Proceedings of the ASME 2015 International Mechanical Engineering Congress & Exposition
IMECE2015
November 13-19, 2015, Houston, Texas

IMECE2015-50654

A CONSTRAINED L_2 BASED ALGORITHM FOR STANDARDIZED PLANAR DATUM ESTABLISHMENT

Craig M Shakarji
Physical Measurement Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
craig.shakarji@nist.gov

Vijay Srinivasan
Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
vijay.srinivasan@nist.gov

ABSTRACT

For years (decades, in fact) a definition for datum planes has been sought by ASME and ISO standards writers that combines the contacting nature of traditional surface plate mating with a means of balancing rocking conditions when there is a centrally positioned extreme point or edge on the datum feature. This paper describes a completely self-balancing method for datum plane establishment that matches traditional surface plate mating while automatically stabilizing rocker conditions. The method is based on a constrained L_2 (L2) minimization, which, when seen mathematically, elegantly combines the desirable contact properties of the constrained L_1 (L1) minimization with the stable properties of the unconstrained least-squares and does so in a manner that avoids the drawbacks of either of those two definitions. The definition is shown along with proofs of a mathematical development of an algorithm that relies on a strategically chosen singular value decomposition that allows for an elegant, robust solution. Concise code is included for the reader for actual use as well as to full clarify all the algorithmic details.

Testing has shown the definition defined here does indeed provide attractive balancing of full contact with rocker stability, leading to guarded optimism on the part of the key standards committees as an attractive default definition. Since both the ISO and ASME standardization efforts are actively working to establish default datum plane definitions, the timing of such a rigorously documented study is opportune.

1. BACKGROUND AND INTRODUCTION

In the world of Geometric Dimensioning and Tolerancing (GD&T), datums are used extensively to locate and orient tolerance zones [1-7]. Datum planes in particular are common and are established by mating planes to imperfect datum features on parts during inspection [3] (see Fig. 1). Distances and orientations on drawings and three-dimensional models are established from these datum planes, relative to which tolerance zones are located and oriented. In many cases there is a need for more than one datum plane. In fact a full Cartesian coordinate system in three dimensions is often established using datums. Datum planes, in particular, are widely used for this. The importance and prevalence of datum planes in specifications are given in greater detail in [8] and will not be revisited in this paper.

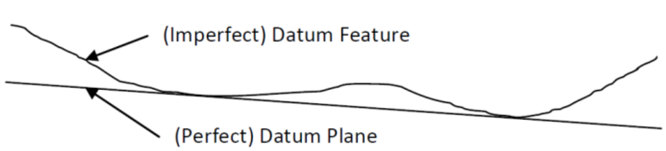


Fig. 1. Deriving a datum plane from a datum feature.

Given that datum planes are ubiquitous, it might be surprising that—short of standardization—there are several different yet reasonable approaches by which a datum plane can be established from a datum feature [9]. Furthermore, the International Organization for Standardization (ISO) and the American Society for Mechanical Engineering (ASME) are actively working to establish default datum plane definitions.[1]

[1] The constrained L_2 planar datum definition, as described in this paper, has been adopted as the default planar datum definition for by ISO for the Draft International Standard ballot to take place in 2015 for the revision of ISO 5459

Consequently, the timing of this paper is opportune, since we seek to demonstrate an algorithm that naturally combines a correspondence to physical, surface plate mating with automatic balancing in the case of rockers.

Till now the ASME definitions (ASME Y14.5, Y14.5.1) have employed a complex "candidate" datum system, which they now desire to replace or supplement with a default, unambiguously defined planar datum. The ISO working group is also seeking to improve its default planar datum definition in its emerging replacement of the ISO 5459 standard. The ISO definition (ISO 5459) had, since 1982, relied on non-rigorous language that implies using the full contact of a surface plate with balancing in the case of rocking conditions (and an intermediate "improvement" has its own issues). Both standards groups seek a mathematical definition that makes sense in ordinary cases of surface plate mating but one that also balances rocking conditions. The purpose of this paper is to document a new and advantageous definition and algorithm for establishing a datum plane from a datum feature—one that is appropriate for national and international standard definitions.

In Section 2 of this paper, we define what the L_2 norm is in the context of datum planes. Section 3 gives details of another planar datum definition based on a constrained L_1 norm that will give the appropriate context to understand the benefit of the constrained L_2 solution. Section 4 details the constrained L_2 algorithm and gives mathematical details that show how it is actually a combination of traditional least-squares fitting and the constrained L_1 datum. That section also gives mathematical means for an efficient algorithm. Section 5 is an important part of the paper, as it answers why L_2 the constrained datum definition is appealing in that it automatically gives the desired result of a full contact or balancing solution. Section 6 gives our conclusions. Matlab code for the 2D case is included in the appendix for any readers who wish to independently examine the effects of the algorithm on various data sets.

2. L_2 NORM DEFINED IN THE CONTEXT OF DATUM PLANES

First, we describe what is meant by a constrained L_2 fit in our context.[2] To fit a one-sided L_2 plane to a surface patch in space, we pose the following optimization problem (with reference to Fig. 2): Given a bounded surface S, and a direction $\boldsymbol{a}^*$ (that points into the material), find the plane P that minimizes $\int_S |d^2(\boldsymbol{p},P)|ds$, subject to the constraint that P lies entirely to one side (as determined by $\boldsymbol{a}^*$) of the surface S.

Here $d(\boldsymbol{p},P)$ denotes the signed perpendicular (to P) distance of a point $\boldsymbol{p}$ on surface patch S from the plane P that will be fitted. We note that $\int_S ds$ is the area of the surface patch. If the surface consists of several patches, then the integrals can be evaluated over each patch and then summed.

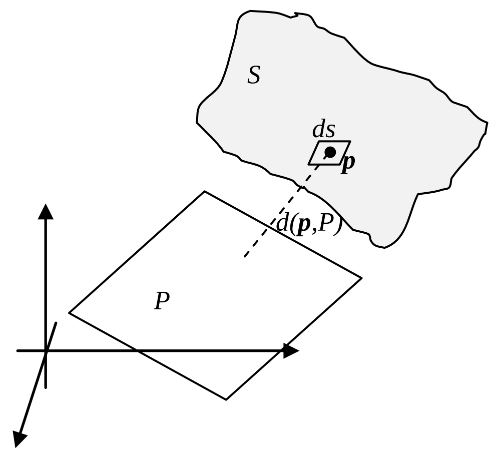


Fig. 2. Fitting a plane to a surface patch.

