Local triangle choice for impact computation in the tactile exploration of a virtual surface

Francesco Domenicucci Massimo Ferri
Dip. di Matematica,
Piazza di Porta S.Donato, 5 I–40126 Bologna ITALY

Giorgio Nicoletti
Dip. di Matematica per le Scienze Economiche e Sociali,
Piazza Scaravilli, 2 I–40126 Bologna ITALY

Abstract

Evaluating the intersection of the trajectory of the exploring finger, with the virtual surface representing the scene, is a key problem in the VIDET project of an aid for the visually impaired. A substitute for Delaunay triangulation, which permits of local computation for that goal, is proposed.

1 Introduction

This paper is concerned with the intersection of a straight line with a surface, interpolating a set of points labelled with a distance. Although some techniques for dealing with height fields [1] and spatial data structures [2] are available, we chose to develop a specific algorithm for the peculiar needs of project VIDET, within which the research has been accomplished.

VIDET (VIsual DEcoder by Touch) will be a portable device by which a visually impaired person will explore a virtual bas–relief of the environment [3]. This will be performed by a camera pair, a portable computer and a robotic apparatus (*WireMan*, a robotic device actuated by means of threads [4, 5]). The project is presently under development at DEIS (Electronics, Informatics and Systems Department), at DIEM (Department of Mechanical

Engineering) and at our Departments at the University of Bologna, under the direction of Prof. Claudio Bonivento (DEIS).

The idea is to bring far objects within reach of the user. So, blind persons will use the sense of touch for understanding shapes — what they already do quite well for close objects (see Figure 1). This was quite impossible before the recent big advances in virtual reality; in particular, this possibility was suggested by the existence of very sophisticated force–feedback systems [6].

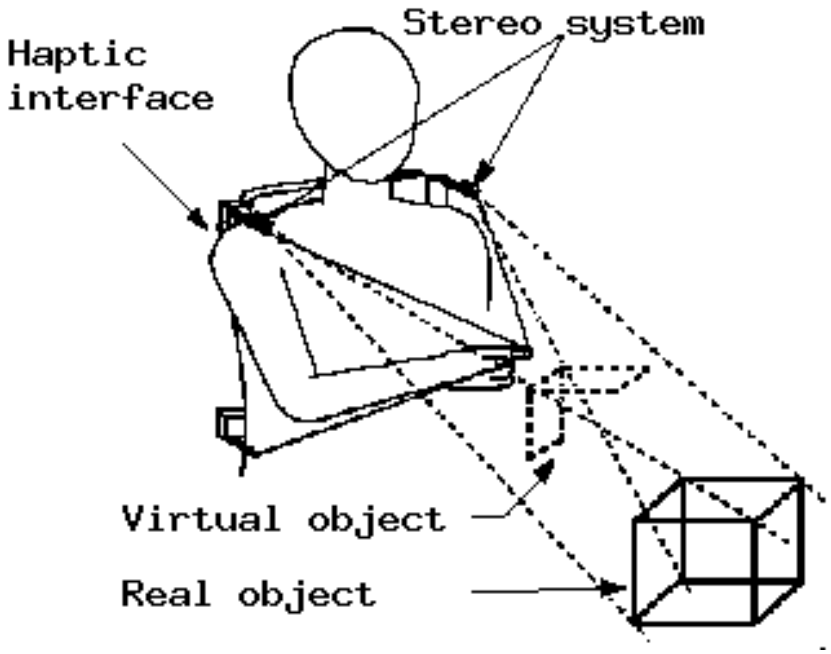

Figure 1: The VIDET system.

VIDET will consist of three phases:

1. A depth measuring device (most likely a pair of cameras together with a program for stereo computing) solid with the user's body (either head or shoulders) yields a grid of points labelled with depth values;

2. a (virtual) surface is interpolated to fit the given points;

3. WireMan, a force–feedback device, is mounted in a rucksack; it consists of three wires coming out of three corners of the rucksack, and converging to a thimble, where the user inserts his/her finger. WireMan then realizes the surface by inhibiting the finger to cross the virtual surface,

so giving the user the impression to touch a bas–relief which represents the framed scene [4, 5].

