COMP 765 – Project Report
A Pseudo-physical Framework for Mapping Outdoor Terrains

Emmanuel Piuze

McGill University

May 4, 2011

1 Introduction

At least until the field of artificial intelligence is considered completely solved, robots will have an aversion to surprises. The thought processes involved for adequate decision making in a human, when facing an unexpected event, are so complex and unpredictable that no machine learning, statistical or AI methods can claim to be able to model them at this point in time [RNC+10]. Until then, it seems like the best way to deal with unexpected events resides in proper planning and sufficiently robust decision mechanisms when facing exceptional circumstances [ID04]. In practice, this implies carefulness when bringing a robot into a situation it was not programmed, trained, or constructed to deal with. When navigating a robot in terra incognita, these situations arise naturally, by presenting the robot with too few, too much, unexpected or sometimes ill-posed information. In this project, we are interested in one category of environment that may present the robot with such situations, outdoor terrains.

Outdoor terrains can present such singular events as their surface can be hard to map at a distance due to occlusion (craggy or bumpy area) or low information content and repeatability of some natural patterns (take a desert with full illumination on a cloudless day for instance)[RL93]. Although aerial robots can be used for the mapping of these environments, we are interested in ground-based robots since they couple more naturally with the pseudo-physical framework we develop.

Considering these issues, we investigate a method for constructing the map of an irregular outdoor terrain, using only local odometry information and global positioning. The system combines local topological and contextual information about the terrain with physical forces into a pseudo-physical framework that drives the robot into new areas and incrementally builds an implicit representation of the surface. Applications include terrain surveying and surface reconstruction, which could be used in the path planning of other more specific robots in the covered region.

The approach is similar to that of Whitaker [Whi98] and others [OF03] in the level set representation of the reconstructed surface but differs in the use of pseudo-physical energy and forces for determining salient regions of exploration.

We make the following assumptions and simplifications about our robot and terrain of interest:

- The robot is an idealized point-robot.

- The robot has unlimited energy reserves.
- The robot odometry is error-free and continuous absolute positional information is available.
- The robot has 3 DOF.
- The robot does not slip or compact the ground.
- The robot has excellent manoeuvrability.
- The surface is topologically smooth if averaged locally (either large-scale or small-scale features).

These assumptions are adopted for the sake of simplicity and time, and in order to better focus on specific aspects of the framework. All of these assumptions could be relaxed by adapting the framework that is to be described next. Some adaptations and extensions are described in the conclusion of this report.

Section 2 first introduces the thin-plate implicit surface representation that we use, and derived quantities. Then, section 3 describes the pseudo-physical implicit outdoor terrain reconstruction framework. Next, section 4 discusses implementation details. Section 5 presents experiments demonstrating the performance of the framework in idealized situations. Finally, section 6 concludes and proposes directions for extensions and future work.

2 Interpolating Implicit Surfaces

An implicit surface is defined by a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ where $n = 3$ ($n = 1$ refers to implicit curves and $n \geq 3$ to implicit hypersurfaces). The locus of the zero-crossings $\{\mathbf{x}|F(\mathbf{x})-c=0\}$ represents the level sets of the implicit surface, for varying c. Figure 1 shows different level sets (implicit curves) of a 2D gaussian-like function. In 3D, these level sets would actually correspond to different surfaces, of which only one (for a fixed c, determined through some constraints with respect to the original surface) is considered. In the context of surface interpolation, an implicit function is sought such that at least one of its level sets passes through (or within an epsilon-radius of) a set of constraints (control points). The nature of the implicit function determines how the surface varies in the vicinity of the control points. [GVJ+09]

In Turk and O'brien [TO02], the energy function

$$E(F) = \int_{\Omega} F_{xx}^2(\mathbf{x}) + 2F_{xy}^2(\mathbf{x}) + F_{yy}^2(\mathbf{x}) d\mathbf{x} \tag{1}$$

is used to evaluate the smoothness of the implicit function f over the region of interest Ω. This energy measures how much the surface changes in the region, and is often used in the context of regularization, in the computer vision literature. Points of high curvature contribute to a large value of E, while smooth regions tend to minimize it. Different applications yield different smoothness criterion regarding E. If the implicit function satisfies the set of constraints implied by the interpolation problem, and minimizes the energy E defined in equation 1, then it is denoted a *thin-plate* solution.

The thin-plate solution gets its name from an analogy with a thin sheet of metal, laid flat and bent such that it grazes the ends of a collection of vertical poles distributed following the constraints

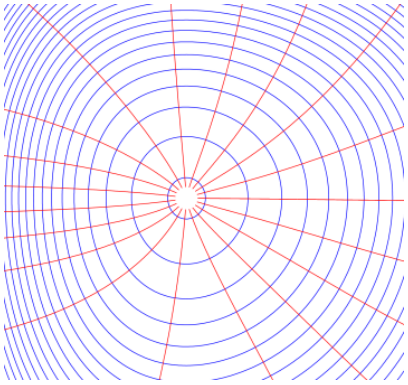


Figure 1: Visualization of an implicit function. Each blue curve is a level set $\{\mathbf{x}|F(\mathbf{x})-c=0\}$ with a different values of c. Red curves follow the direction of the gradient of F.

of the interpolation problem. A metal plate naturally resists bending such that it smoothly changes in shape in between the poles, which is exactly the kind of neighborhing behavior that minimizes the energy function E defined above.

In the 3D interpolation case, the thin-plate radial basis function (RBF), $\phi(\mathbf{x}) = |\mathbf{x}|^3$, for which the positional constraints are automatically satisfied and the smoothness is enforced, is often used. It is the thin-plate function defined by Turk and O'brien [TO02], and the one adopted in this project.

