Finding the intersection point of many lines in 3D (point closest to all lines)

I have many lines (let's say 4) which are supposed to be intersected. (Please consider lines are represented as a pair of points). I want to find the point in space which minimizes the sum of the square distances to all of the lines or in other words, the point which is closest to all the lines.

I want to formulate this as a Least Squares Problem, but I'm not quite sure how it would be. I found the way to compute the distance between line and point. So, I need help to go further.

(As I can't access for optimization packages, I want to implement this with least squares.)

(algorithms) (geometry) (regression)

edited Sep 9 '11 at 23:21
Mateen Ulhaq
640 ● 2 ■ 11 ▲ 27

asked Sep 3 '11 at 23:05
niro
393 ■ 4 ▲ 18

migrated from stackoverflow.com Sep 4 '11 at 2:37

This question came from our site for professional and enthusiast programmers.

There is not necessarily *one* such point (there may be many). – bitmask Sep 3 '11 at 23:34

You may find interesting "Distance between Lines" softsurfer.com/Archive/algorithm_0106/algorithm_0106.htm – alexm Sep 3 '11 at 23:45

@bitmask, yes. but my lines are supposed to be met. but may be, these are met at different points, which are very closely to each other. so, i need to find the point which is closest to all the lines. – niro Sep 3 '11 at 23:55

hi niro, have you solved with the valdo's answer? If yes, could you please be so kind to explain to me how you get the three equation out of the last formula? Thanks in advance – elect Jun 22 '15 at 10:10

3 Answers

In some degenerate cases there may be no such a one point (for instance, if all the lines are parallel). However there's a single solution in the general case.

I assume you're trying to solve a more general problem where all the lines are not required to intersect exactly (otherwise there's a much simpler solution than the least squares).

Derivation:

You say the every line is represented by two points. Let's rather work in the convention where a line is represented by one point and a direction vector, which is just a vector subtraction of those two points. That is, instead of describing a line by points $\mathbf{a}$ and $\mathbf{b}$ we'll describe it by a point $\mathbf{a}$ and a vector $\mathbf{d}$ whereas $\mathbf{d} = \mathbf{b} - \mathbf{a}$.

Our point (which we're trying to find) is $\mathbf{c}$.

The distance of this point to the line is:

$H = \frac{\|(\mathbf{c}-\mathbf{a})\times\mathbf{d}\|}{\|\mathbf{d}\|}$

Using identity $(\mathbf{a}\times\mathbf{b})\cdot(\mathbf{a}\times\mathbf{b}) = \|\mathbf{a}\|^2\|\mathbf{b}\|^2 - (\mathbf{a}\cdot\mathbf{b})^2$

we have:

$H^2 = \frac{\|\mathbf{c}-\mathbf{a}\|^2\|\mathbf{d}\|^2 - \|(\mathbf{c}-\mathbf{a})\cdot\mathbf{d}\|^2}{\|\mathbf{d}\|^2}$

$H^2 = \|\mathbf{c}-\mathbf{a}\|^2 - \frac{\|(\mathbf{c}-\mathbf{a})\cdot\mathbf{d}\|^2}{\|\mathbf{d}\|^2}$

The square sum of the distances of the point $\mathbf{c}$ to all the lines is just the sum of the above expressions for all the lines. The problem is to minimize this sum. This sum depends on a variable $\mathbf{c}$ (which is actually 3 variables, the components of $\mathbf{c}$). This is a standard least squares problem, which generally has a single solution (unless there's a degeneracy).

Solving the least squares for this specific case.

Since we want find such a $\mathbf{c}$ that minimizes this sum, its derivative with regard to $\mathbf{c}$ should be zero. In other words:

$\frac{d(H^2)}{d\mathbf{c}} = 2(\mathbf{c}-\mathbf{a}) - 2\mathbf{d}\frac{(\mathbf{c}-\mathbf{a})\cdot\mathbf{d}}{\|\mathbf{d}\|^2}$

$0 = \sum_{i=0}^{m} \mathbf{c} - \mathbf{a}^{(i)} - \mathbf{d}^{(i)} \frac{(\mathbf{c}-\mathbf{a}^{(i)})\cdot\mathbf{d}^{(i)}}{\|\mathbf{d}^{(i)}\|^2}$

This gives 3 equations (since it's a vector equation) with 3 unknowns (components of $\mathbf{c}$).

edited Sep 4 '11 at 11:39
Christian Rau
890 6 13

answered Sep 4 '11 at 0:32
valdo
541 2 5

1 Are you sure your computation of the distance is correct? The dot product computes the cos of the angle, whose sin we want, therefore we need the cross product (or rather it's norm) instead of the dot product. – Christian Rau Sep 4 '11 at 1:20

