Environment recovery by range scanning with a next-best-view algorithm *

J.M. Sanchiz †, R.B. Fisher §

†Departament d'Informàtica, Universitat Jaume I, E-12071 Castelló, Spain

§Division of Informatics, Institute of Perception, Action and Behaviour

University of Edinburgh, 5 Forrest Hill, Edinburgh EH1 2QL, UK

Abstract

We present an algorithm for determining the next best position of a range sensor in 3D space for incrementally recovering an indoor scene. The method works in five dimensions: the sensor navigates inside the scene and can be placed at any 3D position and oriented by a pan-tilt head. The method is based on a mixed exhaustive search and hill climbing optimisation, and outputs the next position in usable time. Results are shown with a simulated mobile range sensor navigating in CAD models of environments (closed rooms).

1 Introduction

Environment recovery by a sequence of range scans has had some research in recent years [1] [2]. Its aims are to build a realistic computerised model of an indoor or outdoor scene including surface information and texture, so that a user can navigate inside the model. Such models can be placed on the internet, allowing remote access. Among the applications we can mention are virtual

*Work supported by the European Union funded SMART II (ERB4061PL950841) and CAMERA (ERB FMRX-CT97-0127) research networks

reality, remote scene inspection, maintenance, 3D tourist models, 3D building catalogue, architecture, archaeology, etc.

Range sensors used in environment recovery have a wide (panoramic in some cases) field of view, and the task of completely recovering an environment is performed usually by placing the sensor on a tripod, taking some views by moving the sensor manually, and merging the views together to build the model.

Although general interesting environments, like heritage buildings, are not usually suitable for a mobile robot to navigate inside, in some cases it is feasible to fully automate the process of scan acquisition by using a sensor mounted on such a robot. The robot is instructed to do the work in a more intelligent way by taking into account some considerations to find the next best position to place the sensor. Such cases could include, for example, wide corridors or empty rooms with interesting features such as in a monastery, palace or castle; or a large interesting object to be fully mapped around.

This paper describes a method for obtaining a complete and accurate three-dimensional recovery of an unknown indoor environment, placing the sensor at the best position each time. Different views are used to build up an incremental environment model. So the problem is stated as finding the next view that would best improve the current recovered model.

We consider a mobile range sensor. In general 3D motion this is a problem with six degrees of freedom, since the sensor can be placed anywhere in space, and can be oriented by three rotations: pan (rotated around the vertical axis), tilt (rotated around the horizontal axis), and roll (rotated around the optical axis). Sensor rolling can be useful because the horizontal and vertical apertures of the sensor may be different, so different scene areas can be recovered by rolling the sensor. In this work, we assume that rolling is not allowed because conventional pan-tilt heads do not perform this motion. So we can think of a mobile base which moves over a floor, with a vertical bar to lift or lower the sensor, which is mounted on a pan-tilt head at the end of the bar. This makes the problem of finding the next best view a five-dimensional one.

Other research has addressed the next best view problem for object recon-

struction where the outsides of objects are seen, but not for 3D environment recovery, where the inside of a scene is explored and the sensor can navigate into the scene. Massios and Fisher [3] compute the next best position of a range sensor for object acquisition with orthogonal projection. The sensor position space was two-dimensional: a sphere at a fixed radius enclosing the object with the sensor always pointing to the centre. As in our approach they defined a quality criterion and used a voxel map for view reasoning. García et al. [4] also addressed a similar 2D sensor space, although discretised in a different way. They used a voting scheme to compute the next view, maximising the observation of occluded areas, and used a triangular mesh to model the object.

Reed at al. [5] presented a method to recover object models including the computation of the next view to maximise occluded areas. They computed visibility volumes from where occluded areas are fully visible, following the method presented by Tarabanis et al. [6]. But that research did not solve for the best position of the sensor inside the visibility volumes. In that work Reed et al. assumed a 2D sensor position space, an enclosing sphere, and computed its intersection with the viewing volumes. Pito [7] proposed a method for view planning in object modelling with 4 degrees of freedom. The sensor position space was a cylinder, and it could be oriented within a range of pan-tilt angles. He used a voting scheme to maximise the observation of occluded areas.

In summary, the question that this paper addresses is: **Is it possible to define an effective and efficient algorithm for scanning an environment such that all surfaces are observed with high quality measurements?** The results shown here answer the question positively, and we believe this is the first implemented algorithm to do so.

2 Scene representation

Our objective is not to recover a surface model of the scene. Instead, we aim at finding the set of sensor poses that best acquire the 3D points according to some criteria (explained later). Our recovered scene model has to be accurate enough

to compute these criteria. The surface model can later be recovered from the 3D points.

We use a voxel map representation. Voxels, volume elements, are small cubes of a fixed size. The voxel map is a 3D rectangle whose size depends on the available memory, size of the scene to be modelled, and resolution at which we work. The voxel map is implemented as a 3D circular buffer, and it can be placed anywhere in space, so that if new 3D points are sensed that do not fit in the voxel map, new space can be allocated for the new area without moving data in memory.

Thus, a voxel map can only represent a quantised scene model, made of cubes. Nevertheless, it is possible to compute a realistic triangular-mesh scene model using the "Marching Cubes" algorithm [8] on the voxel map, from the 3D points that fall inside the *occupied* voxels. Thus we could obtain a triangular mesh where the triangle size is of the order of the voxel size. This dense triangular mesh could later be simplified by a mesh optimisation algorithm [9]. Other works have computed realistic surface models from range data, for example Wolfart et al [10] found a triangular mesh from every range image and did a registration of the new mesh with the scene model (also a triangular mesh). They included texture information to make it more realistic.