The need to render the surface in "real time" compels us to reduce the computational burden as much as possible. One big challenge is the stereo phase, but also a complete interpolation would take too long a time. Therefore we have chosen to make the third phase drive the second one.

The algorithm we are going to describe is implemented on a standard Pentium based PC. The computation is in real time, and enables us to conduct experiments with the WireMan on synthetic and real data.

2 The specific interpolation problem

A whole interpolation surface is needed, and not only a set of isolated points, however dense. In fact, the finger trajectory might miss a set of isolated points and pass beyond them, if a surface is not there.

The problem of interpolating a set of points endowed with an elevation value is generally solved, in cartography, by means of a Delaunay triangulation, or of some of its variants [7, 8, 9]. In the VIDET problem, a still better solution might be the use of B-splines [10] or of superquadrics [11], in order to give the user a smooth surface to feel, instead of an edgy, triangulated one. This, anyway, is far from our present possibilities, and will be considered only in future developments of the project. An interpolation by means of triangles is presently the only feasible solution. In this context, the global computation of a Delaunay triangulation would yield the most faithful approximation of the real scene surface. However, at least at the present development stage, it is useless to keep the computer busy with this global task (even in a simplified, constrained version [12]), while the use of the surface will be essentially local. In fact, for each frame shot by the cameras, only a very limited part of the resulting surface will be explored by the user's finger.

It should be noted that, whatever depth measuring device is used, the surface presented to the user (and actually also to a normal seeing person) could be considered as the graph of a function from the set of directions to the real numbers, yielding depth as a value for each direction. We prefer the use of a Cartesian reference frame with the xy–plane vertical (actually parallel to the plane of the rucksack), and the z–axis pointing away from the user. The surface will then be considered as the graph of a function from the xy–plane to the real numbers.

A very serious problem is, that the stereo algorithm [13] does not yield a regular grid of points labeled with depth. Our algorithm, on the contrary, assumes exactly that. Hence, everything works fine with synthetic data. Anyway, we are also using this algorithm with real data, after a first (not too costly) interpolation, which yields the required regular grid.

3 Local choice

The input data of the process are

1. *stable* data:
   - a rectangular grid of points on the vertical xy–plane (the points considered by the stereo computation)
   - a real value z for each point of the grid (the depth values computed by the stereo process)
   - a maximum allowed trajectory length;
2. *dynamic* data:
   - the three position coordinates (P_x, P_y, P_z) of the position P of the user's finger at a given instant
   - the three components (v_x, v_y, v_z) of the velocity vector v (given by the coordinate difference between the present position and the one at the previous sampling).

Output of the process are

1. coordinates of the impact point T with the interpolated surface (in the hypothesis of constant velocity)
2. unit vector, normal to the triangle of impact
3. distance to impact
4. time to impact (in the hypothesis of constant velocity).

The main idea of this local process is to follow the estimated trajectory of the finger, and to compare the finger third coordinate (depth) with the same coordinate of the vertical and horizontal lines through the grid points.

The process performs the choice of an interpolating triangle pair along the projection of the trajectory, until a change of sign of the coordinate difference occurs. After that, a standard intersection computation yields the impact point, and consequently the time to impact.

Four points in 3-space (in general position) determine a tetrahedron; this exhibits, altogether, six pairs of adjacent triangles (one for each edge). If a closed polygonal of four edges is imposed as a boundary, then only two of these pairs are left for a choice. This is our case (See Figure 2a). The polygonal in space is the one which projects to the boundary of one grid square. The choice of the triangle pair corresponds to the choice of one of the two possible diagonals of the projected square.