The objective function cannot, in general, be evaluated in closed form. So we resort to numerical integration over the surface S. We can sample points on a surface patch after dividing up the patch into discrete areas ΔA_i and approximate the objective function as

$$\int_S d^2(\boldsymbol{p},P)ds \approx \sum_{i=1}^{N} d^2(\boldsymbol{p}_i,P)(\Delta A_i)\,, \qquad (1)$$

where $\boldsymbol{p}_i$ are the N sampled points, one in each subdivision. Thus we are led to minimizing $\sum_{i=1}^{N}[|d(\boldsymbol{p}_i,P)| \cdot \Delta A_i\,]$ over the parameters of the plane P, where ΔA_i's are treated as the weights.

The distance from a point $\boldsymbol{p}$ to a plane P defined by a point on the plane, $\boldsymbol{p_0}$, and the unit normal to the plane, $\boldsymbol{a}$, is

$$d(\boldsymbol{p},P) = \boldsymbol{a} \cdot (\boldsymbol{p} - \boldsymbol{p_0}).$$

The two-dimensional case is a readily-apparent restriction from the three-dimensional case shown above.

3. A BRIEF LOOK AT THE CONSTRAINED L1 MINIMIZATION DATUM PLANE DEFINITION

Before examining the advantageous properties of the constrained L_2, it is helpful to understand the constrained L_1 datum plane definition—both its advantages and disadvantages. Doing so will highlight how the new, constrained L_2 definition largely keeps the advantages of the constrained L_1, along with elegantly removing the issues with the constrained L_1.

on datums. Thus it is likely that this datum plane definition will be adopted for worldwide use.

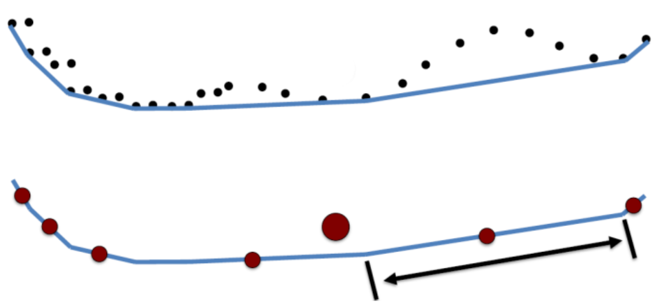
[2] The L_2 norm is also known as a least-squares norm. However, in this paper, in order to avoid confusion, the datum definition we propose is consistently called the constrained L_2 datum. It is not called a constrained least-squares plane (though correct) in order to emphasize that this is different than the normal least-squares plane and also different from a shifted least-squares plane.

In an earlier paper [8] and then improved in [10], we presented the theory and algorithms for datum plane establishment using a constrained minimization search based on the L_1 norm. In short, the algorithm worked as follows: Given a surface (or set of sampled points), the datum plane was defined as the plane that (1) is constrained to lie on the nonmaterial side of the surface (or points), and (2) minimizes the integral (or sum) of absolute distances between the plane and the surface (or points). We showed that finding such a plane actually turns out to be quite simple, since we proved that it is equivalent to finding the plane that minimizes the distance between the centroid of the surface (or of the weighted points) and the plane. This simplification led to efficient algorithms (and code provided) for the primary and secondary planar datums (the tertiary case being trivial).

The reader is encouraged to fill in details as desired from the earlier paper itself [10], but we give a summary of the constrained L_1 algorithm as follows:

1) Given a set of points sampled on a surface, compute the lower convex envelope of those points. This surface is the part of the convex hull of those points that lies to the outside of the material. The constrained L_1 definition will now be applied to this surface (as opposed to the points)
2) Compute the centroid of the surface as the weighted combination of the centroids of the triangles making up the convex surface. In 2D, the centroid of the convex, piecewise linear curve would be computed as the weighted combination of the centroids (midpoints) of the line segments that it is comprised of. The weights are the relative areas of the triangles (or relative lengths of the line segments in 2D, one such length shown in fig. 3, middle picture).
3) Find the plane containing a triangular facet of the convex hull closest to the computed centroid (or, in 2D, find the line containing a line segment of the curve closest to the centroid).

Figure 3 shows these three steps in a 2D case.



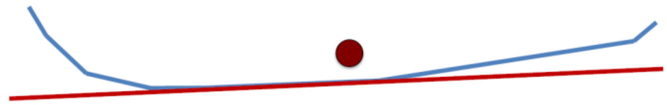


Fig. 3. The three main steps of computing the constrained L_1 datum plane, given a discrete set of points.

Theorems were proved in [10] that showed that the algorithm summarized above is an efficient means of exactly obtaining the constrained L_1 datum plane. Some of the appealing properties of this method are:

1) It mimics the contact achieved by the effect of gravity, if the surface were placed onto a mathematically perfect, horizontal plane.
2) In a 3-2-1 datum reference frame, the primary datum plane always contacts three data points (minimum) and the secondary, always two minimum. This is in the context of discrete, sampled points.
3) The method works well even for non-uniformly sampled data without needing any weights to be provided for the points or any part information.
4) The method yields pleasing results for several example cases studied.

Other advantages are given in [10], but these should suffice for our needs here. In summary, the appeal of the constrained L_1 definition is how closely it mimics many uniform-thickness, real parts sitting on surface plates under the influence of gravity.

However, common practice with a surface plate also employs balancing rocker conditions as shown in fig. 4.

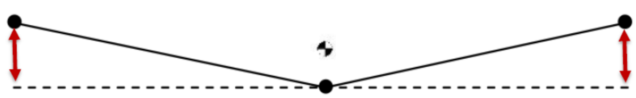


Fig. 4. A planar datum feature of a wedge shape being stabilized to avoid rocking, thus giving the dashed line shown as the datum.

If the constrained L_1 definition were applied to the wedge shape shown in fig. 4, the datum plane would lie coincident with one side or the other of the datum feature. This drawback manifests itself in a few important ways. First, if the part were convex (bowl shaped) and sampled with five points (one in each corner and one in the middle) then the effect would be that of an upside-down pyramid, and the constrained L_1 plane would coincide with one of its triangular faces. In a symmetric case, the choice of which triangular face would be chosen would depend on something as little as measurement error during the time of inspection.

