CJFIT: Cylindrical Jig Fit Test

Emanuele Messina

2024

Abstract

This article presents an approximate solution to the following problem: given a cylinder and a jig, determine whether the jig can fit inside the cylinder. Alternatively, the problem can be restated as: given a cylinder and a 3D object, determine if a rotation and translation exist such that the object is entirely enclosed by the cylinder. This solution also provides one such possible orientation and translation as a byproduct. The solver is implemented as a Blender add-on, available on GitHub. The first section discusses the thought process and initial attempts leading to the final solution, described in detail in the second section.

Contents

1 Initial attempts

Problem (*CJFIT*). *Given a cylinder C with height h and diameter d, and a set $J = \{v_1, v_2, \ldots, v_n\}$ of vertices in 3D space, determine whether a rotation* $\mathbf{R}$ *and a translation* $\mathbf{T}$ *exist s.t.* $\mathbf{RT}J \subset C$.

The initial attempts consist in a sequence of trials, of increasing complexity, to be performed on the jig J to determine a definitive positive or negative outcome to *CJFIT*.

1.1 Largest distance

It is evident that the maximum allowed dimension for an object to fit inside C is its diagonal $l = \sqrt{d^2 + h^2}$.
Thus, if the maximum distance between two points of J is bigger than l, the jig cannot fit inside C:

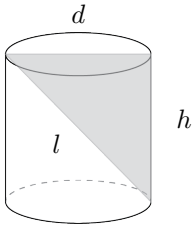


Figure 1: The cylinder C

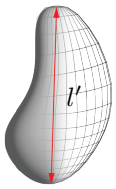


Figure 2: An arbitrary jig J

Trial 1. $l' = \max_{v \in J} |v_i - v_j|\,, \quad l' > l \implies$ *CJFIT negative*

1.2 Minimum Bounding Box extent

Obviously there are cases where $l' < l$ but J still doesn't fit inside C.
Consider, for example (see Fig. 3), J to be a square prism with height h' slightly less than h, WLOG take $h' = \frac{2}{3}h$, and sidelength $s_1 < h$ such that the base diagonal $d' = s_1\sqrt{2} > d$.
It follows that $l' = \sqrt{2s_1^2 + h'^2}$, the main diagonal of the prism. Now consider the prism height to be aligned with the cylinder axis, and the prism to be centered inside the cylinder: this is the best, albeit trivial, orientation and translation configuration that minimizes the volume of the prism outside the cylinder.
Immediately: $l' < \sqrt{d^2 + \frac{4}{9}h^2} < l$, so trial 1 passes. However, the square base intersects the cylinder base, so the prism cannot fit.

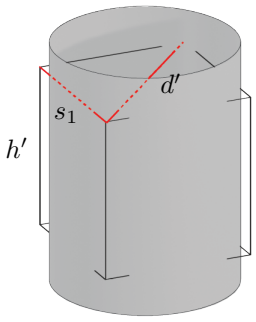


Figure 3: The example above

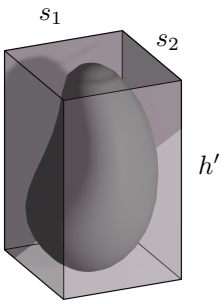


Figure 4: The minimum bounding box B of J

Due to these considerations, we take the Minimum Bounding Box B of J as an estimator of its extent in space. Given h', s_1, s_2 the height and sidelengths of B:

Trial 2. $h' < h \quad \wedge \quad d' = \sqrt{s_1^2 + s_2^2} < d \implies$ *CJFIT positive*

1.3 Carver's test

There are cases in which the bounding box is sufficiently thin but $h' > h$, while still having $l' < l$. In this situation, trial 2 fails at the first condition, but the

jig might still fit if tilted by an appropriate amount.

A necessary condition for B to fit inside C is that one of the lateral faces of B (a rectangle) must fit inside the lateral projection of C (see Fig. 5).
This check is performed between the rectangles $s_2 \leq s_1 \times h'$ and $d \times h$. Since wider rectangles (with the same height) are less likely to fit, and more likely to collide with the cylinder top face when titled, we select the lateral side of B with the shortest base sidelength.
Carver's theorem [1] provides necessary and sufficient conditions on the sidelengths of two rectangles for one to fit into the other.