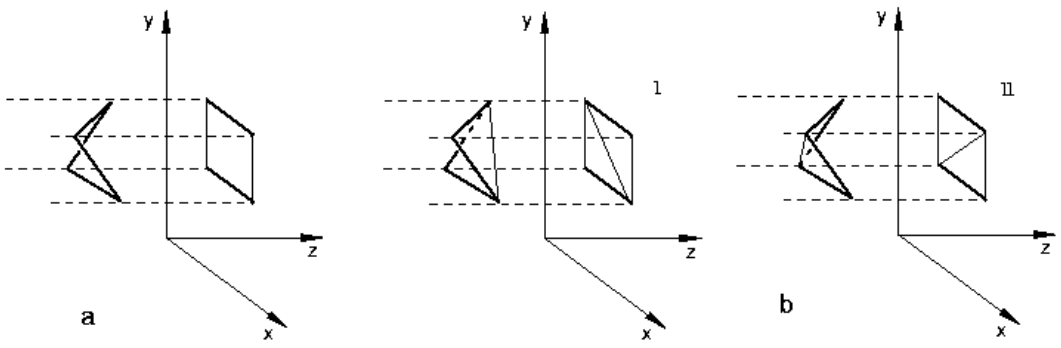

Figure 2: The two possible subdivisions of a square into triangles, and the corresponding triangle pairs in space.

This is actually a very heavy simplifying hypothesis. More explicitly, we are strongly constraining the triangulation, by imposing that each triangle has one of its edges projecting to a horizontal segment (which joins two consecutive points of the grid) and one projecting to a vertical segment (same thing, but vertically). No mathematical reason can actually assure that a true (unconstrained) Delaunay triangulation would admit exactly those triangles. On the other hand, preliminary experiments show that the error (if any) is not very inconvenient in standard use situations: horizontal and vertical edges are pre–eminent in a human made environment.

4 The algorithm

Length units are changed so as to make the grid points in the xy–plane be of integer coordinates. Let P, of coordinates (P_x, P_y, P_z), be the point in space at the present instant, and Q be its projection on the xy–plane. We determine the square to which Q belongs, by truncating its coordinates. It is straightforward to determine whether Q lies on an edge (resp. is a vertex) of the grid. In Subsection 4.4 we shall see how to determine to current square in these cases.

Up to 4.4 we shall assume that Q lies in the interior of a square, which we call the *first square*. We have then to divide the first square into two triangles and to find which of the two contains Q: This is done by the following routine.

4.1 Delaunay

There are two possible subdivisions (I and II) of the square into a pair of triangles, according to which of the two diagonals is chosen (see Figure 2b). Fix one of them (I): the two resulting triangles are projections of two space triangles T_1, T_2; compute the difference of the maximum and minimum side length of each, $d(T_i)$, and take the minimum of the two differences $d_I = \min\{d(T_1), d(T_2)\}$. Do the same for subdivision II, so obtaining d_{II}. Then choose the subdivision which yields the lower value between d_I and d_{II}. Now, the coordinates of Q allow us to select which of the two triangles contains it. Note that Q may belong to both, i.e. to their common edge; we can arbitrarily choose either triangle.

4.2 Starting point

We now have to find the projection R of P on the surface (i.e. the surface point which projects to Q too). Once we have determined the triangle to which Q belongs, we compute the equation of the plane through the points of the selected space triangle. This allows immediate computation of the depth R_z of R. Of course, the other two coordinates R_x, R_y are the same as those of Q and of P. Now, we need the difference $\Delta z = P_z - R_z$ (see Figure 3).

- If $\Delta z > 0$, then P is *beyond* the surface, and we suspend the search. An alert signal is given to the user.

- If $\Delta z = 0$ (within the given approximation), then $P = R$, and this is already the impact point T.
- If $\Delta z < 0$, then we have to find the sequence of intersection points. Note that this is immediate if vector v is along the z–axis: In this case $T = R$. In the following routine we exclude this case.

