marchand. *ViSP: A Software Environment for Eye-in-Hand Visual Servoing.* In IEEE International Conference on Robotics and Automation (ICRA), pages 3224–3229, 1999. 97

[Marchand 2005] E. Marchand, F. Spindler and F. Chaumette. *ViSP for visual servoing: a generic software platform with a wide class of robot control skills.* IEEE Robotics and Automation Magazine, Special Issue on Software Packages for Vision-Based Control of Motion, vol. 12, no. 4, pages 40–52, 2005. 97

[Marin-Hernandez 2005] A. Marin-Hernandez, M. Devy and V. Ayala-Ramirez. *Visual Planning for Autonomous Mobile Robot Navigation.* In MICAI 2005: Advances in Artificial Intelligence, volume 3789 of *Lecture Notes in Computer Science*, pages 1001–1011. Springer Berlin/Heidelberg, 2005. 25

[Martin 1983] W.N. Martin and J.K. Aggarwal. *Volumetric descriptions of objects from multiple views.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 5, no. 2, pages 150–158, 1983. 87

[Martinez-Cantin 2009] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos and A. Doucet. *A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot.* Autonomous Robots, vol. 27, pages 93–103, 2009. 10.1007/s10514-009-9130-2. 22

[Massios 1998] N. Massios and R. Fisher. *A Best Next View Selection Algorithm incorporating a Quality Criterion*. In British Machine Vision conference, 1998. 19, 61

[Matas 2002] J. Matas, O. Chum, M. Urban and T. Pajdla. *Robust wide-baseline stereo from maximally stable extremal regions*. In British Machine Vision Conference, pages 384–393, 2002. 29

[Maver 1991] J. Maver and R. Bajcsy. *Occlusions as a guide for planning the next view*. Technical report, university of pennsylvania, 1991. 16

[Maver 1993] J. Maver and R. Bajcsy. *Occlusions as a Guide for Planning the Next View*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, pages 417–433, 1993. 16

[Maver 1995] J. Maver. *Collecting visual information using an active sensor system*. PhD thesis, University of Ljubljana, Slovenia, 1995. 16

[McKinnon 1998] K. McKinnon. *Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point*. SIAM Journal on Optimization, vol. 9, no. 1, pages 148–158, 1998. 22, 73

[Meger 2008] D. Meger, P-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little and D. G. Lowe. *Curious George: An attentive semantic robot*. Robotics and Autonomous Systems, vol. 56, no. 6, pages 503–511, 2008. 14

[Michel 2008] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner and T. Kanade. *Humanoid navigation planning using future perceptive capability*. In IEEE/RAS International Conference on Humanoid Robots (Humanoids), december 2008. 25, 97

[Morel 2009a] J.M. Morel and G.Yu. *ASIFT: A New Framework for Fully Affine Invariant Image Comparison*. SIAM Journal on Imaging Sciences, vol. 2, pages 438–469, 2009. 88

[Morel 2009b] J.M. Morel and G. Yu. *Affine SIFT*. `http://www.ipol.im/pub/algo/my_affine_sift/`, 2009. 88

[Morisawa 2007] M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko and H. Hirukawa. *Experimentation of Humanoid Walking Allowing Immediate Modification of Foot Place Based on Analytical Solution*. In IEEE International Conference on Robotics and Automation, pages 3989–3994, 2007. 99, 110, 114

[Nakhaei 2008] A. Nakhaei and F. Lamiraux. *Motion Planning for Humanoid Robots in Environments Modeled by Vision.* In IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2008. 87

[Nelder 1965] J. Nelder and R. Mead. *A Simplex Method for Function Minimization.* Computer Journal, vol. 7, pages 308–313, 1965. 73

[Newcombe 2010] R. A. Newcombe and A. J. Davison. *Live dense reconstruction with a single moving camera.* In IEEE Conference on Computer Vision and Pattern Recognition, pages 16–22, 2010. 27