Another case to consider is a 3D concave datum feature, where a rectangular feature has four low spots, one at each corner. In this case, the part would naturally sit on a horizontal plane like a four-legged chair. That is, it would rest along one diagonal (contacting two opposite corners) and rock between contacting either of the two corners off that diagonal. Here again, the desire among many in the standards communities is to balance that rock, a feature the constrained L_1 definition does not employ.

It does little good to seek to remedy the various rocking situations described by simply stating in words that the constrained L_1 definition holds except in rocking situations, where the rock should be balanced. This is insufficient (1) due to the lack of rigor in defining a rocking condition and (2) due to the lack of rigor in defining how the rock should be balanced. But even if crisp definitions are added to the above words, there would still be discontinuities at the thresholds of rocking/non-rocking states that could lead to instability in the resulting datum plane from one measurement to the next.

In contrast to the problems just described with the constrained L_1 datum definition, it is well known that the traditional least-squares fitting plane is a smoothly varying, stable association to a planar feature.[3] We will show that the constrained L_2 takes the best of both worlds. It exactly matches the L_1 solution when there is not a rocker condition and also (naturally and automatically) balances rocker conditions smoothly (like traditional least-squares) without any special "if" statements employed to do so.

4. THE CONSTRAINED L_2 DEFINTION AND EFFICIENT ALGORITHM

As in the constrained L_1 defintion above, the proposed constrained L_2 datum plane definition first forms the lower convex surface of the datum feature and then finds the plane that minimizes the sum-of-squares (or integral, in the continuous case) of the distances from the plane to that convex surface.

The reasons for forming the convex envelope first are given in detail in [10], but are summarized by these three points: (1) it represents the actual interaction of a plane with the feature (if one rocks a datum feature on a perfect plane, the plane never contacts the concave sections), (2) it prevents the need for weights or part information when given discrete data points, since the convex envelope allows appropriate weighting to be included in the algorithm itself, and (3) it better handles broken surfaces.

For simplicity sake, the remainder of this section will often deal with the two-dimensional case, though we will still use the terms "plane" and "surface" instead of "curve" and "line" since all these concepts will apply to the 3D case as well.

Given a set of points (as shown in fig. 5), we compute the lower convex surface as shown.

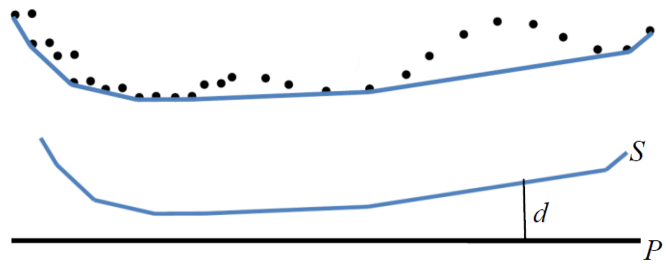


Fig. 5. Above: The lower convex envelope computed from a set of points. Below: A candidate datum plane P is shown along with its distance to a point of the surface.

It is important to emphasize that we now seek find the plane that minimizes the constrained L_2 objective function between the plane and <u>convex surface</u>, not the original points. So then, applying the constrained L_2 norm to the convex surface, we seek to minimize, from Eq. (1),

$$\int_S d^2(\boldsymbol{x}, P)\,ds\,, \tag{2}$$

where the plane P is constrained to lie on the non-material side of the convex surface S. It is immediately clear that the P that minimizes the objective function will contact S, since, if it did not, the objective function could be lowered by shifting P closer to S.

If S is obtained as the convex surface formed from discrete input points, then it is a piecewise linear surface. (In 3D it is a union of discrete triangles). For any candidate plane, P, the solution to equation (2) can be found by summing individual integrals along each line segment of S. But the solution to (2) over each line segment will be a 3rd degree polynomial.

However, the problem can be converted into a least-squares problem, which will allow a much more efficient numerical solution. Simpson's rule [11] is a numerical integration technique that uses three function values at the left, right, and middle of an interval to approximate the integral of a function over an interval (fig. 6) and a similar method for integrating over a triangle in our 3D case. While Simpson's rule is generally an approximation, it has been proved that it is exact for integrals of functions that are polynomials of degree 2, which is the case here. Therefore, we can solve (2) exactly over each line segment (or triangle) that comprises S in order to solve a minimum sum-of-squares problem using well known methods.

[3] We do not go into detail here about the disadvantages of a least-squares or shifted least-squares datum definition. That has been done in [10]. We only note here the advantage of its stability in order to show that the constrained L_2 definition contains a similar appealing property.

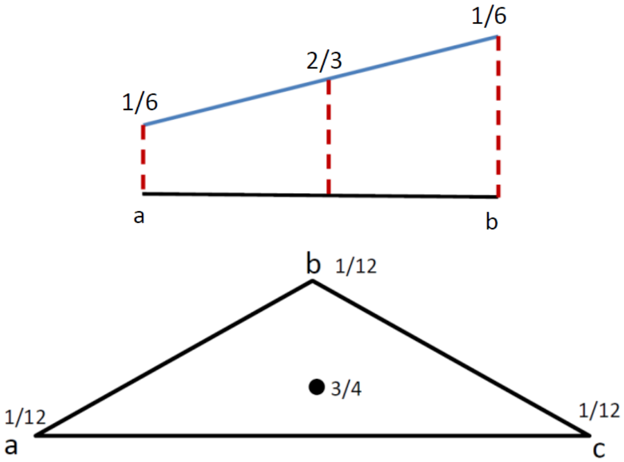


Fig. 6. The locations and weights for function evaluations for numerical integration using Simpson's rule over an interval and triangle.

