Representing the Motion of Objects in Contact using Dual Quaternions and its Applications

George V Paul and Katsushi Ikeuchi

August 1 1997
CMU-RI-TR-97-31

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

This work was done partly at the Robotics Institute, Carnegie Mellon University, Pittsburgh PA, and partly at the Institute of Industrial Science, University of Tokyo, Tokyo, Japan.

Abstract

This report presents a general method for representing the motion of an object while maintaining contact with other fixed objects. The motivation for our work is the assembly plan from observation (APO) system. The APO system observes a human perform an assembly task. It then analyzes the observations to reconstruct the assembly plan used in the task. Finally, the APO converts the assembly plan into a program for a robot which can repeat the demonstrated task.

The position and orientation of an object, known as its configuration can be represented using dual vectors. We use dual quaternions to represent the configuration of objects in 3D space. When an object maintains a set of contacts with other fixed objects, its configuration will be constrained to lie on a surface in configuration space called the c-surface. The c-surface is determined by the geometry of the object features in contact. We propose a general method to represent c-surfaces in dual vector space. Given a set of contacts, we choose a reference contact and represent the c-surface as a parametric equation based on it. The contacts other than the reference contact will impose constraints on these parameters. The reference contact and the constraints constitute the representation of the c-surface. We show that the use of dual quaternions simplifies our representation considerably. Once we define our c-surface representation, we propose methods to compute the projection of a point in configuration space onto the c-surface and to interpolate between points on the c-surface.

We used our theory to implement the APO system. We used our c-surface representation to correct approximate configurations of the objects at each observed instant of the demonstrated task. We interpolated between corrected points on the c-surface to obtain segments of the assembly path. The complete assembly path used in the observed task is then a concatenation of these path segments. Finally, we used the reconstructed assembly path to program a six axis robot arm to repeat the observed assembly task.

Table of Contents

1 Introduction

Consider two cases of a pair of objects in contact in a plane as shown in Figure 1. Let the shaded object be the moving object. Let the object beneath the moving object be fixed and let its position and orientation be known. We are given the approximate position and orientation of the moving object in the two cases. Let us assume that we can deduce the contacts made in each case as indicated by the ellipses in the figure. Using these contacts we want to describe the freedom of the moving object. We also want to correct the position and orientation of the moving object in each case as shown at the bottom of Figure 1. Finally we want to find the continuous motion of the object between the two corrected poses.

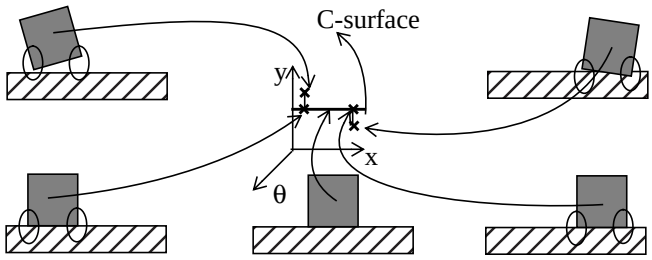


Figure 1 C-surface projection and interpolation

We can do this in the three steps;

1. Find the set of positions and orientations *C*, of the moving object at which the contacts are physically feasible.

2. Find the closest positions and orientations in this set *C* to each of the given approximate positions and orientations.

3. Find the smooth path between the two corrected positions and orientations such that all the points on the path lie in the computed set *C*.

The position and orientation of an object in space is known as its *configuration*. The configuration can be specified by a set of *n* numbers. The *n* dimensional space is called the configuration space of the object and denoted as the *c-space*[32]. When an object is free to

move in space, its configuration can be any point in the c-space. But when there are other fixed objects in space, the configurations at which the moving object overlaps with the fixed objects are physically infeasible. The set of forbidden configurations in c-space is called the *c-obstacle*. The set of configurations where there is no overlap between the objects is called the *free space*. Now, in between free space and c-obstacles are configurations where a set of geometrical features of the moving object makes contact with a set of geometrical features of the fixed objects. This space is sometimes called the *contact space* [17]. The geometrical features are vertices and edges for polygonal objects and vertices, edges and faces for polyhedral objects. The set of configurations in c-space at which a certain set of contacts are maintained is defined to be the *c-surface* [36]. The c-surface can be computed from the geometry of the feature pairs in contact.

In this report we develop a general method to represent the c-surface for a set of contacts between polygonal objects in a plane and polyhedral objects in space. This solves step one of our example problem and the set C is the c-surface. Once we have the representation of the c-surface we develop a method to project any point in c-space onto the c-surface. The projection finds the closest point in the set C to the given point. This solves the second part of the problem. Finally we explain how to obtain a smooth path between two points on a c-surface. This solves the final part of the example problem stated earlier.

The computation of the c-surface is directly dependent on the representation of the c-space. The simplest c-space representing planar displacements is the (x,y,θ) space. Here, (x,y) are the two translation parameters and θ is the rotation angle. For spatial displacement the extension of this approach is to use the six dimensional (x,y,z,θ,ϕ,ψ) c-space. Here (x,y,z) are the translation parameters and (θ,ϕ,ψ) are the three angles representing the orientation of the object in 3D space.

The disadvantage of this conventional representation of c-space is that the space is not homogenous which makes the representation of c-surfaces difficult. Another approach proposed by Ge and McCarthy[12] is to use planar quaternions to represent the c-space for planar displacements and dual quaternions to represent the c-space for displacements in 3D space. The advantage of using planar and dual quaternions in representing rigid body displacements is that all the group properties of displacements are preserved in it unlike the simple c-space representation[35]. Thus the algebra of planar and dual quaternions can be used to manipulate displacements. This greatly simplifies the representation of c-surfaces.

Ge and McCarthy[12] used planar quaternions to represent the configuration of polygonal objects in a plane. The c-surface of the two basic contacts, (*ve* and *ev*) between polygonal objects is parametrized using a rotation angle and a translation parameter. They extended their theory to use dual quaternions to represent the configurations of objects in 3D space. The three basic contacts, between polyhedral objects (*vf*, *fv*, and *ee*) can be parametrized

using three Euler angles for rotation and two parameters for translation. They used their representations to algebraically compute the c-surfaces for an articulated robot arm moving among fixed objects. The c-surfaces are the result of a single contact between a geometrical feature of a robot link with a geometrical feature of an obstacle.

In our work we use the representation of basic contacts developed by Ge and McCarthy to build a general representation of the c-surface for a set of contacts. The use of dual vectors to represent the c-space (planar quaternions for the planar case and dual quaternions for the 3D case) was key in simplifying the representation of the c-surface. The ease of projection and interpolation on the c-surface is a consequence of this representation of the c-space.

In order to represent the c-surface for a set of contacts, we choose a reference contact. The c-surface can be represented as a parametric equation using this reference contact. We formulate the constraint on these parameters for each remaining contact. Each of these constraints can be represented as a matrix equation. The reference contact and the constraint equations constitute the representation of the c-surface for multiple contacts.

We use our representation of the c-surface to compute the projection of a point in c-space onto the c-surface. We can formulate the error in a contact using the constraint equation. The projection can then be formulated as a minimization of the sum of the squared errors. This gives us an unambiguous solution to the projection problem.

Let the moving object maintain the identical set of contacts with the fixed objects in two distinct configurations. These two configurations will correspond to two points lying on the same c-surface. A smooth path which lies on the c-surface and passes through the two points will correspond to a motion of the moving object which maintains the set of contacts. We extend our projection technique to interpolate a smooth path between points lying on the same c-surface.

A major advantage of our proposed approach is that it is possible to handle any number of contacts without having to enumerate all possible configurations. Our representation and projection theory is general enough to work in the planar space as well as in 3D space.

We implemented our theory on the assembly plan from observation (APO) system. The APO is the primary motivation for our work. In the following section we explain the APO system and its components.

1.1 The Assembly Plan from Observation (APO) System

Conventional methods of programming and operating robots are teach-by-showing, teleoperation, textual programming, and automatic programming. This work is motivated by a system which is based on a novel method of programming robots called Assembly Plan from Observation (APO)[28].

When we consider a task such as assembly, the conventional methods all have drawbacks. Teach-by-showing [29] has the disadvantage that the control is low-level, typically at the joint level. Hence it is not robust enough for programming robots for assembly tasks. Textual programming [11] needs a programmer trained in specialized robot programming languages such as VAL, who converts the assembly task into basic robot commands which can accomplish the task. Teleoperation [14][46][53] needs a human operator as an essential part of the system. So, a system based on it cannot operate autonomously. Unlike the previous methods, automatic programming [31][34][32] needs no human intervention after the specification of the task. The system can identify and compute the solutions for all aspects of the task automatically. But the disadvantage of this method is the overwhelming complexity of the subtasks.

The APO system observes a human perform an assembly task in front of a perception system. It then understands what the human did and how it was done. This information is then used to program a robot. Thus it incorporates the simplicity of teach-by-showing and teleoperation. In addition, the robot programs are generated automatically. By using the operator's solutions for the subtasks like assembly planning and grasp planning instead of computing them, the APO system avoids the complexity of automatic robot programming.

When an object is assembled into another, there are two major aspects of the task that can be observed. They are, the object motion resulting in the assembly, and the grasping strategy used to manipulate the object. In this work, we concentrate on the reconstruction of the motion used by the operator to assemble the object.

The schematic of the APO system is shown in Figure 2. The three major components of the APO system are the observation module, the task modeling module and the task execution module. We will indicate the input, output, and the method of operation of each component.

The human operator performs the assembly task by manipulating objects in a sequence of subtasks. The observation module observes the operator perform each subtask. The task recognition module then understands what was done, how it was done, and uses subtask models to represent the observed subtasks. Finally, the task execution module converts the

modeled subtasks into a program that will enable the robot to perform the assembly task in its workspace.

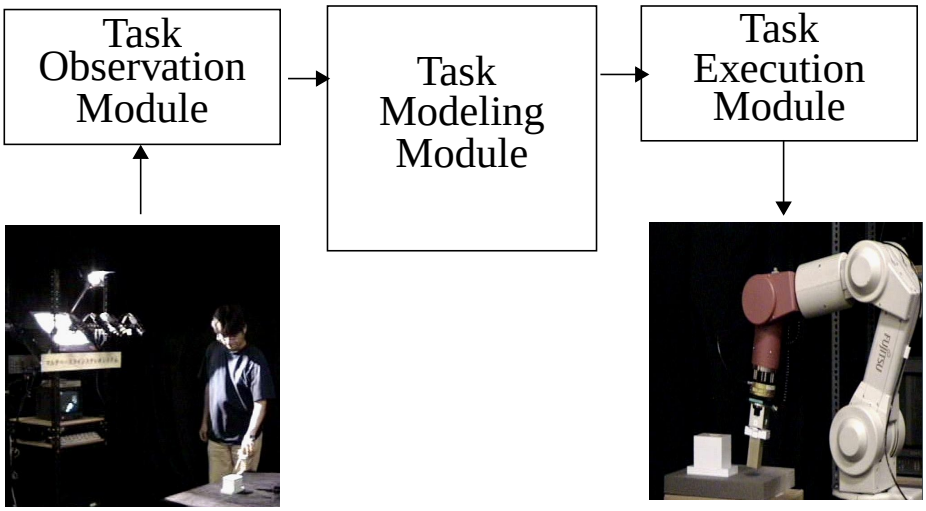


Figure 2 : Assembly Plan from Observation System

Observation Module

The observation module uses a perception (presently, vision only) system to observe the human operator performing the assembly task. The vision system records the scene at sufficiently close instants in time. The module then tracks the assembled objects and the human hand from the start to the end of each assembly task. The output of the observation modules is a rough estimate of the object positions and orientations at each recorded instant of time during the assembly.