[O. 2008] O., T. Foissotte, D. Larlus, A. Kheddar and K. Yokoi. *Treasure hunting for humanoids robot.* In IEEE RAS/RSJ International Conference on Humanoids Robots, Workshop on Cognitive Humanoid Vision. IEEE RAS/RS, december 2008. 82

[Obdrzalek 2005] S. Obdrzalek and J. Matas. *Sub-linear indexing for large scale object recognition.* In British Machine Vision Conference, pages 1–10, 2005. 88

[Ohtake 2003] Y. Ohtake, A. Belyaev and H-P. Seidel. *A Multi-scale Approach to 3D Scattered Data Interpolation with Compactly Supported Basis Functions.* In SMI '03: Proceedings of the Shape Modeling International, page 153, Washington, DC, USA, 2003. IEEE Computer Society. 27

[Okada 2006] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato and M. Inaba. *Vision based behavior verification system of humanoid robot for daily environment tasks.* In IEEE/RAS International Conference on Humanoid Robots (Humanoids), pages 7–12, 2006. 7

[O'Rourke 1987] J. O'Rourke. Art gallery theorems and algorithms. Oxford University Press, 1987. 24

[Pan 2009] Q. Pan, G. Reitmayr and T. Drummond. *ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition.* In British Machine Vision Conference, London, September 2009. 27

[Panier 1987] E. Panier and A. Tits. *A superlinearly convergent feasible method for the solution of inequality constrained optimization problems.* SIAM Journal on Control and Optimization, vol. 25, no. 4, pages 934–950, 1987. 35

[Pito 1996] R. Pito. *Automated Surface Acquisition using Range Cameras.* PhD thesis, University of Pennsylvania, USA, 1996. 16

[Pito 1999] R. Pito. *A Solution to the Next Best View Problem for Automated Surface Acquisition.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, pages 1016–1031, 1999. 16, 23, 40

[Powell 2004] M. J. D. Powell. *The NEWUOA software for unconstrained optimization without derivatives.* Technical report, University of Cambridge, 2004. 56, 57

[Powell 2009] M. J. D. Powell. *The BOBYQA algorithm for bound constrained optimization without derivatives.* Technical report, University of Cambridge, 2009. 73, 124

[Reed 1997a] M. Reed, P. Allen and I. Stamos. *3-D modeling from range imagery: an incremental method with a planning component.* In International Conference on Recent Advances in 3-D Digital Imaging and Modeling, pages 76–83, 1997. 21

[Reed 1997b] M. Reed, P. Allen and I. Stamos. *Automated Model Acquisition from Range Images with View Planning.* In International Conference on Computer Vision and Pattern Recognition, pages 72–77, 1997. 21

[Rusinkiewicz 2001] S. Rusinkiewicz and M. Levoy. *Efficient Variants of the ICP Algorithm.* In International Conference on 3-D Digital Imaging and Modeling, 2001. 100

[Rybski 2007] P. Rybski and D. DeMenthon. *Semantic Robot Vision Challenge.* `http://www.semantic-robot-vision-challenge.org/index.html`, 2007. 14

[Saidi 2007a] F. Saidi, O. Stasse and K. Yokoi. Active visual search by a humanoid robot, chapter 1, pages 171–184. Springer Berlin / Heidelberg, 2007. 23, 40, 82

[Saidi 2007b] F. Saidi, O. Stasse, K. Yokoi and F. Kanehiro. *Online Object Search with a Humanoid Robot.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1677–1682, 2007. 23, 40

[Sanchiz 1999] J. M. Sanchiz and R. B. Fisher. *A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom.* In British Machine Vision Conference, pages 163–172, 1999. 22, 23, 31, 32, 60, 73

[Savarese 2007] S. Savarese, M. Andreetto, H. Rushmeier, F. Bernardini and P. Perona. *3d reconstruction by shadow carving: Theory and practical evaluation.* International Journal of Computer Vision, vol. 71, no. 3, pages 305–336, 2007. 87

[Scott 2003] W. Scott, G. Roth and J-F. Rivest. *View planning for automated three-dimensional object reconstruction and inspection.* ACM Comput. Surv., vol. 35, no. 1, pages 64–96, 2003. 15