A voxel map representation allows ray tracing by the 3D Bresenham algorithm [11] using only integer operations. It also allows a straightforward registration of the new sensed points with the recovered scene model [3] (voxel map update) if we assume that the voxel size is bigger than the errors that may arise in the sensed point positions due to inaccurate sensor placement (navigation errors).

In our scene model a voxel consists of a *label* indicating its type, a *surface normal*, and a *quality*, indicating how accurately this voxel has been sensed so far. The voxel labels include:

- *Unmarked*. A voxel that has never been observed by the sensor.
- *Empty*. A voxel that has been observed and found to be empty.

- *Occupied.* A voxel in which 3D sensed points have fallen.
- *Occluded.* A voxel so far occluded by an occupied voxel.
- *Occlusion Plane.* A special kind of *Occluded* voxel adjacent to an *Empty* voxel through any of its six faces.

A voxel's *surface normal* and *quality* are only defined for *Occupied* voxels. Normals are estimated at every point in a range image. The voxel *surface normal* is the average of the normals of all the range points that have updated this voxel. The *sensed quality* of an *Occupied* voxel is the cosine of the angle formed by the *surface normal* and the viewing ray. The voxel *quality* is the best *sensed quality* of the voxel so far.

3 Voxel map update

Every time a new range image is taken the voxel map is updated, incrementing our knowledge of the scene.

A range image is a matrix $[d_{uv}]$ ($u \in [0..N-1]$, $v \in [0..M-1]$) of distances sensed in the direction $\mathbf{R}_{\phi}^{\vec{y}}\mathbf{R}_{\theta}^{\vec{z}}\vec{n}_{uv}$ from the optical centre $\vec{c} = (x_0, y_0, z_0)'$ (see Fig. 1). (ϕ, θ) are the pan-tilt angles the sensor is oriented, and $\mathbf{R}_{angle}^{axis}$ represents a 3D rotation of $angle$ about $axis$.

x_0, y_0, z_0, ϕ and θ are the 5 extrinsic parameters of the sensor, the 5 degrees of freedom of the problem of sensor placement. $\vec{n}_{uv}$ are normalised vectors computed for a specific sensor geometry (spheric, tilted line scan, line striper) and for a value of the sensor intrinsic parameters. Details on the sensor model used will be given later.

The 3D co-ordinates of a sensed point, $\vec{p}_{uv}$, are computed by adding to the optical centre $\vec{c}$ the distance d_{uv} in the direction of the corresponding ray $\vec{n}_{uv}$ rotated by the pan and tilt angles of the sensor:

$$\vec{p}_{uv} = \vec{c} + d_{uv}\mathbf{R}_{\phi}^{\vec{y}}\mathbf{R}_{\theta}^{\vec{z}}\vec{n}_{uv} \tag{1}$$

A voxel is identified by its three indices in the voxel map $(i, j, k)'$. The centre voxel $(0, 0, 0)'$ is placed in space at world co-ordinates $(x_w, y_w, z_w)'$ and

the voxel array is aligned with the world co-ordinate axes. A voxel is a cube of side $scale$, so a point in 3D space $(x, y, z)'$ falls inside voxel
$(RoundScale(x - x_w), RoundScale(y - y_w), RoundScale(z - z_w))'$,
where

$$RoundScale(x) = Round(\frac{x + sgn(x)\frac{scale}{2}}{scale}) \tag{2}$$

The voxel map is updated with the following method:

- Compute the camera position in the voxel map $c^{voxel} = RoundScale(c)$.
- For every range image point p_{uv} compute its voxel co-ordinates $p_{uv}^{voxel} = RoundScale(p_{uv})$ using (1–2).
  - Compute the intersection l_{uv}^{voxel} of the ray $\vec{c} + \lambda \mathbf{R}_{\phi}^{\vec{y}} \mathbf{R}_{\theta}^{\vec{z}} \vec{n}_{uv}$ with the limits of the voxel map in voxel co-ordinates.
  - Do ray tracing from c^{voxel} to p_{uv}^{voxel} marking as *Empty* the voxels that are not *Occupied*. Due to the digital nature of the approach (voxel map and Bresenham ray tracing), it is possible that a ray crosses already *Occupied* voxels, which are not modified.
  - Do ray tracing from p_{uv}^{voxel} to l_{uv}^{voxel} marking as *Occluded* the voxels that are still *Unmarked*.
  - Mark voxel p_{uv}^{voxel} as *Occupied*.
- Traverse the voxel map marking as *Occlusion Plane* the voxels that are of type *Occluded* and have a face touching an *Empty* voxel.

4 Fitness function

In order to compute the best next position we have set some criteria for the goodness of a sensor pose, which are formulated as a mathematical function to maximise. The criteria are defined on the scene area that a test position covers, and are:

- Providing sufficient overlap with previously acquired data for fine registration of the data (as wheel slip on the vehicle is likely to introduce dead-reckoning registration errors)
- Eliminating occlusion plane areas
- Observing new unseen areas

Let a_{ov} be the proportion of voxels intersected by the sensor rays that are already occupied (i.e. the proportion of the new view that overlaps previously acquired data), a_{op} be the proportion of voxels marked occlusion plane, and a_{us} be the proportion of voxels marked unseen area. $a_{ov}, a_{op}, a_{us} \in [0..1]$ and $a_{ov} + a_{op} + a_{us} = 1$. The data for computing these values comes from projecting the current scene model, when viewed from a viewpoint and direction, onto an internal image plane. The function to be maximised has been designed with these characteristics:

- A unique maximum at a certain value of a_{ov} (we have fixed 40% for this value) and for $a_{op} = a_{us}$ (thus favouring at the same time the equal sensing of occlusion plane and unseen areas).
- Zero at $a_{ov} = 0$, forcing always some overlap.
- A fixed value greater than zero at $a_{ov} = 1$, to make possible views with no occlusion planes or unseen areas. This can occur at late stages of the scene recovery, when all parts have been observed and new views aim at increasing the quality of the sensed data.