Task Recognition Module

In each stage of the APO system operation, the operator manipulates a moving object into the previously assembled objects in the scene. The aim of the task recognition module is to understand the assembly task performed by the operator during each stage and how it was done. This involves the following:

- Modeling the Assembly Plan - This is the part of the APO system that motivates this report. The aim is to extract the motion plan used by the human while assembling a part into the other parts. This is called modeling the assembly subtask. The complete assembly task can be represented as a series of such subtask models.

- Modeling the Grasp Plan- While performing the assembly task, it is useful to understand where and how the operator grasps the assembled objects. This can be used for planning the grasp for the robot. In addition to the grasp, we can also obtain the motion of the hand before and after the grasp. This is called the grasping strategy. A detailed description of modeling these aspects of the task from observation can be found in [23].

Task Execution Module

Once the assembly plan and the grasping plan is reconstructed from the human assembly observations, the output is a trajectory of the robot hand and its fingers. The task execution module takes this as input and can generate the motion control commands for the robot to reproduce the trajectory in its workspace. This nominal motion has to be augmented by feedback strategies to account for uncertainities in the world model.

1.2 Related Work

This work is one in a line of works related to its motivation, the APO system. The first work in this series was by Ikeuchi et al[48]. The idea in this work was to reconstruct the assembly plan by observing the beginning and end of each assembly task. Once they computed the contact states at these two instants, they used a pre-computed skill library to obtain the procedure for achieving the contact state transitions. The theory could handle assembly motions using only translation motions. Further work on the system by Suehiro [48] resulted in an iterative method to correct the configuration of an assembled object given a set of contacts. Further improvements in the system using curved objects can be found in a later version of the APO[27]. Kuniyoshi et al. [26] also has a system which also can teach robots by demonstration, but it is limited to simple pick and place tasks.

A earlier part of our work dealt with the extension of the work by Suehiro to include rotation[41]. In this work we developed the theory of contact states. We used the theory of polyhedral convex cones by Tucker and Goldman[25] to partition the contact space into a set of finite topologically distinct states. The draw back of this system as with its predecessors was that the gross assembly motion cannot be reconstructed from just the contact states without a lot of limiting assumptions.

Another aspect of the APO is the reconstruction of the grasp and the grasping strategy from observation. This work was done by Kang[23]. Here the vision system and a polyhemous device records the actions of the human operator continuously. This work was further extended by Jiar et al[21] to obtain the grasp strategy by using just the vision system and without the polyhemous device.

This work differs from the contact state approach of the past. We recognize that the two observations of the task is not sufficient to reconstruct the compliant motion. Instead we obtain as many observations of the task as is possible. We then compute the c-surfaces corresponding to the contacts and correct the observations. The assembly motion plan can then be reconstructed as smooth paths on these c-surfaces.

We implemented this new approach for the planar assemblies using Brost's work[6] to compute c-surfaces[42]. To extend the idea to 3D space we looked at work in robot motion planning mainly the analytical representation of the c-space obstacle. Most of the related work is in the field of motion planning for robots. The main idea in all such work is the analytic computation of the configuration space obstacles given the geometry of the features in contact. One essential difference in these works is the type of representation of the configuration space.

Work by Lozano-Perez originated the concept of c-space in the field of robot motion planning. The c-space used in the work was the three dimensional (x,y,θ) c-space for the planar case and the six dimensional (x,y,z,θ,ϕ,ψ) c-space for the spatial robots. The major focus was finding a collision free path. Donald[9] extended this work to use $(x,y,z,R(\theta))$ to represent c-space, where $R(\theta)$ is the 3x3 rotation matrix. The complexity of the c-surfaces are such that they required symbolic mathematics software.

Brost[6] derived the c-surfaces for polygonal objects in a plane for the (x,y,θ) space. This work considers all possible contact configurations and individually derives the equations for the c-surfaces corresponding to them. We used the results to implement a previous version of the APO system[42]. Unfortunately the work has no counterpart in 3D. Avnaim et al. [2] also developed a general method to analytically compute the c-surfaces for polygonal objects in a plane. Bajaj et al. [3] presents an algebraic method to compute the c-surface for curved convex objects. The limitation is that they consider only translation motions and convex objects.

Other work such as by Canny[7] deals with finding the shortest collision free paths in c-space. The work does not explicitly deal with computing the c-surfaces. The shortest paths are connected path segments whose end points can be computed by solving a system of homogenous polynomial equations. Canny provides an elegant solution to the problem. Our work also involves the system of polynomial equations, but the equations are not homogenous.

Work by Popplestone [44] and Koutsou [24] use the 4x4 transformation matrix representation of displacement to compute the position of objects in contact. Popplestone's motivation is quite similar to our work, in that the qualitative relations between geometrical features of the object in an assembly are used to obtain the compute the possible configura-

tion of the assembled object. Koutsou's work is in the same genre and proposes a method to compute the c-surfaces for objects in contact using the matrices. Koutsou uses the graph searching technique suggested by Hopcroft et al. [17]. Hopcroft et al. proposed that finding a compliant path in c-space can be reduced to searching in a graph where the nodes are zero dimensional c-surfaces and the edges are one dimensional c-surfaces. Hwang[18] also uses the *T* matrix representation of the c-space to compute the c-surface for robot manipulators. The work assumes stick like manipulators and triangulated obstacles.

Ge and McCarthy[12] introduced planar quaternions to represent c-surfaces. This work specifically deals with generating the c-surfaces for a planar manipulator in contact with objects in the plane. They extended their theory to represent c-surfaces for spatial robots in 3D space using dual quaternions[13]. Their work shows how basic contacts can be represented using dual quaternions. They use it to compute the algebraic equations of c-surfaces. Both the 4x4 matrix representation and the dual quaternion representation have the valuable property of being a group. Theoretically the ease of representation must be equivalent, but the key difference is that the number of parameters in the a 4x4 matrix are far more than those in the dual quaternion. This makes the dual quaternion a better alternative.

Our work uses the essentials of the basic contact representation of Ge and McCarthy. We use it to build a general representation of c-surfaces. Unlike their approach our general method can represent the c-surface for a set of contacts. We then show how the representation can be used to compute the projection onto a c-surface and finally, how to interpolate on c-surfaces.

1.3 Outline

In section 2 we explain the basics of dual quaternions and how they can be used to represent the basic *vf, fv* and *ee* contacts between polyhedral objects in space. We explain our general method for representing c-surfaces in section 3. In this chapter we our representation to project and interpolate on c-surfaces. In section 4 we explain how we implemented our theory on the APO system. Finally in section 5 , we summarize our work and draw our conclusions and suggest directions of future research.

2 Dual Quaternions

The dual quaternion can be used to represent rigid displacements of objects in space. In this section we introduce the work of Ge and McCarthy [13] which uses dual quaternions to represent the three basic contacts between polyhedral objects.

2.1 Representation of Displacement

The displacement of an object in 3D space can be represented by its translation parameters *(x,y,z)* and the Euler angles *(*θ,ϕ,ψ) [8]. The dual quaternion can be used for representing rigid displacements in space. The algebra of dual quaternions has a product operation which has the property of linearity and associativity. In addition the dual quaternion space has a unit element and an inverse element. These properties make dual quaternions a group. Rigid displacements in space also possesses the same group properties. This makes dual quaternions suitable for representing and manipulating rigid spatial displacements.

Any rigid displacement of an object in space can be represented by a rotation about an axis in space and a translation along the same axis as illustrated in Figure 3. This axis and the rotation angle together with the translation distance is called a screw. The screw is an intrinsic property of the displacement and is independent of the coordinate systems of the objects. The dual quaternion can be obtained from this screw representation as shown below..

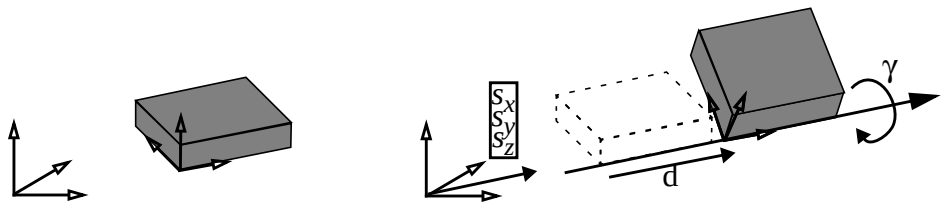


Figure 3 Screw representation

The dual quaternion representation of displacement $Q=(\bar{q},\bar{q}^0)$ is a combination of two

four element vectors. The four elements of $\bar{q}$ are related to the screw axis direction (s_x,s_y,s_z) and rotation angle(γ) as shown in (EQ1). This is the quaternion which is commonly used to represent rotations in 3D space.

$$\bar{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} s_x \sin\left(\frac{\gamma}{2}\right) \\ s_y \sin\left(\frac{\gamma}{2}\right) \\ s_z \sin\left(\frac{\gamma}{2}\right) \\ \cos\left(\frac{\gamma}{2}\right) \end{bmatrix} \quad \text{(EQ1)}$$

If the rotation is expressed using Euler angles (φ,θ,ϕ) where φ is the rotation angle about the original x axis, θ is the rotation about the new z' axis, ϕ is the rotation angle about the final x" axis. The quaternion $\bar{q}$ in terms of the Euler angles can be written as shown in (EQ2) [13].

$$\bar{q} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi+\psi}{2}\right) \\ -\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\varphi-\psi}{2}\right) \\ \sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi-\psi}{2}\right) \\ \cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\varphi+\psi}{2}\right) \end{bmatrix} \quad \text{(EQ2)}$$

The remaining part of the dual quaternion, $\bar{q}^0$ is defined to be a combination of the translation (x,y,z) and $\bar{q}$ as shown in (EQ3).

$$\bar{q}^0 = \begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -z & y & x \\ z & 0 & -x & y \\ -y & x & 0 & z \\ -x & -y & -z & 0 \end{bmatrix}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = [D]\bar{q} \quad \text{(EQ3)}$$

This $\bar{q}^0$ can be conveniently expressed as a matrix equation as shown in (EQ4).

$$\bar{q}^0 = x[D_x]\bar{q} + y[D_y]\bar{q} + z[D_z]\bar{q}$$

$$[D_x] = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \qquad [D_y] = \frac{1}{2}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \qquad [D_z] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad \text{(EQ4)}$$

Given a dual quaternion $Q=(q,q^0)$ we can compute the axis-angle (s_x,s_y,s_z,γ) representation of the rotation from $\bar{q}$ as shown in (EQ5).

$$\gamma = 2\,\text{atan}2(\sqrt{q_1^2 + q_2^2 + q_3^2}, q_4)$$

$$\begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = \frac{1}{\sin\left(\frac{\gamma}{2}\right)}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad \text{(EQ5)}$$

We can obtain the translation (x,y,z) from $\bar{q}^0$ using equations (EQ6)-(EQ8). Given the axis-angle representation of the rotation and the translation, we can compute any of the other representations of displacement.

$$x = 2(q_1q_8 - q_2q_7 + q_3q_6 - q_4q_5) = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \bar{q}4D_x\bar{q}^0 \quad \text{(EQ6)}$$

$$y = 2(q_1q_7 + q_2q_8 - q_3q_5 - q_4q_6) = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \bar{q}4D_y\bar{q}^0 \quad \text{(EQ7)}$$

$$z = 2(-q_1q_6 + q_2q_5 + q_3q_8 - q_4q_7) = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix} 2 \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} = \bar{q}4D_z\bar{q}^0 \quad \text{(EQ8)}$$

