# A Next-Best-View Algorithm for Autonomous 3D Object Modeling by a Humanoid Robot

T. Foissotte [1,2], O. Stasse [2], A. Escande [2], A. Kheddar [1,2]

[1] *CNRS-LIRMM, France*
[2] *CNRS/AIST JRL, Japan*

***Abstract*— A novel solution is presented which allows humanoids to build autonomously geometric models of unknown objects. Although good methods have been proposed for the specific problem of the next-best-view during the modeling and the recognition process; our approach is different and takes advantage of humanoid specificities in terms of embedded vision sensor and redundant motion capabilities.**

**The problem to select the best next view of interest at each modeling step is formulated as an optimization problem where the whole robot posture needs to be defined jointly with the robot cameras' position and orientation. To achieve this, we propose a differentiable formula that expresses the amount of unknown data visible from a specific viewpoint, given only knowledge acquired in previous steps. In addition, a specific stability constraint is introduced to allow the robot to reach a configuration where its feet can be moved away from their initial position.**

## I. Introduction

### A. Problem statement

One requirement for an autonomous robot to explore and interact fully in an unknown environment with humans is its ability to model and recognize new objects and environments. The work presented in this paper is a part of an ongoing project called 'treasure hunting', where the robot should retrieve an object in an unknown environment [1] based on a model that it previously build and stored [2]. This paper deals specifically with new objects modeling, with the future aim of being able to be robustly detected and recognized. Three main problems need to be solved to ensure a successful modeling process: (i) object/environment distinction, (ii) object features processing and memorizing, and (iii) object manipulation or sensor movement so as to model different faces. Currently we are simplifying the first problem by putting the object on a known table in front of the robot. For the second problem, we take advantage of results from a previous work [1] using an occupancy grid and disparity maps obtained by stereo vision, coupled with scale invariant features (SIFT) detection [3] which already proves their robustness for object recognition. Finally, this paper deals more particularly with the third problem by proposing an algorithm to move a humanoid robot around the object to be modeled. However, the object manipulation aspect of the problem is not addressed in this work.

### B. Overview of related work

Many existing works focus on the environment exploration [4] or object recognition problems [5]. The modeling part usually relies on a supervised method where different views of an object are taken manually by a human and served as an input to the algorithm. A number of works are dedicated to planning of sensor positions in order to create an accurate 3D model an unknown object, see for example [6], [7] or [8]. Hypothesis and limits of such works are detailed in these two surveys: [9] and [10]. The most usual assumptions 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. The object to analyze is also considered to be inside a sphere or on a turntable so that 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. The main aim is to get an accurate 3D reconstruction of an object, using voxels or polygons, while reducing the number of viewpoints required.

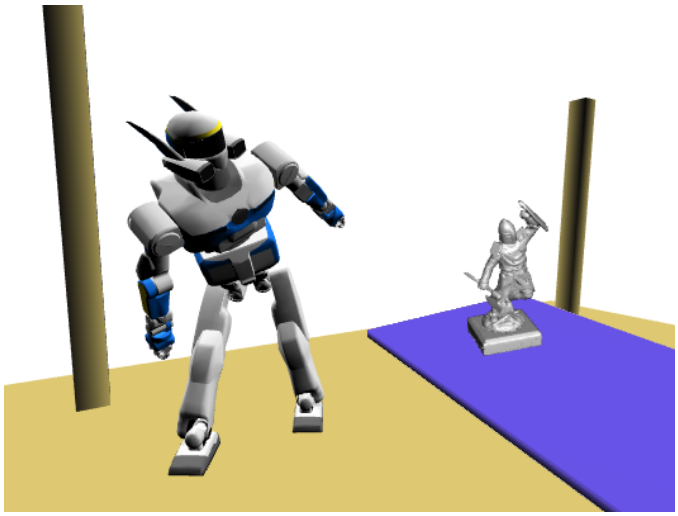

Fig. 1. Object modelization setting.