2.1 Thin-plate Fitting

Constraints $\mathbf{c}_j$ on the reconstructed surface are obtained from the robot odometry and sensors as it travels on the terrain. More precisely, we are interested in obtaining a measurement $\mathbf{p} = (x, y, h, n_x, n_y, n_z)$, where (x, y, h) is the local coordinate of the robot, and $\mathbf{n} = (n_x, n_y, n_z)$ is the direction of the surface normal. A global positioning system coupled with an altitude sensor, such as a barometric pressure sensor, in order to compensate for GPS's poor elevation accuracy and repeatability[Lei04], would provide such a measurement.

One interior and one exterior constraint are added to the surface to fix its orientation and are found by determining locations far under, and far above the surface.

The implicit function that generates the interpolation surface is written as

$$F(\mathbf{x}) = \sum_{j=1}^{k} w_j \phi(\mathbf{x} - \mathbf{c}_j) + P(\mathbf{x}) \tag{2}$$

where $\mathbf{w}_j$ are the weighting factors to be found for each RBF, $\mathbf{c}_j$ is the location of each constraint, and $P(x) = p_0 + p_1 x + p_2 y + p_3 z$ is a polynomial of degree one that describes the constant behavior of F. A constraint system that solves for the weights w_j's can be obtained:

$$h_i = \sum_{j=1}^{k} w_j \phi(\mathbf{c}_i - \mathbf{c}_j) + P(\mathbf{c}_i). \tag{3}$$

Now let $\mathbf{c}_i = (c_i^x, c_i^y, c_i^z)$ and $\phi_{ij} \equiv \phi(\mathbf{c}_i - \mathbf{c}_j)$. Equation 3 can then be expanded as the linear system

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \cdots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \cdots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \cdots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \cdots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

which can be rewritten in compact form:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{G} \\ \mathbf{G}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{h} \\ 0 \end{bmatrix} \tag{5}$$

or

$$\mathbf{M} \cdot \mathbf{W} = \mathbf{H} \tag{6}$$

Eq. 6 can be solved by computing the pseudoinverse $\mathbf{M}^+$ of the constraint matrix from its singular value decomposition:

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{7}$$

$$\mathbf{M}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T \tag{8}$$

where $S = diag(s_1, s_2, ..., s_k)$ is a diagonal matrix containing the singular values of $\mathbf{M}$. Zeroing singular values that fall below a certain threshold effectively smoothes the implicit surface at the expense of reduced fitting accuracy. Other least-squares methods can be used by exploiting the symmetrical structure of the constraint matrix, such as LU factorization.

2.2 Implicit Evaluation and Curvature Estimation

The thin-plate function

$$F(\mathbf{x}) = \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j|^3 + P(\mathbf{x}) \tag{9}$$

that represents the implicit surface needs to be manipulated in order to reveal underlying topological and differential information. In this particular case, we are interested in estimating the normal to the surface, its mean curvature, and its first and second derivatives. These quantities are required for projecting points onto the implicit surface, for determining its salient regions, and in the computation of physical quantities. In order to estimate these quantities, we need to calculate the gradient and Hessian of $F(x)$.

2.2.1 Gradient Evaluation

The gradient of the implicit surface at a point $\mathbf{x} = (x, y, z)$, $\nabla F(\mathbf{x}) = (F_x, F_y, F_z)$ is oriented in the direction of the normal to the surface. Applying the differential operator ∇ on 9, we obtain

$$\nabla F(\mathbf{x}) = \nabla P(\mathbf{x}) + \nabla \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j|^3 \tag{10}$$

$$= (p_1, p_2, p_3) + 3 \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j| (\mathbf{x} - \mathbf{c}_j), \tag{11}$$

by using the following for the partial derivative of the sum (and similarly for $\frac{\partial}{\partial y}, \frac{\partial}{\partial z}$)

$$\frac{\partial}{\partial x} \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j|^3 = \frac{\partial}{\partial x} \sum_{j=1}^{k} w_j \left[(x - c_{j,x})^2 + (y - c_{j,y})^2 + (z - c_{j,z})^2 \right]^{\frac{3}{2}} \tag{12}$$

$$= \sum_{j=1}^{k} w_j \cdot \frac{3}{2} \left[(x - c_{j,x})^2 + (y - c_{j,y})^2 + (z - c_{j,z})^2 \right]^{\frac{1}{2}} \cdot 2 \cdot (x - c_{j,x}) \tag{13}$$

$$= 3 \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j| \cdot (x - c_{j,x}). \tag{14}$$

2.2.2 Hessian Evaluation

The Hessian of $F(\mathbf{x})$, $H(F(\mathbf{x}))$ is a 3×3 matrix composed of the derivates and partial derivatives of ∇F:

$$H(F(\mathbf{x})) = \begin{bmatrix} F_{xx} & F_{xy} & F_{xz} \\ F_{yx} & F_{yy} & F_{yz} \\ F_{zx} & F_{zy} & F_{zz} \end{bmatrix}. \tag{15}$$

First derivatives F_x, F_y, F_z are obtained directly from eq. 11:

$$\frac{\partial F}{\partial x} = p_1 + 3 \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j| (x - \mathbf{c}_{j,x}), \tag{16}$$

$$\frac{\partial F}{\partial y} = p_2 + 3 \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j| (y - \mathbf{c}_{j,y}), \tag{17}$$

$$\frac{\partial F}{\partial z} = p_3 + 3 \sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j| (z - \mathbf{c}_{j,z}). \tag{18}$$