One of the advantages of using dual quaternions to represent displacements is the ease of composing displacements. The composition of two displacements represented by quaternions $X=(\bar{x},\bar{x}^0)$ and $Y=(\bar{y},\bar{y}^0)$ will be given by the quaternion product $Z = XY = [X^+]Y = [Y^-]X$, which is defined as a matrix product shown in (EQ9)[13].

$$\begin{bmatrix}\bar{z}\\ \bar{z}^0\end{bmatrix} = [X^+]Y = \begin{bmatrix}[x^+] & [0]\\ [x^{0+}] & [x^+]\end{bmatrix}\begin{bmatrix}\bar{y}\\ \bar{y}^0\end{bmatrix}$$

$$\begin{bmatrix}\bar{z}\\ \bar{z}^0\end{bmatrix} = [Y^-]X = \begin{bmatrix}[y^-] & [0]\\ [y^{0-}] & [y^-]\end{bmatrix}\begin{bmatrix}\bar{x}\\ \bar{x}^0\end{bmatrix} \quad \text{(EQ9)}$$

$$[x^+] = \begin{bmatrix} x_4 & -x_3 & x_2 & x_1\\ x_3 & x_4 & -x_1 & x_2\\ -x_2 & x_1 & x_4 & x_3\\ -x_1 & -x_2 & -x_3 & x_4\end{bmatrix} \quad [y^-] = \begin{bmatrix} y_4 & y_3 & -y_2 & y_1\\ -y_3 & y_4 & y_1 & y_2\\ y_2 & -y_1 & y_4 & y_3\\ -y_1 & -y_2 & -y_3 & y_4\end{bmatrix}$$

The sub-matrices $[x^{0+}]$ and $[y^{0-}]$ are constructed from x^0 and y^0 in the same way as $[x^+]$ and $[y^-]$ are constructed from x and y as shown in (EQ9).

Note that the rotation part of the dual quaternions in (EQ9) is isolated from the rest of the equation. We will show how this characteristic can be taken advantage of when representing c-surfaces.

We also note that the dual quaternion has 8 elements while rigid displacement in space has only six degrees of freedom. This means that in order to directly use dual quaternions to represent rigid spatial displacements, the dual quaternion must satisfy two constraints[35]. The first constraint on the rotation part of the quaternion $\bar{q}$ is shown in (EQ10). This constrains the $\bar{q}$ to be a unit vector.

$$\bar{q}^T\bar{q} = q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \quad \text{(EQ10)}$$

The other constraint on q^0 is shown in (EQ11) is the result of the definition of $\bar{q}^0$ as given in (EQ1).

$$\bar{q}^T\bar{q}^0 = q_1q_5 + q_2q_6 + q_3q_7 + q_4q_8 = 0 \quad \text{(EQ11)}$$

Mathematically this means that the set of dual quaternions in R^8 space which can repre-

sent spatial displacements will have to lie on a six dimensional algebraic sub-manifold of R^8 called the *image space* of spatial displacements[35].

2.2 Representation of Basic Contacts

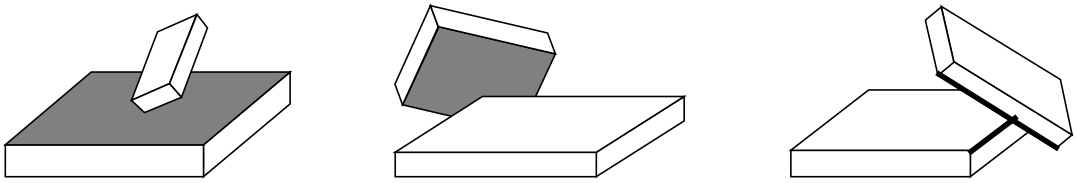
For polyhedral objects in contact there are three basic contacts[32] as shown in Figure 5. They are;

1. A vertex of the moving object is in contact with the face of the fixed object. This is known as the (*vf*) contact. An example is shown in Figure 5(a).

2. A face of the moving object is in contact with the vertex of the fixed object. This is known as the (*fv*) contact. An example is shown in Figure 5(b).

3. An edge of the moving object is in contact with the edge of the fixed object. This is the *(ee)* contact. An example is shown in Figure 5(c).

All other contact configurations can be expressed as a combination of these basic contacts. In this section we introduce the work of Ge and McCarthy [13] and show how dual quaternions can be used to represent the c-surface for basic contacts in a very uniform manner. The representation of all three contacts can be done using a parametric equation or by using an implicit algebraic equation.



(a) vf-contact **(b) fv-contact** **(c) ee-contact**

Figure 4 Basic contacts

We will demonstrate this uniform representation of the c-surface for all the three basic contacts in the following sub-sections.

2.2.1 *vf*-contact

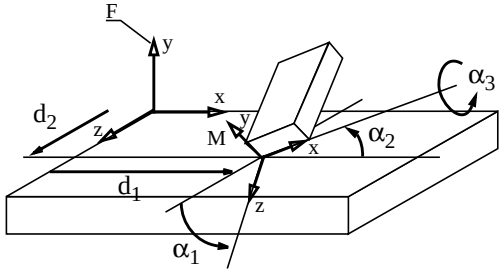


Figure 5 *vf* contact [13]

Consider the vertex of a smaller object in contact with the face of the larger fixed object as shown in Figure 5. Let the coordinate systems on the moving object *(M)* and the fixed object *(F)* be defined as shown in Figure 5. The relative configuration of the two coordinate systems can be expressed as the following sequence of displacements of *M* with respect to *F*. A translation by d_1 along the *x* axis. A translation by d_2 along the *z* axis. An orientation by Euler angles $(\alpha_1,\alpha_2,\alpha_3)$. We can compute the dual quaternion of each of these displacements using (EQ2)-(EQ4). Using (EQ9) we can compute the dual quaternion representing the combined displacements as shown in (EQ12)-(EQ14).

$$Y(d_1, d_2, \alpha_1, \alpha_2, \alpha_3) = \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix} = X(d_1)Z(d_2)S(\alpha_1, \alpha_2, \alpha_3) \tag{EQ12}$$

$$\bar{y} = \begin{bmatrix} \cos\left(\frac{\alpha_2}{2}\right)\sin\left(\frac{\alpha_1 + \alpha_3}{2}\right) \\ -\sin\left(\frac{\alpha_2}{2}\right)\sin\left(\frac{\alpha_1 - \alpha_3}{2}\right) \\ \sin\left(\frac{\alpha_2}{2}\right)\cos\left(\frac{\alpha_1 - \alpha_3}{2}\right) \\ \cos\left(\frac{\alpha_2}{2}\right)\cos\left(\frac{\alpha_1 + \alpha_3}{2}\right) \end{bmatrix} \tag{EQ13}$$

$$\bar{y}^0 = \left[\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}\frac{d_1}{2} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}\frac{d_2}{2}\right]\bar{y} = \left[\left[U_{vf}\right]d_1 + \left[V_{vf}\right]d_2\right]\bar{y} \tag{EQ14}$$

This is the parametric form of the c-surface equation for a single *vf* contact. The parameters (d_1, d_2) are the translation parameters and (α_1,α_2,α_3) are the rotation parameters. Note that we can write the translation part of the planar quaternion $\bar{y}^0$ in terms of the rotation part $\bar{y}$ and the d_1 and d_2 parameters as a matrix equation shown in (EQ14). This is a valuable property of this representation as will be shown later.

Constraint Equation

The c-surface for the *vf* contact can also be represented using an implicit equation. In the *vf* contact case this implicit equation essentially means that the displacement of *M* along the *y* axis of *F* will be zero. Using (EQ7) we can form the constraint on the dual quaternion $Y=(\bar{y},\bar{y}^0)$ as shown in (EQ15).

$$y_1y_7 + y_2y_8 - y_3y_5 - y_4y_6 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y} T_{vf} \bar{y}^0 = 0 \qquad \text{(EQ15)}$$

2.2.2 *fv*-contact

Consider the face of a moving object in contact with the vertex of a fixed object as shown in Figure 6[13]

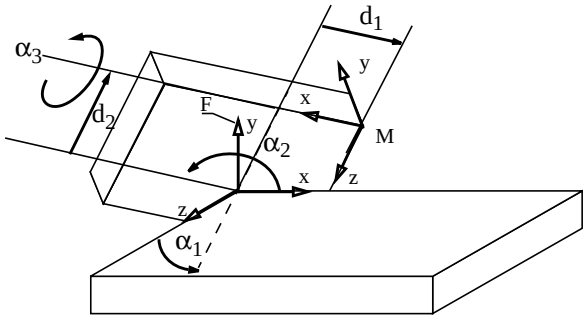


Figure 6 fv contact [13]

Let the coordinate systems on the moving object *(M)* and the fixed object *(F)* be as shown in Figure 6. The relative displacement of the two coordinate systems can be expressed as the following sequence of displacements of *M* with respect to *F*. An orientation by Euler angles (α_1,α_2,α_3). A translation by $-d_1$ along the new *x* axis. A translation by $-d_2$ along the latest *z* axis. Using the same technique as for the *vf* contact we obtain the dual

quaternion as shown in (EQ16).

$$Y(d_1, d_2, \alpha_1, \alpha_2, \alpha_3) = \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix} = S(\alpha_1, \alpha_2, \alpha_3)X(-d_1)Z(-d_2) \quad \text{(EQ16)}$$

The definition of $\bar{y}$ will be exactly same as (EQ13). The definition of $\bar{y}^0$ in terms of the rotation part $\bar{y}$ and the d_1 and d_2 parameters is given by (EQ17).

$$\bar{y}^0 = \left[\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \frac{d_1}{2} + \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \frac{d_2}{2} \right] \bar{y} = \left[\left[U_{fv} \right] d_1 + \left[V_{fv} \right] d_2 \right] \bar{y} \quad \text{(EQ17)}$$

Constraint Equation

The constraint for the fv contact is given by (EQ18). The equation essentially states that the *y* coordinate of the vertex in the face coordinate system is zero.

$$y_1 y_7 - y_2 y_8 - y_3 y_5 + y_4 y_6 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y} T_{fv} \bar{y}^0 = 0 \quad \text{(EQ18)}$$

2.2.3 *ee*-contact

Consider the edge of a moving object in contact with the edge of a fixed object as shown in Figure 7[13]

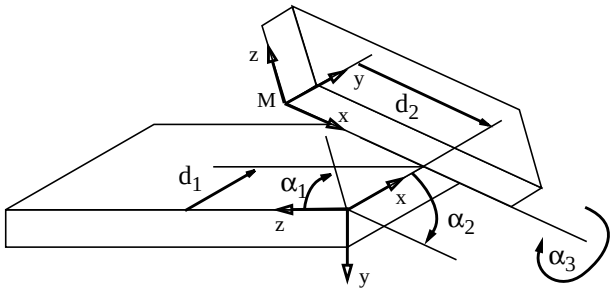


Figure 7 *ee* contact [13]

Let the coordinate systems on the moving object *(M)* and the fixed object *(F)* be as shown in Figure 7. The relative configuration of the two coordinate systems can be expressed as the following sequence of displacements. A translation by $+d_1$ along the *x* axis. An orientation by Euler angles $(\alpha_1,\alpha_2,\alpha_3)$. A translation by $-d_2$ along the *x* axis. Using the same technique as for the *vf* contact we obtain the dual quaternion as in (EQ19).

$$Y(d_1, d_2, \alpha_1, \alpha_2, \alpha_3) = \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix} = X(d_1)S(\alpha_1, \alpha_2, \alpha_3)X(-d_2) \qquad \text{(EQ19)}$$