Theorem (Carver [1]). *Let T be an $a \times b$ rectangle with $a \geq b$, and R be a $p \times q$ rectangle with $p \geq q$. R fits into $T \iff$ either:*

$$p \leq a \ \wedge \ q \leq b \quad , \quad \textit{or} \tag{a}$$

$$p \geq a \ \wedge \ q \leq b \ \wedge \ \left(\frac{a+b}{p+q}\right)^2 + \left(\frac{a-b}{p-q}\right)^2 \geq 2 \tag{b}$$

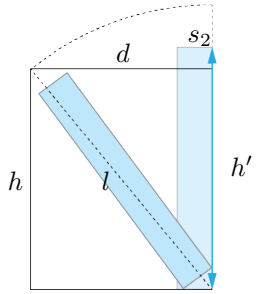


Figure 5: Application of Carver's Theorem

If Carver's theorem is satisfied, we confirm that there exists an orientation for which no collisions occur in this projection.

Next, we must find an actual orientation to make the chosen projection of B fit into the one of C, to then look at the top projection for a collision. If there is no lateral collision, the trial result is positive, and the jig fits.
To do this, we place the view on the lateral projection of the C, we align B with C's axis, align B's chosen lateral face to the view, and lastly we place B at the center of C.
Now we choose a preferred quadrant from this local view (in this example, the first). Then we take the vertex in this quadrant and we calculate the angle to rotate B such that this vertex will land on the top face of C. This guarantees antisymmetry with respect to the local x axis. This is the final transformation of B. Having positioned B at the center of C, rather than elsewhere on the horizontal axis, ensures that B is placed in the region with the most free space inside the cylinder, minimizing the chances of lateral collisions (see Fig. 6).
The final step is to check the top projection for collisions (see Fig. 7).

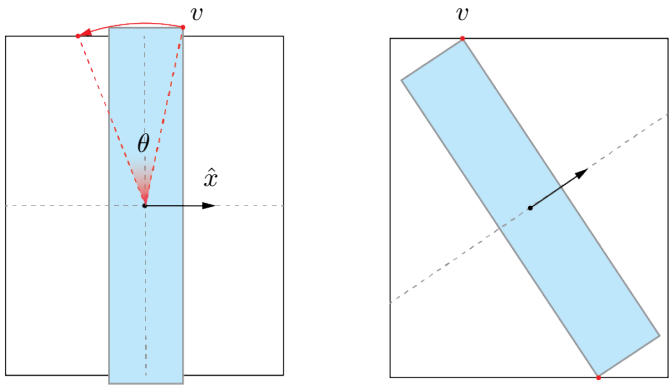


Figure 6: Finding **R** and **T** for trial 3

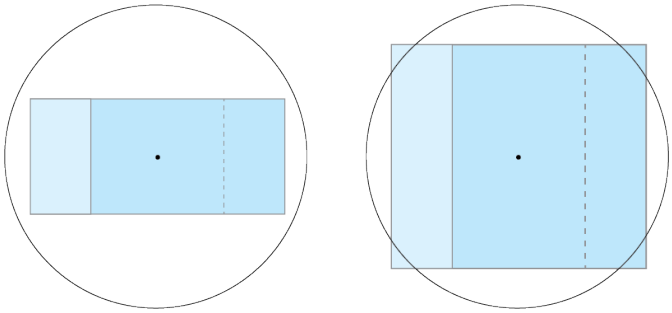


Figure 7: Checking the top projection for collisions (the two possible cases)

Trial 3. *Apply Carver's Theorem to the lateral face $s_2 < s_1 \times h'$ of B and the lateral projection $d \times h$ of C. If the theorem is verified, center B in C and rotate it to fit the lateral projection of B into the lateral projection of C (as described in section 1.3). Check the top projection: if there are no collisions $\Longrightarrow$ CJFIT positive.*

2 Convex Hull Intersections

2.1 Introduction

Unfortunately the MBB method is overly simplistic and can result in false negatives for most shapes. While using auxiliary structures such as an elliptical prism (constructed with the base sides of B as base diagonals) could reduce the false negatives, it makes more sense to develop a universal solution that works for any shape. Nevertheless, the bounding box method provided valuable insight into the workings of the final solution.