### C. Contribution

Though our modeling process also requires a next best view solution, it appears that working hypothesis are quite specific for a humanoid robot which needs only to characterize not all but the specific useful object's parts for its detection and recognition. Our work aims at getting rid of the human intervention in the modeling phase taking into account maneuverability and

constraints of a humanoid robot equipped with stereo cameras. In [2], we already stepped toward the object modeling by the robot, yet with human supervision. Our goal is thus to improve this work by guiding the modeling process using a new visual criterion.

Section II recalls some previous work necessary to introduce the new posture generation. Section III details the new stability constraint designed to ensure a statically stable posture while not specifying an artificial constraint on the feet as it has been done in previous work [2]. Our new optimization function which measures the visible area of the object's unknown parts depending on the robot posture is then introduced in section IV. Section V presents the simulation experiment results and section VI concludes this paper.

## II. Posture generation

The posture generation is realized by taking advantage of the posture generator (PG) proposed as part of the work in [11] and [2]. This posture generator is based on FSQP. Let us recall the problem to be minimized on our previous work when assuming:

1) that a Next Best View algorithm provides to the vision system with pose $\mathbf{H}$, a point to look at $\mathbf{x}$ with a given direction $\mathbf{v}$, and the vision system is at a distance inside the following interval $[d_{\min}, d_{\max}]$.
2) an arbitrary vector $\mathbf{f}$ sets a constant rigid transformation between the left foot $\mathbf{F}_l$ and the right foot $\mathbf{F}_r$.

then the problem can be written:

$$\min_{\mathbf{q}\in X} f_1(\mathbf{q}) \tag{1}$$

where $\mathbf{q} = [r \;\; w \;\; \Theta]^\top$, $r$ the position of the free-floating body, $w$ its orientation, and $\Theta = \{\theta_0 \dots \theta_d\}$ the robot's joints. Moreover $X$ is a set of constraints:

$$\begin{cases} \Theta_{\min} < \Theta < \Theta_{\max} & (2)\\ a < d(B_i(\mathbf{q}), B_j(\mathbf{q})) \; \forall (i,j) \in \mathcal{C} & (3)\\ F_l^z(\mathbf{q}) = F_r^z(\mathbf{q}) = 0 & (4)\\ \mathbf{F}_l(\mathbf{q}) - \mathbf{F}_r(\mathbf{q}) = \mathbf{f} & (5)\\ \mathbf{h}_z(\mathbf{q}) \times (\mathbf{x} - \mathbf{h}(\mathbf{q})) = 0 & (6)\\ \mathbf{h}_z(\mathbf{q}).(\mathbf{x} - \mathbf{h}(\mathbf{q})) \leq 0 & (7)\\ \mathbf{h}_z(\mathbf{q}) \times \mathbf{v} = 0 & (8)\\ \mathbf{h}_z(\mathbf{q}).\mathbf{v} \leq 0 & (9)\\ d_{\min} \leq \| \mathbf{h}(\mathbf{q}) - \mathbf{x} \|^2 \leq d_{\max} & (10)\\ \mathbf{A}_{\mathcal{S}(\mathbf{q})}\mathbf{c}(\mathbf{q}) \leq \mathbf{b}_{\mathcal{S}(\mathbf{q})} & (11) \end{cases}$$

with $\mathbf{c}(\mathbf{q})$ the CoM of the robot, $\Theta_{\min}$ and $\Theta_{\max}$ the joint limits, $d(B_i(\mathbf{q}), B_j(\mathbf{q}))$ the $\mathcal{C}^1$ distance between two bodies introduced by Escande et al. [12], $\mathcal{C}$ the set of collision pairs which are tracked to avoid non desirable collisions and auto-collisions. It is important to note that here $B_i$ is not constrained to be a robot's body but could be an object of the environment [2]. (4) imposes to the feet to be on the ground, while (5) imposes the relationship between the feet given by $\mathbf{f}$. The vector $\mathbf{h}_z(\mathbf{q})$ is the optical axis of the camera system, and $\mathbf{h}$ its position. (7) and (6) enforces the vision system to look towards $\mathbf{x}$. (9) and (8) constraints the vision system to be aligned with the vector $\mathbf{v}$. (10) imposes the vision system distance to $\mathbf{x}$ to be within a predefined interval. The stability constraint of our previous work uses the convex support polygon of the robot, obtained from the convex hull of the footprints, at pose $\mathbf{q}$, which is given by the set of points $\mathcal{S}(\mathbf{q})$, and is represented by the convex polytope $\mathbf{A}_{\mathcal{S}(\mathbf{q})}\mathbf{w} \leq \mathbf{b}_{\mathcal{S}(\mathbf{q})}$, with