The definition of $\bar{y}$ will be exactly the same as (EQ13). The definition of $\bar{y}^0$ in terms of the rotation part $\bar{y}$ and the d_1 and d_2 parameters will be as given by (EQ20).

$$\bar{y}^0 = \left[\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \frac{d_1}{2} + \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \frac{d_2}{2} \right] \bar{y} = \left[\left[U_{ee}\right] d_1 + \left[V_{ee}\right] d_2 \right] \bar{y} \qquad \text{(EQ20)}$$

Constraint Equation

The constraint equation for the *ee* contact is given by (EQ21). This constraint is obtained by setting the displacement along the direction of the cross product of the two edges in contact to be zero.

$$y_1y_5 - y_2y_6 - y_3y_7 + y_4y_8 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y} T_{ee} \bar{y}^0 = 0 \qquad \text{(EQ21)}$$

2.2.4 General Form of the Contact Equations

The parametric equations for the basic contacts, (EQ13) and (EQ14) for *vf* contact, (EQ13) and (EQ17) for *fv* contact, and (EQ13) and (EQ20) for *ee* contact, have the same structure. They can be written as a general equation as shown in (EQ22). The matrices U_c and V_c depend on the type of contact.

$$Y(d_1, d_2, \alpha_1, \alpha_2, \alpha_3) = \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix}$$
$$\bar{y} = S(\alpha_1, \alpha_2, \alpha_3) \quad \text{(EQ22)}$$
$$\bar{y}^0 = \left[\left[U_c\right] d_1 + \left[V_c\right] d_2\right] \bar{y}$$

We can see that the constraint equations for the three basic contacts also have the same form and can be written conveniently as a matrix product as shown in (EQ23). The matrix T_c depends on the type of contact.

$$\begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} [T_c] \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}[T_c]\bar{y}^0 = 0 \quad \text{(EQ23)}$$

These two general representation of the basic contacts are a valuable feature of using the dual quaternions to represent contacts. They simplify the general representation of the c-surface we propose considerably.

2.2.5 Representation in World Coordinates

If the configuration of the objects are defined in terms of a world coordinate system as shown in Figure 8. Then, the local contact quaternion can be related to the configuration of the object in the world coordinate system. Let T_{bf} represent the transformation of the fixed object feature with respect to the world. Let T_{ml} represent the transformation of the moving feature with respect to the moving object coordinates. Let Y be the dual quaternion representing the contact. The configuration of the moving object with respect to the world is Z_{bl}.

The relation between the coordinate systems is shown in Figure 8,

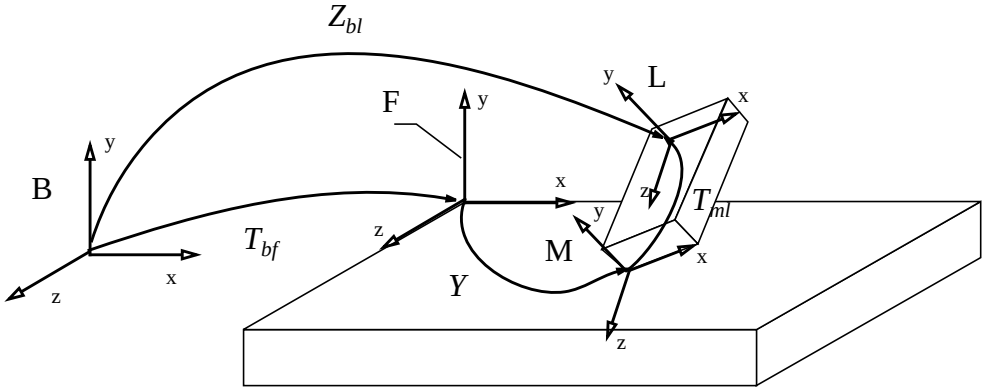


Figure 8 World Coordinates

We can express the configuration of the object Z_{bl} in terms of the contact quaternion Y as shown in (EQ24)[13]. The matrices B and C can be computed from T_{bf} and T_{ml}, both of which can be computed from the geometry of the features in contact.

$$Z_{bl} = T_{bf} Y T_{ml} = [T_{bf}^{+}][T_{ml}^{-}]Y$$

$$\begin{bmatrix} \bar{z} \\ \bar{z}^0 \end{bmatrix} = \begin{bmatrix} [C] & 0 \\ [B] & [C] \end{bmatrix} \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix} \qquad \text{(EQ24)}$$

$$[C] = [t_{bf}^{+}][t_{ml}^{-}] \qquad [B] = [t_{bf}^{0+}][t_{ml}^{-}] + [t_{bf}^{+}][t_{ml}^{0-}]$$

Once again we see that the rotation is separate from the rest of the quaternion. This makes it possible to split (EQ24) into two parts as shown in (EQ25) and (EQ26). The translation part $\bar{z}^0$ can be expressed in matrix form in terms of $(\bar{y},d_1,d_2)$ by substituting (EQ22) into (EQ24) to obtain (EQ26).

$$\bar{z} = [C]\bar{y} \qquad \text{(EQ25)}$$

$$\bar{z}^0 = [C][U_c] d_1 \bar{y} + [C][V_c] d_2 \bar{y} + [B]\bar{y} \qquad \text{(EQ26)}$$

2.2.6 Limits on Parameters

In order for the contact to be legal the parameters $(\alpha_1,\alpha_2,\alpha_3)$ and (d_1,d_2) can vary only within limits defined by the geometry of the features in contact [13]. These limits for the

translation parameters will be as shown in (EQ27).

$$0 \le d_1 \le \overline{d_1} \qquad 0 \le d_2 \le \overline{d_2} \tag{EQ27}$$

The limits on the rotation parameters can be found indirectly from the rotation quaternion $\bar{y}$. The limits on $\bar{y}$ can be determined by the fact that the neighboring vertices of one feature of the contact cannot invade the other feature of the contact. As will be explained in the following chapter, this condition is related to the constraint equation (EQ24). This allows us to write the condition as shown in (EQ28).

$$\bar{y}^T T_c [v_k^{0-}] \bar{y} \ge 0 \tag{EQ28}$$

If v_k is the coordinates of the k-th neighboring vertex in the local feature coordinate system, then the matrix $[v_k^{0-}]$ is obtained using (EQ9). The legal range of $\bar{y}$ has to satisfy (EQ28) for all the k neighboring vertices. The limits on the rotation parameters (α_1,α_2,α_3) will be determined from the legal range of $\bar{y}$.

***vf* contact**

The limits on the translation parameters are as shown in (EQ27). The values d_1 and d_2 are determined by the length of the sides for a rectangular face. For a non-rectangular face with axes not aligned to the face coordinate axes, the limits will be more complicated.

The limits on the rotation quaternion $\bar{y}$ are determined by the fact that the neighboring vertices of the vertex in contact cannot invade the plane of the face. This condition can be written as shown in (EQ28). The matrix T_c will correspond to T_{vf}.

***fv* contact**

For the *fv* contact, The limits on the translation parameters will be similar to (EQ27). The values d_1 and d_2 are determined by the length of the sides of the moving face.

The limits on the rotation quaternion $\bar{y}$ are determined as for the *vf* contact. The matrix T_c will be replaced by T_{fv}. The neighboring vertices will be that of the fixed vertex.

***ee* contact**

The limits on the translation parameters for the *ee* contact will be similar to (EQ27). The values d_1 and d_2 are determined by the length of the edges in contact.

The limits on the rotation quaternion $\bar{y}$ are determined as for the *vf* contact. The matrix T_c will be replaced by T_{ee}. The neighboring vertices will be that of the faces forming the moving edge.

3 C-Surface : Representation, Projection and Interpolation

When there is a contact between a single geometrical feature of the moving object with a single geometrical feature of the fixed object, the c-surface representation will be identical to the corresponding basic contact as explained in the previous chapter. In this chapter we explain how to represent the c-surface when there are multiple basic contacts between the moving object and fixed objects. We show how we can combine the c-surfaces of the individual basic contacts to obtain the representation of the c-surface for multiple contacts.

The projection of a point in c-space onto a given c-surface can be defined as the closest point on the c-surface to the given point in c-space. In order to find this point we formulate the projection as a minimization problem. The solution of this minimization problem will be the projection. We then show how we can interpolate between two points on the c-surface by using the projection technique.

3.1 Representation

When there is just a single elementary contact between the moving object and the fixed object, the parametric c-surface representation will be identical to the basic c-surface equation given by (EQ15). The five independent parameters will be α_1, α_2, α_3, and d_1, d_2 and their limits will be given by (EQ20).

Instead of using the Euler angles $(\alpha_1,\alpha_2,\alpha_3,)$ to represent the rotation we use the rotation part of the contact dual quaternion $\bar{y}$. We keep the d_1 and d_2 parameters to define the translation. The advantage of using $\bar{y}$ instead of Euler angles is that it provides a homogenous parameterization of the rotation space which becomes indispensable when we have to differentiate with respect to rotation and when we have to interpolate between rotations.

When there is more than one contact between the moving object and the fixed objects, the c-surface will be intersection of the c-surfaces of the N individual contacts. The indepen-

dent parameters defining this intersection will be fewer than the number of parameters of a single contact. In addition any point on the c-surface must now satisfy the limits of all the individual contacts simultaneously. Instead of finding the independent parameters of the c-surface and its limits, we represent the c-surface as follows.

First, we choose a reference contact, which can be any contact. We choose the parametric equation (EQ22) of this reference contact as the c-surface equation. We use the $\bar{y}$ quaternion of this reference contact to represent the rotation and the d_1 and d_2 parameters of this reference contact to be the translation parameters of the c-surface. Each of the remaining *(N-1)* contacts will enforce a constraint on $(\bar{y},d_1,d_2)$. We show that each of these constraints can be expressed compactly as a matrix equation. As for the limits on $(\bar{y},d_1,d_2)$, in addition to the original limits of the reference contact, the parameters now have to simultaneously satisfy the limits of the *(N-1)* additional contacts. We convert the limits of the *(N-1)* contacts to the limits of the reference contact. Thus the c-surface representation for multiple contacts have the same $(\bar{y},d_1,d_2)$ parameters of the reference contact, except that, now the parameters are subject to *(N-1)* constraints.

3.1.1 C-Surface Equation

Consider the objects shown in Figure 9, where the smaller moving object makes two contacts with the larger object. Let *Z* be the configuration of the moving object with respect to the world coordinates. Let the parameters of the dual quaternion representing the first contact c_1 be $(\bar{y},d_1,d_2)$. Let c_1 be the reference contact. Let c_2 be the second contact and let its parametric equation have parameters, $(\bar{y},d_1,d_2)_2$. Now the parameters of the c-surface will be $(\bar{y},d_1,d_2)$. The parameter of the contact c_2, $(\bar{y},d_1,d_2)_2$ can be related to $(\bar{y},d_1,d_2)$ as explained below.