In fact, the final solution is really just the 3D heuristic version of the *Rectangle in Rectangle (Carver)* problem. It relies on a deterministic clever alignment of the jig relative to the cylinder, which serves as the starting point for a heuristic algorithm. This approach tries to eliminate any remaining intersection by fine-

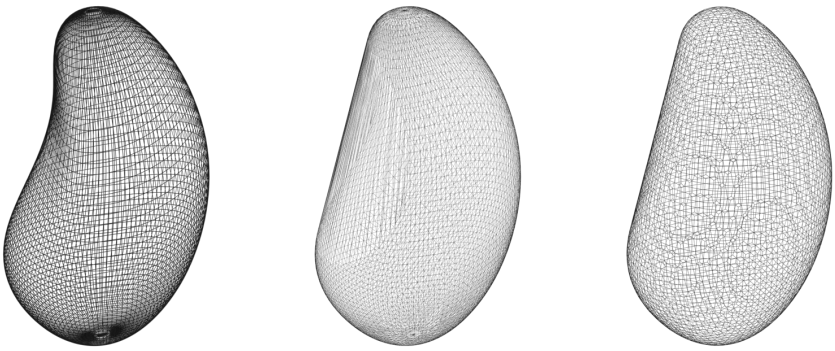
tuning the position and rotation of the jig inside the cylinder.

We begin by creating the convex hull of the object: J fits in C if and only if its convex hull fits. The convex hull will help in defining the spatial extent of J.

2.2 $\mathbf{R_1}, \mathbf{T_1}$

PCA We first need to find a local orthogonal frame of reference for J where the basis vectors are collinear with the directions that best fit the point cloud: this is exactly what PCA does.

In practice, depending on how the convex hull was generated (in this case via a Blender binding) it may need remeshing (also applied via Blender) to evenly distribute the vertices on the surface for PCA to be effective (see Fig. 8). Furthermore, PCA is applied repeatedly to make sure it converges.



(a) (b) (c)

Figure 8: (a) J (b) Convex Hull generated by Blender: uneven point distribution (c) Remesh modifier: even point distribution

OABB Once we obtain the three principal components of the convex hull, we construct an Object Aligned Bounding Box (OABB) with respect to the principal components frame of reference. It's not the MBB but it's a good approximation and it's fast.
To do this, we project J in the plane normal to the largest principal component (giving us a *top view* of the object), onto which we see the 2D basis formed by the other two components. We select the second largest component's direction and we put two tangent planes to J to bound it along this direction.

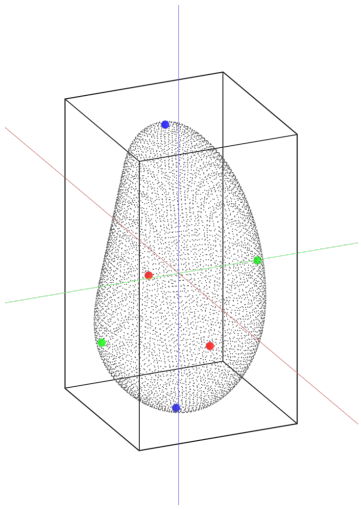
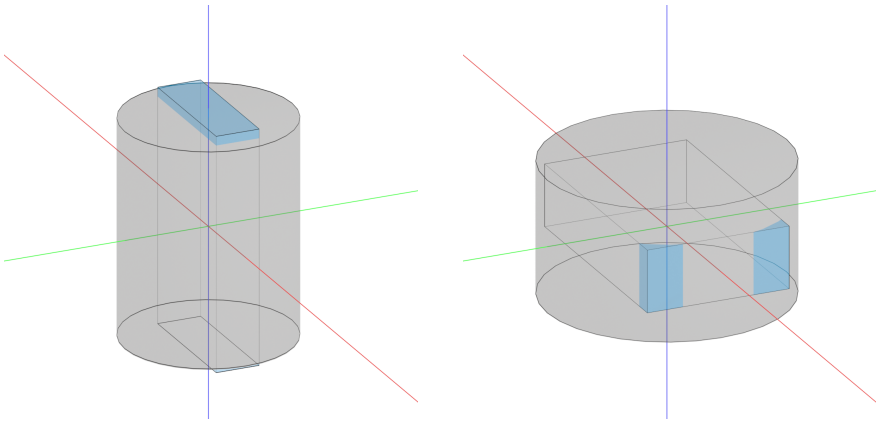