A simple (polynomial) function that satisfies the above criteria is

$$f_{area} = \left(5a_{ov}^3 - 10.5a_{ov}^2 + 6a_{ov}\right)\left(1 - \frac{1}{2}\left|a_{op} - a_{us}\right|\right) \tag{3}$$

This function has a maximum of 1.04 at $a_{ov} = 0.4$ and $a_{op} = a_{us}$, a local minimum of 0.5 at $a_{ov} = 1$, value 0 at $a_{ov} = 0$, and decreases as a_{op} differs from a_{us}. In the space defined by axes (a_{ov}, a_{op}, a_{us}), the domain of f_{area} is the triangle defined by the plane $a_{ov} + a_{op} + a_{us} = 1$ and the conditions $a_{ov}, a_{op}, a_{us} \in [0..1]$. Fig. 2 shows the domain triangle and the shape of f_{area}.

The area-based evaluation has the advantage that small occluded areas will tend to be examined more closely, since the overlapping area attempts to be about 40%. This forces the sensor to approach unobserved areas until they occupy about 60% of the image. On the other hand, and for the same reason, big occluded areas, like those hidden by a salient corner, will tend to be imaged from a further distance, and if new detail appears (in the form of occlusion planes), it will be examined closer in further views.

We have qualified as *basic* the above area-based criteria. Other criteria can be represented by factors $f_i \in [0..1]$ which multiply by f_{area}, thus increasing the total evaluation, $f = f_{area} \prod_i (1 + f_i)$. These secondary criteria may include:

- Quality improvement, $f_{quality}$.
- Structure of the overlapping area, $f_{structure}$. The purpose of this factor is to favour the sensing of areas where the surface has non-degenerate shape, thus easing the registration.
- Navigation cost, $f_{navigation}$, modelling the cost of reaching a new position and orientation from the current position of the sensor.

From these proposed criteria we have only implemented and tested $f_{quality}$, leaving the other two to further work. $f_{structure}$ could be computed from the variance of the surface normals at the new sensed points. $f_{navigation}$ should rely on robot path planning, reasoning about the known obstacles in the scene, the distance to travel, and trajectory of the robot. We have implicitly introduced a simple navigation factor in the definition of the feasible space when optimising the fitness function, but a reliable navigation factor should be computed by a navigation module.

For $f_{quality}$ we use the ratio of *Occupied* voxels that would improve quality from this view to the total number of *Occupied* voxels updated, multiplied by the mean quality improvement. Clearly $f_{quality} \in [0..1]$. The total fitness function becomes then

$$f = f_{area}(1 + f_{quality}) \tag{4}$$

5 Optimisation

The feasible space is related to the physical characteristics of the sensor. If the sensor is mounted on a mobile base that moves on the floor, the mobile base cannot even move safely unless areas of the floor have been scanned and an obstacle-free path is found. Nevertheless, a navigation reasoning module is not taken into account in the present work, so we define the feasible space by these simple constraints:

1. The sensor must lie inside an *Empty* voxel.
2. The nearest *Occupied* voxel is not closer than $K \times scale$ (K times the voxel size) so the sensor does not collide with obstacles ($K = 4$ has been used).

The search strategy to optimise the fitness function could be one of the following:

- Exhaustive search. This will find a global maximum but it would be extremely costly due to the five degrees of freedom of the problem.
- Hill climbing methods. They will end up in a local maximum close to the starting position of the search.
- Statistical methods: simulated annealing. These would need lots of fitness function computations, and are not guaranteed to end up in a global maximum.
- Evolutionary methods: genetic algorithms. It would require an extremely large number of function computations to maintain a population of test positions.

To choose the search method one has to consider that our goal is to provide an answer, the next best position, in a reasonable amount of time. Several dozens of views will be necessary to recover a normal-sized room, so response times of the order of a minute, at most, are desirable. The fitness function is based on ray tracing on a subsampled range image used for view prediction,

while the full range image is later used for modelling. Sensor sizes may be about 50-250 thousand points, and after subsampling for view prediction the number of points may be still of about two thousand (64×32 for example).

We use a mixed method: exhaustive search in the 2D space formed by the pan-tilt angles, and a hill climbing method in the 3D space of sensor positions.

To perform an exhaustive (coarse) search in the pan-tilt space, we choose the centres of the 20 faces of an icosahedron as the values to test. These orientations are evenly distributed around a sphere, and in case there is spare time, a face can be subdivided as shown in Fig. 3 providing four new faces, which can be further subdivided to the desired resolution. So, provided the next best position and orientation of the sensor is worked out at this resolution, the orientation can be refined as desired.

For the hill climbing optimisation we use the N-dimensional *simplex* method [12]. The method starts from the current position of the sensor, and finds a nearby local maximum. A simplex in 3D space is a tetrahedron (Fig. 3). The vertices of the initial simplex are set as follows: the first vertex is set as the current sensor position, the other three are set randomly choosing an icosahedron face [1..20] and a ray within this face. These directions are projected a random distance (within a range).