Simpson's rule for integrating over an interval or triangle depends only on the weighted values of the function at the endpoints (vertices) and centroid. Over an interval, Simpson's rule is given by:

$$\int_a^b f(x)dx \approx (b-a)\left(\frac{1}{6}f(a) + \frac{2}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{6}f(b)\right),$$

and for integrating over a triangle, T, as shown in fig. 6,

$$\int_T f(\boldsymbol{s})dT \approx$$
$$\text{Area}(T)\left(\frac{1}{12}f(a) + \frac{1}{12}f(b) + \frac{1}{12}f(c) + \frac{3}{4}f\left(\frac{a+b+c}{3}\right)\right).$$

If each line segment of S is called S_i having left endpoint $\boldsymbol{x_i}$, right endpoint $\boldsymbol{x_{i+1}}$, midpoint, $\boldsymbol{m_i}$, and length L_i, (i = 1, 2, …, N, the number of edges and where L denotes the total length, $L = \sum_{i=1}^{N} L_i$) then Simpson's rule gives the integral evaluation as

$$\int_{S_i} |d^2(\boldsymbol{x}, P)| ds = \frac{L_i}{6}[d^2(\boldsymbol{x_i}) + 4d^2(\boldsymbol{m_i}) + d^2(\boldsymbol{x_{i+1}})]. \quad (3)$$

Because Simpson's rule is exact for functions of degree 2, we note that in Eq (3) this is an exact calculation of the integral and not a mere approximation. (Simpson's rule is also exact for our 3D case). The framing of this problem as a weighted sum-of-squares now allows us to solve the objective function as a singular value decomposition problem. See [12] for a general treatment of the singular value decomposition as a method for minimizing the total least-squares problem, and [13] for an application of it applied to planar fitting with weighted points, which is our case here.

In the Appendix, we prove theorems 1 and 2, which when applied to our applications give us the remarkable result, that (in 2D) the objective function for any candidate plane P is given by the efficient formula:

$$\sigma_1^2 \text{Cos}^2\theta + \sigma_2^2 \text{Sin}^2\theta + L d_c^2, \quad (4b)$$

or equivalently

$$\sigma_1^2 a^2 + \sigma_2^2 b^2 + L d_c^2, \quad (4a)$$

where (see fig. 7) d_c is the distance from the plane P to the centroid, σ_1 and σ_2 are the singular values from the singular value decomposition (SVD, of the matrix $\boldsymbol{M}$ below), and θ represents the angle P makes with the singular vector corresponding to the smallest singular value, σ_1. (Eq. (4b) is just a restatement of (4a), where $(a, b) = (\text{Cos}\theta, \text{Sin}\theta)$ is the unit normal to the candidate plane expressed as dot products with the singular vectors.) The $3N \times 2$ matrix, $\boldsymbol{M}$, that is used in the singular value decomposition comes from the elements of Eq. (3), repeated for each of the N line segments:

$$\boldsymbol{M} = \sqrt{\frac{1}{6}}\begin{bmatrix} \sqrt{L_1}(x_1) & \sqrt{L_1}(y_1) \\ 2\sqrt{L_1}\left(\frac{x_1+x_2}{2}\right) & 2\sqrt{L_1}\left(\frac{y_1+y_2}{2}\right) \\ \sqrt{L_1}(x_2) & \sqrt{L_1}(y_2) \\ \vdots & \vdots \\ \sqrt{L_N}(x_N) & \sqrt{L_N}(y_N) \\ 2\sqrt{L_N}\left(\frac{x_N+x_{N+1}}{2}\right) & 2\sqrt{L_N}\left(\frac{y_N+y_{N+1}}{2}\right) \\ \sqrt{L_N}(x_{N+1}) & \sqrt{L_N}(x_{N+1}) \end{bmatrix}$$

(The construction of $\boldsymbol{M}$ is done with the data translated so the centroid is at the origin. This translation is not shown explicitly in the matrix due to lack of space. See Theorem 1 in the appendix for further details.)

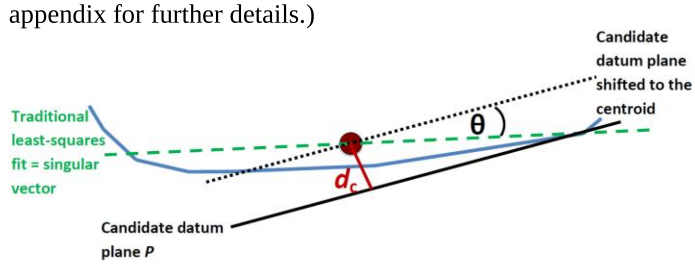


Fig. 7. The objective function for any candidate datum can be found simply by finding the angle θ and distance d_c and using Eq. (4).

Using Eq. (4) to compute the objective function means that the singular value decomposition only has to be computed once and its result can be applied to any given candidate datum plane. This makes for a much more efficient minimization algorithm.

What is fascinating about Eq. (4) is that the first two terms are exactly the objective function used in a traditional least-squares minimization while the last term is the objective function in an L_1 fit. And we will see that the objective function indeed does manifest itself as having the properties of both, which is what is desired.

This can extend to 3D as well, since we showed that there is an extension of Simpson's rule that applies to integration over a triangular region. For the 3D case, the objective function for any candidate plane P is given by the efficient formula:

$$\sigma_1^2 a^2 + \sigma_2^2 b^2 + \sigma_3^2 c^2 + A d_c^2, \quad (5)$$

where d_c is the distance from the plane P to the centroid, σ_1, σ_2 and σ_3 are the singular values from the singular value decomposition (SVD, of the matrix $\boldsymbol{M}$ below), and (a, b, c) is the unit normal to the candidate plane P expressed as the dot product of that normal with each of the three singular vectors. Applying Simpson's rule for each of the N triangles, the $4N \times 3$ matrix, $\boldsymbol{M}$, that is used in the singular value decomposition is:

$$\boldsymbol{M} = \sqrt{\frac{1}{12}} \begin{bmatrix} \sqrt{A_1} x_{1\mathrm{A}} & \sqrt{A_1} y_{1\mathrm{A}} & \sqrt{A_1} z_{1\mathrm{A}} \\ \sqrt{A_1} x_{1\mathrm{B}} & \sqrt{A_1} y_{1\mathrm{B}} & \sqrt{A_1} z_{1\mathrm{B}} \\ \sqrt{A_1} x_{1\mathrm{C}} & \sqrt{A_1} y_{1\mathrm{C}} & \sqrt{A_1} z_{1\mathrm{C}} \\ 3\sqrt{A_1} \bar{x}_1 & 3\sqrt{A_1} \bar{y}_1 & 3\sqrt{A_1} \bar{z}_1 \\ \vdots & \vdots & \vdots \\ \sqrt{A_N} x_{N\mathrm{A}} & \sqrt{A_N} y_{N\mathrm{A}} & \sqrt{A_N} z_{N\mathrm{A}} \\ \sqrt{A_N} x_{N\mathrm{B}} & \sqrt{A_N} y_{N\mathrm{B}} & \sqrt{A_N} z_{N\mathrm{B}} \\ \sqrt{A_N} x_{N\mathrm{C}} & \sqrt{A_N} y_{N\mathrm{C}} & \sqrt{A_N} z_{N\mathrm{C}} \\ 3\sqrt{A_N} \bar{x}_N & 3\sqrt{A_N} \bar{y}_N & 3\sqrt{A_N} \bar{z}_N \end{bmatrix}$$