Then, the second derivatives F_{xx}, F_{yy}, F_{zz} can be computed:

$$\frac{\partial^2 F}{\partial x^2} = \frac{\partial}{\partial x}\left(p_1 + 3\sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j|(x - \mathbf{c}_{j,x})\right) \tag{19}$$

$$= 3\sum_{j=1}^{k} w_j \left(\frac{(x - c_{j,x})^2}{|\mathbf{x} - \mathbf{c}_j|} + |\mathbf{x} - \mathbf{c}_j|\right), \tag{20}$$

$$\frac{\partial^2 F}{\partial y^2} = 3\sum_{j=1}^{k} w_j \left(\frac{(y - c_{j,y})^2}{|\mathbf{x} - \mathbf{c}_j|} + |\mathbf{x} - \mathbf{c}_j|\right), \tag{21}$$

$$\frac{\partial^2 F}{\partial z^2} = 3\sum_{j=1}^{k} w_j \left(\frac{(z - c_{j,z})^2}{|\mathbf{x} - \mathbf{c}_j|} + |\mathbf{x} - \mathbf{c}_j|\right). \tag{22}$$

Similarly, partial derivatives F_{xy}, F_{xz}, Fyz can be obtained:

$$\frac{\partial^2 F}{\partial x \partial y} = \frac{\partial}{\partial x}\left(p_1 + 3\sum_{j=1}^{k} w_j |\mathbf{x} - \mathbf{c}_j|(y - \mathbf{c}_{j,x})\right) \tag{23}$$

$$= 3\sum_{j=1}^{k} w_j \frac{(x - c_{j,x})(y - c_{j,y})}{|\mathbf{x} - \mathbf{c}_j|}, \tag{24}$$

$$\frac{\partial^2 F}{\partial x \partial z} = 3\sum_{j=1}^{k} w_j \frac{(x - c_{j,x})(z - c_{j,z})}{|\mathbf{x} - \mathbf{c}_j|}, \tag{25}$$

$$\frac{\partial^2 F}{\partial y \partial z} = 3\sum_{j=1}^{k} w_j \frac{(y - c_{j,y})(z - c_{j,z})}{|\mathbf{x} - \mathbf{c}_j|}. \tag{26}$$

Since $F_{xy} = F_{yx}$, $F_{xz} = F_{zx}$, and $F_{yz} = F_{zy}$, the full Hessian can be constructed.

2.2.3 Curvature Evaluation

The mean curvature K_m at location $\mathbf{x}$ on the surface can be obtained using the following formula [Gol05]:

$$K_M = \frac{\nabla F(\mathbf{x}) \cdot H(F(\mathbf{x})) \cdot \nabla F(\mathbf{x})^T - |\nabla F(\mathbf{x})|^2 \text{Trace}(H)}{2|\nabla F(\mathbf{x})|^3}, \tag{27}$$

where $\cdot$ denotes the usual matrix product and ^{T} the transpose of a matrix. The mean curvature can be estimated continuously and at any location as the implicit surface evolves.

2.3 Surface Projection

The Newton-Raphson method can be used for projecting a point $\mathbf{x}$ towards the implicit surface, at a new location $\mathbf{x}'$. Each iteration of the Newton-Raphson projects the point in the direction of the

gradient to the surface by a magnitude h, thus bringing it closer to its boundary:

$$\mathbf{x}' = \mathbf{x} - h\frac{\nabla F(\mathbf{x})}{|\nabla F(\mathbf{x})|}, \tag{28}$$

$$h = \frac{F(\mathbf{x})}{|\nabla F(\mathbf{x})|}. \tag{29}$$

A stopping criterion based on the surface evaluation can be used for determining when the magnitude of the projection step becomes negligible:

$$\left| \frac{F(\mathbf{x})}{|\nabla F(\mathbf{x})|} \right| \leq \epsilon, \tag{30}$$

where ϵ is the desired accuracy of the implicit projection.

3 Pseudo-physical Navigation Framework

We use a pseudo-physical framework for driving the robot in its environment. Forces are accumulated on the robot from the local topology of the terrain, from existing knowledge of obstacles, from implicit constraints, and from the local exploration target. The robot's velocity is obtained by integrating the equations of motions for the accumulated forces at each time step using the Velocity Verlet algorithm (see Appendix B). When the robot enters a region of interest, it places a control point $\mathbf{c}_j$ at that location, thus populating the terrain and further reconstructing the implicit surface. This section describes the different energies that are used for determining and exploring regions of interest on the surface. The accumulated forces F can then be obtained from the gradient of the local potential energy field E:

$$F = -\nabla E. \tag{31}$$

3.1 Exploration Energy

In order to determine which regions of space should be explored further, a measure of relevance needs to be introduced. This measure is expressed in the form of energy terms, which need to be minimized.

3.1.1 Obstacle Repulsion Energy – E_r

A Gaussian repulsion energy E_r is defined to avoid collisions with obstacles on the terrain. For an obstacle located at $\mathbf{x}_0$ and a point $\mathbf{x}$, the function has the form

$$E_r(\mathbf{x}) = A_r e^{-\frac{|\mathbf{x}-\mathbf{x}_0|^2}{\sigma_r^2}}, \tag{32}$$

where σ_r is the variance of the energy functional and A_r a scalar controlling the magnitude of the repulsion. A piecewise energy function could also be used, by defining a distance below which the obstacle is so close that this repulsion should rule over all other energy terms.