A simplex evolves in 3D space changing its shape, size and position, aiming at high values of the fitness function. This is done by performing reflections of the worst point through the opposite face, expansions of a point along the direction of the opposite face, 1D contractions of the worst point toward the opposite face, and 3D contractions of all but the best point toward this point. A deeper explanation of the simplex optimisation algorithm can be found in [12].

The simplex optimisation is stopped when the range of change of the fitness function among the four vertices of the simplex is below a threshold. This will always be reached eventually since the simplex gets smaller as it contracts toward vertices where the fitness function is better, and at a certain iteration the whole simplex will be contained inside a unique voxel, so the fitness of its four vertices will be the same, and the range will be zero.

The combination of the exhaustive search in the pan-tilt space and the simplex method in position space is done by computing the fitness of all 20 directions (faces of the icosahedron) at every position tested (a simplex vertex), and keeping the best evaluation as the fitness for that position.

The termination criterion is aimed at ensuring that the whole scene is recovered, and it can be:

1. No more unseen area is covered.
2. No more unseen area is covered and the quality of every pixel is above a threshold.
3. No more unseen area is covered and no more quality improvements are achieved.

We used criterion 2.

The covered area and the occlusion plane area can be roughly computed as the number of voxel faces that touch an empty voxel, times the area of a voxel face.

6 Results

Although the proposed approach does not guarantee the selected best view is globally the best, which could be computed by exhaustive search in five dimensions, it provides a feasible solution which:

- is locally a maximum of the fitness function,
- is near to the previous sensor position,
- improves quality of the covered area, and covers occlusion planes and new unseen areas.

To show the goodness of the method, experiments have been carried out using a simulated range sensor and mobile base. The base is able to move forward/backwards, left/right, and lift the sensor up/down. It can rotate, thus

panning the sensor, besides the sensor can tilt from 0 to 180 degrees. The simulated sensor navigates in a scene model built with a CAD tool, accepting commands through a UNIX socket to perform the motion and to take range images, which are also piped through the socket.

The sensor simulator provides a $2\frac{1}{2}$D range image, implementing a tilted line scan geometry. Its intrinsic parameters are: $N = 64$, $M = 30$ (observing 64×30 points), horizontal angular aperture $\alpha = 60$ deg, and vertical angular aperture $\beta = \tan^{-1}(\frac{M}{N} \tan \alpha)$. The sensor rays, Fig. 4, are computed as:

$$\begin{cases} \phi_u = \alpha(\frac{1}{2} - \frac{u}{N-1}) \\ \theta_v = \beta(\frac{1}{2} - \frac{v}{M-1}) \\ \vec{a} = \mathbf{R}^{\vec{x}}_{\theta_v} \vec{z} \\ \vec{n}_{uv} = \mathbf{R}^{\vec{a}}_{\phi_u} \mathbf{R}^{\vec{x}}_{\theta_v} \vec{y} \end{cases} \tag{5}$$

Where again $\mathbf{R}^{axis}_{angle}$ represents a 3D rotation of $angle$ about $axis$.

The test environments used consisted of closed rooms, Fig. 5, a room of $5 \times 3 \times 3$ metres with some boxes inside, and a room of $6 \times 4.5 \times 3\, m$ with three columns. The voxel size was $scale = 10\, cm$ and the voxel map was of $64 \times 48 \times 32$ voxels. We carried out two experiments with each environment, one with the quality factor switched off ($f_{quality} = 0$), and another taking this factor into account. The experiments were run till the 5D space to optimise the fitness function was almost flat (60 views with the quality factor off and 300 views with the factor on). The main difference was that in the second experiment the sensor reexamined previously scanned walls that had low quality scores, by new almost-vertical views.

Fig. 6 shows the sensed points in a sample range image, and the positions tested by the optimisation method as the simplex evolved in the 3D space. The lines start at the position of the vertices (dots) and have the direction of the best evaluation, with length proportional to the fitness function.

Figs. 7 and 8 show several plots giving information of how the method works, they include: *Fitness function*, which shows that the optimisation method finds good poses for the sensor, but declines as more of the scene is recovered. *Number*

of iterations of the optimisation method: the number of function evaluations is this number multiplied by twenty (number of faces of an icosahedron). *Mean quality* of *Occupied* voxels: in the second experiment quality improves monotonically after the scene is initially completely observed, and it reaches a fairly big value. *Occupied area* in m^2: this figure stabilises after about 35 and 60 views respectively, when the whole environment has been observed at least once.

Fig. 9 shows the recovered environments after 15 views, showing only the *Occupied* voxels and the sensor positions. There is no important difference between the acquired models at this stage.

The test results show that the scene scanning is virtually complete after a reasonable number of views with or without the quality measure. A drawback of the approach is that when almost the whole scene is recovered and just few isolated occlusion planes remain, the space is almost flat regarding the fitness function, and the local hill-climbing method cannot find a good direction to "climb". In this case the simplex evolves randomly until a timeout is signaled. A few additional scans are needed to obtain the remaining isolated unobserved voxel faces, or to improve the quality. We envisage two approaches to cope with this problem:

- Direct a final "tiding-up" phase by a deterministic approach (detecting holes in the scene by morphology analysis and computing unoccluded viewpoints for them)
- Apply a morphological operator to fill up the small one voxel-wide holes and take no additional views