(The construction of $\boldsymbol{M}$ is done with the data translated so the centroid is at the origin. This translation is not shown explicitly in the matrix due to lack of space. See Theorem 1 in the appendix for further details.)

The notation used in showing $\boldsymbol{M}$ (just above) assumes the surface is comprised of N triangles T_i , each having area A_i and vertices ($x_{i\mathrm{A}}$, $y_{i\mathrm{A}}$, $z_{i\mathrm{A}}$), ($x_{i\mathrm{B}}$, $y_{i\mathrm{B}}$, $z_{i\mathrm{B}}$), and ($x_{i\mathrm{C}}$, $y_{i\mathrm{C}}$, $z_{i\mathrm{C}}$), their average being $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$

We can summarize the 3D constrained L_2 algorithm as follows (the 2D case being similar):

Given:

1) Data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots, \boldsymbol{x}_M$, where each $\boldsymbol{x}_i = (x_i, y_i, z_i,)$, and
2) A direction, $\boldsymbol{a}^*$ that indicates the direction into the material,

then the datum plane is established using the following steps:

1) Compute the convex hull of the data points and represent it by the union of a set of triangles.
2) Select the N triangles (where $N < M$) that are exterior to the material (i.e., the triangles that comprise the lower convex envelope). This can be accomplished by computing the normal to each triangle (pointing into the hull) and comparing its direction to $\boldsymbol{a}^*$. (The sign of the dot product can easily be used here).
3) Compute the centroid, $\bar{\boldsymbol{x}}$, of the convex surface of Step 2. The centroid of each triangle can be trivially computed as the average of its vertices. The sum of these centroids when weighted by their relative areas is the centroid of the lower convex envelope. If the N triangles each has area A_i, then each relative weight is $w_i = A_i / \sum_{i=1}^{N} A_i$.
4) Construct the matrix $\boldsymbol{M}$ as defined above and compute its singular value decomposition to obtain the singular values σ_1 , σ_2 and σ_3 and their corresponding singular vectors.
5) The objective function can now be used efficiently in a minimization algorithm to find the optimal plane that is constrained to lie on one side of the material. Given any candidate orientation, the candidate plane can be found easily by shifting it just to the outer edge of the material. The objective function of this candidate plane can be easily computed using Eq. (5).

Before moving on to the next section that highlights why the algorithm is so appealing to the standards writers, we note that the only three nontrivial mathematical functions needed for implementation of this algorithm are (1) a convex hull function, (2) a singular value decomposition function, and (3) a minimization function. All three of these are well researched, documented, and available to the numerical community. In fact, the minimization algorithm (3) can be eliminated, as is explained in the code in the appendix, where an even more efficient solution is explained.

5. THE APPEALING PROPERTIES OF THE CONSTRAINED L_2 DATUM PLANE DEFINITION

When we saw that the L_2 constrained objective function in Eq. (4) was in fact a combination of L_1 and traditional least-squares objective functions, we suspected that this datum plane definition might manifest itself as combining the advantageous properties of them both. This turns out to be the case. Figure 8 shows two typical cases where, on the left, one would seek to balance the rocking condition, and on the right, one would seek for the datum plane to be stably flush with the edge of the datum feature. This is what the constrained L_2 solution does automatically.

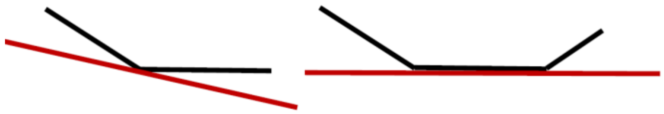


Fig. 8. Two typical cases of datum features with the associated constrained L_2 datums shown. The balanced rocking case is on the left and the stable, flush case is on the right.

For the rocker condition pictured on the left side of fig. 8, if the line segment on the right were made longer, the constrained L_2 datum plane would roll to the right smoothly. For the stable case pictured on the right side of fig. 8, if the line segment on the right were made somewhat longer, the L_2 constrained datum plane would not move from its stable state. It would remain flush with the edge of the datum feature until the line segment on the right grew long enough to make a rocker condition, at which point the L_2 constrained datum would smoothly begin to roll to the right to balance the rocker.

In contrast, the shifted least-squares solution would achieve a flush mating with the datum feature (as pictured on the right of fig. 8) for only an instant. That is, as the line segment on the right began to be extended, there would only be one length that resulted in a flush mating. This contrast shows the fascinating feature of the constrained L_2, which stays flush with the datum feature—even while the line segment extends—until it reaches such a length that a rocking condition exists, like shown in fig. 9.

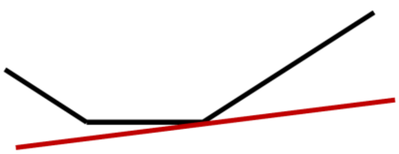


Fig. 9. The line segment on the right is long enough for the constrained L_2 datum to treat it as a rocking condition and separate from the flush contact it had in the right hand picture of fig. 8.

6. CONCLUSIONS

The constrained L_2 datum definition for planes has the remarkable benefit of combining desired properties from both the constrained L_1 definition and traditional least-squares definition, which each have their deficiencies by themselves. We have shown that the objective function in the constrained L_2 definition actually can be mathematically broken down to be seen (perhaps unexpectedly) as a combination of the objective functions of the constrained L_1 and traditional least-squares. Furthermore, a careful application of Simpson's rule and singular value decomposition (which is widely available) allows for the objective function to be evaluated efficiently and solved with popular optimization algorithms. 2D code in Matlab is provided in the appendix for the reader and has been evaluated in numerous test cases to be found appealing in its behavior and stable in its results.