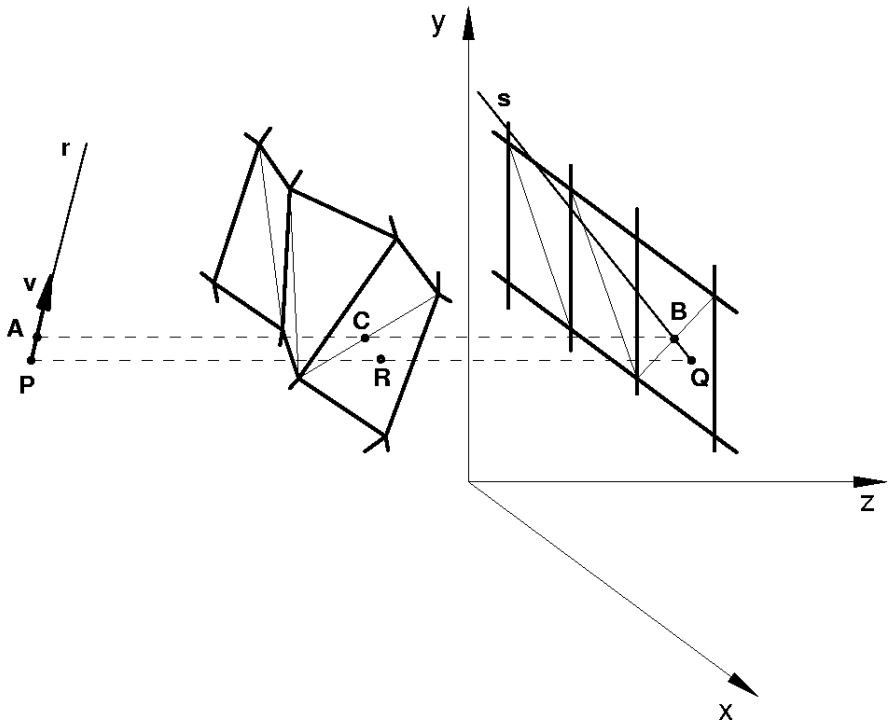

Figure 3: Trajectory, surface, and their projections.

4.3 Exploring the first square

Consider the half–line r starting in P and with vector v (best expressed by parametric equations); call s the half–line which is its projection on the xy–plane (obviously starting from Q). Recall that the square to which Q belongs has already been divided while executing the Delaunay routine of 4.1, so a diagonal of it has been chosen.

The half–line s intersects either an edge or the diagonal of the square. Find this intersection B. Finally, find point A of coordinates (A_x, A_y, A_z) of r, and point C of coordinates (C_x, C_y, C_z) of the surface, which both project to B (i.e. such that $A_x = B_x = C_x, A_y = B_y = C_y$). This is easily done by

using the parametric equations of r and the equations of the line projecting to the given edge or diagonal. Compute the difference $\Delta z = A_z - C_z$.

- If $\Delta z > 0$, then A is *beyond* the surface, so the impact occurs within the present square, nay within the present triangle. Compute the coordinates of the impact point T by intersecting r with the space triangle.
- If $\Delta z = 0$ (within the given approximation), then $A = C$, and this is already the impact point T.
- If $\Delta z < 0$, then the impact occurs outside of the triangle, and we go on with the search of the next triangle. B may belong to the diagonal; then determining if the impact is in the other triangle and — if not — determining the next square to search, requires intersection with the edges of the present square and the computation of a new Δz.

Assume that the impact does not occur in either triangle; then we have a new point, the one of intersection with the edge, from which to start again the procedures of the present and preceding Subsections. Note, however, that we have gone back to the situation we are in, when the point is not in the interior of a square.

4.4 Particular cases

The point from which we start the whole exploration, or the exploration of a new square, might project to

1. a vertex of the grid (Figure 4a), or
2. a point of the interior of an edge of the grid (Figure 4b).

First, note that coordinate z of the point can very easily obtained in case 1: It is part of the input data. In case 2, the z coordinate is computed by using the equations of the space straight line which projects to the line containing the edge.

In both cases, we must select the square to search; the choice is among four squares, in case 1, between two in case 2. Assume that vector v is not directed along an edge; then we have to test the intersection of the half–line s with eight edges in case 1, with six in case 2.

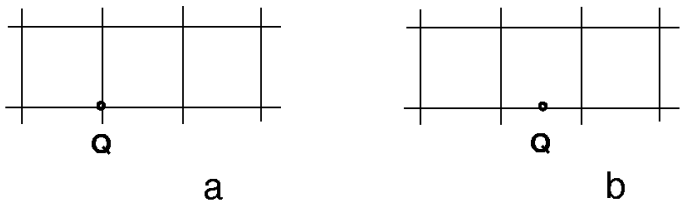

Figure 4: Particular cases.

If v is directed along an edge, either of the adjacent squares can be arbitrarily chosen as the current square. The same applies to the case of v along the selected diagonal of a square: either triangle can be chosen.