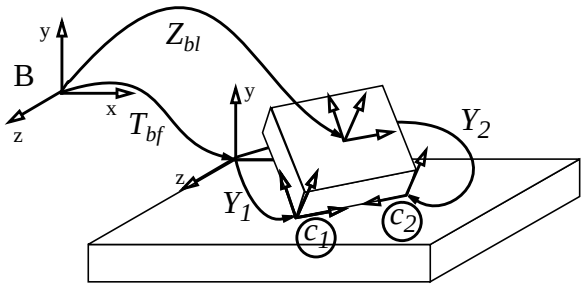


Figure 9 Multiple contacts

Let *Z* be the configuration of the moving object in world coordinates. Both the dual quaternion of the first contact and the dual quaternion of the second contact can be related to *Z* using (EQ24) as shown in (EQ29). The sub-matrices *[C]*, and *[B]* correspond to the first

contact and can be obtained from its geometry. The sub-matrices $[C]_2$, and $[B]_2$ correspond to the second contact and can be obtained from the geometry of its features.

$$\begin{bmatrix} \bar{z} \\ \bar{z}^0 \end{bmatrix} = \begin{bmatrix} [C] & 0 \\ [B] & [C] \end{bmatrix} \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix} = \begin{bmatrix} [C]_2 & 0 \\ [B]_2 & [C]_2 \end{bmatrix} \begin{bmatrix} \bar{y} \\ \bar{y}^0 \end{bmatrix}_2 \tag{EQ29}$$

If we equate the $\bar{z}$ half of (EQ29) we get (EQ30). This equation essentially relates the rotation of the second contact to the rotation of the reference contact.

$$\bar{y}_2 = [C]_2^{-1}[C]\bar{y} \tag{EQ30}$$

Equating the $\bar{z}^0$ half of (EQ29) and substituting (EQ26) we get (EQ31). This equation relates the translation half of the dual quaternion to the translation parameters d_1 and d_2 and the rotation parameter $\bar{y}$ of the first contact.

$$\bar{y}_2^0 = [C]_2^{-1}\left([C][U]d_1 + [C][V]d_2 + [B] - [B]_2[C]_2^{-1}[C]\right)\bar{y} \tag{EQ31}$$

Now the constraint on the dual quaternion representing contact c_2 will be given by (EQ23) as a matrix equation $\bar{y}_2 T_{c_2} \bar{y}_2^0 = 0$. If we substitute the values of $\bar{y}_2$ and $\bar{y}^0{}_2$ given above into (EQ23) we can translate the constraint equation for the second contact into a constraint on the reference contact parameters. This is a constraint on the $(\bar{y},d_1,d_2)$ parameters due to the second contact is shown in (EQ32).

$$e_2 = \bar{y}^T\left[M(d_1, d_2)\right]_2\bar{y} = 0$$
$$\left[M(d_1, d_2)\right]_2 = [C]^T[C]_2[T_c]_2[C]_2^{-1}\left([C][U]d_1 + [C][V]d_2 + [B] - [B]_2[C]_2^{-1}[C]\right) \tag{EQ32}$$

For the example with two contacts this is the only constraint. For *N* contacts there will be *(N-1)* constraints. The generalized constraint due to the *i*-th contact will be given by the general equation (EQ33).

$$e_i = \bar{y}^T \left[M(d_1, d_2) \right]_i \bar{y} = 0 \qquad i = 2 \rightarrow N$$

$$\left[M(d_1, d_2) \right]_i = \left[C \right]^T \left[C \right]_i \left[T_c \right]_i \left[C \right]_i^{-1} \left(\left[C \right] \left[U \right] d_1 + \left[C \right] \left[V \right] d_2 + \left[B \right] - \left[B \right]_i \left[C \right]_i^{-1} \left[C \right] \right) \quad \text{(EQ33)}$$

$$\left[M(d_1, d_2) \right]_i = \left[D_1 \right]_i d_1 + \left[D_2 \right]_i d_2 + [E]_i$$

Note that these are *(N-1)* simultaneous non-linear equations in $(\bar{y},d_1,d_2)$, and their solution defines the c-surface for *N* contacts. The matrix coefficients in the constraints (E,D_1,D_2) define the properties of the c-surface and are defined as shown in (EQ34).

$$\left[D_1 \right]_i = \left[C \right]^T \left[C \right]_i \left[T_c \right]_i \left[C \right]_i^{-1} \left(\left[T_c \right]_i \left[C \right]_i^{-1} \left[C \right] \left[U \right] \right)$$

$$\left[D_2 \right]_i = \left[C \right]^T \left[C \right]_i \left[T_c \right]_i \left[C \right]_i^{-1} \left(\left[T_c \right]_i \left[C \right]_i^{-1} \left[C \right] \left[V \right] \right) \quad \text{(EQ34)}$$

$$[E]_i = \left[C \right]^T \left[C \right]_i \left[T_c \right]_i \left[C \right]_i^{-1} \left(\left[B \right] - \left[B \right]_i \left[C \right]_i^{-1} \left[C \right] \right)$$

The $(N-1)$ sets of (E,D_1,D_2) coefficient matrices together with the reference contact will completely define the c-surface. The convenience of their use will become apparent when we have to interpolate on the c-surface.

3.1.2 Limits

The limits on the parameters will be affected by the limits on the additional contacts. Now all the limits of the individual contacts have to satisfied simultaneously. The limits on the *i*-th contact can be mapped into the $(\bar{y},d_1,d_2)$ of the reference contact.The limits on the local contact angles can be mapped into the reference contact by using (EQ35).

$$\bar{y} = \left[C \right]_i^{-1} \left[C \right] \bar{y}_i \qquad \bar{y}_i = 0 \rightarrow \alpha_{v_i} \quad \text{(EQ35)}$$

If we map the limits of $\bar{y}$ of the *i*-th contact back to the reference contact, we can obtain the legal range of $\bar{y}$. If we use the definition of (d_1,d_2) as given by (EQ8) we can convert the limits of $(d_1,d_2)_2$ to limits on (d_1,d_2) as shown in (EQ36).

$$x_i = \bar{y}^T \left[L_x(d_1, d_2) \right]_i \bar{y} = 0 \rightarrow \bar{d}_{1_i} \qquad i = 2 \rightarrow N \quad \text{(EQ36)}$$

$$\bar{z}_i = \bar{y}^T \left[L_z(d_1, d_2) \right]_i \bar{y} = 0 \rightarrow \bar{d}_{2_i} \qquad i = 2 \rightarrow N \tag{EQ37}$$

The matrices $L_x(d_1,d_2)$ and $L_y(d_1,d_2)$ can be obtained in a similar procedure as was employed for $M(d_1,d_2)$.

3.2 Projection

Mason[36] defines the projection on a c-surface in the context of hybrid force-position control of the robot. According to this definition the projection is the intersection of the orthogonal space of the c-surface passing through the point with the c-surface. The difficulty with implementing this definition is the concept of distance in this non-euclidean c-space. Since rotation and translation are represented in the same space the distance along an axis representing rotation will be different from a distance along an axis representing translation.

Despite this non-Euclidean c-space, the result of a certain motion in c-space on any point on the moving object in physical space is obvious. The movement in c-space will translate to movement of points of the moving object in 3D space. We show that if we consider the distance travelled by the physical points instead of the distance in c-space the definition of projection becomes unambiguous. First, we reassert that the c-surface is defined by the set of features in contact between the moving object and the fixed objects. Thus we can define the projection of a point P in c-space to the c-surface to be that point P_c on the c-surface at which the cumulative squared distance travelled by the points in contact is minimum.

We can formulate the distance to a contact using the dual quaternions. The projection can then be formulated as a minimization of the sum of the squared distances. The solution gives us an unambiguous solution to the projection problem.

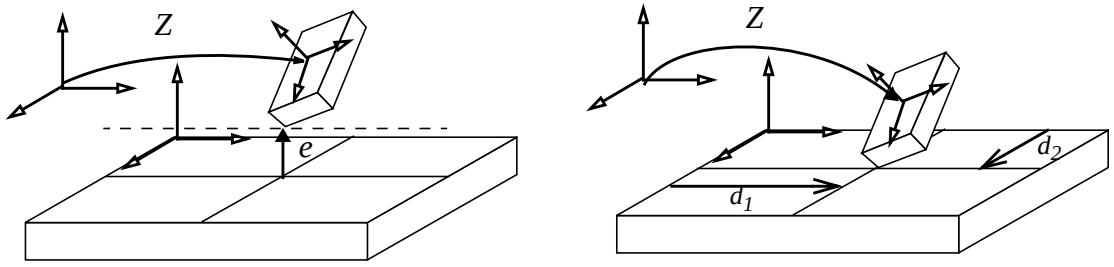
If the c-surface is represented using dual quaternions as explained in the previous section, then the projection of a point onto the c-surface will be completely determined by the values of the $(\bar{y},d_1,d_2)$ parameters defining the c-surface. In this section we first illustrate the projection technique for a single contact case and then develop the general method to project onto the c-surface for multiple contacts.

3.2.1 Single Contact

We will first compute the projection for each of the three basic contacts. The equations

for projection using dual quaternions for all three cases have the same structure. This following subsections will show that the LHS of the constraint equations of the c-surface (EQ23) for all the three basic contacts is actually the signed distance of a point in c-space to the c-surface.

3.2.1.1 *vf* Contact



(a) (b)

Figure 10 Projection for single *vf* contact

Consider the object in a configuration Z shown in Figure 10(a). Now assume that the vertex v is in contact with the face f. The closest configuration to the given configuration which makes this contact is when the vertex is at the intersection of the perpendicular from the vertex to the face. The projected configuration is shown in Figure 10(b).

The projection on the c-surface will be completely determined by the values of $(\bar{y},d_1,d_2)$. These values of the projection can be obtained by substituting the given configuration Z into (EQ24). The value of the rotation parameter $\bar{y}$ can be obtained directly as shown in (EQ38).

$$\bar{y} = \left[C\right]^{-1}\bar{z} \qquad \bar{y}^0 = \left[C\right]^{-1}\left(\bar{z}^0 - \left[B\right]\bar{y}\right) \tag{EQ38}$$

The value of the translation parameters d_1 can be obtained by simply finding the x coordinates using (EQ6) as shown in (EQ39).

$$d_1 = 2(y_1y_8 - y_2y_7 + y_3y_6 - y_4y_5) = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}[P_{vf}]\bar{y}^0 \tag{EQ39}$$

The value of the translation parameters d_2 can be obtained by finding the z coordinates

using (EQ8) as shown in (EQ40).

$$d_2 = 2(-y_1y_6 + y_2y_5 - y_3y_8 + y_4y_7) = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}[Q_{vf}]\bar{y}^0 \quad \text{(EQ40)}$$

The perpendicular distance *d* between the vertex and the face is exactly the *y* coordinate of the vertex in the face coordinates. It can seen that it is exactly the LHS of the constraint equation for the *vf* contact (EQ13) and can be written as (EQ41). When the c-space point lies exactly on the c-surface this distance *e* will be zero.

$$e = y_1y_7 + y_2y_8 - y_3y_5 - y_4y_6 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}T_{vf}\bar{y}^0 \quad \text{(EQ41)}$$

3.2.1.2 *fv* Contact

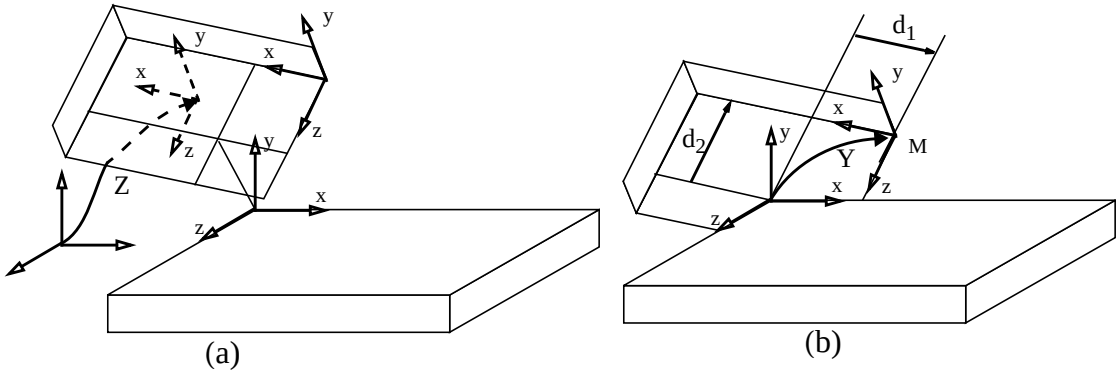


Figure 11 Projection for single *fv* contact

Consider the object in a configuration *Z* shown in Figure 10(a). Now assume that the moving face *f* is in contact with the fixed vertex *v*. The closest configuration to the given configuration which makes this contact is when the vertex is at the intersection of the perpendicular from the vertex to the face. The projected configuration is shown in Figure 11(b).