[Seara 2001] J. F. Seara, O. Lorch and G. Schmidt. *Gaze Control for Goal-Oriented Humanoid Walking*. In IEEE/RAS International Conference on Humanoid Robots (Humanoids), 2001. 25

[Seara 2003] J. Seara, K. Strobl and G. Schmidt. *Path-Dependent Gaze Control for Obstacle Avoidance in Vision Guided Humanoid Walking*. In International Conference on Robotics and Automation, 2003. 25

[Seara 2004] J.F. Seara and G. Schmidt. *Intelligent gaze control for vision-guided humanoid walking: methodological aspects*. Robotics and Autonomous Systems, vol. 48, pages 231–248, october 2004. 25

[Seitz 1999] S. M. Seitz and C. R. Dyer. *Photorealistic Scene Reconstruction by Voxel Coloring*. International Journal of Computer Vision, vol. 35(2), pages 151–173, 1999. 87

[Shi 1994] J. Shi and C. Tomasi. *Good features to track*. In IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, 1994. 29, 96

[Shubina 2010] K. Shubina and J. Tsotsos. *Visual search for an object in a 3D environment using a mobile robot*. Computer Vision and Image Understanding, vol. 114, no. 5, pages 535 – 547, 2010. Special issue on Intelligent Vision Systems. 24

[Siciliano 1991] B. Siciliano and J-J. Slotine. *A general framework for managing multiple tasks in highly redundant robotic systems*. In International Conference on Advanced Robotics, pages 1211–1216, 1991. 113

[Sinha 2005] S. Sinha and M. Pollefeys. *Multi-View Reconstruction Using Photo-consistency and Exact Silhouette Constraints: A Maximum-Flow Formulation*. In Tenth IEEE International Conference on Computer Vision (ICCV), pages 349–356, Washington, DC, USA, 2005. IEEE Computer Society. 87

[Stasse 2006a] O. Stasse, A. Davison, R. Sellaouti and K. Yokoi. *Real-time 3D SLAM for Humanoid Robot considering Pattern Generator Information*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 348–355, October 9-15 2006. 79

[Stasse 2006b] O. Stasse, J. Semere, N. Ee Sian, T. Yoshimi and K. Yokoi. *Vision-based Virtual Information and Semi-autonomous Behaviours for a Humanoid Robot*. In International Conference on Intelligent Autonomous Systems (IAS-9), pages 794–803, Marc 7-9 2006. 80

[Stasse 2007] O. Stasse, D. Larlus, B. Lagarde, A. Escande, F. Saidi, A. Kheddar, K. Yokoi and F. Jurie. *Towards autonomous object reconstruction for visual* *search by the humanoid robot HRP-2*. In IEEE/RAS International Conference on Humanoid Robots (Humanoids), Pittsburg, November 29 - December 1 2007. 14, 35, 36, 40, 82

[Stasse 2008] O. Stasse, F. Lamiraux, A. Kheddar, K. Yokoi, R. Ruland and W. Prinz. *Integration of Humanoid Robots in Collaborative Working Environment: A case study on motion generation*. In International Conference on Ubiquitious Robots and Ambient Intelligence, pages 211–216, 2008. 108

[Stasse 2009] O. Stasse, P. Evrard, N. Perrin, N. Mansard and A. Kheddar. *Fast foot prints re-planning and motion generation during walking in physical human-humanoid interaction*. In IEEE/RAS International Conference on Humanoid Robotics(Humanoids 09), pages 284–289, 2009. 110

[Strasdat 2010] H. Strasdat, J. M. M. Montiel and A. J. Davison. *Scale-drift Aware Large Sclale Monocular SLAM*. In Proceedings of Robotics Science and Sytems (RSS), Zaragoza, Spain, June 2010. 97

[Stühmer 2010] J. Stühmer, S. Gumhold and D. Cremers. *Real-time Dense Geometry from a Handheld Camera*. In DAGM-Symposium, page accepted, 2010. 27