3.1.2 Curvature Energy – E_k

The mean curvature of the surface at a location $\mathbf{x}$,

$$K_M(\mathbf{x}) = \frac{\nabla F(\mathbf{x}) \cdot H(F(\mathbf{x})) \cdot \nabla F(\mathbf{x})^T - |\nabla F(\mathbf{x})|^2 \text{Trace}(H)}{2|\nabla F(\mathbf{x})|^3}, \tag{33}$$

is used to define the curvature energy E_k:

$$E_k = \frac{1}{1 + |K_M|}. \tag{34}$$

3.1.3 Stationary Energy – E_s

The stationary energy E_s is related to points of zero curvature that are either saddle points or points of inflection. These points are determined by performing a circular sampling in a region and determining if either the curvature flips in sign or reaches a local minima. The energy is determined as

$$\frac{1 - E_s}{E_s} = \begin{cases} |K_{max} - K_{min}| & \text{if } \text{sgn}(K_{max}) \neq \text{sgn}(K_{min}) \\ |K_0| & \text{if } |K_{max}| = \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

where K_{max} is the largest curvature in the sampling and K_{min} the smallest, ϵ is close to zero, and K_0 is a free parameter for local minima / maxima.

3.1.4 Constraint Energy – E_c

In order to prevent the robot from unnecessarily exploring known regions of space, a repulsion energy, similar to E_r, is associated nearby existing implicit constraint points $\mathbf{c}_j$:

$$E_r(\mathbf{x}) = A_c e^{-\frac{|\mathbf{x} - \mathbf{c}_j|^2}{\sigma_c^2}}, \tag{36}$$

where σ_r is the variance of the energy functional and A_r a scalar controlling the magnitude of the repulsion.

3.2 Moving Target

The robot responds to the accumulated forces applied on it by moving towards the current energy gradient, inherently favoring the direction of a moving target, which represents the current region of interest (ROI) or an energy minima. This ROI, location at a point $\mathbf{p}$, is described by an energy functional $E(\mathbf{p})$, which is a sum of the different energy terms described previously:

$$\mathbf{p} = \underset{\mathbf{x} \in C_{free}}{\text{argmin}} \; E_r(\mathbf{x}) + E_k(\mathbf{x}) + E_s(\mathbf{x}) + E_c(\mathbf{x}) \tag{37}$$

where E_r is the obstacle repulsion energy, E_k is the curvature energy, E_s is the stationary energy, E_c is the constraint energy, and C_{free} is the space of allowed robot configurations.

3.3 External Forces

A drag force,

$$\mathbf{F}_d = -c_d \mathbf{v}, \tag{38}$$

where c_d is the drag coefficient, is applied to the robot. This force effectively smoothes out the trajectory of the robot, and prevents it from moving unnecessarily in regions of low potential energy. A gravitational force

$$\mathbf{F}_g = m\mathbf{g}, \tag{39}$$

where m is the mass of the robot, is added. Other forces such as wind and friction could also be considered by adding them to the force accumulator.

4 Implementation

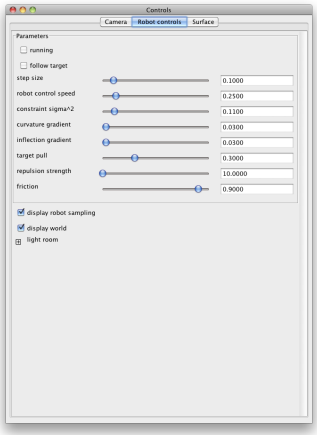
The framework was built from the ground up in Java, including implicit surface fitting (thin-plate spline) and visualization (marching tetrahedron), energy minimization, curvature and stationary point estimation and numerical derivatives (central differences) and integration (Velocity Verlet). A basic matrix package (Java3D) was used for matrix representations but matrix operations were coded by hand, including SVD decomposition and LU factorization. Java OpenGL (JOGL) was used for rendering. Fig. 2 shows the user interface for the framework.

5 Results

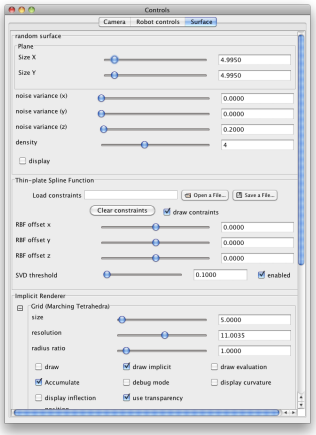
We first show results for fitting a thin-plate spline to a cloud of points. Figure 3 illustrates examples of interpolated implicit surfaces from randomly generated constraints. As discussed in Section 2, the thin-plate surface resembles a bent metal plate.

Next, we show how the curvature and stationary point estimation methods discussed in Section 3.1.2 and 3.1.3 can reveal topological information about the surface. Figure 4 shows the curvature evaluation for randomly generated implicit surfaces containing different number of constraints. A purple color gradient is used to represent the magnitude of the curvature. As expected, the color gradient indicates higher curvature around the control points, which represent a sharp bending in the thin-spline sheet. A set of stationary points can be obtained by sampling the implicit surface in a grid and determining whether each point of the grid satisfies the stationary criteria. Figure 5 shows a 40×40 sampling of inflection points, shown in cyan, for a randomly generated implicit surfaces containing 9 constraints.

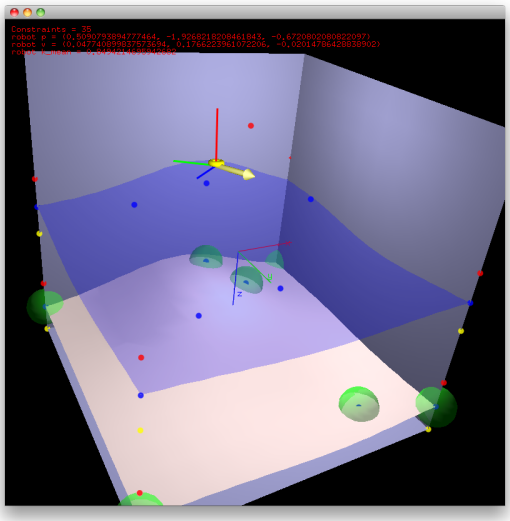
Lastly, we show examples of iteratively reconstructed surface in Fig. 6 and 7. The outdoor terrain is modeled as a random implicit surface with added zero-mean Gaussian random noise of variance $\sigma = 0.1$ (the simulation bounding box has dimensions $1 \times 1 \times 1$ unit). The robot continuously explores the area and drops constraints on the surface when it reaches energy minima. The target surface is rapidly reconstructed in both cases. The robot naturally transits from one area to the next without getting suck, although there is no mechanism implemented for determining when this is the case, as discussed in the conclusion of this report.