The projection will be completely determined by the values of $(\bar{y},d_1,d_2)$. These values of the projection can be obtained by substituting the given configuration *Z* into (EQ22). The

value of the rotation parameter $\bar{y}$ can be obtained directly as shown in (EQ42).

$$\bar{y} = [C]^{-1}\bar{z} \qquad \bar{y}^0 = [C]^{-1}\left(\bar{z}^0 - [B]\bar{y}\right) \tag{EQ42}$$

The value of the translation parameter d_1 can be computed to be as shown in (EQ43).

$$d_1 = 2(y_1y_8 - y_2y_7 + y_3y_6 - y_4y_5) = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}[P_{fv}]\bar{y}^0 \tag{EQ43}$$

The value of the translation parameters d_2 can be computed to be as shown in (EQ44).

$$d_2 = 2(-y_1y_6 + y_2y_5 - y_3y_8 + y_4y_7) = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}[Q_{fv}]\bar{y}^0 \tag{EQ44}$$

As with the *vf* contact the perpendicular distance *d* between the vertex and the face is exactly the *y* coordinate of the vertex in the face coordinates. It can shown that it is exactly the LHS of the constraint equation for the *fv* contact (EQ18) and can be written as (EQ45). When the c-space point lies exactly on the c-surface this distance *e* will be zero.

$$e = y_1y_7 - y_2y_8 - y_3y_5 + y_4y_6 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}T_{fv}\bar{y}^0 \tag{EQ45}$$

3.2.1.3 *ee* Contact

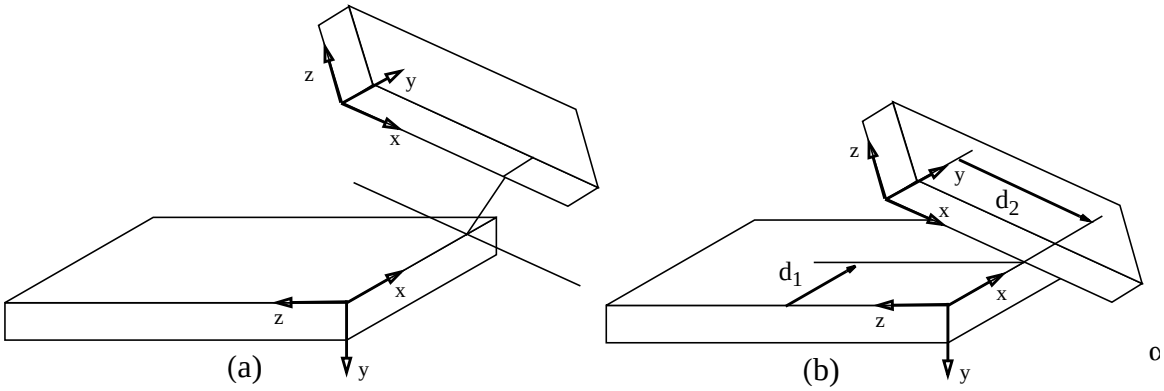


Figure 12 Projection for single *ee* contact

Consider the object in a configuration Z shown in Figure 12(a). Now assume that the edge e_1 is in contact with the edge e_2. The closest configuration to the given configuration which makes this contact is when the moving edge moves along the cross product of the two edge vectors. The is the projected configuration shown in Figure 12(b).

The projection will be completely determined by the values of $(\bar{y},d_1,d_2)$. These values of the projection can be obtained by substituting the given configuration Z into (EQ2). The value of the rotation parameter $\bar{y}$ can be obtained directly as shown in (EQ46).

$$\bar{y} = \left[C\right]^{-1}\bar{z} \tag{EQ46}$$

The values of the d_1 and d_2 parameters are not as simple as for the *vf* and *fv* contacts. These values can be obtained by additional manipulation. The result for d_1 will be as shown in (EQ47).

$$d_1 = \frac{2(y_1y_7 - y_2y_8 + y_3y_5 - y_4y_6)}{(y_1y_2 + y_3y_4)} = \left[y_1\ y_2\ y_3\ y_4\right]\left(\frac{2}{(y_1y_2 + y_3y_4)}\begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\\ 1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\end{bmatrix}\right)\begin{bmatrix}y_5\\ y_6\\ y_7\\ y_8\end{bmatrix} \tag{EQ47}$$

$$= \bar{y}[P_{ee}]\bar{y}^0$$

The value of the translation parameters d_2 can be obtained as shown in (EQ48).

$$d_2 = \frac{2(-y_1y_7 - y_2y_8 + y_3y_5 + y_4y_6)}{(y_1y_2 + y_3y_4)} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} \left(\frac{2}{(y_1y_2 + y_3y_4)} \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} \quad \text{(EQ48)}$$

$$= \bar{y}[Q_{ee}]\bar{y}^0$$

As with the *vf* contact the perpendicular distance *d* between the moving edge and the fixed edge can be obtained as a distance along the cross product of the two edge directions. It can shown that it is exactly the LHS of the constraint equation for the *ee* contact (EQ21) and can be written as (EQ49). When the c-space point lies exactly on the c-surface this distance *e* will be zero.

$$e = y_1y_5 - y_2y_6 - y_3y_7 + y_4y_8 = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} 2 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \bar{y}T_{ee}\bar{y}^0 \quad \text{(EQ49)}$$

3.2.2 Multiple Contacts

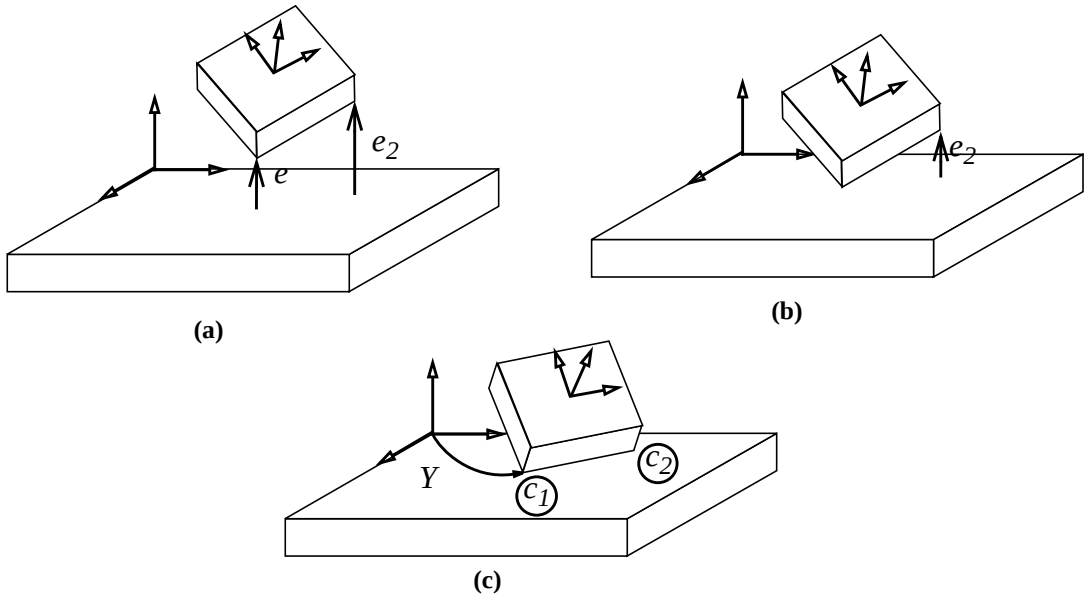


Figure 13 Projection for multiple contacts

Suppose we are given a point in c-space which corresponds to the position and orientation of the moving object as shown in Figure 13(a). Now we need to compute the projection of this configuration onto the c-surface *C*. Let the c-surface correspond to the multiple contact c_1 and c_2.

We obtain the projection onto the c-surface using the same steps as for the representation. We first choose a reference contact and project the point onto the single contact c-surface to obtain the configuration as shown in Figure 13(b). Let the parameters of this projection be $(\bar{y},d_1,d_2)_0$.

We then setup *(N-1)* simultaneous non-linear equations (EQ33) which define the c-surface. Recall that after the projection onto the first contact, each of the errors in the *(N-1)* remaining contacts e_i shown in Figure 13(b) can be written as one of the equations (EQ41), (EQ45), and (EQ49).

The direct solution of these *(N-1)* simultaneous non-linear equations is difficult. Instead, we use an iterative method. We first set up an objective function which gives a distance measure of the errors in contacts. This measure is the sum of the squares of the e_i values for the (*N-1)* contacts as given by (EQ51).

$$L = \sum_{2}^{N} (e_i)^2 = \sum_{2}^{N} \left(\bar{y}^T \left[M(d_1, d_2) \right]_i \bar{y} \right)^2 \qquad \text{(EQ50)}$$

Since $\bar{y}$ represents rotation it is constrained to be a unit vector. If this constraint is not enforced the iteration will rapidly get $\bar{y}$ to zero. To bring the constraint into the picture we use Lagrange constant λ as shown in (EQ51)[47].

$$Q = \sum_{2}^{N} \left(\bar{y}^T \left[M(d_1, d_2) \right]_i \bar{y} \right)^2 + \lambda(\bar{y}^T \bar{y} - 1) \qquad \text{(EQ51)}$$

Now the value of $(\bar{y},d_1,d_2)$ at which the objective function Q is minimum will be the projection we seek. This can be obtained by solving the simultaneous equations obtained by setting the partial differentials to zero as shown in (EQ52).

$$\underbrace{Min(Q(\bar{y}, d_1, d_2, \lambda))}$$

$$\begin{aligned}
\frac{\partial}{\partial \bar{y}} Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial d_1} Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial d_2} Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial \lambda} Q(\bar{y}, d_1, d_2, \lambda) &= 0
\end{aligned} \qquad \text{(EQ52)}$$

We obtain the solution of (EQ52) in $(\bar{y},d_1,d_2,\lambda)$ space using the Newton-Raphson's method.The method compute the solution iteratively using (EQ53). The starting point being $(\bar{y},d_1,d_2,\lambda)_0$. The first and second order partial differentials *Q'* and *Q"* are computed as shown in Appendix B.

$$\begin{bmatrix} \bar{y} \\ d_1 \\ d_2 \\ \lambda \end{bmatrix}_{k+1} = \begin{bmatrix} \bar{y} \\ d_1 \\ d_2 \\ \lambda \end{bmatrix}_k - \left(Q'' \Big|_k \right)^{-1} Q' \Big|_k \qquad \text{(EQ53)}$$

3.3 Interpolation

Consider two distinct configurations of objects in contact as shown in Figure 14. The smaller moving object makes two *vf* contacts with the top face of the larger fixed object in both cases. The c-surface corresponding to the two cases will be the same. The representation of this c-surface will be determined by the geometry of features in contact as explained in 3. The two configurations in Figure 14 will correspond to two distinct points, P_1 and P_2 lying on this c-surface. If *P* is a continuous curve in c-space which lies completely on this c-surface and passes through P_1 and P_2, then the motion along *P* from P_1 to P_2 in the c-space will correspond to a smooth transition from the configuration in the first case to the configu-

ration in the second case while maintaining the contacts.

