ABSTRACT
We present three different methods for fitting CAD models to point clouds. Constructive Solid Geometry (CSG) is used for representing the models because of its flexibility and compactness. A given CSG tree is converted to a Boundary Representation (B-rep) or a triangular mesh or a point cloud for approximating the orthogonal distance of a given point from the model surface. We name the resulting methods as ICS (Iterative Closest Surface-point), ICT (Iterative Closest Triangle-point) and ICP (Iterative Closest Point-cloud-point) respectively. Levenberg-Marquardt algorithm is used for the minimization of the sum of orthogonal distances. We introduce the notion of Internal Constraints representing the geometric relationships among constituent components of a CSG tree. We show that their inclusion reduces the number of free parameters and results in a more faithful and robust estimation as complete a priori geometric information is utilized. The results of applying these methods to different objects are presented and compared, which bring out the necessity of using a hybrid methodology for achieving accuracy at a practical speed.

KEY WORDS
Surface fitting, Reverse Engineering, 3D Modelling, Point cloud processing, Object Recognition

1. Introduction
The problem of fitting CAD models to point clouds arises in many applications like model-based object recognition, surface reconstruction, reverse engineering and quality control. Recent advances in laser scanning technologies have also added to their importance, as acquisition of dense point clouds has become both faster and cost-effective.

Many of the fitting methods reported in the literature [1,2,3,4] have focused on recovering patches of simple geometric surfaces, which are then connected together resulting in a B-rep model. This approach assumes that all patches necessary for the model were completely visible in the captured data. Besides, after fitting some automatic or manual editing is necessary to fill in the gaps at edges and joints.

Many other approaches fit simple unbounded geometric objects like planes and cylinders to a segmented point cloud using one of the standard segmentation techniques [5,6]. This fitting is based on minimizing the sum of the orthogonal distances of the given points to the model surface. This is in contrast to fitting implicit models given by a zero set, where the comparison of the speed and the accuracy of using the algebraic vs. the orthogonal distance is a recurring theme [4,7,8,9]. A survey of recovering quadric surface patches in meshes is given in [10]. Another option is to use B-splines as surface representation. A method for fitting NURBS based models is discussed in [11]. Once again the recovery of surface patches is followed by a heuristics based editing resulting in a B-rep model.

There has been a lot of work on using the zero set of the signed distance field for the recovery of a smooth manifold from a given point cloud [12,13,14]. Such a manifold is essentially a freeform surface, and thus cannot be represented by a CAD model.

In many problems of surface reconstruction, especially those involving industrial and man-made environments, a library of CAD models usually exists which was used during the design stage. These models contain information about the geometric relationships of constituent surfaces as well as the valid bounds of their parameters. By using these models directly for fitting we can get rid of the model editing stage [3], which is necessary for B-rep based approaches mentioned above. In this paper, we present such a procedure for fitting CAD models represented by Constructive Solid Geometry (CSG) to a given point cloud. CSG is one of the most commonly used CAD design and representation schemes. It is especially suited for fitting due to its flexibility and compactness. Furthermore, it results in a small number of unknown object parameters during least-squares estimation. For example, to represent the shape of a box, we need to specify three sizes in CSG, while in case of B-rep explicit sizes and geometric relationships of all faces and edges are necessary, making its use for fitting very cumbersome. The objects in CSG are represented as Boolean operations between some simple geometric primitives, like cylinders, cones, boxes, spheres etc. This enables us to represent a large number of objects using a few simple primitives.

The use of CAD models for fitting to images was pioneered by Lowe [15]. He estimated the pose and shape
parameters by minimizing the distance of the visible edges from the hidden-line projection of the estimated model. An extension of this approach for fitting CSG objects to image gradients and point measurements for industrial reconstruction has been reported in [16], where geometric constraints are also used. In this paper we extend this fitting paradigm from images to point clouds. The key problem in fitting a given point cloud to a selected CAD model is finding the orthogonal distance of a point from the model surface. Moreover, the partial derivatives of the distance with respect to shape and pose parameters are needed for non-linear least squares. We present three different methods to solve this problem. We name these methods ICS (Iterative Closest Surface-point), ICT (Iterative Closest Triangle-point), and ICP (Iterative Closest Point-cloud-point). These methods use a B-Rep, a triangle mesh, and a point cloud respectively to calculate the orthogonal distance of a given point from the CSG model surface. These methods provide different trade-offs between speed and accuracy and are compared in the final results section.