$$\mathbf{A}_{\mathcal{S}(\mathbf{q})}\mathbf{w} = \begin{pmatrix} a_0 & -1 & 0\\ \vdots & \vdots & \\ a_i & -1 & 0\\ \vdots & \vdots & \\ a_n & -1 & 0 \end{pmatrix} \quad \mathbf{b}_{\mathcal{S}(\mathbf{q})} = \begin{pmatrix} b_0\\ \vdots\\ b_i\\ \vdots\\ b_n \end{pmatrix}$$
$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \quad b_i = -(y_i - a_i x_i) \quad \forall i \in \{0, ..., n-1\}$$
$$a_n = \frac{y_0 - y_{n-1}}{x_0 - x_{n-1}} \quad b_i = -(y_{n-1} - a_i x_{n-1}) \tag{12}$$

and $\mathcal{S}(\mathbf{q}) = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. Finally the function to minimize is:

$$f_1(\mathbf{q}) = \| \; \mathbf{pc}(\mathbf{q}) - \text{CoG}(\mathcal{S}(\mathbf{q})) \; \|^2 \tag{13}$$

where the $\text{CoG}(\mathcal{S}(\mathbf{q}))$ is the barycenter of the convex support polygon, and $\mathbf{pc} = [c_x \;\; c_y]\top$ is the projection of the CoM on the floor. This criterion seeks for the most statically stable posture that satisfies the constraints described previously. (11) makes sure that the stability criterion is never violated.

## III. Stability constraint

The robot is required to be statically stable while taking pictures of the object. Indeed, when walking, induced motion might result in a blurred image. This happen especially during landing of the foot; resulting impact's propagation creates oscillations at terminal points such as the head. In our previous work [2], the stability is ensured by both a constraint (11) and a criterion (13). However these have two limitations: (i) the poses of the feet relatively to each other cannot be modified, and (ii) a margin is necessary in the constraint implementation. In fact, practically if $\mathbf{pc}$ is close to the limits of the convex hull of $\mathcal{S}(\mathbf{q})$ the robot can be in an unstable position due to the flexibility in its ankle. In this paper, our original approach is to set the robot stability as a constraint where the distance from $\mathbf{pc}$ to the segment between both feet must be null. Though this is more restrictive than the previous approach, this has three advantages: (i) a dedicated criterion is not required, (ii) we are sure that the posture generated is stable, and (iii) the feet pose can be freely modified. Let us note this distance $g(\mathbf{q})$, thus the constraint to comply with is: $g(\mathbf{q}) = 0$.

### A. Mathematical formulation

The formulation of this constraint can be expressed in a 2D coordinate system as we work with points on a horizontal floor. First, the distance between the CoM projection, $\mathbf{pc}$ and the segment between the robot left foot's center $\mathbf{pF}_l =$

$[F_l^x \ F_l^y]^\top$ and right foot's center $\mathbf{pF}_r = [F_r^x \ F_r^y]^\top$ needs to be computed. When a specific robot pose results in a null distance (or practically when the distance is below a chosen threshold), then the robot stability constraint is solved. The computation of $g(\mathbf{q})$ depends on the relative position of the 3 points. Three cases are possible: the closest geometric object to $\mathbf{pc}$ 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, and thus 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:

$$\mathbf{ps} = \mathbf{pF}_l + \alpha_p \left(\mathbf{pF}_r - \mathbf{pF}_l\right) \tag{14}$$

$$(\mathbf{pc} - \mathbf{ps}) \cdot (\mathbf{pF}_r - \mathbf{pF}_l) = 0 \tag{15}$$

