Transformations and Algorithms for Least Sum of Squares Hypersphere Fitting

Michael A. Burr* Alan C. Cheng† Ryan G. Coleman‡ Diane L. Souvaine§

Abstract

A problem from shape fitting is finding the sphere which has the least sum of squares fit to a set of points. Most current algorithms for this use an iterative process, with randomized restarting. We present, instead, two geometric transformations, and algorithms derived using them, which present alternate methods of solving the problem. We discuss running of these algorithms in three dimensions, and note the algorithms have favorable time complexities in high dimension. Finally, we present directions for future work.

1 Introduction

Some problems related to least sum of squares fitting have simple direct solutions, including that of finding a hyperplane through points. Other problems do not have such a simple solution, including that of fitting a hypersphere through points.

Least sum of squares fitting to a set of points offers one intuitive notion of the shape of the points. Certain problems, like fitting a line to points in $\mathbb{R}^2$, are easily solved by, for instance, taking the average of the points in $\mathbb{R}^2$, and then setting up a matrix and finding the eigenvalues and eigenvectors. The least significant eigenvalue corresponds to the eigenvector which is normal to the line, and the average of the points is a point on the line, which together completely defines the least sum of squares fitting line. This algorithm generalizes to finding a hyperplane through points in $\mathbb{R}^d$ [5, 3].

However, algorithms for fitting other shapes are not as direct. For the case we are concerned with, fitting a circle, sphere, or hypersphere to points in the co-dimension one case, all algorithms found are variations on a non-linear iterative algorithm [5, 3]. The idea of the algorithm is to place a center in space, evaluate the fit of the circle and which direction the center should be moved in to produce a better fit. The center is moved slightly and the process repeated until the center has converged. Since there can be local minima, the process must be restarted from several different locations to ensure the least sum of square fit is obtained.

*Department of Computer Science, Tufts University

†Research Technology Center, Pfizer Global Research and Development

‡Department of Computer Science, Tufts University and Pfizer Research Technology Center colemanr@eecs.tufts.edu

§Department of Computer Science, Tufts University

This algorithm has drawbacks. The running time is unbounded, and no upper bounds on the running time have been found. The algorithm requires a definition of converged when run with floating-point arithmetic. No known proof of correctness exists, and most implementations simply call for a number of restarts until the same local minima have been found.

Here we use ideas from geometry to convert the generalized sphere fitting problem into a problem more easily solved. Two particular transformations, inversive transformation and stereographic projection, are applied to the problem. Algorithms using both transformations are described. A preliminary practical implementation for the three-dimensional case is discussed, as are results from comparing these two algorithms with the non-linear iterative procedure.

2 Inversive Transformation and Algorithm

Inversive geometric transformations are well established as both geometrically interesting and useful from a computational perspective [1, 2]. We explain the one we use in detail. Consider any point (call it the inversive point) p, q, r and all points x_i, y_i, z_i. Around the inversive point we imagine an inversive sphere (not to be confused with the sphere we are trying to find) with a radius k. Each data point is now transformed according to the following equations:

$$x_i \mapsto \frac{k^2(x_i - p)}{(x_i - p)^2 + (y_i - q)^2 + (z_i - r)^2} + p$$

$$y_i \mapsto \frac{k^2(y_i - q)}{(x_i - p)^2 + (y_i - q)^2 + (z_i - r)^2} + q$$

$$z_i \mapsto \frac{k^2(z_i - r)}{(x_i - p)^2 + (y_i - q)^2 + (z_i - r)^2} + r$$

This transformation has several useful and interesting properties. First, it should be noted the inversion point p, q, r will be undefined under this transformation, since the denominator will be zero. Geometers solve this by taking it to infinity, and taking the point at infinity back to the inversion point. We, however, can ignore the inversion point's transformation. Second, the transformation, without the inversion point or infinity, is self-dual. Points transformed twice will return to the same point.

Finally, the most interesting property, and the one we exploit for the algorithm, is that for spheres that pass through the inversion point (*i.e.* if all the data points and inversion point lie on a sphere) the transformed points will lie on a plane. It follows that points distributed roughly on a sphere, with the least sum of squares fit sphere passing through the inversion point, will also lie near a plane.