We borrow the notion of Internal Constraints from [16] and use them for encoding geometric relationships between constituent primitives of a CSG tree. These internal constraints also contain valid ranges of parameter values, also called Bound Constraints. The inclusion of these constraints removes extra degrees of freedom and makes the fitting procedure more robust. The rest of the paper is organized as follows. In Section 2 we define the problem, and present the three different distance calculation methods. Section 3 discusses Internal Constraints and their enforcement. Details about fitting method and its implementation are given in Section 4. The results of applying the presented algorithms to some test objects are presented in Section 5. Finally we conclude in Section 6 and give some directions for future work.

2. Distance Approximation Algorithms

The problem we will address can be defined as follows. We are given a CAD model represented as a CSG tree having some unknown shape and pose parameters and a set of 3D points, which are known to belong to it. We also assume that the CSG tree contains information about geometric as well as bound constraints for parameters of the constituent primitives (collectively called Internal Constraints). We want to estimate those values of the parameters, which minimize the sum the squares of the orthogonal distance of the points from the estimated CSG object.

\[
\min \sum_{i=1}^{N} \Omega^2 [p_i, \Gamma, \tau_1, \tau_2, \ldots, \tau_M]
\]  

(1)

Where \( \Omega \) defines the shortest distance of a given point \( p \) to the surface of the CSG model \( \Gamma \) which has \( M \) shape and pose parameters given by \( \tau_1, \tau_2, \ldots, \tau_M \). The point cloud consists of \( N \) points, \( p_1, p_2, \ldots, p_N \).

To solve this optimization problem we need a method to find value of distance function \( \Omega \) in (1) and its partial derivatives with respect to the CSG parameters i.e.

\[
\frac{\partial \Omega}{\partial \tau_1}, \frac{\partial \Omega}{\partial \tau_2}, \ldots, \frac{\partial \Omega}{\partial \tau_M}
\]

For unbounded quadric objects such as cylinders, spheres etc. it is possible to get closed-form expressions for the orthogonal distance as well as its partial derivatives [17]. As CSG uses bounded objects and also employs Boolean operations among them, it is not possible to derive an analytical expression for distance of a point from an arbitrary CSG object. Consequently, we resort to numerical approximation, and try the following three methods for the calculation of the distance of a given point from the surface of a CSG model:

1. Get the B-rep of the CSG model and calculate the distance of a given point to the closest surface in the B-rep. We call this method Iterative Closest Surface-point (ICS).
2. Approximate the CSG model with a triangulated mesh and calculate the distance of a given point from the closest triangle. We name this method Iterative Closest Triangle-point (ICT).
3. Convert the CSG model to a point cloud and get the distance of a given point to the closest point contained therein. We call this method Iterative Closest Point-cloud-point (ICP).
These methods decrease in computational complexity as well as accuracy from top to bottom. Conversion from CSG tree to one of the three representations is shown in Fig. 1, while Fig. 2 shows the distance with arrows marked from points to CSG surface. For ICS we use ACIS (A commercial geometric modelling package [18]) for converting CSG to B-rep as well as for calculating the distance of points from the closest surface. This method is geometrically most accurate as it uses the exact mathematical formulation of the surface limited only by the numerical precision of the computer. However, this comes at a high computational cost making it the slowest method of all three.