By solving these 2 formulas we can deduce the value of $\alpha_p$

$$\alpha_p = \frac{(\mathbf{pc} - \mathbf{pF}_l) \cdot (\mathbf{pF}_r - \mathbf{pF}_l)}{(pF_r^x - pF_l^x)^2 + (pF_r^y - pF_l^y)^2} \tag{16}$$

The value of $\alpha_p$ determines the closest geometric object to $\mathbf{pc}$:

$$g(\mathbf{q}) = \mathbf{o}(\mathbf{q})\mathbf{o}(\mathbf{q})^\top \tag{17}$$

with

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

### B. Gradient for the stability constraint

In order to generate a pose which satisfies our stability constraint, FSQP relies on a gradient descent method and thus needs the partial derivatives formulation of the constraint. Three formulations are possible, depending on the value of $\alpha_p$. For simplicity, let us write $\dot{f}(\mathbf{q}) = \partial f(\mathbf{q})/\partial \mathbf{q}$. From (17) it is possible to write:

$$\dot{g}(\mathbf{q}) = 2\mathbf{o}(\mathbf{q})\dot{\mathbf{o}}(\mathbf{q}) \tag{18}$$

with

$$\begin{cases} \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{pF}}_l(\mathbf{q}) & \text{if } \alpha_p \leq 0 \quad (19) \\ \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{pF}}_r(\mathbf{q}) & \text{if } \alpha_p \geq 1 \quad (20) \\ \dot{\mathbf{o}}(\mathbf{q}) = \dot{\mathbf{pc}}(\mathbf{q}) - \dot{\mathbf{ps}}(\mathbf{q}) & \text{if } 1 \geq \alpha_p \geq 0 \quad (21) \end{cases}$$

The gradient's continuity has been verified by ensuring that (19) and (21) are equivalent to the same expression when $\alpha_p = 0$, and that the same is true for (20) and (21) when $\alpha_p = 1$.

## IV. $\mathcal{C}^1$ function for Unknown quantification

### A. Introduction

The goal of this function is to find a next pose for a camera, at a given instant, using an occupancy grid obtained from stereo vision and updated through space carving [13]. The grid's voxels can be assigned a normal vector and are set to one of three possible states: known (i.e. perceived), unknown (i.e. occluded by perceived voxels or out of fields of vision used), and empty. The normal vector of known voxels are computed by using a normal map created from the disparity map following a common method used in computer graphics to perform bump mapping [14]. Unknown voxels are assigned a normal vector when they have at least one emtpy neighbor by considering that the normal go through the barycenter of all empty neighbors.

We want to maximize the area of unknown voxels that will be visible from the next robot's pose in order to reduce the number of required viewpoints and motions. A new formula to quantify the amount of unknown voxels visible depending on the camera pose was thus used. Although this amount can be effectively deduced with basic algorithms, in order to use it as a criterion to minimize in the PG, we need a function which is at least of class $\mathcal{C}^1$. Our new function is inspired by the splatting algorithm [15] where voxels projection on the image plane are represented by a pre-defined kernel.