Synthesis of the inverse transformation defined above, the technique for fitting data to a plane, and an assumption made about the data and the least sum of squares sphere, result in a quadratic time algorithm for finding the least sum square sphere fitting three dimensional data points. The assumption we make is that the least sum of squares sphere we are searching for passes through at least one of the data points. We find this to be a reasonable assumption in practice. This algorithm is good due to guaranteed runtime and accuracy as shown below. The possibility for improvements, and the generalization to any dimension, may prove useful for other applications.

The algorithm is straightforward and easy to implement with a eigensystem solver to do the least squares fitting to a plane.

1. Consider any set of points P to find the least sum of squares sphere of.
2. For each point $p_i \epsilon P$.
   a. Let p_i be the inversion point p, q, r and data points x_i, y_i, z_i be all the other points in P.
   b. Invert the data points according to the above transformations with $k = 1$. Call the resulting points t_i.
   c. Solve for the least sum of squares plane fit to the points t_i using standard techniques.
   d. Find the point on the plane closest to p_i. Call this a.
   e. Transform a according to the above transformation, with $k = 1$. Call this a'.
   f. p_i and a' are two points on a diameter of sphere. Their average is the center c_i.
   g. Find the best radius for the sphere given this center.
   h. Keep this center and radius as best if they fit better than the current least sum of squares sphere.
3. Output the best found center and radius.

Conceptually, the algorithm asks, if the best sphere passes through it, what would the center of that sphere be? It generates candidate centers for each point, and tests each one to see which fits best. The overall complexity of the algorithm is $O(n^2d^2)$ where n is the number of input points and d is the dimension of the problem.

However, this is not the entire story for this algorithm. Ideally, during each iteration, the fit of the plane in inversive space should correlate to the fit of the sphere in the normal space. To check this, some sample data sets were checked for the case of fitting a sphere to points in $\mathbb{R}^3$. However,

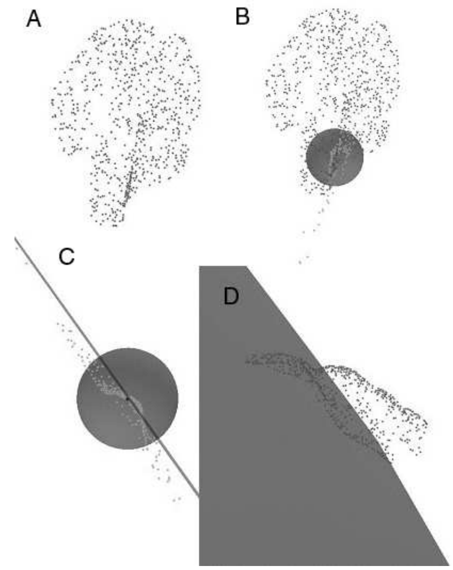

Figure 1: The inversive algorithm in $\mathbb{R}^3$. A. The data points in darker gray. B. The inversive sphere for one step shown, the inverted points shown in light gray. C. The weighted plane found shown with a rotated version of the inverted data points. D. The sphere fit from this single iteration.

when a correlation line was fit to the least sum square fits in both spaces, the R^2value calculated was 0.23.

This low R^2value can be attributed to the following: points very close to the inversion point but slightly off the sphere being searched for will end up very far away from the plane in the inversive space. This large distance can skew the plane fit. The solution is to assign the points a weighting based on their distance in the normal space. However, an ideal weighting scheme was not found from the geometry, and several options were tested empirically. The best weighting scheme weights each point by d_i^4 where d_i is the distance in the normal space from the inversion point.

This weighting changes the setup of the matrix to solve in step c. The problem is now a weighted least sum of square plane fit, but can be solved without asymptotically more work. The R^2value of the correlation between the weighted sum of squares fit of the plane to the sum of squares fit of the sphere using the d_i^4 weighting scheme was 0.9252, which is not perfect and suggests there is room for improvement.