[Tarabanis 1995] K.A. Tarabanis, P.K. Allen and R.Y. Tsai. *A survey of sensor planning in computer vision*. In IEEE Transactions on Robotics and Automation, 1995. 15

[Thrun 2005] S. Thrun, W. Burgard and D. Fox. Probabilistic robotics. MIT press, 2005. 23

[Tongphu 2009] S. Tongphu, N. Thongsak and M. Dailey. Rapid detection of many object instances, volume 5807 of *Lecture Notes in Computer Science*, pages 434–444. Springer Berlin / Heidelberg, 2009. 88

[Triggs 1995] B. Triggs and C. Laugier. *Automatic Task Planning for Robot Vision*. In International Symposium Robotics Research, 1995. 24, 32, 61

[Triggs 2000] B. Triggs, P. McLauchlan, R. Hartley and A. Fitzgibbon. *Bundle Adjustment – A Modern Synthesis*. In B. Triggs, A. Zisserman and R. Szeliski, editeurs, Vision Algorithms: Theory and Practice, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag, 2000. 92

[Tsuji 2010] T. Tsuji, K. Harada, K. Kaneko, F. Kanehiro and K. Maruyama. *Grasp Planning for a Multifingered Hand with a Humanoid Robot*. Journal of Robotics and Mechatronics, vol. 22, no. 2, pages 230–238, 2010. 13

[Ude 2008] A. Ude, D. Omrcen and G. Cheng. *Making Object Learning and Recognition an Active Process.* International Journal Humanoid Robotics, vol. 5, no. 2, pages 267–286, 2008. 13, 24

[Vahrenkamp 2009] N. Vahrenkamp, C. Boge, K. Welke, T. Asfour, J. Walter and R. Dillmann. *Visual Servoing for Dual Arm Motions on a Humanoid Robot.* In IEEE/RAS International Conference on Humanoid Robots (Humanoids), pages 208–214, 2009. 13

[Vatti 1992] B. Vatti. *A generic solution to polygon clipping.* Commun. ACM, vol. 35, no. 7, pages 56–63, 1992. 41

[Wang 2003] P. Wang and K. Gupta. *Computing c-space entropy for view planning for a generic range sensor model.* In IEEE International Conference on Robotics and Automation (ICRA), 2003. 22

[Wang 2006] P. Wang and K. Gupta. *A Configuration Space View of View Planning.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1291–1297, 2006. 22

[Wedel 2008] A. Wedel, T. Pock, C. Zach, H. Bischof and D. Cremers. *An improved algorithm for TV-L1 optical flow computation.* In Proceedings of the Dagstuhl Visual Motion Analysis Workshop, 2008. 27

[Welke 2010] K. Welke, J. Issac, D. Schiebener, T. Asfour and R. Dillmann. *Autonomous Acquisition of Visual Multi-View Object Representations for Object Recognition on a Humanoid Robot.* In IEEE International Conference on Robotics and Automation, 2010. 13

[Westover 1989] L. Westover. *Interactive volume rendering.* In Symposium on Volume Visualization - proceedings of the 1989 Chapel Hill workshop on Volume Visualization, 1989. 43

[Whaite 1991] P. Whaite and F. Ferrie. *From uncertainty to visual exploration.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 3, pages 1038–1049, 1991. 20

[Whaite 1997a] P. Whaite. *A curious Machine for Autonomous visual exploration.* PhD thesis, McGill University, Canada, 1997. 20

[Whaite 1997b] P. Whaite and F. Ferrie. *Autonomous Exploration: Driven by Uncertainty.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 3, pages 193–205, 1997. 20, 40

[Wong 1999] L. M. Wong, C. Dumont and M. A. Abidi. *Next Best View System in a 3-D Object Modeling Task.* In International Symposium on Computational Intelligence in Robotics and Automation, pages 306–311, 1999. 19