REFERENCES

[1] Srinivasan, V., "Reflections on the role of science in the evolution of dimensioning and tolerancing standards," Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 227, No. 1, pp. 3-11, 2013. DOI: 10.1177/0954405412464012

[2] Tandler, W. "All Those Datum Things" *Inside Metrology*, Quality Digest, Quality Digest Magazine, 13 February 2008.

[3] Tandler, W. "Establishing Datum Reference Frames," *Inside Metrology*, Quality Digest, 12 March 2008.

[4] ANSI/ASME Y14.5.1M-2009 "Dimensioning and Tolerancing," The American Society of Mechanical Engineers, New York.

[5] ANSI/ASME Y14.5.1M-1994 "Dimensioning and Tolerancing," The American Society of Mechanical Engineers, New York.

[6] ISO 5459:2011. "Geometrical product specifications (GPS)—geometrical tolerancing—datums and datum systems." Geneva: International Organization for Standardization, 2011.

[7] Zhang, Xuzeng, and Roy, Utpal "Criteria for establishing datums in manufactured parts" Journal of Manufacturing Systems, 12(1), pp 36–50, 1993.

[8] Shakarji, C. M., and Srinivasan V., "Theory and Algorithms for L1 Fitting Used for Planar Datum Establishment in Support of Tolerancing Standards," DETC2013-12372, Proceedings, ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR, 2013.

[9] Hopp, T. H., 1990, "The Mathematics of Datums," ASPE Newsletter, September 1990, American Society for Precision Engineering, Raleigh, NC. A reprint is available at: http://www.mel.nist.gov/msidlibrary/doc/hopp90.pdf

[10] Shakarji, C. M., and Srinivasan V., "An improved L1 based algorithm for stnadardized planar datum establishment," DETC2014-35461, Proceedings, ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Buffalo, NY, 2014.

[11] Horowitz, A., "A version of Simpson's rule for multiple integrals," Journal of Computational and Applied Mathematics 134 (2001) 1–11.

[12] VanHuffel, S., and Vandervalle, J., 1991 The Total Least Squares Problem: Computational Aspects and Analysis, SIAM, Philadelphia, PA.

[13] Shakarji, C. M., and Srinivasan, V., "Theory and Algorithms for Weighted Total Least-Squares Fitting of Lines, Planes, and Parallel Planes to Support Tolerancing Standards," ASME Journal of Computing and Information Science in Engineering, 13(3), 2013.

[14] "Singular Value Decomposition" Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 10 Aug. 2015. https://en.wikipedia.org/wiki/Singular_value_decomposition

APPENDIX: PROOF OF THE EFFICIENT OBJECTIVE FUNCTION FORMULAS

Two theorems need to be proved in order to justify Eqs (4) and (5). They are closely related to the well-known principal axis theorem and parallel axis theorem. The 3D and 2D proofs are similar, and one can infer one straightforwardly from the other, so to minimize cumbersome notation, we show the 3D case.

Theorem 1. *Assume that we are given a set of data points* $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, *where* $\boldsymbol{x}_i = (x_i, y_i, z_i)$, *and the corresponding positive weights:* $w_1, w_2, \cdots w_N$, *where all the weights are positive and where the centroid (i.e. the weighted centroid,* $\frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i}$*) is expressed as* $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y}, \bar{z})$. *Then the sum of the squares of the distances from these points to a plane passing through the centroid is* $\sigma_1^2 a^2 + \sigma_2^2 b^2 + \sigma_3^2 c^2$, *where* σ_1, σ_2 *and* σ_3 *are the singular values of* $\boldsymbol{M}$ *(as defined below) and* (a, b, c) *is the unit normal to the plane expressed in terms of the eigenvectors of* $\boldsymbol{M}$.

Proof: For a plane passing through the centroid, having unit normal $\boldsymbol{n} = (n_1, n_3, n_3)$, define the sum-of-squares of the distances as

$$F(\boldsymbol{n}) = \sum_{i=1}^N w_i d_i^2 = \sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})]^2.$$

Let $G(\boldsymbol{n}) = 0$ be the constraint that $\boldsymbol{n}$ be a unit vector (where $G(\boldsymbol{n}) = |\boldsymbol{n}|^2 - 1$). Using the method of Lagrange multipliers, we know that the critical points of $F(\boldsymbol{n})$ subject to the constraint that $G(\boldsymbol{a}) = 0$ occurs when $\nabla F = \lambda \nabla G$. In this case we have,

$$\nabla F = \begin{bmatrix} \frac{\partial F}{\partial n_1} \\ \frac{\partial F}{\partial n_2} \\ \frac{\partial F}{\partial n_3} \end{bmatrix},$$

which, when expanded becomes:

$$2 \begin{bmatrix} \sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})](x_i - \bar{x}) \\ \sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})](y_i - \bar{y}) \\ \sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})](z_i - \bar{z}) \end{bmatrix},$$

which can be rewritten as

$$2 \begin{bmatrix} w_i (x_i - \bar{x})^2 & w_i (x_i - \bar{x})(y_i - \bar{y}) & w_i (x_i - \bar{x})(z_i - \bar{z}) \\ w_i (x_i - \bar{x})(y_i - \bar{y}) & w_i (y_i - \bar{y})^2 & w_i (y_i - \bar{y})(z_i - \bar{z}) \\ w_i (x_i - \bar{x})(z_i - \bar{z}) & w_i (y_i - \bar{y})(z_i - \bar{z}) & w_i (z_i - \bar{z})^2 \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix},$$

where each entry in the 3×3 matrix is understood as being summed from 1 to N. However, the 3×3 shown can be written as $\boldsymbol{M}^T \boldsymbol{M}$ where $\boldsymbol{M}$ is defined as the $N \times 3$ matrix

$$\boldsymbol{M} = \begin{bmatrix} \sqrt{w_1}(x_1 - \bar{x}) & \sqrt{w_1}(y_1 - \bar{y}) & \sqrt{w_1}(z_1 - \bar{z}) \\ \sqrt{w_2}(x_2 - \bar{x}) & \sqrt{w_2}(y_2 - \bar{y}) & \sqrt{w_2}(z_2 - \bar{z}) \\ \vdots & \vdots & \vdots \\ \sqrt{w_N}(x_N - \bar{x}) & \sqrt{w_N}(y_N - \bar{y}) & \sqrt{w_N}(z_N - \bar{z}) \end{bmatrix}.$$

