Geometry Special 2018 — Editorial

Victor Lecomte

May 6, 2018

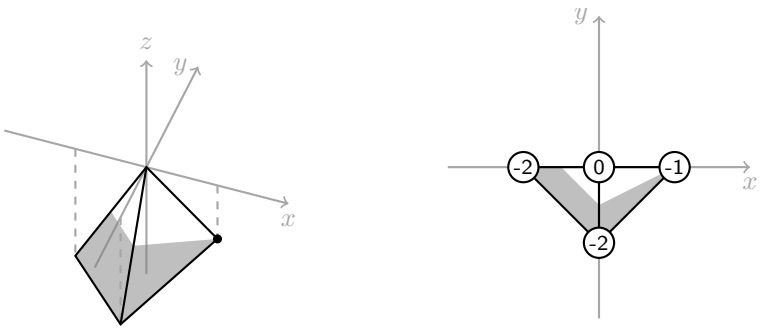
Problem A: Sledding down the mountain

In this problem I wanted to experiment with graph-like problems on a polyhedral mesh. My aim was to show that the added dimension of height could make interesting ideas possible.

Idea

The first observation to make is that if we can reach a triangle at height h, then we can reach all points of the triangle at height $\leq h$. So for each triangle i, we want to find the highest height h_i at which it can reached from $P_{1,1}$.

A first idea would be to build a graph from the vertices of the triangles and determine for each of them whether it is reachable, with some kind of flood fill. However, this is insufficient, as sometimes the critical height h_i is not the height of any vertex of the triangle, like in the left triangle below.



A better idea is to build a graph on the triangles of mountain, where two triangles are linked if they share an edge. Let's figure how to do the relaxations between two triangles. Suppose triangles i and j share an edge, such that the lowest point of the edge has height h_{lo} and the highest point

of the edge has height h_{hi}. If triangle i can be reached at h_i, what height can we reach in j using that edge?

There are three cases:

- if $h_i < h_{\text{lo}}$, triangle j cannot be reached from triangle i;
- if $h_{\text{lo}} \leq h_i \leq h_{\text{hi}}$, it can be reached with a height of h_i;
- if $h_i > h_{\text{hi}}$, it can be reached with a height of h_{hi}.

Since in all cases the height that is reached is smaller or equal to h_i, we can use Dijkstra's algorithm to efficiently compute h_i for all i. We maintain the best known height for every triangle, and always select the unvisited triangle that has the highest height.

Cutting the triangles

Now that we found h_i for all triangles i, we need to find the part of the triangle that is under plane $z = h_i$. This is a similar exercise to cutting a polygon by a line in 2D: we need to keep the points for which $z \leq h_i$ and add points whenever a edge goes through the plane. This can be simply implemented this way:

```
// Assume p3 is a 3D point structure with basic operators

p3 planeInter(p3 a, p3 b, int h) {
    return (a*(b.z-h) - b*(a.z-h))/(b.z-a.z);
}

vector<p3> polygonUnder(vector<p3> ps, int h) {
    vector<p3> under;
    for (int i=0, n=ps.size(); i<n; i++) {
        p3 a = ps[i], b = ps[(i+1)%n];

        // Keep points under h
        if (a.z <= h)
            under.push_back(a);

        // Add intersections
        if ((a.z-h)*(b.z-h) < 0) // if a and b on either side
            under.push_back(planeInter(a,b,h));
    }
    return under;
}
```

Finding the area

It turns out finding the area of a polygon in 3D is pretty much the same as in 2D. In 2D, we can find the area of a polygon $P_1 \cdots P_n$ as

$$\frac{|P_1 \times P_2 + P_2 \times P_3 + \cdots + P_n \times P_1|}{2}$$

where $\times$ is the 2D cross product (a scalar). In 3D, if $P_1, \ldots, P_n$ are coplanar, the vector

$$\vec{S} = P_1 \times P_2 + P_2 \times P_3 + \cdots + P_n \times P_1$$

is a vector perpendicular to their plane of polygon $P_1 \cdots P_n$ and whose norm is twice the area of the polygon. So we can similarly compute the area as $\frac{1}{2}\|\vec{S}\|$.

So we just need to use this formula and for every triangle sum up the area of the part that is under h_i.

Problem B: Pen spinning accident

There were two ways to approach this problem (that I know of), with wildly different difficulties of implementation. The first is interesting in that we transform a 3D problem into a 1D problem. The second shows that good knowledge of geometric primitives can help you simplify a problem and avoid a lot of edge cases.