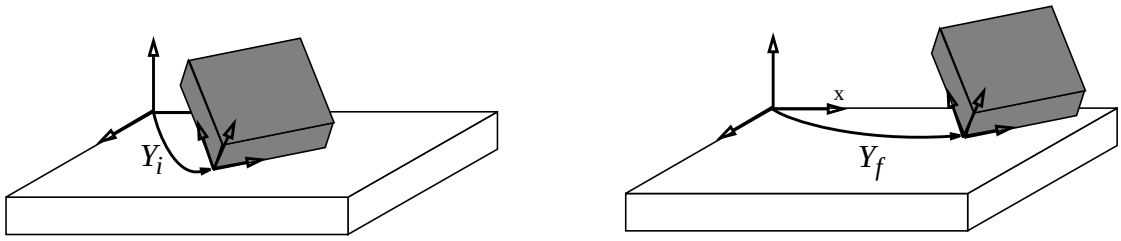


Figure 14 Interpolation on the C-surface

We obtain this smooth curve from P_1 to P_2 by interpolation. If the c-surface is parameterized by a few independent parameters, we can interpolate the parameters independently. The c-space point corresponding to the interpolated parameters will be guaranteed to remain on the c-surface.

Mathematically, the one dimensional interpolated path can be described using a single parameter *t*. The method of obtaining this path *P* given P_i and P_f is basically first order interpolation subject to the constraints imposed by the c-surface. The interpolated point *P(t)* will be given in terms of the variable *t* as shown in (EQ54).

$$\begin{aligned} P(t) &= f(t, P_i, P_f) \qquad t = 0 \rightarrow 1 \\ \bar{y}^T \left[M(d_1, d_2) \right]_i \bar{y} &= 0 \qquad i = 2 \rightarrow N \end{aligned} \tag{EQ54}$$

The general c-surface representation we explained in section 5 is parameterized by $(\bar{y},d_1,d_2)$. Interpolating in the parameter space will mean interpolating in this $(\bar{y},d_1,d_2)$ space. We show how to independently interpolate in this space for the single contact case. However, for the multiple contact case, the parameters are not independent. They have to satisfy the *(N-1)* constraints. We explain how to interpolate in this case.

3.3.1 Linear Interpolation of Translation

If d_{1i} is the initial value of the d_1 parameter and d_{1f} is the final value. Then we can linearly interpolate (lerp) the d_1 parameter using (EQ55). We can do the same for the d_2 parameter as shown in (EQ56).

$$d_1 = d_{1_i} + t(d_{1_f} - d_{1_i}) \qquad t = 0 \rightarrow 1 \tag{EQ55}$$

$$d_2 = d_{2_i} + t(d_{2_f} - d_{2_i}) \qquad t = 0 \rightarrow 1 \tag{EQ56}$$

3.3.2 Spherical Linear Interpolation of Rotation

Interpolating $\bar{y}$ between two values of the rotation in space $\bar{y}_i$ and $\bar{y}_f$ cannot be done linearly, because the $\bar{y}$ is constrained to lie on the unit hyper-sphere in the four dimensional quaternion space. We can interpolate on the surface of the hyper-sphere using a technique called spherical linear interpolation (slerp)[50]. The idea is to find the shortest path between $\bar{y}_i$ and $\bar{y}_f$ which lies on the hyper-sphere. The shortest path will be the arc traced by rotating $\bar{y}_i$ to $\bar{y}_f$ about an axis perpendicular to the hyper-plane containing $\bar{y}_i$ and $\bar{y}_f$. The quaternion $\bar{y}$ obtained using spherical interpolation can be written as (EQ57).

$$\bar{y} = \frac{\sin(\theta(1-t))}{\sin(\theta)}\bar{y}_i + \frac{\sin(\theta t)}{\sin(\theta)}\bar{y}_f \qquad t = 0 \rightarrow 1$$
$$\cos(\theta) = \bar{y}_i \cdot \bar{y}_f \qquad \text{(EQ57)}$$

3.3.3 Single Contact

Consider a single basic contact between the objects at two configurations P_i and P_f. The parameters defining the c-surface will be $(\bar{y},d_1,d_2)$ as explained earlier. Let $(\bar{y},d_1,d_2)_i$ be the parameter values at the initial position P_i, and $(\bar{y},d_1,d_2)_f$ be the parameter values at the final position P_f. The rotation parameter $\bar{y}$, and the translation parameters d_1 and d_2 are independent.

The values of d_1 and d_2 can easily be linearly interpolated between the initial and final values as shown in (EQ55) and (EQ56). The quaternion $\bar{y}$ obtained using spherical interpolation can be expressed as shown in (EQ57). The actual configuration of the moving object can then be obtained by substituting the parameter values into (EQ22).

3.3.4 Multiple Contacts

As explained earlier the parameters of the c-surface for multiple contacts are $(\bar{y},d_1,d_2)$ of the reference contact, which are the same parameters of the single contact case. But, the key difference is that in the multiple contact case, the parameters are not independent. This means that the parameters cannot be interpolated independently. The dependence of the parameters are given by the *(N-1)* nonlinear constraint equations as explained in section 5. If we independently interpolate the $(\bar{y},d_1,d_2)$ parameters they may not satisfy these *(N-1)* constraint equations. Nevertheless, if we project these tentative interpolated points onto the c-surface using these constraints as explained in section 6, we can obtain the exact interpolated points which will satisfy the constraints.

Despite being unable to analytically solve the *(N-1)* non-linear equations for the general case, we show that for certain special cases we can do exactly that. One major criterion for such cases is that the translation parameters d_1 and d_2 be independent of the rotation parameter $\bar{y}$. In addition, these special cases can only occur when the rotation $\bar{y}_i$ and $\bar{y}_f$ have the same fixed axis of rotation. The system of simultaneous non-linear equations in this case reduces to a linear system in d_1 and d_2. The planar case is one such example and we can exploit this linearity as explained in our previous work [43].

Special Cases

We can check if d_1 is an independent parameter by checking if all the D_1 matrices shown in (EQ58) are skew symmetric or not. If it is skew-symmetric then d_1 is an independent parameter and its value can be interpolated linearly between P_i and P_f using (EQ55).

$$D_{1_i} = \left[C\right]^T \left[C\right]_i \left[T_c\right]_i \left[C\right]_i^{-1} \left(\left[T_c\right]_i \left[C\right]_i^{-1} \left[C\right] \left[U\right] \right) \qquad i = 2 \rightarrow N \tag{EQ58}$$

The independence of d_2 can be checked exactly as for d_1 except that we check for the skew-symmetry of the D_2 matrices as shown in (EQ59).

$$D_{2_i} = \left[C\right]^T \left[C\right]_i \left[T_c\right]_i \left[C\right]_i^{-1} \left(\left[T_c\right]_i \left[C\right]_i^{-1} \left[C\right] \left[V\right] \right) \qquad i = 2 \rightarrow N \tag{EQ59}$$

The result of these checks can result in either one of three cases. Both d_1 and d_2 are independent. Second case is when either d_1 or d_2 is independent while the other is dependent on $\bar{y}$. The third case is when both d_1 and d_2 are dependent on $\bar{y}$.

***d_1 and d_2* are independent**

One such instance is shown in Figure 16. Note that the two translation parameters are independent of the orientation of the moving object.

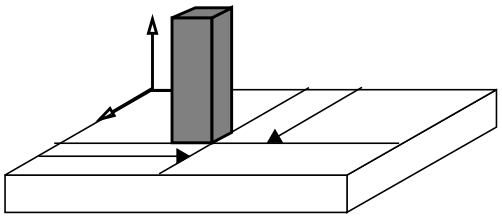


Figure 15 *d_1* and *d_2* are independent

Either d_1 *or* d_2 is independent

An instance is shown in Figure 16. Here the parameter d_2 is independent of the orientation of the moving object while the parameter d_1 is dependent on it.

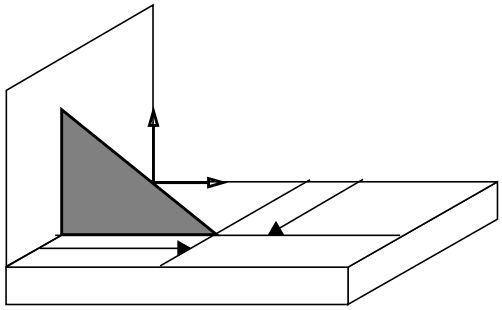


Figure 16 d_2 is independent

Both d_1 *and* d_2 are dependent

For the moving object shown in Figure 17, both parameters d_1 and d_2 are dependent on the orientation of the object.

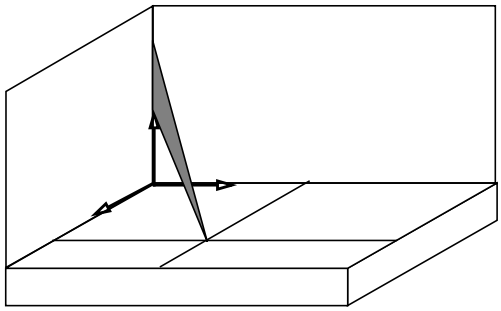


Figure 17 d_1 and d_2 are dependent

4 Implementation

We have implemented the theory developed in the previous sections on our assembly plan from observation (APO) system. This section explains the details of this implementation. The APO system observes a human perform an assembly task, analyzes the observations and reconstructs the compliant motion plan used in the assembly task. The motion plan can then be used to program a robot to repeat the assembly task.

The APO system contains a number of modules. First among them is the multibaseline stereo system which provides us with 3d range images of each observed scene of the assembly task. The second module is VANTAGE, a geometric modeling system which provides us with the geometric details of all the features of a modeled object. The third module is the tracking system which uses the models of the objects to localize the assembled objects through the range image sequence. The output of these modules is the location of the assembled objects at each discrete instants of time. This forms the input to our task modeling system.

4.1 Observation Module

The aim of the observation module is to record the actions of the human performing the assembly task. The output of the module will be the locations of the assembled objects at each observed instant. We use a multibaseline stereo system to record the assembly task. Once we have the sequence of stereo images we run a stereo algorithm on each set of images to get the range image off-line. The next step is to track the assembled objects in each scene from the start of the assembly to the end of the assembly.

4.1.1 Multibaseline stereo system.

The schematic of the stereo system is shown in the Figure 18.

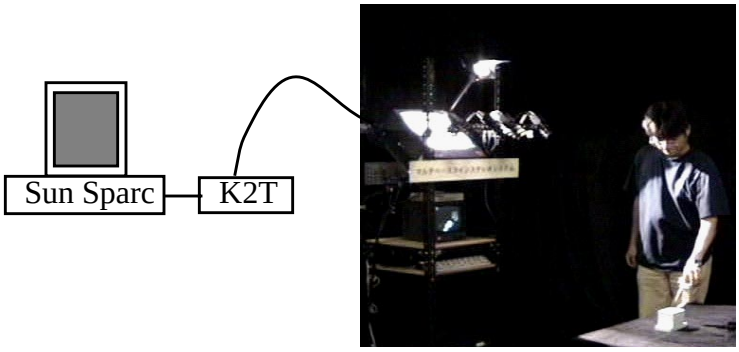


Figure 18 Multibaseline stereo system

The system contains three monochrome cameras arranged such that their optical axes converge to approximately the space where the assembly is performed. The system is calibrated to compute the parameters of each camera. The scene is illuminated by a stripe pattern using a slide projector. This provides sufficient texture to the intensity images which make the stereo algorithm work efficiently. The camera outputs are connected to the RGB input of a K2T digitizer. The K2T situated inside a sun workstation can record approximately 10 sets of images each second. This is sufficient for recording significant contact configurations in a typical assembly sequence. The capture rate is not uniform, so we cannot reliably extract the velocity information from the data.