Figure 9: Point cloud, principal axes found by PCA, and OABB. Tangency points highlighted

To find the tangency points, we locate the most distant point of the projection from the geometric center along the positive and negative directions given by the principal component. The convexity of the hull ensures that these are indeed tangency points.
We repeat this process with another pair of planes along the third principal direction. Then we project J onto the plane normal to one of the last two components and find the bounding planes along the first principal direction in the same way.
The intersections between all these planes are the vertices of the OABB (Fig 9).

For simplicity, in this implementation C is always aligned and centered to the global frame of reference. Therefore, we rotate J using the transpose (inverse) eigenvector matrix as the rotation matrix ($\mathbf{R_1}$), to let the PCA basis correspond to the global basis. In this case we make the three principal components correspond, in order, to: z, x, y (Fig 10a). If the cylinder is wider than it is tall ($d > h$) the the order is x, y, z (Fig 10b).
Consequently, the OABB is just the Axis Aligned Bounding Box with respect to the global basis, which Blender can already calculate with a built in function. In practice, the specific PCA algorithm may be inconsistent in assigning the horizontal components x and y. Thus, we also manually correct the assignment by checking the sidelengths of the bounding box instead of the eigenvalues.

At this point, the translation $\mathbf{T_1}$ is just the translation from the center of the OABB to the center of C. This method, in a similar fashion to what described in section 1.3, increases the symmetry of the problem by already aligning J in a way that naively best fits C.



(a) (b)

Figure 10: (a) z, x, y alignment (b) x, y, z alignment

2.3 $\mathbf{R_2}, \mathbf{T_2}$

Now we enter the stage of trial and error, due to the arbitrary shape of J. We distinguish between two kinds of intersections: axis-aligned and radial.

Axis-Aligned Intersections This is the simplest case and is analogous to the rectangle-in-rectangle problem. We consider the intersections of J with the lateral projection of C on the xz and yz planes.

We start by eliminating the vertical intersections. We rotate J along the largest horizontal component (normal to the shortest side of the BB, providing the most space available for rotation in this projection) until the vertical intersections disappear, for a maximum of 90° (after which the problem would repeat symmetrically). If vertical intersections cannot be removed, then the test fails.

If horizontal intersections appear after removing the vertical ones, we continue the rotation for another cycle. If intersections still exist at the end of this second cycle (whether remaining horizontal or newly created vertical ones, or both), the test fails. Fig 11 proposes a method to find the rotation angles.

If there are no more intersections in this projection, we proceed to check for Radial Intersections by examining the top view (the projection of C on the xy plane).

Actually, instead of making the test fail immediately if intersections cannot be removed by rotations along the largest horizontal component, we give the smallest horizontal component a chance and we restart the previous steps by rotating along it. If neither component can remove the axis aligned intersections, the test fails.

The `rotate-adjust` block Due to the bounding box centering, the intersections should typically appera and disappear in pairs across opposite quadrants.