(a) Robot controls

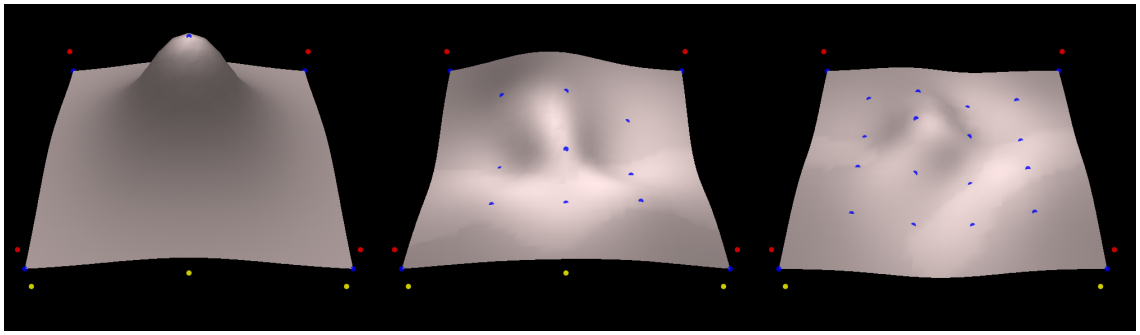


(b) Surface controls



(c) OpenGL rendering

Figure 2: GUI and user-controllable parameters for the robot and surface controls. The robot is shown as a yellow torus and its local coordinate frame is represented as red, green, and blue lines. Surface constraints and repulsion radius are shown as green spheres.



(a) 1 constraint (b) 9 constraints (c) 16 constraints

Figure 3: From left to right, an implicit surface reconstructed from 1, 9 and 16 randomly generated boundary control points respectively (shown in blue, some of which are hidden due to occlusion). Not shown are one exterior constraint (red) placed above, and one interior constraint placed beneath the surface. The corners of the surface are further constrained using triplets of interior-boundary-exterior constraints in order to enforce flatness of the thin-plate spline outside of the bounding volume.

6 Conclusion

A pseudo-physical framework was designed for navigating and reconstructing an unknown outdoor terrain. The terrain reconstruction is done using interpolated implicit surfaces based on the thin-plate spline radial basis function. The robot is driven by forces that reflect local topological and contextual information about the surface. Constraints are deposited on the implicit surface as the robot travels to regions of interest, thus iteratively giving it its shape.

Some issues remain to be addressed further, such as entrapment prevention (if the local energy gradient prohibits any movement), a more sophisticated way of dealing with dangerous or unreachable areas by associating them with some appropriate energy measure (i.e. steep terrain, crevasses, rivers, etc), and better collision detection and response.

6.1 Assumption and Simplifications

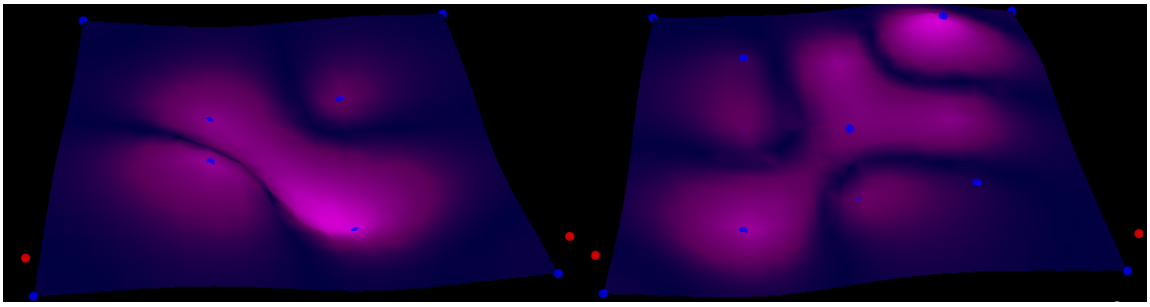
Many simplifications made to the robot and surface could be further investigated, paving the way for many directions of future work.

6.1.1 Unlimited energy reserves

Following the same idea as a vacuum cleaner robot, when the robot has low energy reserves, it could follow the geodesic (computed on the implicit surface) to the nearest charging station.

6.1.2 Error-free odometry and absolute positional information

As always, noise models for the sensors could be designed and adapted to the framework. Since the framework highly depends on accurate absolute positioning of the robot, this is probably the point



(a) 4 constraints (b) 9 constraints

Figure 4: The left figure shows an implicit surface reconstructed from 4 boundary control points (in blue). The right figure uses 9 boundary control points, some of which are hidden due to occlusion. The mean curvature of the surface is computed and shown in a purple color gradient.

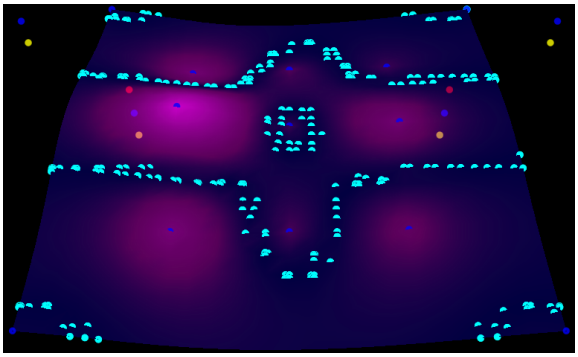


Figure 5: An implicit surface reconstructed from 9 boundary control points (in blue). Stationary points are sampled on the surface and shown in cyan.

that needs to be addressed with the most care.