Test environment "boxes" has a total inside viewing surface of $89\,m^2$, and environment "columns" has $134\,m^2$. A rough estimate of the minimum number of views to cover these areas, assuming a 40% overlap, and that the sensor stays at $2\,m$ from the surfaces, covering an area of $2(2\tan\frac{\alpha}{2})(2\tan\frac{\beta}{2}) = 2.5\,m^2$ ($\alpha = 60\,deg$, $\tan\beta = \frac{M}{N}\tan\alpha$, $M = 30$, $N = 64$), would be about $\frac{89}{2.5\times 0.6} \simeq 60$ and $\frac{134}{2.5\times 0.6} \simeq 90$ views respectively. On the other hand, the maximum number of views is given by visiting all the voxels inside the room, that is

$\frac{volume(environment)}{volume(voxel)}$, giving 45000 and 81000 views respectively. As we can see the number of views taken by the present approach is quite reasonable, as it takes only about 60 views before most of the scenes are fully scanned. 60 is less than the estimated minimum number of views for the second scene; however the lower bound estimates used very approximate values for the overlap percentage and scan distance. The key point to note is that the algorithm is clearly producing an efficient scanning sequence.

The time to deduce the next observation position depends on the complexity of the recovered environment, but the experiments reported here took approximately 4421 sec for "boxes" and 4926 sec for "columns" (300 views, quality factor on) on a *Pentium* processor at 166 MHz. This is less than 16.5 sec per view on average. The storage requirements for the voxel representation were about 3/4 of a M-byte, which is also low enough for practical use.

7 Conclusions

We have presented an approach to full 3D environment recovery by a range sensor mounted on a mobile platform. The recovered model is represented by a voxel map at just enough resolution for computing the criteria to find the next best view. This allows an easy update of the model with the new views (voxel map update), while a realistic surface representation based on a triangular mesh can be computed off-line from the voxel map. During the environment recovery, one could also acquire higher resolution surface and texture data. This data would not be needed for the best next view planning process, but could be used for scene modelling.

The next best view is recovered by mixed exhaustive search and hill climbing optimisation, and the fitness function is based on area proportions of the new image to be sensed. This means that detailed areas are examined closer. We have envisaged other criteria that can modify the basic one, aimed at improving the quality of the sensed data, or at favouring the recovery of highly structured parts to ease the registration, if a realistic recovery of the scene is to

be performed by another task.

We have presented results that show the feasibility of the method. The experiments have been carried out on a simulated sensor and mobile platform navigating in CAD scenes, with and without the aim of quality improvement of the data. Recovering our sample scene with a mean quality of 0.9, for example, takes 4 times the number of views required for recovering it with any data quality. We expect to perform real-scene experiments in the future.

References

[1] D. Leevers, P. Gil, F.M. Lopes, J. Pereira, J. Castro, J. Gomes-Mota, M.I. Ribeiro, J.G.M. Gonçalves, V. Sequeira, E. Wolfart, V. Dupourque, V. Santos, S. Butterfield, D. Hogg, and K. Ng. An autonomous sensor for 3d reconstruction. In *Proc of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST'98)*, pages 26–28, 1998.

[2] V. Sequeira, K. Ng, S. Butterfield, J.G.M. Gonçalves, and D. Hogg. *3D textured models of indoor scenes from composite range and video images*, Proceedings of SPIE 3313, pages 46–58. SPIE–The International Society for Optical Engineering, 1998.

[3] N.A. Massios and R.B. Fisher. A best next view selection algorithm incorporating a quality criterion. In *Proc of the 6th British Machine Vision Conference*, pages 780–789, 1998.

[4] M.A. García, S. Velázquez, and A.D. Sappa. A two-stage algorithm for planning the next view from range images. In *Proc of the 6th British Machine Vision Conference*, pages 720–729, 1998.

[5] M.K. Reed, P.K. Allen, and I. Stamos. Automated model acquisition from range images with view planning. In *Proc of the Int Conf on Computer Vision and Pattern Recognition, CVPR'97*, pages 72–77, 1997.

[6] K.A. Tarabanis, R.Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):279–292, 1996.

[7] R. Pito. A sensor-based solution to the next best view problem. In *Proc of the 13th International Conference on Pattern Recognition, ICPR'96*, volume I, pages 941–945, 1996.

[8] W.E. Lorensen and H.E. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[9] H. Hoppe, T. de Rose, T. Du Champ, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proc of SIGGRAPH'93*, pages 19–26, 1993.

[10] E. Wolfart, V. Sequeira, K. Ng, S. Butterfield, and J.G.M. Gonçalves. *Hybrid approach to the construction of triangulated 3D models of building interiors*, Lecture Notes in Computer Science 1542, pages 489–508. Springer-Verlag, 1999.

[11] J.E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[12] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P.Flannery. *Numerical Recipes in C, the art of scientific computing.* Cambridge University Press, 1995.