We also have $\nabla G = 2\boldsymbol{n}$ making $\nabla F = \lambda \nabla G$ become a 3×3 eigen-problem given by

$$\boldsymbol{M}^T \boldsymbol{M} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix} = \lambda \begin{bmatrix} n_1 \\ n_2 \\ n_3 \end{bmatrix}.$$

These three equations can be written:

$$\sum_{i=1}^N w_i (x_i - \bar{x})[\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})] = \lambda n_1$$
$$\sum_{i=1}^N w_i (y_i - \bar{y})[\boldsymbol{n} \cdot (\boldsymbol{y}_i - \bar{\boldsymbol{y}})] = \lambda n_2$$
$$\sum_{i=1}^N w_i (z_i - \bar{z})[\boldsymbol{n} \cdot (\boldsymbol{z}_i - \bar{\boldsymbol{z}})] = \lambda n_3$$

Multiplying these equations by n_1, n_2, and n_3 respectively, then summing the equations gives

$$\sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_{\boldsymbol{A}})]^2 = \lambda |\boldsymbol{n}|^2 = \lambda \qquad (6)$$

But the sum on the left is just the objective function, $F(\boldsymbol{a})$, hence the sum of squares of the distances to a plane passing through the centroid (when the plane's normal is an eigenvector of $\boldsymbol{M}^T \boldsymbol{M}$) is equal to the eigenvalue (λ) corresponding to that eigenvector. We note $\boldsymbol{M}^T \boldsymbol{M}$ is a real, symmetric matrix and thus its eigenvectors are orthogonal (and can be assumed to orthonormal by simple scaling).

Now consider the more general case of any plane P passing through the centroid (not necessarily having its normal as one of the eigenvectors) whose unit normal is $\boldsymbol{a} = (a, b, c)$ when expressed in terms of the orthonormal eigenvectors of $\boldsymbol{M}^T \boldsymbol{M}$.

(The eigenvectors are being used as a basis to express the normal to the plane.) The Pythagorean Theorem can be used to show the square of the orthogonal distance from each point to the plane is equal to the sum of the squares of the distances from the point to the three orthogonal planes formed by the eigenvectors. (Figure 10 shows a 2D depiction).

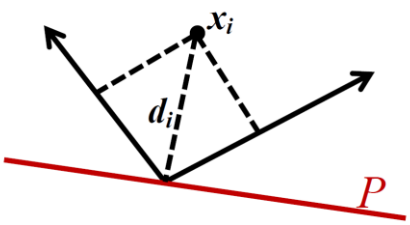


Fig. 10. The distance from a point to a plane is decomposed into separate distances to the orthogonal planes formed as normal to the eigenvectors.

Therefore, the sum of the squares of the distances from the points to P can be grouped by distances to each orthogonal plane and then the sum of the squares for each group can be replaced by the eigenvalue associated with its plane as we showed. Thus if the eigenvalues are labeled λ_1, λ_2, and λ_3, then the sum of the squares of the orthogonal distances to the plane P is simply

$$a^2\lambda_1 + b^2\lambda_2 + c^2\lambda_{3.}$$

Because the singular vectors from the singular value decomposition of $\boldsymbol{M}$ are the same as the eigenvectors of $\boldsymbol{M}^T\boldsymbol{M}$ [14], and since the singular values of $\boldsymbol{M}$ are the square root of the singular values of $\boldsymbol{M}^T\boldsymbol{M}$ [14], we have that the sum of the squares of the distances can be restated as

$$a^2\sigma_1^2 + b^2\sigma_2^2 + c^2\sigma_3^2. \qquad \blacksquare$$

Theorem 1 is related to the principal axis theorem. The following theorem, related to the parallel axis theorem, states that when a plane is translated away from passing through the centroid, the increase to the sum-of-squares of the distances increases by an easily computed amount, namely the square of the distance moved times the sum of the weights. In our application, the sum of the weights is the total area (in 3D) or the total length (in 2D).

Theorem 2. *Assume that we are given a set of data points* $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, *where* $\boldsymbol{x}_i = (x_i, y_i, z_i)$, *and the corresponding positive* weights*:* $w_1, w_2, \cdots w_N$, *where all the weights are positive and where the centroid (i.e. the weighted centroid,* $\frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i}$*) is expressed as* $\bar{\boldsymbol{x}} = (\bar{x}, \bar{y}, \bar{z})$. *Assume also that* α *represents the sum of squares of the distances from that plane to a plane P passing through the centroid. If P* is parallel to P but separated from by a distance* d_c, *then the sum of squares of the distances from the points to P* is* $\alpha + d_c^2 \sum_{i=1}^N w_i$.

Proof: Given that $\alpha = \sum_{i=1}^N w_i d_i^2$, we seek to find $\sum_{i=1}^N w_i (d_i + d_c)^2$. Expanding the square yields

$$\sum_{i=1}^N w_i d_i^2 + d_c^2 \sum_{i=1}^N w_i + 2d_c \sum_{i=1}^N w_i d_i.$$

But the first term is just α and the last term is zero. The last term must be zero, since

$$\sum_{i=1}^N w_i d_i = \sum_{i=1}^N w_i [\boldsymbol{n} \cdot (\boldsymbol{x}_i - \bar{\boldsymbol{x}})] =$$

$$\boldsymbol{n} \cdot \left[\sum_{i=1}^N w_i \boldsymbol{x}_i - \bar{\boldsymbol{x}} \sum_{i=1}^N w_i\right],$$

And substituting $\frac{\sum_{i=1}^N w_i x_i}{\sum_{i=1}^N w_i}$ for $\bar{\boldsymbol{x}}$ causes the bracketed term to vanish. Thus the shift by d_c caused the sum of squares to become

$$\alpha + d_c^2 \sum_{i=1}^N w_i \qquad \blacksquare$$

In our application of this theorem, the weights are the areas of the triangles (or the the lengths of the line segments in 2D) so the sum of the weights is the total area A (or total length L in 2D). The shift of a plane by an amount d_c as shown in fig. 7 results in an addition to the sum-of-squares of Ad_c^2 in 3D or Ld_c^2 in 2D.