4.5 End of the algorithm

The algorithm stops in either of the following cases:

- An impact point T has been reached. Then the output (coordinates, normal vector, distance to impact) are easily computed and passed to the control section of the system.
- The maximum allowed distance has been reached while exploring the sequence of squares. This information is passed to the control section of the system.

Remark - One might think that it is not necessary to triangulate all squares along the trajectory: it might seem sufficient to compute Δz just at the intersection with the square edges. It is not so: s can cross two edges of a square and have a negative Δz at both places, but can yield a positive Δz at the intersection with the selected diagonal, so revealing impact within the square.

5 Conclusions

We have described a technique for locally interpolating points in space, whose projections on a plane form a regular rectangular grid. The interpolation actually reduces to the choice of a triangle in a pair — for the few squares

met along the trajectory — and is driven by the movement of the point, whose impact with the surface is then computed.

The procedure has already been inserted into a program for controlling the motion of the end-effector (i.e. the thimble of WireMan, and finally the user's finger) by force-feedback.

Acknowledgements

Research accomplished under support of CNR–GNSAGA, of ASI, and of the University of Bologna, funds for selected research topics. Unfortunately, we can no more thank David Searby, whose helpful discussions on the subject showed once more that he cared more for others than for himself.

References

[1] D.W. Paglieroni, "Parametric height field ray tracing", Proc. Graphics Interface '92, Vancouver, Morgan–Kaufmann Publ. Inc. (1992), 192–200.

[2] J. Nievergelt, P. Widmayer, "Spatial data structures: Concept and design choices", in: Handbook of computational geometry, Eds. J.–R. Sack et al., North–Holland (2000) 725–764.

[3] L. Di Stefano, A. Eusebi, M. Ferri, C. Melchiorri, M. Montanari, and G. Vassura, "A robotic System for Visually Impaired People based on Stereo–Vision and Force–Feedback", IARP Int. Workshop on Medical Robots, Vienna (1996), 39–46.

[4] C. Bonivento, A. Eusebi, C. Melchiorri, M. Montanari, and G. Vassura, "WireMan: A Portable Wire Manipulator for Touch-Rendering of Bas-Relief Virtual Surfaces", 8th Int. Conf. on Advanced Robotics (ICAR97), Monterey (1997).

[5] C. Melchiorri, M. Montanari, and G. Vassura, "Control Strategies for a Defective, Wire-Based, Haptic Interface", Int. Conference on Intelligent Robots and Systems (IROS'97) Grenoble (1997).

[6] M. Bergamasco, B. Allotta, L. Bosio, L. Ferretti, G. Parrini, F. Salsedo and G. Sartini, "An Arm Exoskeleton System for Teleoperation and Virtual Environments Applications" IEEE Int. Conf. on Robotics and Automation, San Diego (1994).

[7] I. Babuska and A.K. Aziz, "On the angle condition in the finite element method", SIAM J. Numer. Anal. 13 (1976), 214–226.

[8] E. Bruzzone, M. Cazzanti, L. De Floriani and F. Mangili, "Applying two-dimensional Delaunay triangulation to stereo data interpolation", Proc. ECCV '92, S. Margherita Ligure, Springer–Verlag (1992), 368–372.

[9] J.D. Boissonnat and M. Teillaud, "On the randomized construction of the Delaunay tree", INRIA, Sophia Antipolis, Res. Rep. 1140 (1991).

[10] L. Piegl, "On NURBS: A survey", IEEE Proc. on Comp. Graph. and Appl. 11 (1991), 55–71.

[11] R. Bajcsy, F. Solina, "Recovery of parametric models from range images: the case for superquadrics with global deformations", IEEE Trans PAMI 12 (1990), 131–147.

[12] L. De Floriani and E. Puppo, "An on-line algorithm for constrained Delaunay triangulations", CVGIP: Graph. Mod. and Image Process. 54 (1992), 290–300.

[13] P. Fua, "Combining stereo and monocular information to compute dense depth maps that preserve depth discontinuities", Proc. 12th Int. Joint Conf. on Artificial Intelligence, Sydney (1991), 1292–1298.