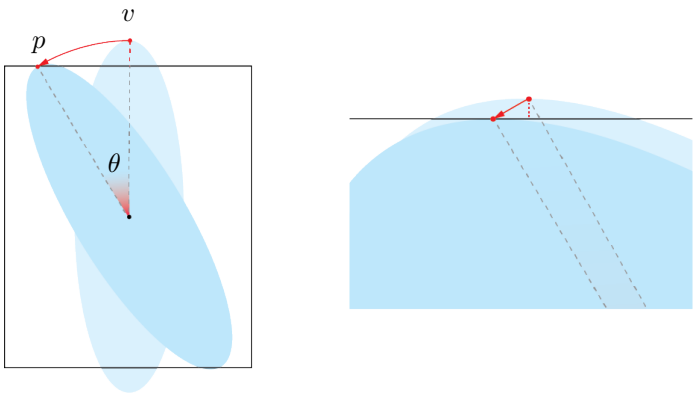


Figure 11: Axis-Aligned intersection removal: take the most distant outlier vertex v_+ from the rectangle upper bound and the most distant outlier v_- from the rectagle lower bound; select the nearest of the two (e.g. $v = v_+$); trace a circumference passing from it centered in the bounding box center; find the intersection p between the circumference and the bound crossed by v (in this case the upper bound); calculate the angle θ between v and p; rotate J by θ; perform an axial Center Adjust; repeat the process until the intersections along z are removed. The process for the intersections along an horizontal axis is analogous.

However, if during a rotation cycle at most one pair of adjacent quadrants in the current projection plane is affected, we perform a Center Adjust before further rotating J. This ensures that the available space inside C is maximally utilized before attempting to remove intersections through additional rotations.

We thus define a `rotate-adjust` block that will act on J as a roto-translation aiming to make it non-intersecting with respect to a given 2D shape, lying in a particular 3D plane, used as the intersection boundary.

We perform the trials by constructing a tree of `rotate-adjust` blocks, each with different rotation axes and shapes. If one branch successfully eliminates all the intersections, *CJFIT* is positive and the accumulated rotations and translations along that branch constitute, respectively, R_2 and T_2. The final matrices are the combination of the heuristic solution with starting point: $\mathbf{R} = \mathbf{R_2 R_1}$ and $\mathbf{T} = \mathbf{T_2 T_1}$.

Radial Intersections Having removed the Axis-Aligned Intersections (necessary condition to proceed), the radial intersections are not visible from the lateral projections. We thus look at the top projection of C on the xy plane, where this time we see an intersecting circle instead of a rectangle.

We open three branches with `rotate-adjust` blocks on the three components as rotation axes. Note that, having (possibly) tilted J along an horizontal component, the rotations along the horizontal components still correspond to rotations along the global x and y axes, while the vertical component is now just the *local* z axis of the bounding box. For this reason, we allow 180° of maximum rotation along the local z axis. If there were no axis-aligned intersections in the first

place, we skip the local z rotation, as it would be unnecessary given that the intersecting shape is a circle. Fig 12 proposes a method on how to find the rotation angle for local z rotations. The method for finding the angle for rotations along the horizontal components is the same.

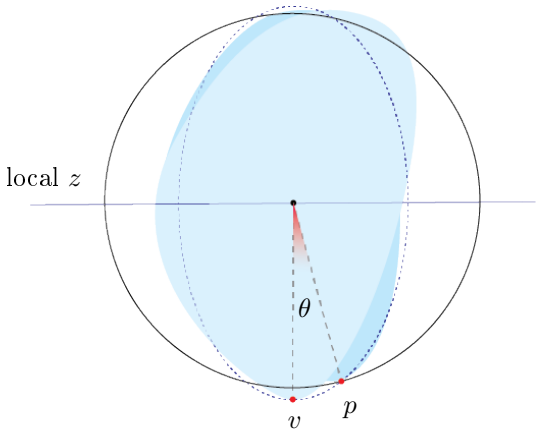


Figure 12: Radial Intersection removal via local z rotation: analogously to what described in Fig 11, between the farthest outlier along the positive y direction and the farthest in the negative direction, select the closest, v; trace a circumference centered in the bounding box centered, passing through v, normal to the local z axis (highlighted in blue); find the intersection p between the circumference and C (it will land on the side of the cylinder, otherwise the test fails because no rotation can remove the intersection); calculate the angle θ between v and p; rotate J by *theta* along the local z axis; perform a radial Center Adjust; repeat the process until the radial intersections are removed.