APPENDIX: 2D CODE IN MATLAB

The algorithm documented in this paper showed how the objective function could be efficiently expressed and used in a minimization algorithm. However, the (2D) code below employs an even faster method. Specifically, Eq. (4) is used to find the line coincident with a line segment of the convex surface that minimizes the objective function. Then both endpoints of that line segment are evaluated to see if balancing a rocker condition on either vertex improves the objective function. These tests on the two endpoints are achieved using two other calculations of the singular value decomposition. While the details of this are not gone into in this paper, it has been tested in over 250,000 test cases with simulated data sets to ensure the exact equivalence. Thus the faster algorithm is given here.

Though 3D code is not included here, Eq. (5) can be used to find the triangle that minimizes the objective functions. Then similar tests could be used to check the vertices and edges of that triangle to see if balancing a rocker condition (on an edge or point) improves the sum-of-squares.

```
function [point, direction] = L2C2Dline(originalpts,
refdir)
% L2C2Dline returns the line that minimizes the sum-
% of-squares of distances between the line and the
% lower convex envelope of a set of points ("lower"
% as determined by refdir) and such that the line is
% constrained to lie on the lower side of the convex
% envelope. The function can be used, for example, as
% [point, direction] = L2C2Dline(originalpts, refdir)
```

```
%
%  The function returns [point, direction] where
% "point" is a point on the line and "direction" is a
% unit vector giving the direction of the line.
% "orignalpts" is an N X 2 matrix of points: [x1
% y1;x2 y2;...;xN yN] and "refdir" is a direction [x
% y] (not = {0, 0}) that points into the material.
% "refdir" allows the algorithm to know on which side
% of the points the line must lie. Generally, refdir
% does not need to be known very accurately. The
% number of points, N, must be at least two.
%
% Check for the two point case:
%
if (size(originalpts,1) == 2)
    point = sum(originalpts)/size(originalpts,1);
    direction = originalpts(2,:)-originalpts(1,:);
    direction = direction/norm(direction);
    if ([refdir(2) -refdir(1)]*[direction]' < 0)
        direction = -direction;
    end
    else

%
% Translate and rotate the original data set, so that
% the points are close to the origin and so that
% refdir points in the direction of the +y-axis.
%
translation = sum(originalpts)/size(originalpts,1);
pts = bsxfun(@minus,originalpts,translation);
dir = refdir/norm(refdir);
pts = pts*[dir(2) dir(1);-dir(1) dir(2)];
%
% Now that the point lie somewhat along the x-axis,
% sort them according to increasing x-values
%
[~,indices]=sort(pts(:,1));
pts = pts(indices,:);

indices = convhull(pts(:,1),pts(:,2));
pts = pts(indices,:);

midpts = (pts(2:end,:) + pts(1:end-1,:))/2;
vectors = pts(2:end,:) - pts(1:end-1,:);
pts = pts(1:end-1,:);
normals = [-vectors(:,2) vectors(:,1)];
indices = normals(:,2) > 0;
pts = pts(indices,:);
midpts = midpts(indices,:);
vectors = vectors(indices,:);
normals = normals(indices,:);
pts = [pts;pts(end,:)+vectors(end,:)];
%
% Now "pts" contains only the vertices of the lower
% convex envelope. We now compute a single Singular
% Value Decomposition that can be used to obtain the
% objective function values for all the lines
% containing edges of the lower convex envelope.
%
normals = bsxfun(@rdivide,normals,rssq(normals,2));
lengths = rssq(vectors,2);
L = sum(lengths);
centroid = lengths'*midpts/L;

weights = ([lengths;0] + [0;lengths])/6;
shiftedpts = bsxfun(@minus,pts,centroid);
weightedpts =
bsxfun(@times,shiftedpts,sqrt(weights));
shiftedmidpts = bsxfun(@minus,midpts,centroid);
weightedmidpts =
bsxfun(@times,shiftedmidpts,sqrt((2/3)*lengths));
allweightedpts = [weightedpts; weightedmidpts];

[~,S,V] = svd(allweightedpts,0);
ssq1 = S(2,2)^2;
ssq2 = S(1,1)^2;
direction = V(:,1)';
if direction(1) < 0
    direction = -direction;
end

angleSVD = atan2(direction(2),direction(1));
angles = atan2(vectors(:,2),vectors(:,1));
anglediffs = angles - angleSVD;
objfunangle = ssq1*(cos(anglediffs)).^2  +
ssq2*(sin(anglediffs)).^2;
ds = dot(normals',-shiftedmidpts')';
objfunedges = objfunangle + L*ds.^2;

[bestobjvalue,minedgeindex] = min(objfunedges);
bestdirection =
[cos(angles(minedgeindex)),sin(angles(minedgeindex))]
;
bestpoint = midpts(minedgeindex,:);
%
% Now that the best edge has been found, we look at
% the endpoints of that edge to see if a line
% (external to the material) passing through either
% endoint gives a better objective function. This
% will involve one Singular Value Decomposition for
% each of the two endpoints.
%
for ii = minedgeindex:minedgeindex+1
    pt = pts(ii,:);
    shiftedpts = bsxfun(@minus,pts,pt);
    weightedpts =
(bsxfun(@times,shiftedpts,sqrt(weights)));
    shiftedpts = bsxfun(@minus,midpts,pt);
    weightedmidpts =
bsxfun(@times,shiftedpts,sqrt((2/3)*lengths));
    allweightedpts = [weightedpts; weightedmidpts];

    [~,S,V] = svd(allweightedpts,0);
    direction = V(:,1)';
    if direction(1) < 0
        direction = -direction;
    end

    normal = [-direction(2) direction(1)];
    shiftedpts=(bsxfun(@minus,pts,pt));
    dists = (normal*shiftedpts')';
    [~,minindex]=min(dists);

    if (minindex == ii && S(2,2)^2 < bestobjvalue)
        bestobjvalue = S(2,2)^2;
        bestdirection = direction;
        bestpoint = pt;
    end
end
point = translation + bestpoint*[dir(2) -
dir(1);dir(1) dir(2)];
direction = bestdirection*[dir(2) -dir(1);dir(1)
dir(2)];
end
end
```