Approach 1: Position on the intersection line

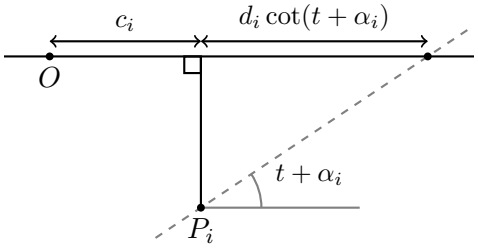
We can observe that when the pens spin around, they each sweep a distinct plane. So if the pens touch at some point, it must happen on the intersection of the planes. If it doesn't exist, the answer is `never`, otherwise because of the constraints of the problem, it must be a line l that touches neither P_1 nor P_2.

Since we're only interested in positions on l, let's set an arbitrary starting point O on l, and figure out how far from O the lasers of the i^{th} pen touch l. We'll call this $f_i(t)$.

We can easily convince ourselves that $f_i(t)$ is of the form

$$f_i(t) = c_i + d_i \cot(t + \alpha_i)$$

where c_i is the position of the projection of P_i on l, d_i is the distance from P_i to l, and α_i depends on the angle between l and $\vec{v_i}$.



Instead of trying to find a root straight away (which would be very tricky), we can use the summation formula:

$$\cot(t + \alpha_i) = \frac{\cot t \cot \alpha_i - 1}{\cot t + \cot \alpha_i}$$

By substituting and eliminating the denominators gives this (rather lengthy) quadratic equation in $\cot t$ which can be used to find t more easily:

$$\begin{aligned} 0 &= (\cot t)^2(c_2 - c_1 + d_2 \cot \alpha_2 - d_1 \cot \alpha_1) \\ &+ (\cot t)[(c_2 - c_1)(\cot \alpha_1 + \cot \alpha_2) + (d_2 - d_1)(\cot \alpha_1 \cot \alpha_2 - 1)] \\ &+ [(c_2 - c_1) \cot \alpha_1 \cot \alpha_2 - (d_2 \cot \alpha_1 - d_1 \cot \alpha_2)] \end{aligned}$$

Some caveats:

- Since the values of c_i, d_i and $\cot \alpha_i$ are most likely imprecise, we need to do an epsilon comparison to test the value of the discriminant and determine the existence of solutions.
- We also need to handle the cases when $\cot \alpha_1$ or $\cot \alpha_2$ don't exist. One trick is to multiply the whole equation by $\sin \alpha_1 \sin \alpha_2$, which replaces $\cot \alpha_i$ with $\cos \alpha_i$ and adds some $\sin \alpha_i$ in the mix.
- Note that we don't care about the times when $\cot t$ doesn't exist, that is, $t = k\pi$, since the lasers are guaranteed not to touch at the start.

Approach 2: Vector magic

Given 4 points A, B, C, D in space, we can define the expression

$$\text{orient}(A, B, C, D) = \left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot \overrightarrow{AD}$$

which has many properties similar to its 2D counterpart

$$\text{orient}(A, B, C) = \overrightarrow{AB} \times \overrightarrow{AC}$$

For example, $\text{orient}(A, B, C, D) = 0$ iff A, B, C, D are coplanar, its sign determines on which side of plane ABC point D lies, and its norm is 6 times the volume of tetrahedron $ABCD$. The property we are interested in here is that it is zero iff lines AB and CD are either parallel or intersecting (remember that in space, lines can be neither). Since in our case the lasers are never parallel, we can use this to determine easily whether the lines intersect.

We can reshuffle it this way:

$$\text{orient}(A, B, C, D) = \left(\overrightarrow{AB} \times \overrightarrow{AD}\right) \cdot \overrightarrow{AC} = \left(\overrightarrow{AB} \times \overrightarrow{CD}\right) \cdot \overrightarrow{AC}$$

so that in our case, the lasers touch iff

$$\text{orient}\left(P_1, P_1 + \vec{d_1}(t), P_2, P_2 + \vec{d_2}\right) = \left(\vec{d_1}(t) \times \vec{d_2}(t)\right) \cdot \overrightarrow{P_1P_2} = 0$$

If we plug in the definition of $\vec{d_i}(t)$, we obtain

$$\begin{array}{rcll} 0 & = & \cos^2 t & (\vec{v_1} \times \vec{v_2}) \cdot \overrightarrow{P_1P_1} \\ & + & \cos t \sin t & (\vec{v_1} \times \vec{w_2} + \vec{w_1} \times \vec{v_2}) \cdot \overrightarrow{P_1P_2} \\ & + & \sin^2 t & (\vec{w_1} \times \vec{w_2}) \cdot \overrightarrow{P_1P_2} \end{array}$$

If we divide the equation by $\sin^2 t$, we obtain (again) a quadratic equation in $\cot t$, but this time with no special cases, and all computations can be done with integers.