Center Adjust This sub-block pushes J from its regions outside C towards the inside, either until the intersections with the given shape disappear or if equilibrium is reached.
For horizontal projections, the intersecting shape is a rectangle in either the xz or the yz plane, so we compare the horizontal coordinates of the vertices $v \in J$ with the rectangle bounds. An intersection occurs if $v_x > \frac{d}{2} \vee v_x < -\frac{d}{2}$ and $v_y > \frac{h}{2} \vee v_y < -\frac{h}{2}$.
For the top projection, the intersecting shape is a circle, so we compare the norm of the vertices with the radius. An intersection occurs if $v_x^2 + v_y^2 > r^2$.

An imbalance in the intesections distribution is present when at most two adjacent quadrants in the current projection plane contain an intersection.

For axis-aligned intersections in the lateral projections, we separate the problem into the vertical and horizontal axes. This way, an axis-specific imbalance is present if either the pair of quadrants in the positive axis direction or the pair in the negative direction contain an intersection, but not both pairs at once. In such cases, we simply translate up/down and left/right (with respect to the current horizontal axis) until both pairs of quadrants contain an intersection or none. Fig 13 presents a method to find a suitable stride length.

For radial intersections in the top projection, we pick the most distant outlier (the one with the highest norm among all the outliers) and translate J along $-\widehat{v_h}$ where $\widehat{v_h} = \frac{1}{\sqrt{v_x^2+v_y^2}} \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ is the normalized top projection of the selected outlier. This translation attracts J towards the center of C. We keep attracting until the intersections are cleared or there is balance. Fig 14 presents a method on how to find a suitable stride length.
Some radial imbalances may be impossible to clear, and the procedure could potentially spiral into a deadlock, so in practice we limit the number of iterations to 4.

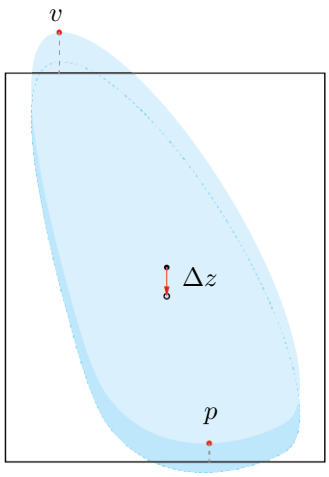


Figure 13: Axial Center Adjust: find the outlier vertex v in the quadrant pair that contains the imbalance as the farthest from the rectangle bound lying in those quadrants; calculate the distance Δv from v to the bound (e.g. for a vertical imbalance in the positive z direction: $\Delta v = \frac{h}{2} - v_z$); find the nearest inner vertex p to the rectangle bound lying in the opposite quadrant pair with respect to the one that contains the imbalance (and thus contains v); calculate the distance Δp between p and the bound (in this example it's the lower bound, so $\Delta p = \frac{h}{2} - |p_z|$); translate J along the imbalanced axis (in this example z) towards the attracting bound (in this example the lower bound) by $\Delta z = \Delta p + \frac{\Delta v}{2}$; the imbalance is removed since both the upper and lower bound are now intersecting with J, the `rotate-adjust` block can proceed with further rotating and then checking for new imbalances.

Considerations While this solution tries to cover more shape cases than the initial attempts, it is still a first order refinement of the former, hence still an approximate solution.

The tree structure of the heuristic part allows for parallel computation.