6.1.3 3 DOF

Additional degrees of freedom could be included in the Velocity Verlet update, and constraints between them could be modeled accordingly. Implicit integration methods or conjugate gradient approaches could be used to deal directly with hard constraints.

6.1.4 No slipping or ground compaction

Depending on the accuracy of the global positioning system, slipping is less of an issue. Compaction, however, can yield incorrect local height measurements on multiple robot passes, which in turn would falsify the implicit surface representation.

6.1.5 Excellent manoeuvrability

This can either be seen as a hardware problem or as a computational one. A robot with in-place rotational capabilities solves this problem, as also does an algorithm that considers the configuration state of the robot when integrating the equations of motion.

6.1.6 Smooth surface topology

This would probably be the hardest simplification to drop, as implicit surfaces are notoriously bad, especially the thin-plate spline, in dealing with sharp edges, pimples, or dimples. Other types of radial basis functions could be used, for instance asymmetric RBFs [FCOS05].

A Derivative Approximation

The framework requires the computation of the gradient of many numerical and estimated quantities on the implicit surface. We adopt the explicit central difference method. The derivative of a function F at a point x is approximated to fourth-order accuracy by

$$F'(x) \approx \frac{\frac{1}{12}F(x-2h) - \frac{2}{3}F(x-h) + \frac{2}{3}F(x+h) - \frac{1}{12}F(x+2h)}{h}, \tag{40}$$

for sufficiently small h. The difference points $x-2h$, $x-h$, $x+h$, and $x+2h$ are projected back onto the surface prior to evaluation.

B Velocity Verlet Integration

The robot state $\mathbf{x}$ is integrated at each time step from its accumulated forces using the Velocity Verlet algorithm [MTTK96]. The algorithm is implemented as follows:

1. Half velocity update step: $\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t) + \frac{1}{2}\mathbf{a}(t)\Delta t$
2. Position update step: $\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t+\frac{1}{2}\Delta t)\Delta t$
3. Evaluation of acceleration $\mathbf{a}(t+\Delta t)$ from updated $\mathbf{x}(t+\Delta t)$
4. Velocity update step: $\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{a}(t+\Delta t)\Delta t$

C Visualizing Implicit Surfaces

Implicit surfaces are harder to visualize than explicit (polygonal) ones. The difficulty resides in the continuous description of the surface, defined for all points in space. In this case, the level set of interest is the zero-crossing. In a brute-force manner, all points in space could be evaluated using the implicit function describing the implicit surface, and the ones that yield a value of 0 would be marked for rendering. Obviously, this is not an efficient way to go, and different methods have been designed for this purpose. One of the most well-known method for rendering implicit surfaces is the *marching cube* algorithm [LC87]. A similar method, the *marching tetrahedron*, was used. The method works by dividing the rendering space into a grid of tetrahedra, and evaluating the

implicit function at the four corners of the cells composing that grid. The division of a cube into six tetrahedra is shown in Fig. 8. The value (positive, zero, negative) of each cell constitutes a signature, for which only one possibility exists, locally, as the planar approximation of the surface. The cell signature determines which of the cases listed in Fig. 9 applies.

References

[FCOS05] S. Fleishman, D. Cohen-Or, and C.T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (TOG)*, 24(3):544–552, 2005.

[Gol05] R. Goldman. Curvature formulas for implicit curves and surfaces. Technical report, Department of Computer Science, Rice University, 2005.

[GVJ+09] A.J.P. Gomes, I. Voiculescu, J. Jorge, B. Wyvill, and C. Galbraith. *Implicit curves and surfaces: mathematics, data structures and algorithms*. Springer Verlag, 2009.

[ID04] K. Iagnemma and S. Dubowsky. *Mobile robots in rough terrain: Estimation, motion planning, and control with application to planetary rovers*. Springer Verlag, 2004.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, page 169. ACM, 1987.

[Lei04] A. Leick. *GPS satellite surveying*. Wiley, 2004.

[MTTK96] G.J. Martyna, M.E. Tuckerman, D.J. Tobias, and M.L. Klein. Explicit reversible integrators for extended systems dynamics. *Molecular Physics*, 87(5):1117–1157, 1996.

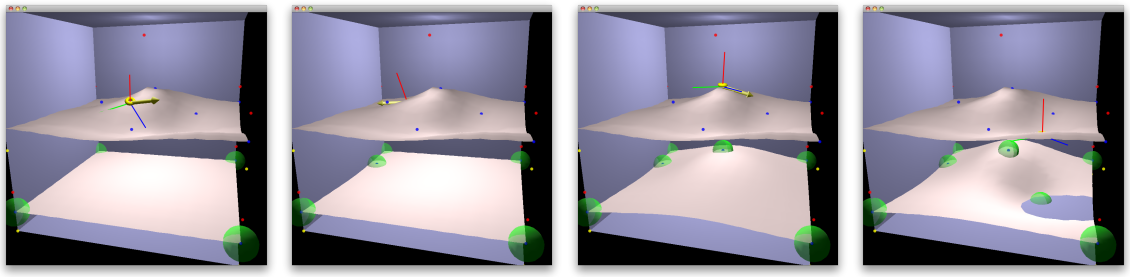
[OF03] S. Osher and R.P. Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer Verlag, 2003.

[RL93] N.S.V. Rao and Oak Ridge National Laboratory. *Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms*. Citeseer, 1993.