For ICT the CSG object is converted to a triangular mesh, again using ACIS. Once this conversion is complete, the distance is estimated using a hand-coded point to triangle distance routine. In 2D, ICT can be compared to approximating a general curve by a set of linear segments. When the points are very close to the surface the relative difference between distances approximated by ICT and ICS would be significant but this difference rapidly falls as we move away from the surface. The difference also depends on the type of the surface, as for objects made of planar faces the ICT gives accurate results but for curved surfaces like spheres and cylinders the performance degrades but can be improved by using more triangles.

For ICP the CSG object is converted to a point cloud, and the distance of the closest point therein to a given data point is used to approximate the distance to CSG object. In contrast to ICT, which is a first order approximation, ICP uses a zeroth order approximation. Consequently, even for planar surfaces the distance has errors. In terms of speed this method is found to be the fastest, as we use a kd-tree for nearest neighbour lookup. The ICP we present here has some similarity to Iterative Closest Point [19,20], which is widely used for registration. However, the problem of fitting is more complex than registration as in addition to the rigid body transformation the object shape parameters for the CSG tree also need to be estimated.

3. Internal Constraints

One of the attractive features of CSG is its ability to represent a large number of complex objects by applying Boolean operations to a few simple geometric primitives. This powerful feature has its downside for an iterative optimization process, as in the absence of extra geometric information it is very easy to get trapped in the local minima. In most of the CAD models this extra geometric information is implicit and is not encoded as part of the design. But for fitting this geometric information must be encoded explicitly as it can help reduce the number of free parameters, and can also aid in avoiding local minima occurring from degenerate and erroneous configurations.

In Fig. 4 we show two fitting scenarios without employing the internal constraints. As initial values are not very good, the optimization procedure assigns all points to one of the primitives, and subsequently only its parameters change while the other the primitives remain unchanged. However, this is not the desired solution, as we would like to enforce the geometric and topological constraints at all stages. For example in the case of Cylinder-box union the solution in Fig. 4(b) fits a box to the whole point cloud while the cylinder gets disconnected from the box. We can avoid this situation by
enforcing Internal Constraints [8], which encode the geometric relationships between constituent primitives of a CSG tree. For example in the case of Cylinder-box union we would like the radius of the cylinder to equal one of the sizes of the box. Furthermore, the axis of cylinder and box must be parallel and the cylinder must touch the box, so that the intersecting surfaces are tangent.

Similarly for a Flanged T-junction (Table 1), consisting of union of five cylinders, by enforcing all geometric relationships explicitly the number of free parameters is reduced from 40 to 13.

The same argument can be applied to valid ranges for geometric relationships between constituent primitives of a CSG tree. For example in the case of Cylinder-box union we would like the radius of the cylinder to equal one of the sizes of the box. Furthermore, the axis of cylinder and box must be parallel and the cylinder must touch the box, so that the intersecting surfaces are tangent.

Similarly for a Flanged T-junction (Table 1), consisting of union of five cylinders, by enforcing all geometric relationships explicitly the number of free parameters is reduced from 40 to 13.

The standard solution for implementing constraints is to use Constrained Optimization. The Lagrange based method doesn’t exploit constraints to reduce the number of unknowns. Conversely, for each constraint we get another unknown in the form of Lagrange Multiplier [21]. This combined with the slow process of distance evaluation and slower nature of constrained optimization procedures like Sequential Quadratic Programming (SQP) makes the fitting process computationally expensive.

To avoid this situation we decided to use explicit constraint enforcement, and solve the resulting unconstrained optimization problem. The concept of explicit Internal constraints can be explained by an example. A Box is represented by ten parameters in our primitive library, four parameter are for quaternion representation of rotation, three for translation, and three for x, y, and z sizes. If we don’t enforce any constraints for Box-minus-Box case we have twenty parameters giving us many erroneous solutions like the one shown in the Fig. 4(d). While by looking closely at the design intent, the second box can be represented just by the width of the resulting L-shaped hollow frame. The Rest of the parameters can be deduced from the first box. This reduces the number of parameters from 20 in the unconstrained case to 11 in the constrained case. In addition we can enforce maximum and minimum sizes of boxes as well as the values of other parameters if we have