### B. Function to minimize

In the present work, a voxel is considered as a sphere, their influence on any pixel $(x, y)$ in the resulting image can then be expressed as a 2D Gaussian function:

$$G_i(\mathbf{q}) = \exp\left(-0.5\left(\frac{(x - X_i(\mathbf{q}))^2}{\sigma_i(\mathbf{q})^2} + \frac{(y - Y_i(\mathbf{q}))^2}{\sigma_i(\mathbf{q})^2}\right)\right) \tag{22}$$

$(X_i(\mathbf{q}), Y_i(\mathbf{q}))$ are the coordinates of the perspective projection of the voxel $i$'s center $v_i$ on the camera image plane. They are computed relatively to the camera focal length $f$, its position $\mathbf{C}(\mathbf{q})$ and its orthonormal basis vectors ($\mathbf{e_i}$, $\mathbf{e_j}$, $\mathbf{e_k}$):

$$Z_i(\mathbf{q}) = (\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_k} \tag{23}$$

$$X_i(\mathbf{q}) = f\,\frac{(\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_i}}{Z_i(\mathbf{q})} \tag{24}$$

$$Y_i(\mathbf{q}) = f\,\frac{(\mathbf{V}_i - \mathbf{C}(\mathbf{q})) \cdot \mathbf{e_j}}{Z_i(\mathbf{q})} \tag{25}$$

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

$$\sigma_i(\mathbf{q}) = f\,\frac{\sigma}{Z_i(\mathbf{q})} \tag{26}$$

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. $S_i = -1$ if the voxel is known or equal to 1 if the voxel is unknown. The empty voxels are ignored in this algorithm. To help deal with occlusions, a weight is defined for each voxel depending on their distance to the robot camera. This weight should get bigger when the voxel is closer to the camera:

$$D_i(\mathbf{q}) = \exp\left(-\sigma_d\left(\frac{Z_i(\mathbf{q}) - Z_{min}}{Z_{max} - Z_{min}}\right)^2\right) \tag{27}$$

The weight value depends on three values arbitrarily set: $\sigma_d$, $Z_{\min}$ and $Z_{\max}$. The $\sigma_d$ parameter influences the discrimination based on distance. $Z_{\min}$ and $Z_{\max}$ are parameters delimiting the maximum and minimum distance between the camera and the object's voxels. They help influence further the

distance-based discrimination but must be coherent with the allowed robot movement space and the relative object position.

Finally another coefficient is added to enhance the voxel occlusions handling by using the voxel's normal vector $\mathbf{n_i}$:

$$N_i(\mathbf{q}) = \exp\left(-0.5\left(\frac{\mathbf{n_i}\cdot -\mathbf{e_k} - 1}{\sigma_n}\right)^2\right) \tag{28}$$

The $\sigma_n$ parameter is chosen so that angles superior to 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:

$$P_{x,y}(\mathbf{q}) = \sum_{i=0}^{N} S_i\, G_i(\mathbf{q})\, D_i(\mathbf{q})\, N_i(\mathbf{q}) \tag{29}$$

Depending on the closest visible voxel status, $P_{x,y}$ is then supposed to be either negative or positive. 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 $P_{x,y}$, the pixel contribution on the total area of unknown currently visible can be found. The continuous threshold function used is a sigmoid defined as:

$$T(x) = \left(1 + \exp\left(-\alpha\, x\right)\right)^{-1} \tag{30}$$

The $\alpha$ 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. 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:

$$A_{tot}(\mathbf{q}) = \sum_{x=0}^{W}\sum_{y=0}^{H} T\left(P_{x,y}(\mathbf{q}) - \epsilon\right) \tag{31}$$

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

### C. Gradient formulation

The optimization method tries to find the minimum of the objective function by using its 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.

$$-\dot{A}_{tot}(\mathbf{q}) = -\frac{\partial}{\partial \mathbf{q}} \sum_{x=0}^{W}\sum_{y=0}^{H} T\left(P_{x,y}(\mathbf{q}) - \epsilon\right) \tag{32}$$

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

$$-\dot{A}_{tot}(\mathbf{q}) = -\sum_{x=0}^{W}\sum_{y=0}^{H} \frac{e^{-\alpha\left(P_{x,y}(\mathbf{q})-\epsilon\right)}}{\left(1 + e^{-\alpha\left(P_{x,y}(\mathbf{q})-\epsilon\right)}\right)^2}\, \alpha\, \dot{P}_{x,y}(\mathbf{q}) \tag{33}$$