[Yamazaki 2004a] K. Yamazaki, M. Tomono, T. Tsubouchi and S. Yuta. *3-D object modeling by a camera equipped on a mobile robot.* In IEEE International Conference on Robotics and Automation (ICRA), 2004. 24

[Yamazaki 2004b] K. Yamazaki, M. Tomono, T. Tsubouchi and S. Yuta. *Object Shape Reconstruction and Pose Estimation by a Camera Mounted on a Mobile Robot.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. 24

[Yemez 2007] Y. Yemez and C. J. Wetherilt. *A volumetric fusion technique for surface reconstruction from silhouettes and range data.* Computer Vision and Image Understanding, vol. 105, no. 1, pages 30–41, 2007. 87

[Zha 1997] H. Zha, K. Morooka, T. Hasegawa and T. Nagata. *Active modeling of 3-D objects: planning on the next best pose(NBP) for acquiring range images.* In Recent Advances in 3-D digital imaging and modeling, pages 68–75, may 1997. 20

Titre en français: Modélisation Visuelle d'un Objet Inconnu par un Robot Humanoïde Autonome

Résumé en français: Ce travail est focalisé sur la construction autonome du modèle 3D d'un objet inconnu en utilisant un robot humanoïde. Nous considérons un HRP-2 guidé par la vision au sein d'un environnement connu qui peut contenir des obstacles. Notre méthode considère les informations visuelles disponibles, les contraintes sur le corps du robot ainsi que le modèle de l'environnement dans le but de générer les postures et mouvements nécessaires.

Le problème de sélection de vue est abordé en se basant sur un générateur de postures qui calcule une configuration par la résolution d'un problème d'optimisation. Une première solution est une approche locale où un algorithme de rendu original à été conçu afin d'être inclus directement dans le générateur de postures. Une deuxième solution améliore la convergence en décomposant le problème en 2 étapes: (i) trouver la pose du capteur tout en satisfaisant un ensemble réduit de contraintes, et (ii) calculer la configuration complète du robot avec le générateur de posture. La première étape repose sur des méthodes d'optimisation globale et locale afin de converger vers des points de vue pertinents dans des espaces de configuration admissibles non convexes.

Notre approche est testée en conditions réelles par le biais d'une architecture cohérente qui inclus différents composants logiciels spécifique à l'usage d'un humanoïde. Ces expériences intègrent des travaux de recherche en cours en planification de mouvements, contrôle de mouvements et traitement d'image, afin de construire de façon autonome le modèle 3D d'un objet.

Mots-clés: robotique, humanoïde, modélisation, sélection de vue, vision, BOBYQA.

Title: Visual Modeling of an Unknown Object by an Autonomous Humanoid Robot

Abstract: This work addresses the problem of autonomously constructing the 3D model of an unknown object using a humanoid robot. We consider a HRP-2 evolving in a known environment, which is possibly cluttered, guided by vision. Our method considers the visual information available, the constraints on the robot body, and the model of the environment in order to generate pertinent postures and motions.

Our two solutions to the Next-Best-View problem are based on a specific posture generator, where a posture is computed by solving an optimization problem. The first solution is a local approach to the problem where an original rendering algorithm is specifically designed in order to be directly included in the posture generator. The rendering algorithm can display complex 3D shapes while taking into account self-occlusions. The second solution seeks more global solutions by decoupling the problem in two steps: (i) find the best sensor pose while satisfying a reduced set of constraints, and (ii) generate a whole-body posture with the posture generator. The first step relies on global sampling and local optimization method to converge toward pertinent viewpoints in non-convex feasible configuration spaces.

Our approach is tested in real conditions by using a coherent architecture that includes various complex software components that consider the specificities of the humanoid robot. This experiment integrates on-going works addressing the tasks of motion planning, motion control, and visual processing, in order to build autonomously the 3D model of the object.

Keywords: robotics, humanoid, object model, next-best-view, vision, BOBYQA.

Discipline: SYAM (Systèmes Automatiques et Microélectroniques)
Laboratoire: CNRS-LIRMM, CNRS-AIST JRL, UMI3218/CRT