Problem C: Polar map

In this problem, we are interested in finding the area of objects of unusual shapes. So first we need to find mathematical tools that can help us do that. The objective was mostly to show this way to compute areas.

Area surrounded by a curve

Given a curve $\mathcal{C}$, how can we find the area it encloses? One way is to compute the following integral:

$$\frac{1}{2}\int_{\mathcal{C}} x\,dy - y\,dx$$

This can be seen as an infinitesimal version of the shoelace formula, and computes the surface sweeped by the segment from the origin O to a moving point on $\mathcal{C}$.

If we know that the curve never goes through O, then we can consider polar coordinates $(r\cos\theta, r\sin\theta)$ and the integral becomes

$$\frac{1}{2}\int_{\mathcal{C}} r^2\,d\theta$$

Indeed, $\frac{r^2}{2}\,d\theta$ is the area of circular sector of radius r and angle $d\theta$.

This means that if we describe $\mathcal{C}$ in a parametric way, that is, if we find functions $r(t)$ and $\theta(t)$ that describe the coordinates of points on $\mathcal{C}$ as parameter t increases in $[a, b]$, then we can compute the area as

$$\frac{1}{2}\int_a^b r^2(t)\,\theta'(t)\,dt$$

where $\theta'(t)$ is the derivative of $\theta(t)$.

For example, let's say we want to find the area of a circle of radius 1.[1] We can define the points of the circle with

$$\begin{cases} r(t) &= 1 \\ \theta(t) &= t \end{cases}$$

for $t \in [0, 2\pi]$.

Then we can find

$$\frac{1}{2}\int_{\mathcal{C}} r^2\,d\theta = \frac{1}{2}\int_0^{2\pi} r^2(t)\,\theta'(t)\,dt = \frac{1}{2}\int_0^{2\pi} 1^2\,1\,dt = \pi$$

Note that the integral can be computed in several parts to handle different parametrizations: for example, this allows us to easily compute the area of shapes formed by line segments and circular arcs.

[1] Groundbreaking, right?

Parametrization

Now that we have a way to compute the area of any parametric curve, let's try to find a parametrization for the country borders on our map.

Let's say the Earth has radius 1 and we want to find the path between two consecutive points A and B on the border of the country. First, we observe that the "straight line" between A and B actually gives the correct direction: if we take point $P(t) = A + t\overrightarrow{AB}$ ($t \in [0, 1]$) and scale it so that it has a norm of 1, then it will lie on the great circle containing A and B. And since the position on our map only depends on the latitude ϕ and longitude λ, not on the altitude, we will use this point $P(t)$ directly.

Let's assume that the axis $z = 0$ is the line going between the poles. First, we find the longitude and latitude of $P(t)$:

$$\begin{aligned}\varphi(t) &= \operatorname{atan2}\left(P_z(t), \sqrt{P_x^2(t) + P_y^2(t)}\right)\\ \lambda(t) &= \operatorname{atan2}\big(P_y(t), P_x(t)\big)\end{aligned}$$

This gives the following polar coordinates on the map:

$$\begin{aligned}r(t) &= \frac{\pi/2 - \varphi(t)}{\pi}\\ \theta(t) &= \lambda(t)\end{aligned}$$

We need to find the derivative of $\theta(t)$. Here we are using formulas for the partial derivatives of atan2, see `https://en.wikipedia.org/wiki/Atan2#Derivative` for more info.

$$\begin{aligned}\theta'(t) = \lambda'(t) &= \Big(\operatorname{atan2}\big(P_y(t), P_x(t)\big)\Big)'\\ &= \frac{P_y'(t)P_x(t) - P_x'(t)P_y(t)}{P_x^2(t) + P_y^2(t)}\\ &= \frac{(B_y - A_y)P_x(t) - (B_x - A_x)P_y(t)}{P_x^2(t) + P_y^2(t)}\\ &= \frac{A_xB_y - A_yB_x}{P_x^2(t) + P_y^2(t)}\end{aligned}$$

Now we can find the part of the integral for this part of the border as

$$\frac{1}{2}\int_0^1 r^2(t)\,\theta'(t)\,dt$$

Integration

We didn't manage to find an antiderivative for the given function to integrate. However, in general numerical integration works very well, so you can just use that.