3 Stereographic Projection and Algorithm

The second algorithm uses a *stereographic projection* also known as a *Riemann sphere transformation*. The transformation maps a d-dimensional space to a $d+1$-hypersphere. In $\mathbb{R}^3$, the transformed points all lie to a 4-sphere centered at the origin with a radius k:

let $a = x^2 + y^2 + z^2$ then $(x, y, z) \mapsto \left(\frac{2x}{a+k}, \frac{2y}{a+k}, \frac{2z}{a+k}, \frac{a-k}{a+k}\right)$

The key property taken advantage of in this algorithm is that points on a sphere in the d-dimensional normal space will lie on a d-dimensional sphere on the $d+1$-dimensional hypersphere. In the case presented, points on a 3-sphere in $\mathbb{R}^3$ will lie on a 3-sphere on the 4-sphere in the transformed space. This 3-sphere defines a unique hyperplane in $\mathbb{R}^4$, and can be easily found again using the technique of least squares fitting to a hyperplane.

The assumption made here is again that points close to a sphere in the original space will lie close to a sphere in the transformed space. Again, this is not completely accurate, as points very far away from the origin map to points very similar, while points close to the origin could skew the plane found. We use a weighting scheme derived for the 2-dimensional case of circle fitting of $\left(1+d_i^2\right)^2$, where d_i is the distance in normal space from the origin [6]. We tested several other weighting schemes experimentally and found this one to have the best R^2value and showing good correlation between the problems in either space.

For this algorithm, points must be shifted so that they are closer to the origin, since if all points are far from the origin, they will map to nearly the same point in the transformed space. With these considerations in mind, we can write an algorithm:

1. Consider any set of points P to find the least sum of squares sphere of.
2. Shift all points in P so that the origin lies at the average of the points.
3. Transform points onto the $d+1$-dimensional Riemann sphere. Let $k=1$.
4. Give each point a weight according to $\left(1+d_i^2\right)^2$ where d_i is the distance from the origin in normal space.
5. Setup and solve the weighted matrix to find the normal to the
plane that best fits the transformed points.
6. With the normal, and the distance from the hyperplane, directly solve for the center in normal space.
7. Find the best radius as the average distance from the center to all the points.
8. Output the found center and radius.

This algorithm is straightforward, and easy to code with an eigensystem solver. The overall time complexity is $O(n \cdot (d+1)^2)$, again where n is the size of the input and d is the dimension. It is generally better than the previous algorithm in terms of the complexity, since it trades an order of magnitude in terms of the input size for the small penalty of a slightly bigger matrix to setup and solve. In effect this algorithm solves the problem by taking it up a dimension and trasforming the answer back down. This is only done once whereas the inversive algorithm loops over every data point.

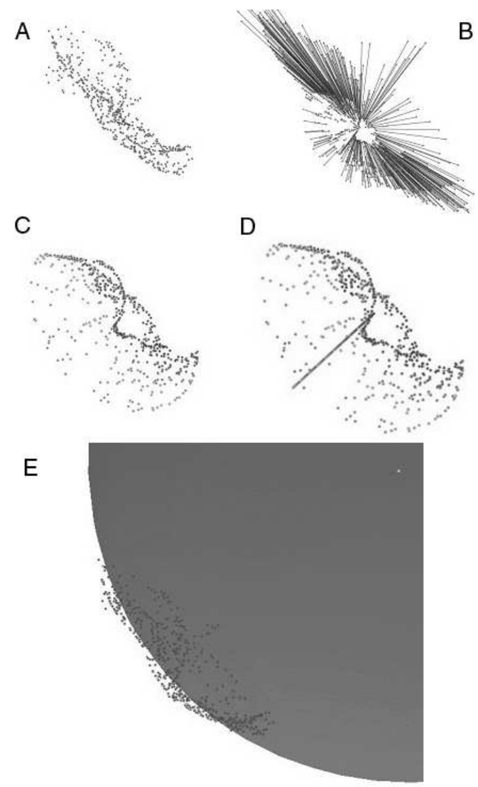

Figure 2: The Riemann algorithm in $\mathbb{R}^3$. A. The data points in dark gray. B. The points transformed onto the hypersphere with lines connecting the original and transformed points. C. The points (with dimension 4 shown as grayscale) on a 4-sphere. D. The weighted normal found. E. The final least sum of squares fit sphere found by the algorithm.