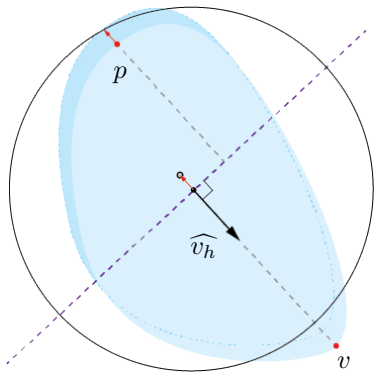
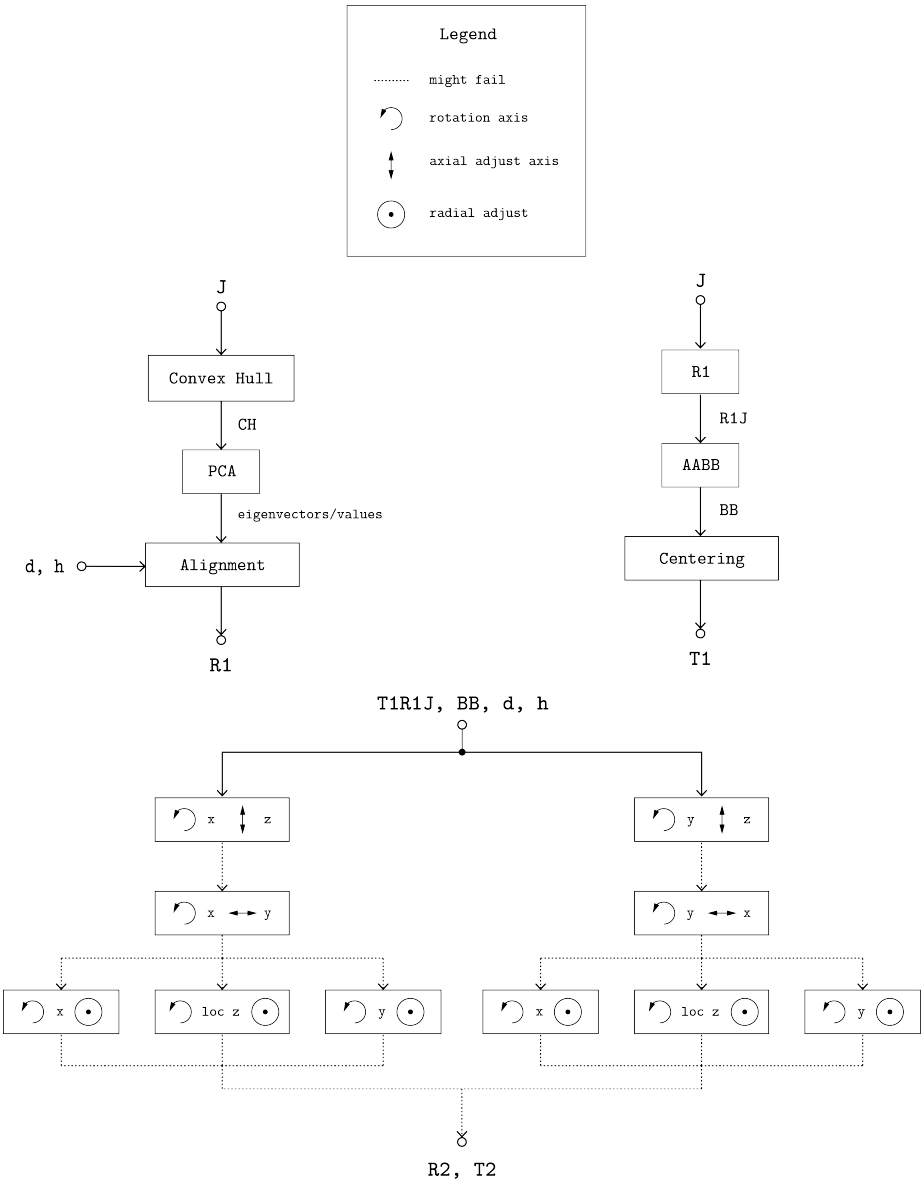


Figure 14: Radial Center Adjust: obtained the versor $\widehat{v_h}$, find the normal versor on the xy plane; the line described by the normal vector divides the plane in to regions, one containing the outlier v; find the farthest inner vertex p from to the line, among the one lying in the opposite region with respect to v; calculate the distance Δr from p to the circumference along the direction $\widehat{v_h}$; translate J by $-\Delta r\,\widehat{v_h}$; repeat the process for a maximum of 4 iterations; the `rotate-adjust` block can now proceed to perform a rotation and recheck for intersections and imbalances.

2.4 Algorithm Recap



References

[1] John Wetzel. "Rectangles in Rectangles". In: *Mathematics Magazine* 73 (June 2000), p. 204.