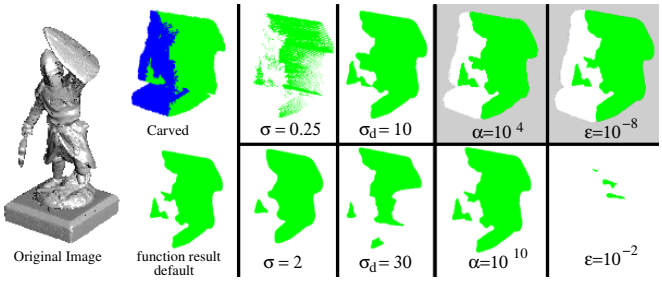


Fig. 2. Influence of the $\mathcal{C}^1$ function's parameters on the unknown area visibility estimation.

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

$$\dot{P}_{x,y}(\mathbf{q}) = -\sum_{i=0}^{N} S_i\, G_i(\mathbf{q})\, D_i(\mathbf{q})\, N_i(\mathbf{q})\, \dot{\Psi}(\mathbf{q}) \tag{34}$$

with

$$\begin{aligned}\Psi(\mathbf{q}) =& \frac{\left(x - X_i(\mathbf{q})\right)^2 + \left(y - Y_i(\mathbf{q})\right)^2}{2\,\sigma_i(\mathbf{q})^2} \\ &+ \frac{\sigma_d\,\left(Z_i(\mathbf{q}) - Z_{\min}\right)^2}{\left(Z_{\max} - Z_{\min}\right)^2} + \frac{\left(\mathbf{n_i}\cdot -\mathbf{e_k} - 1\right)^2}{2\,\sigma_n^2}\end{aligned} \tag{35}$$

As the function depends only on the robot head position and orientation, we can compute first 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:

$$\dot{\Psi}(\mathbf{q}) = \left(\frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{C}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_i}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_j}}\ \frac{\partial\Psi(\mathbf{q})}{\partial\mathbf{e_k}}\right)\left(\dot{\mathbf{C}}\ \dot{\mathbf{e_i}}\ \dot{\mathbf{e_j}}\ \dot{\mathbf{e_k}}\right)^T \tag{36}$$

The gradient's continuity was verified by developing all partial derivatives.

## V. Simulation

### A. Stability

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 robot camera must be looking at a specified point using a specified view direction vector. By modifying the target point and view direction vector, different poses were obtained where the projection of the CoM on the floor was lying on the segment between the feet contact point. A generated pose is illustrated in Fig.1 where the viewing vector is aimed at the object center and is rotated 30 degrees vertically and 15 degrees horizontally.

### B. Unknown area estimation

*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 by judging visually the images obtained. Fig.2 presents some samples of images computed

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

TABLE I
COMPUTATION TIME TABLE

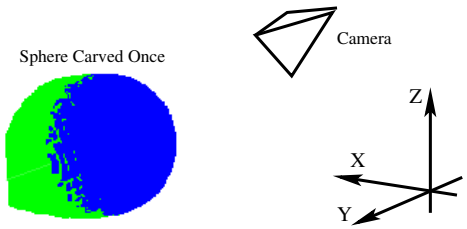


Fig. 3. Setup to test the function variations relatively to the camera movement around a sphere. Known voxels are represented in blue and unknown ones in green.

depending 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 $\epsilon = 10^{-4}$. A small value of $\sigma_d$ results in occlusions by known voxels not correctly rendered when a relatively large amount of unknown voxels is behind. On the other hand, distant voxels may not appear with a big $\sigma_d$ value. Small values for $\alpha$ and $\epsilon$ create a background noise, rendered in light gray in the figure, as the values of the function $P_{x,y}(\mathbf{q})$ equal or are close to 0 resulting then in $T(x) = 0.5$. Big values for $\epsilon$ make the function too restrictive. Setting $\alpha$ to a high value gives a correct rendering but the function reacts as a discrete function with higher variations of the gradient.