4 Implementation Comparison

These three algorithms were implemented and compared using Java, Java3D for vector math and visualization, and JAMA to solve eigenvalues and eigenvectors. We only tested our results for $d = 3$. We were encouraged by good results for the $d = 2$ case of the Weighted Riemann Sphere algorithm done by Strandlie *et al.* [6], and many implementations of the iterative procedure [5, 3]. Our goal was to test the efficiency and accuracy in the three dimensional case.

Our sample point sets were roughly between 1 and 10 units across in any dimension. This produced good results when using inversion or Riemann spheres of radius 1. Further effect of the radius on the accuracy should be evaluated.

Two example runs of the algorithms presented here are shown in Figure 1 and 2. Again, these examples are for the $\mathbb{R}^3$ case.

We compared both the accuracy and speed of the three algorithms. In our implementation of the non-linear iterative procedure, the center was assumed to have converged if it did not move in any direction by more than 0.000005 units. While this value may seem high, it did result in running times that were practical. Also, only 2 restarts were used, with the second forced to start in the opposite direction of the first. This along with a maximum number of iterations set to 10000 allowed the algorithm to report good results in reasonable time. More accurate implementations could slow the overall running time.

No algorithm as implemented performed consistently better than the other two. For low numbers of points, the Inversive and Riemann algorithms performed equally well, while the non-linear iterative algorithm did not match the performance. For large numbers of points, the Inversive and Non-linear algorithms performed equally well, and the Riemann algorithm had much worse accuracy. The effect of Riemann sphere radius being larger may improve the accuracy in cases where the number of points is large.

In terms of speed, the asymptotic bounds shown for the Inversive and Riemann algorithms were verified. The Non-linear algorithm, though having no theoretical upper bounds, did execute faster than the Inversive algorithm. The Riemann algorithm grew linearly in the size of the input, as expected, and was much faster than the other two algorithms, at the expense of being less accurate (as implemented) for large numbers of points.

5 Conclusions & Future Work

The two algorithms presented here as options to a non-linear iterative procedure have been shown to work in test cases, and further work could make them generally useful. A better weighting scheme derived analytically from the problem would be a great improvement, as would be the testing of the effect of varying the sphere radius used in the transformation.

Another possibility is to attempt to use the fast Riemann algorithm as a best guess for the non-linear procedure, with the hope of converging quickly to the global minima which corresponds to the least sum of squares fit sphere.

In terms of dimension, these algorithms have favorable time complexities. Other shape fitting algorithms from computational geometry typically have complexities that are exponential in terms of d [4]. These algorithms only have quadratic complexities, which could be useful in high dimensional cases.

There are many other directions for future research. Transformations for other shape fitting problems, like ellipsoids, could be explored. Also, instead of the least sum of squares fit, a least median squares fit could be explored.

References

[1] D. A. Brannan, M. F. Esplen, and J. J. Gray. *Geometry: Inversive Geometry* pages 199–260. Cambridge University Press, 1999.

[2] D. P. Dobkin, and D. L. Souvaine. 1987. *Advances in Robotics 1: Algorithmic and Geometric Aspects of Robotics: Computational Geometry – A User's Guide*. (eds. J.T. Schwartz, and C. K. Yap), pages 43–93. Lawrence Erlbaum Associates, 1987.

[3] D. Eberly. Least squares fitting of data. `http://www.magic-software.com/Documentation/ Least-SquaresFitting.pdf` pages 1–9. 2001.

[4] S. Har-Peled. and K. R. Varadarajan. High-dimensional shape fitting in linear time. In *Symposium on Computational Geometry*. pages 39–47, 2003.

[5] C. M. Shakarji. Least-Squares fitting algorithms of the NIST algorithm testing system. In *J. Res. Natl. Inst. Stand. Technol.* **103** pages 633–641, 1998.

[6] A. Strandlie, J. Wroldsen, R. Frühwirth, and B. Lillekjendlie. Particle tracks fitted on the Riemann sphere. In *Comp. Phys. Comm.* **131** pages 95–108, 2000.