List of Figures

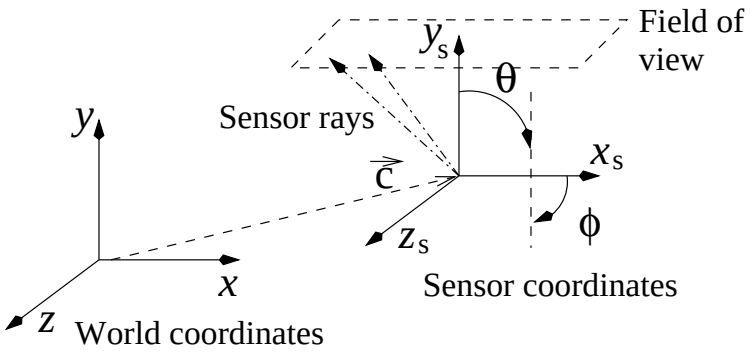

Figure 1: Range sensor extrinsic parameters

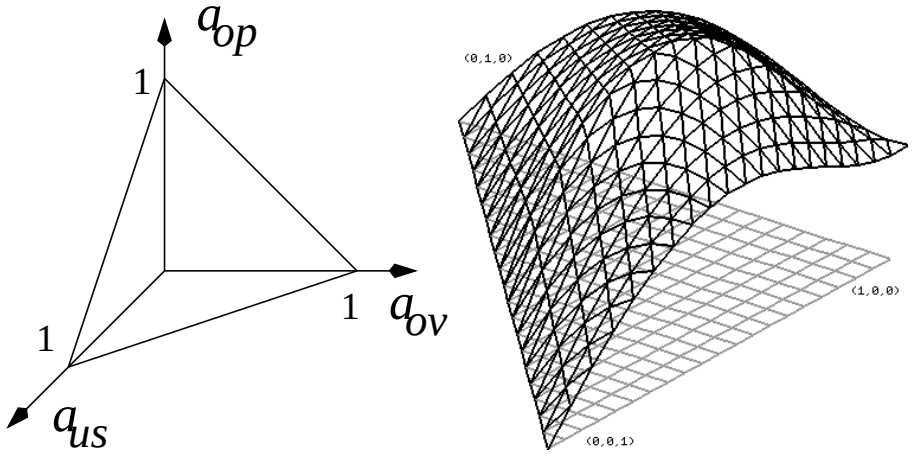

Figure 2: Area-based fitness function, f_{area}, on (a_{ov}, a_{op}, a_{us})

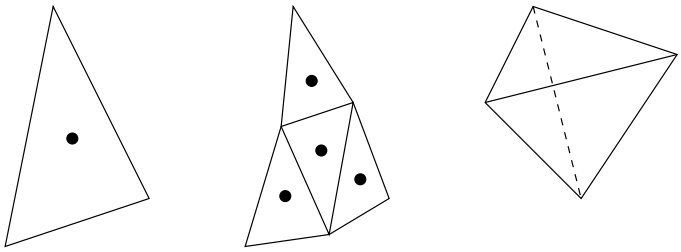

Figure 3: *Left*: face of an icosahedron. *Middle*: refinement. *Right*: simplex (tetrahedron) used in search algorithm

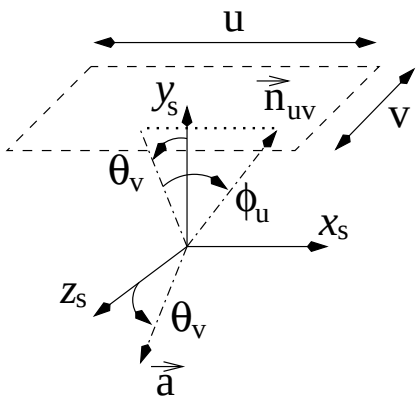

Figure 4: Range sensor geometry

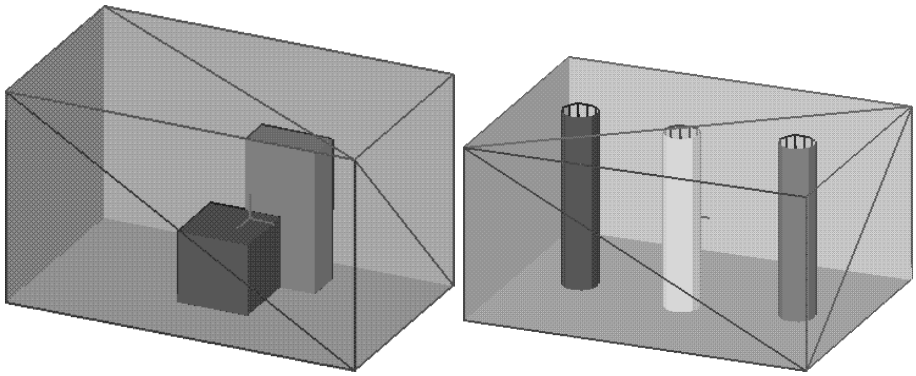

Figure 5: *Left*: test environment 1, "boxes". *Right*: test environment 2, "columns"

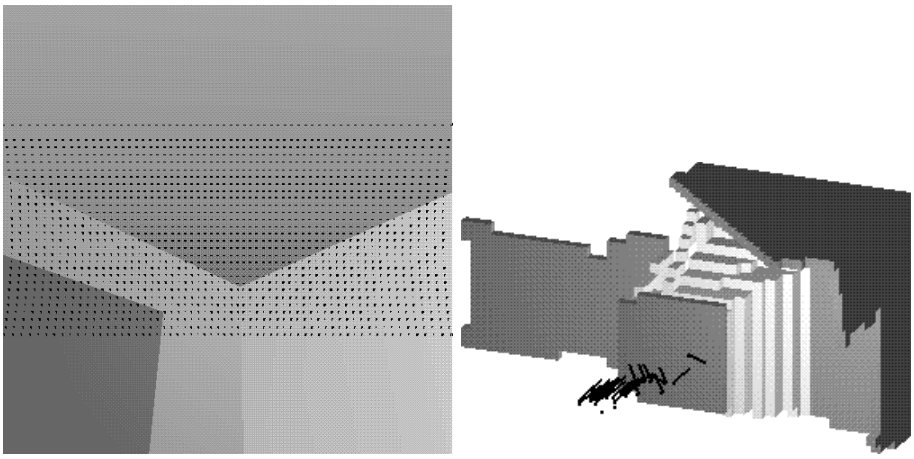

Figure 6: *Left*: sensed points (dots) on environment "boxes". *Right*: positions tested by the simplex during one optimisation cycle: black lines indicate the best direction with length proportional to their fitness