<table>
<thead>
<tr>
<th>Figure</th>
<th>Method (No of elements)</th>
<th>Method</th>
<th>No Noise</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Box Minus Box</td>
<td>ICS (10)</td>
<td>6</td>
<td>1638.4</td>
<td>10.2</td>
</tr>
<tr>
<td></td>
<td>ICP (92)</td>
<td>6</td>
<td>15.6</td>
<td>10.3</td>
</tr>
<tr>
<td></td>
<td>ICP (232)</td>
<td>15</td>
<td>11.7</td>
<td>2904.2</td>
</tr>
<tr>
<td>(b) Cylinder Box Union</td>
<td>ICS (10)</td>
<td>3</td>
<td>687.5</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>ICT (54)</td>
<td>4</td>
<td>5.3</td>
<td>325.1</td>
</tr>
<tr>
<td></td>
<td>ICP (230)</td>
<td>15</td>
<td>9.2</td>
<td>1451.4</td>
</tr>
<tr>
<td>(c) Cylinder Cylinder Intersection</td>
<td>ICS (4)</td>
<td>6</td>
<td>514.0</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>ICT (64)</td>
<td>8</td>
<td>9.8</td>
<td>194.9</td>
</tr>
<tr>
<td></td>
<td>ICP (272)</td>
<td>15</td>
<td>6.4</td>
<td>254.9</td>
</tr>
<tr>
<td>(d) Pipe L Junction</td>
<td>ICS (5)</td>
<td>4</td>
<td>238.2</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>ICT (132)</td>
<td>5</td>
<td>9.3</td>
<td>178.2</td>
</tr>
<tr>
<td></td>
<td>ICP (584)</td>
<td>15</td>
<td>14.0</td>
<td>924.2</td>
</tr>
<tr>
<td>(e) Prism Box Union</td>
<td>ICS (11)</td>
<td>10</td>
<td>2059.7</td>
<td>270.9</td>
</tr>
<tr>
<td></td>
<td>ICT (38)</td>
<td>10</td>
<td>11.9</td>
<td>270.9</td>
</tr>
<tr>
<td></td>
<td>ICP (165)</td>
<td>15</td>
<td>9.2</td>
<td>4019.4</td>
</tr>
<tr>
<td>(f) Flanged T-Junction</td>
<td>ICS (11)</td>
<td>9</td>
<td>2196.8</td>
<td>408.5</td>
</tr>
<tr>
<td></td>
<td>ICT (170)</td>
<td>9</td>
<td>36.9</td>
<td>476.1</td>
</tr>
<tr>
<td></td>
<td>ICP (635)</td>
<td>15</td>
<td>16.4</td>
<td>811.5</td>
</tr>
</tbody>
</table>

Table 1. Comparison of ICS, ICT and ICP.

\[ N = \text{Number of points} \quad D_M = \text{Maximum dimension of object} \quad E_T = \text{Total estimation error} \]

\[ M_U = \text{Number of unconstrained parameters} \quad M_C = \text{Number of constrained parameters} \]
a priori information about them. Table 1 gives a comparison of the number of parameters for different objects for both constrained and unconstrained cases.

4. Fitting Algorithm

Once a mechanism to approximate the distance of one point from a given CSG tree, i.e. the function \( \Omega \), is available, any of the standard non-linear least squares methods [22] can be used to estimate the best shape and pose. We decided to use Levenberg-Marquardt [21] method as it ensures global convergence for non-linear least squares by adaptively combining the steps given by Newton’s method and Steepest descent.

Starting from an initial estimate of CSG parameters \( \Gamma_0 \), at each iteration we get an adjustment given by:

\[ \Delta \Gamma = (J^T J + \lambda J^T J) \Gamma_0 \]

\[ \Gamma_i = \Gamma_0 - \Delta \Gamma \]

Here \( J \) is the Jacobian and \( D \) is the distance given by:

\[ J = \frac{\partial \Omega}{\partial \Gamma} \]

\[ D = \Omega(p, \Gamma_0) \]

Where \( \Omega \) is the distance of the \( i \)th point from the CSG surface, and \( \Gamma \) is the \( i \)th parameter of the CSG tree. In (3) above \( \lambda \) is the Levenberg-Marquardt parameter. When \( \lambda = 0 \) Newton step is taken while for \( \lambda \to \infty \) results in steepest descent step [22].