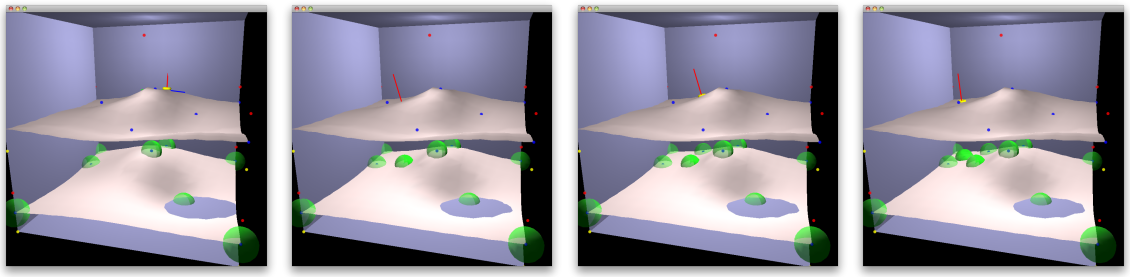
[RNC+10] S.J. Russell, P. Norvig, J.F. Candy, J.M. Malik, and D.D. Edwards. *Artificial intelligence: a modern approach*. Prentice hall, 2010.

[TO02] G. Turk and J.F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics*, 21(4):855–873, 2002.

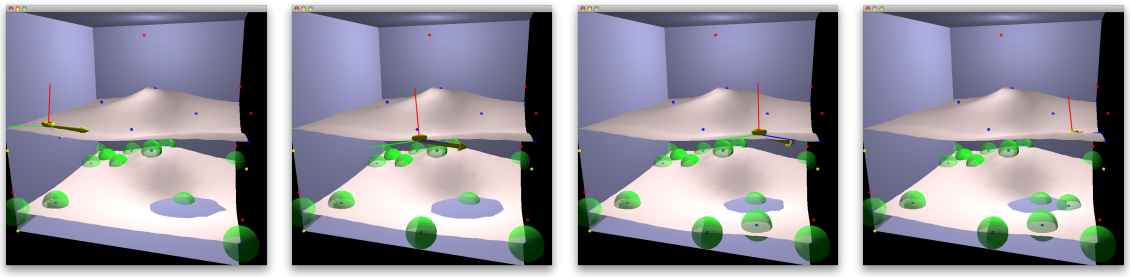
[Whi98] R.T. Whitaker. A level-set approach to 3D reconstruction from range data. *International Journal of Computer Vision*, 29(3):203–231, 1998.