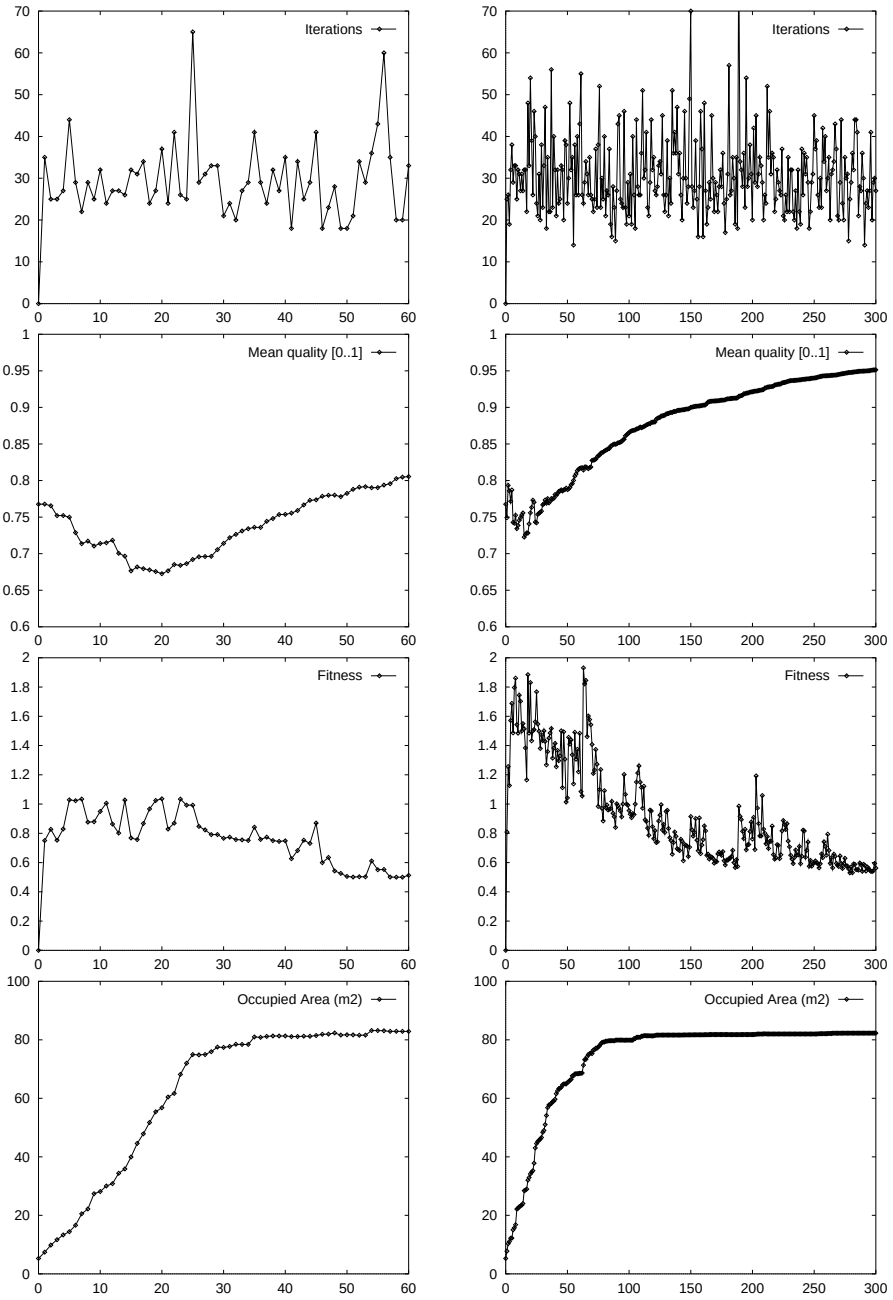

Figure 7: Recovery process for environment "boxes". Plots are versus view number. *Left column*: with the quality factor off. *Right column*: with the quality factor on. *From top to bottom*: number of iterations to locate the next best view. Optimised fitness function at the next best view. Mean quality of *Occupied* voxels. Area of *Occupied* voxels