*2) Computation time:* Our formulation depends on the size of the camera image and the number of voxels. Practically for an accurate result, the process involves a high number of pixels and voxels. Some examples of computation time to evaluate the unknown area are presented in table I for 3 objects with different number of voxels and 2 different image size. Tests were performed with a C implementation of the algorithm with 2 threads, on an Intel Xeon 3.2GHz processor with 1GB of RAM under an Ubuntu OS. Some speed optimization techniques were applied: only pixels of the image which are close to a voxel's center projection are considered, and a parallel implementation of the algorithm was realized.

*3) Comparison of the function results with OpenGL rendering:* To ensure that the function's 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. An example of test setup is illustrated in Fig.3. The corresponding results, obtained with the default parameters for the function, are shown in Fig.4. Though the resulting values

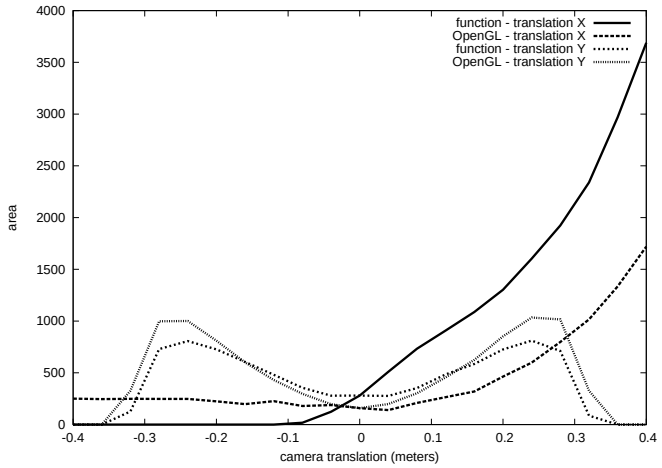


Fig. 4. Comparison of the amount of unknown area visible depending on camera position for our evaluation method and a basic voxel rendering method.

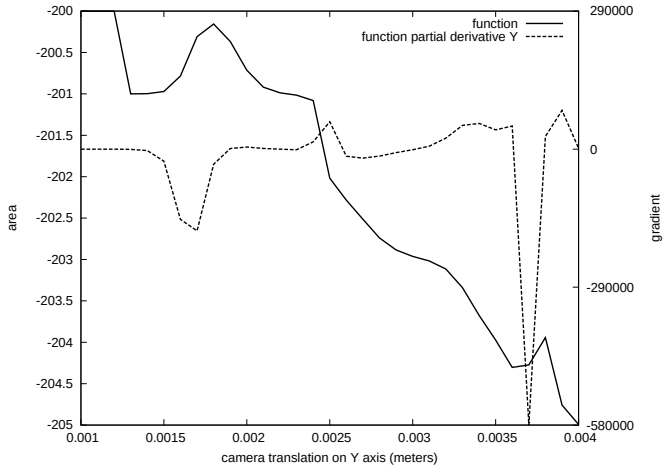


Fig. 5. Close-up of area and gradient depending on camera position on the Y axis.

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.

*4) 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.5 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. Unfortunately this affects badly the optimization process which therefore cannot converge properly, and reflects in the computation time. Our future work is to investigate the resolution of this problem.

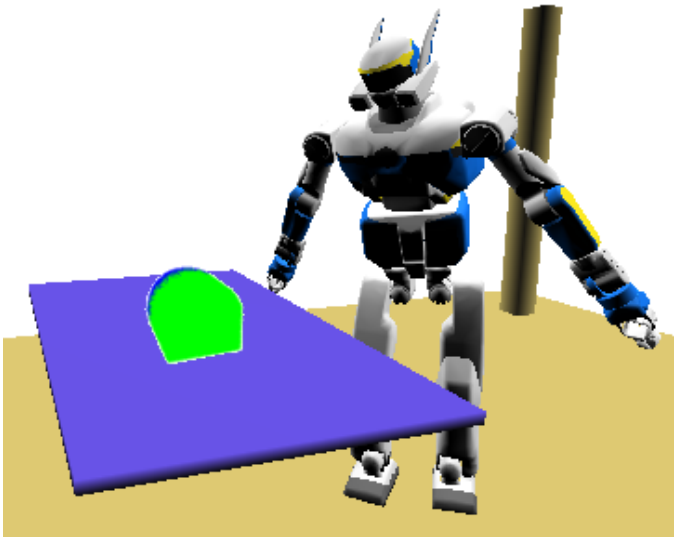

Fig. 6. Pose generated using our NBV algorithm.