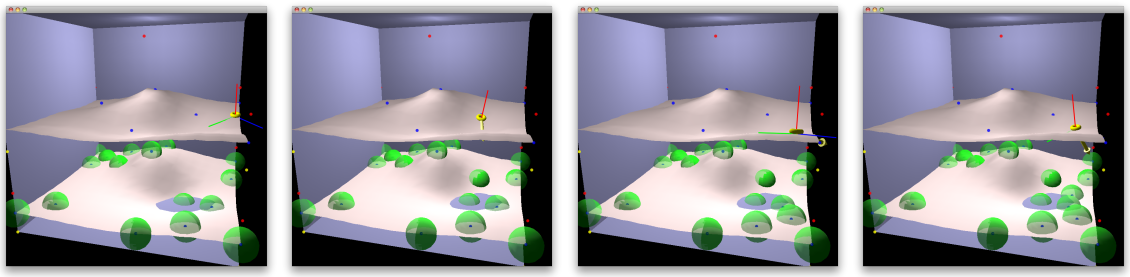
(a) 0 (b) 1 (c) 2 (d) 3



(e) 4 (f) 5 (g) 6 (h) 7

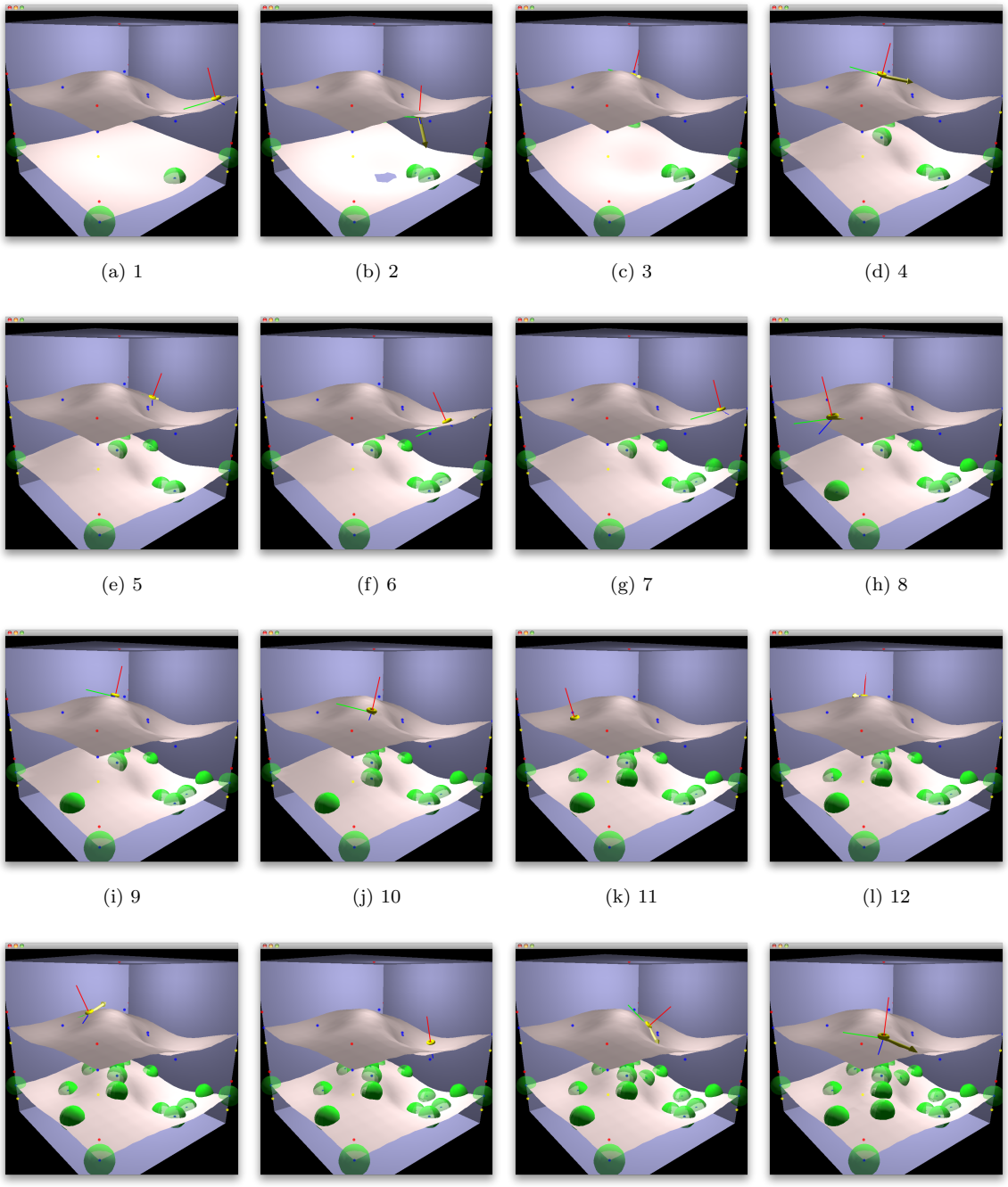


(i) 8 (j) 9 (k) 10 (l) 11



(m) 12 (n) 13 (o) 14 (p) 15

Figure 6: Iteratively reconstructed surface. The caption of each subfigure indicates the number of constraints.



(a) 1 (b) 2 (c) 3 (d) 4

(e) 5 (f) 6 (g) 7 (h) 8

(i) 9 (j) 10 (k) 11 (l) 12

(m) 13 (n) 14 (o) 15 (p) 16

Figure 7: Iteratively reconstructed surface. The caption of each subfigure indicates the number of constraints.

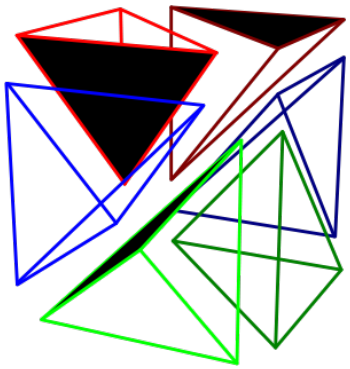


Figure 8: Division of a cubic cell into six tetrahedra.

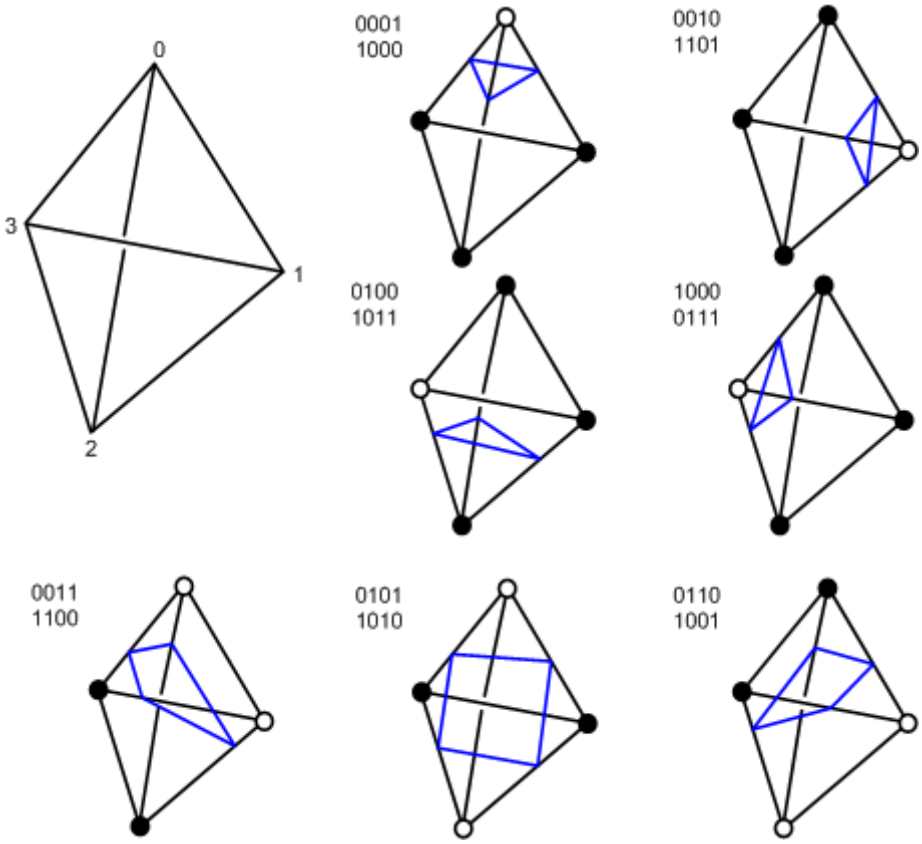


Figure 9: Marching tetrahedra test cases. A binary 0/1 value is assigned at the corners of each tetrahedron depending on whether the evaluation of the implicit function yields a negative or positive value, respectively. By default, evaluations to zero (within one machine-epsilon) are assigned the smallest non-zero positive value.