@Christian Rau: You are 100% right, I noticed that too and fixed. Thanks! – valdo Sep 4 '11 at 1:21

@Christian Rau: a particular example of a degenerate case is when you have only one line. Then any solution of the form **A** + n * **D** (i.e. points that sits on this line) should zero the equation. However it did not. Then I've spotted the mistake. – valdo Sep 4 '11 at 1:25

+1 I tried to work on the distance equation and derive it, but I completely forgot about Lagrange identity. Perfectly simple now. – Christian Rau Sep 4 '11 at 1:32

1 @g_niro: what we did is derived the formula for the H^2 - the square distance of the point **c** from the given line. Then we summed the H^2 over all the lines and argued that this is the sum that we want to minimize. However when we add weights into the problem - the formula is slightly redefined. Now we say that we sum the [H^2 * w] over all the lines, whereas *w* is the weight. And this is now the quantity that we want to mimimize. The rest of the derivation is very similar to what we did, except that every term in the sum should now be multiplied by the appropriate weight. – valdo Sep 4 '11 at 18:51

In order to find the intersection point of a set of lines, we calculate the point with minimum distance to them. Each line is defined by an origin a_i and a unit direction vector, n_i. The square of the distance from a point p to one of the lines is given from Pythagoras:

$$d_i^2 = [||p - a_i||]^2 - \left[(p-a_i)^T * n_i\right]^2 = (p-a_i)^T * (p-a_i) - \left[(p-a_i)^T * n_i\right]^2$$

Where $(p-a_i)^T * n_i$ is the projection of $(p - a_i)$ on the line i. The sum of distances to the square to all lines is:

$$\sum_i d_i^2 = \sum_i \left[(p-a_i)^T * (p-a_i) - \left[(p-a_i)^T * n_i\right]^2\right]$$

To minimize this expression, we differentiate it with respect to p.

$$\sum_i [2*(p-a_i) - 2*[(p-a_i)^T * n_i] * n_i] = 0$$

$$\sum_i (p-a_i) = \sum_i \left[n_i * {n_i}^T\right] * (p-a_i)$$

It results:

$$[\sum_i \left[n_i * {n_i}^T - I\right]] * p = \sum_i \left[n_i * {n_i}^T - I\right] * a_i$$

Where I is the identity matrix. This is a matrix $S * p = C$, with solution $p = S^+ * C$, S^+, being the pseudo-inverse of S.

Straightforward implementation can be found on:
http://www.mathworks.com/matlabcentral/fileexchange/37192-intersection-point-of-lines-in-3d-space

edited Apr 28 '16 at 15:35

answered Apr 28 '16 at 10:22

Alberto Izaguirre
11 ▲3

For each line i, let $\mathbf{p}^i$ be a point on the line and let $\mathbf{u}^i$ be a unit vector parallel to the line. Extend $\mathbf{u}^i$ to an orthonormal basis $\mathbf{u}^i$, $\mathbf{v}^i$, $\mathbf{w}^i$ (e.g., rotate the standard basis, use Gram–Schmidt, etc.). The overdetermined system of equations (in variable $\mathbf{x}$) is, for all i,

$(\mathbf{x} - \mathbf{p}^i) . \mathbf{v}^i = 0$
$(\mathbf{x} - \mathbf{p}^i) . \mathbf{w}^i = 0.$

Solve using the usual methods.

answered Sep 4 '11 at 0:10

quaint

Excuse me, but I don't see the rationale behind this method (i.e. why this should be correct). The Gram–Schmidt constructs the orthonormal basis from the linearly-independent vectors. So what? What's the connection? You may only do this for 3 vectors (all the others are zeroed by Gram–Schmidt). Can you explain please? – valdo Sep 4 '11 at 0:19

@valdo We repeatedly extend one vector to a basis rather than a collection of vectors. The idea is that ((x - pi) . vi)^2 + ((x - pi) . wi)^2 is the distance squared from x to line i. – quaint Sep 4 '11 at 0:27

Sorry, still don't get you. (1) Into which basis do you extend the **ui** vectors. Is it a randomly-chosen basis or what? (2) Why do you need this? The method you suggest is ok without this trick. You may just solve the over-defined equation system of (**x** - **pi**) * **ui** = 0. By changing the basis you insert unneeded complications to the problem (such as square root calculations and etc.) – valdo Sep 4 '11 at 0:42

P.S. The solution of the overdefined problem actually expands to the solution that I posted IMHO. – valdo Sep 4 '11 at 0:44