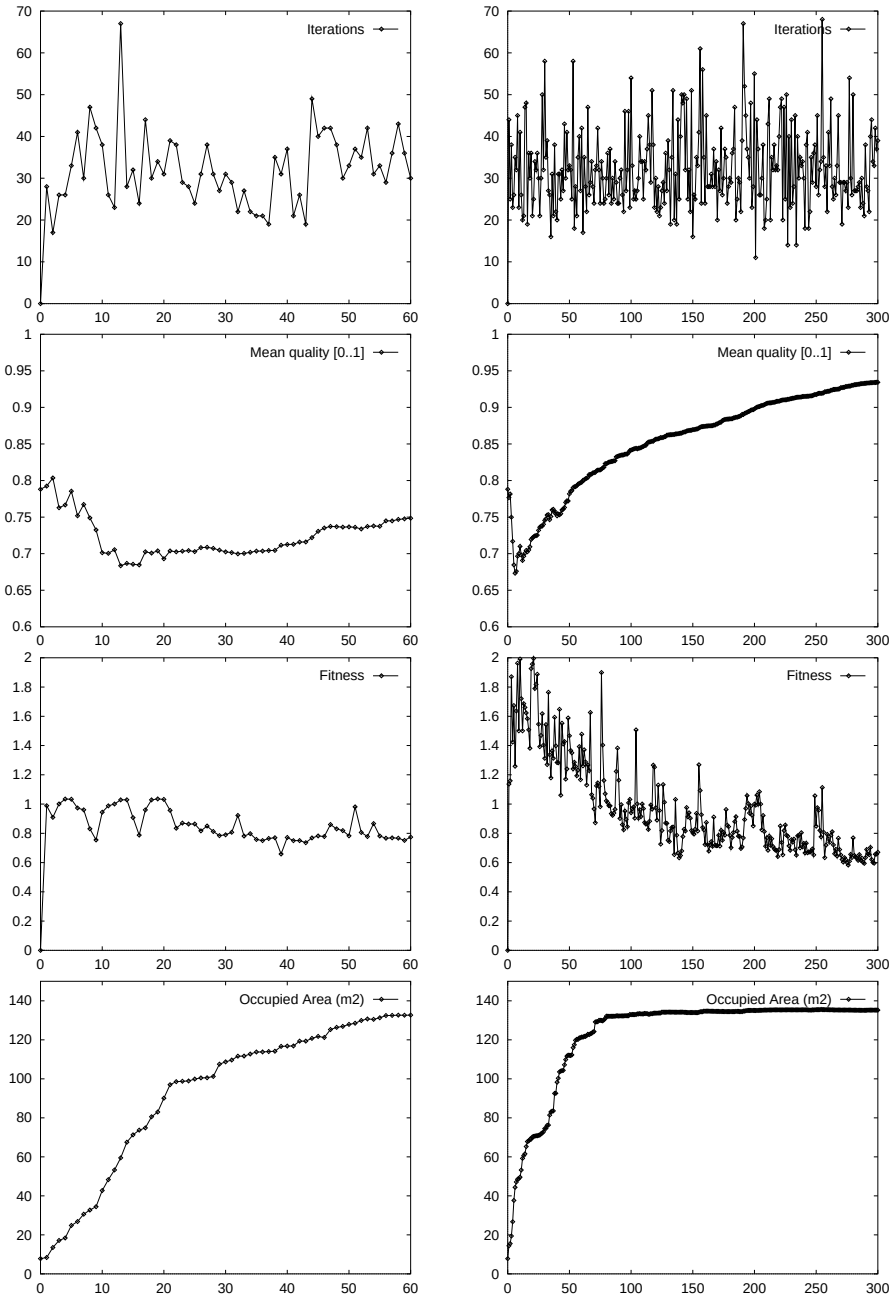

Figure 8: Recovery process for environment "columns". Plots are versus view number. *Left column*: with the quality factor off. *Right column*: with the quality factor on. *From top to bottom*: number of iterations to locate the next best view. Optimised fitness function at the next best view. Mean quality of *Occupied* voxels. Area of *Occupied* voxels

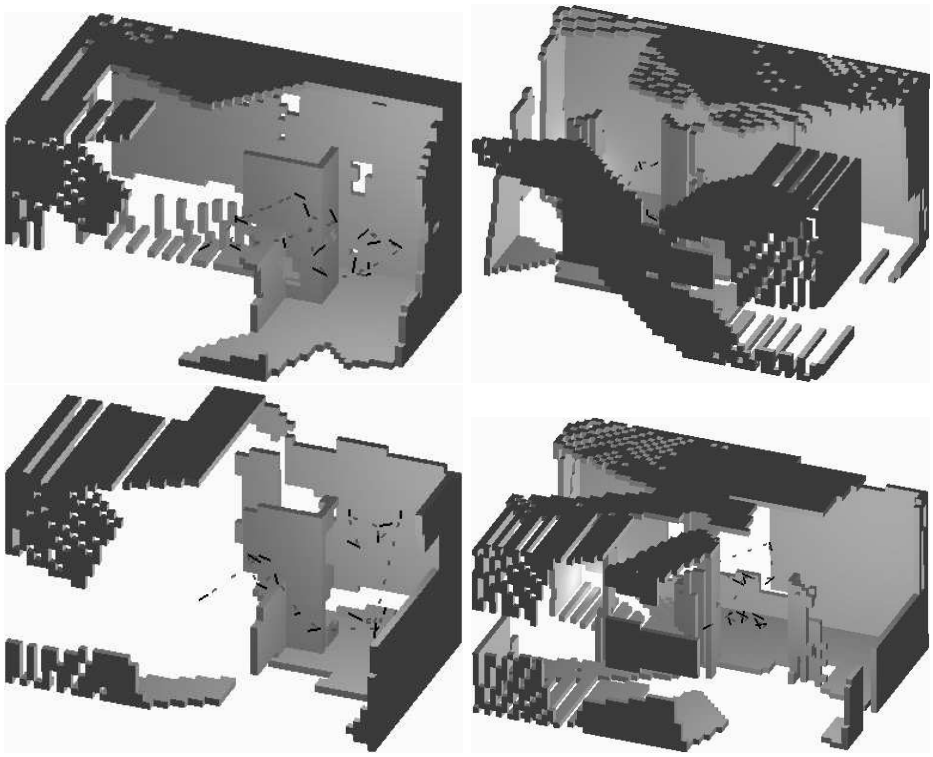

Figure 9: Two views of the recovered environments after 15 scans: "boxes" (left) and "columns" (right). *Top*: with quality factor off. *Bottom*: with quality factor on. Only *Occupied* voxels are shown. The path followed by the sensor is pointed out by dashed lines, sensor positions and orientations by solid lines