For the best precision/time ratio, we advise using something adaptative like adaptative Simpson's method (see `https://en.wikipedia.org/wiki/Adaptive_Simpson%27s_method`). But time limits were lenient enough that regular Simpson's rule and even trapezoidal or midpoint rule worked as well, as long as the step is small enough to detect the brusque changes when the border passes close to the South Pole as in the third sample.

Problem D: Gold paint optimization

Some of the difficulty of this problem was to get used to the notion of solid angle. But actually, an intuitive understanding of it was enough to solve the problem, since the solution doesn't require actually computing a solid angle at any point.

Idea

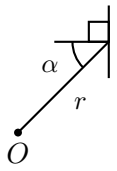
The main insight was thinking in terms of return on investment: given a location, if I paint a small area there, by how much does it increase the solid angle. Thus we consider the ratio

$$\rho = \frac{\text{solidangleadded}}{\text{paintused}}$$

It is clear that surfaces with higher ratios should be painted before surfaces with lower ratios. So we perform a binary search on this ratio ρ, computing the total area of parts that have a ratio $\geq \rho$. If this total area is smaller than l, we should decrease ρ, otherwise we should increase it. The distribution of paint after convergence is the optimal distribution of paint.

Computing the ratio

Intuitively, the ratio is inversely proportional to the square of the distance from $O = (0, 0, 0)$, and it will be lower if the surface is seen at an angle. More precisely, if the surface is seen from distance r and the angle between the direction of observation and the normal of the surface is α, then the ratio is $\rho = \cos\alpha / r^2$.

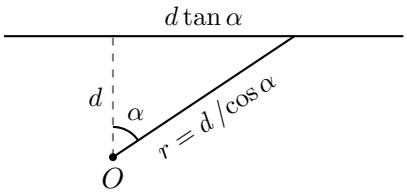


This can be seen from a quick drawing or taken from Wikipedia:

$$\Omega = \int_S \frac{\hat{r} \cdot \hat{n}\, d\Sigma}{r^2}$$

where Ω is the solid angle, S the surface observed, $\hat{r}$ the direction of observation, $\hat{n}$ the normal of the surface and $d\Sigma$ is an infinitesimal area.

Now, given a ratio ρ and a cube face, how do we find the part of the face that has ratio at least ρ? Suppose without loss of generality that the square is on plane $x = d$.



Clearly, on that plane, the closer to $O = (0, 0, 0)$, the better the ratio. In particular, if the observed point is at angle α from the perpendicular, then $r = d / \cos \alpha$, so the ratio is

$$\rho = \frac{\cos \alpha}{r^2} = \frac{\cos^3 \alpha}{d^2}$$

and we can find α as

$$\alpha = \arccos\left(\sqrt[3]{\rho d^2}\right)$$

The time limits were lenient enough to allow solutions to find this value by some binary search instead.

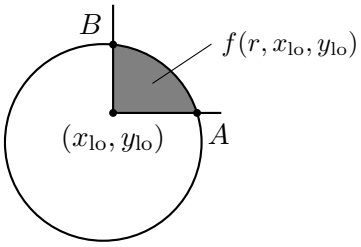
Thus, for the value of α we fond, all points on the plane $x = d$ that are within a disk of radius $d \tan \alpha$ around $(d, 0, 0)$ will have ratio $\geq \rho$. So the relevant points on the cube face are the intersection of that disk and the square of the cube face (a 2D problem).

Circle-square intersection

So the last thing we need to figure out is how to compute the area of the intersection of a disk and a square. This is feasible but tedious in the general case.

Here we are helped by the fact that the square will always be fully within one of the quadrants of the disk. So we can find it easily by inclusion-exclusion. Let's say we have a function $f(r, x_{\text{lo}}, y_{\text{lo}})$ that finds the intersection between the circle of radius r and center $(0, 0)$, and the quadrant $x \geq x_{\text{lo}}, y \geq y_{\text{lo}}$, for $x_{\text{lo}}, y_{\text{lo}} \geq 0$. Then the area of intersection with square $[x, x + 1] \times [y, y + 1]$ can be found as

$$f(r, x, y) - f(r, x + 1, y) - f(r, x, y + 1) + f(r, x + 1, y + 1)$$



Function f can be easily implemented this way, starting by computing intersection points A and B if they exist:

```
double quadrantDisk(double r, double x, double y) {
    if (x*x + y*y >= r*r)
        return 0;
    pt p(x,y), a(sqrt(r*r-y*y), y), b(x, sqrt(r*r-x*x));
    return (cross(p,a) // triangle OPA
        + r*r*asin(min(cross(a,b)/r/r, 1.0)) // circular sector OAB
        + cross(b,p) // triangle OBP
        )/2;
}
```