### C. Pose generation

We tested our posture generation solution in simulation using two virtual objects: a sphere and the soldier shown in Fig. 2. Two main problems with our criterion (31) need to be faced: (i) the computation time as it takes from many seconds to 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 few hours. In such experiments, the optimization algorithm could solve all constraints easily but got 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. 6 gives the result of a typical pose generated from the initial robot posture in front of the object. The humanoid moved 102 cm 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.

## VI. Conclusion

A new stability constraint (17) specific to a humanoid and a new $\mathcal{C}^1$ function (31) for visual unknown quantification were introduced in this work. The stability constraint allows us to generate statically stable postures where feet position and orientation do not need to be specified. The introduced function for quantification is able to compute an estimation of the amount of unknown area visible from a specific camera location by taking into account occlusions between known and unknown voxels. Its result matches those of algorithmic methods confirming its estimation accuracy, and making it a pertinent criterion for a local search of a next best view. With further optimizations and tuning of the formulation, we hope this function can be of particular use coupled with a global planning method in order to complete the modeling of an unknown object by a humanoid. We are actually investigating on possible refinements of the function results and the completion of the whole autonomous modeling process.

## Acknowledgment

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`.

The soldier 3D model used for tests is provided courtesy of INRIA by the AIM@SHAPE Shape Repository `shapes.aim-at-shape.net`.

The visualization of the experimental setup relied on the AMELIF framework presented in [16].

## References

[1] F. Saidi, O. Stasse, K. Yokoi, and F. Kanehiro, "Online object search with a humanoid robot," in *IEEE/RSJ IROS*, 2007.

[2] 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/RSJ Conference on Humanoids Robots*, 2007.

[3] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[4] J. Sanchiz and R. Fisher, "A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom," in *British Machine Vision Conference*, 1999.

[5] D. Lowe, "Local feature view clustering for 3d object recognition," in *IEEE CVPR,*, 2001.

[6] J. Banta, Y. Zhien, X. Wang, G. Zhang, M. Smith, and M. Abidi, "A best-nextview algorithm for three-dimensional scene reconstruction using range images," in *Proceedings SPIE*, 1995.

[7] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[8] K. Yamazaki, M. Tomono, T. Tsubouchi, and S. Yuta, "3-d object modeling by a camera equipped on a mobile robot," in *IEEE ICRA Proceedings*, 2004.

[9] K. Tarabanis, P. Allen, and R. Tsai, "A survey of sensor planning in computer vision," in *IEEE Transactions on Robotics and Automation*, 1995.

[10] W. Scott, G. Roth, and J. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Comput. Surv.*, 2003.

[11] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on hrp-2," in *IEEE/RSJ IROS*, 2006.

[12] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids collision-free optimized postures," in *IEEE RAS/RSJ Conference on Humanoids Robots*, 2007.

[13] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, 1999.

[14] A. Hertzmann, "Introduction to 3d non-photorealistic rendering: Silhouettes and outlines," in *SIGGRAPH '99 Course Notes. Course on Non-Photorealistic Rendering*, 1999.

[15] L. Westover, "interactive volume rendering," in *Symposium on Volume Visualization*, 1989.

[16] P. Evrard, F. Keith, J.-R. Chardonnet, and A. Kheddar, "Framework for haptic interaction with virtual avatars," in *17th IEEE International Symposium on Robot and Human Interactive Communication (IEEE RO-MAN 2008)*, 2008.