To estimate the partial derivatives in (5) we used finite differences, as analytical expressions for the CSG tree are not possible. As noted by Schnabel in [22] for sufficiently small step-size the results obtained from the finite difference approximation of the partial derivatives for the least square solution are indistinguishable from the analytical ones.

We use quaternions as a representation of rotation [23], which have an extra constraint that the sum of the squares of their four elements should equal one. This cannot be enforced due to our use of unconstrained optimization, which often results in ill-conditioned or sometimes singular system. To tackle such conditions we solve (3) by using singular value decomposition [24]. This identifies the linearly dependent columns and removes them from the matrix system.

3. Results

We tested the three methods outlined above on some CSG objects, and compared the results on the basis of total error of fit, iterations required, and time taken. The results are presented in Table 1.

The fitting tests were conducted for three data sets for each object: first one was with no noise, the second one contained \( \pm 5 \) mm uniform random noise added to each of \( x, y \) and \( z \) component of the point cloud. The noise level in the third set was \( \pm 10 \) mm. For each dataset we show the total error between final estimated model and input point cloud. This error is calculated using ICS on the computed model and input point cloud.

Comparing on the basis of speed, we find that on average ICS is 20 to 200 times slower than ICT. ICP is usually faster than ICT, but the final error as well as number of iterations required is higher. This can be explained by erroneous estimation of partial derivatives. Even though the number of points used for ICP is greater than the number of triangles for ICT, small perturbations required by finite differences combined with non-smooth nature of ICP result in noisy partial derivatives of the distance function. The same reason explains converging of ICP to local minima resulting in higher total error compared to ICS and ICT (Fig. 5).

As noted earlier, the error performance of ICT depends on the type of the surface as well as the number of triangles used. We see for objects (a) and (f) in Table 1, the results obtained by ICT and ICS have the same amount of error because they are composed of planar faces. In contrast, for object (d) and (e) the error of ICT is significant compared to ICS, because the objects are composed of curved surfaces.

The higher speed of ICT compared to ICS and better and smooth performance compared to ICP makes it the method of choice, giving best trade-off between speed and accuracy. Additionally, as seen from the plot of average relative error in Fig. 3, for initial iteration ICT
can be safely used, as average relative error is very small. For final iteration either ICS or ICT with higher number of triangles should be employed.

From the table we see that explicit enforcement of internal geometric and bound constraints leads to significant reduction in the number of parameters. Moreover, erroneous configurations like the ones shown in Fig. 4 are successfully avoided, because at the end of each iteration only valid solutions are accepted.

The speed comparison of ICT with ICP does not reflect the true situation; as for ICP we use kd-trees while for ICT exhaustive search is used. By employing faster search methods along with multi-resolution techniques we expect the speed performance of ICT to come much closer to that of ICP.

All the three fitting methods assume that the correspondence between the points and the originating faces will be successfully resolved during iterations of the fitting. From the results we see that such is the case for all objects expect (f) and (g), where a local minima is reached before the correct correspondences are found. For example, in Fig. 6(b) points belonging to blue planar faces have been wrongly assigned to neighbouring cylindrical faces. A possible solution for such cases can use an initial segmentation, combined with segment to face correspondence.

3. Conclusions

We have presented and compared three different methods for fitting CSG objects to point clouds. We showed that for all practical purposes ICT provides best trade-off between speed and accuracy. This fitting approach combined with a database of CSG models can be used for object recognition in point clouds. In future we plan to extend the fitting method to use segmented point clouds and to test its performance and utility for object recognition.

References:


[18] The 3D ACIS® Modeler (ACIS) [www.spatial.com](http://www.spatial.com)