Once we record the sets of stereo images of an assembly task we compute the range images for each scene off-line. Each set of the input images is fed into a stereo algorithm[22] which was optimized to work on this setup. The system delivers 512x640 pixels range images up to an accuracy of one percent, which translates to an error of about 3 mm for each pixel.

4.1.2 Geometric Modeling

We use a constructive solid geometry (CSG) based modeling system called VANTAGE. VANTAGE has a set of primitive objects such as cube, cone, prism, sphere, cylinder etc. VANTAGE also has a set of boolean operations which operate on these primitive objects. The operators are, union, intersection, difference, mirror and move. VANTAGE then constructs a general object by combining or modifying the set of primitives using the set of operators. For example, the CSG definition of the peg used in our assembly will be a tree as

shown in Figure 19. The leaf nodes will be the primitives or nil. The parent nodes will be the boolean operation applied to the children nodes.

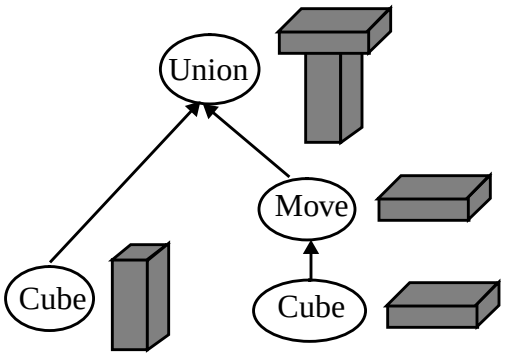


Figure 19 CSG node representation of a peg in VANTAGE

The representation by the CSG tree is the first step in the modeling of an object. The next step is to construct the boundary of the object. The boundary representation of any solid object can be represented as a network of faces, edges and vertices. Each face is a collection of edges. Each edge is defined by a pair of vertices. The vertices are defined by their *(x,y,z)* coordinates. The nodes representing the faces, edges, and vertices are spatially related to reach other. This topological formation is encoded in VANTAGE in the form of the *winged edge* representation.

If we consider every face, edge, and vertex composing an object as a node, then each edge in a *winged edge* representation is connected to eight nodes as shown in Figure 20. The two end vertices of the edge are the *P-vertex* and *N-vertex*. The four neighboring edges of the edge are *PCW, PCCW, NCW, NCCW.* The two adjacent faces of the edge are *P-face* and *N-face*.

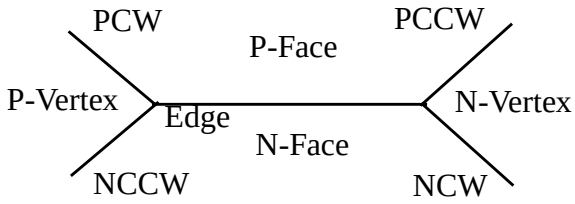


Figure 20 Winged Edge Representation

The boundary representation for the peg shown in Figure 19 will now contain all the

faces, edges and vertices as shown in Figure 21.

pegz:
csg_node: peg
rigid_motion: [[4 1 0 0 0][4 0 1 0 0][4 0 0 1 0][4 0 0 0 1]]
face_list: (B-PEG-2-f-9 B-PEG-2-f-8... B-PEG-2-f-0)
edge_list: (B-PEG-2-e-23 B-PEG-2-e-22... B-PEG-2-e-0)
vertex_list: (B-PEG-2-v-15 B-PEG-2-v-14... B-PEG-2-v-0)

Figure 21 Boundary Representation Structure of the Peg

Since VANTAGE uses the winged edge representation of the boundary for all objects, we can access all the topological and spatial information of every geometrical feature of an object. This information is key to finding contacts in our application.

4.1.3 Localization and Tracking

The stereo algorithm gives us the range images from the stereo algorithm for each observed scene. The next step is to localize the assembled objects in each scene. We do this by manually localizing the objects in the first scene approximately. We then feed this to a localization program to localize the objects in the first scene. Then we let the localization program to track the objects in the remaining scenes automatically.

The 3D localization program is called 3D template matching algorithm (3DTM) [51]. The 3DTM can localize an object in a 3D scene, given a rough estimate of the location of the object in the scene. The algorithm uses sensor modeling, probabilistic hypothesis generation and robust localization techniques to make localization fast and accurate. The input to 3DTM is a range image and the triangulated model of the object in the scene and an approximate location of the object in the scene. The 3DTM uses the triangulated model and the approximate location to compute an error using a special error function. It then deferentially adjusts the location of the model to minimize the error. The definition of this error function is such that it is very robust to noise and occlusion which makes it well suited for our APO.

The original 3DTM program can localize one object in one scene. We modified the 3DTM program to localize multiple objects. We then set up the system to feed the position information of the objects in one scene to initialize the localization in the next scene. This results in automatic tracking of the objects in all but the first scene. The sequence of observed poses of the peg during the peg in hole task is shown in Figure 22. The localized models of the peg and hole are superimposed as dark triangulated meshes on each of the

observed intensity images.

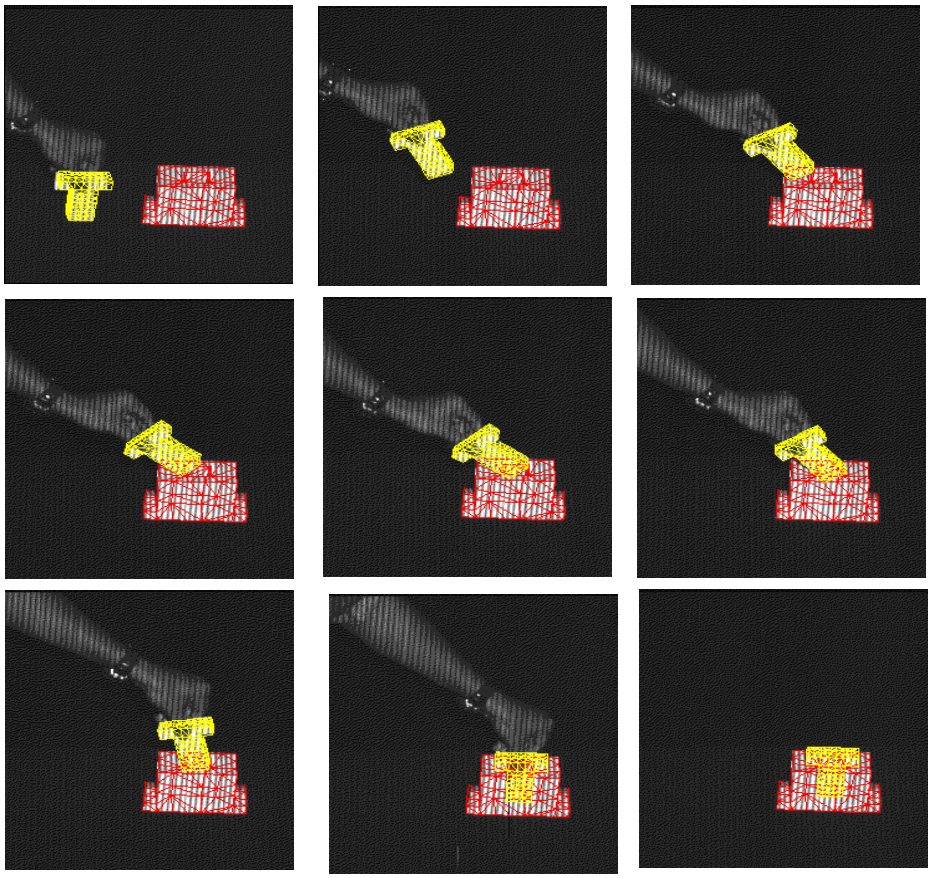


Figure 22 Object tracking (only 9 of the 70 scenes are shown here)

The result of tracking is a list of lists. Each list contains the pointer to the images of the scene, the number of localized models in the scene, their identity, and the their poses, with respect to a global coordinate system as shown in Figure 23. The pose of each object in a scene is given as $P\ (x,y,z,s_x,s_y,s_z,\alpha)$ in a global reference frame. The first three values are the translation parameters of the object coordinate system. The last four numbers is the rotation of the object coordinate system in axis-angle format. The output of the localization program for a single scene is a list of poses of the objects in the scene. The output of the tracking sys-

tem is a list of list of poses of the objects being tracked as shown in Figure 23.

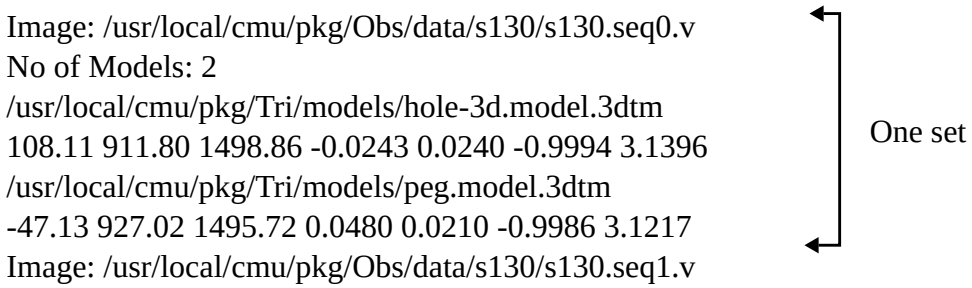


Figure 23 Output of Tracking system

It should be noted that the raw pose obtained from the observation system is error prone. Despite this, we can extract the essential contact information in each observation, and use it to reconstruct the path.

4.2 Modeling the Assembly Path

The task modeling system can be considered as a module which takes in the list of object locations at discrete instants of the assembly and uses it to reconstruct the path of the assembled object. The schematic of the algorithm is shown in Figure 24.

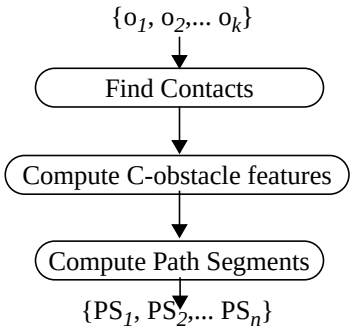


Figure 24 Main steps of the Algorithm

The steps of the algorithm are as follows:

- Instantiate the models of the assembled objects at each observation, o_i. Find the feature pairs which are in contact for each o_i of the assembled object with the environment. Let the contact feature pair set for the observed configuration o_i be FP_i.

- Compute the c-surface C_i corresponding to each observation o_i using the feature pair set FP_i as explained in section 5. Project the observed configuration o_i onto the c-surface to obtain the corrected configuration p_i.

- Segment the observed configurations $\{p_1, p_2, \ldots p_k\}$ into contiguous segments $\{S_1, S_2, \ldots S_n\}$ such that all observations in a segment S_j maintain the same contact pair set c_j and have the same c-surface.

$$S_j = \{p_{j_1}, p_{j_2}, \ldots, p_{j_k}\}$$
$$FP_{j_1} = FP_{j_2} = \ldots = FP_{j_k} = \ldots = c_j \qquad \text{(EQ60)}$$

- Compute the path segment PS_j lying on the c-space obstacle feature C_j through the observed configurations in the segment $S_j = \{p_{j_1}, p_{j_2}, \ldots, p_{j_k}\}$.

- The complete assembly motion path is then the concatenation of the path segments PS_i.

4.2.1 Computing Contacts

The first step in the module is to find the contacts between objects in each observed scene. The basic information extracted from the raw position information from the tracker is the set of contacts between the objects in each scene. As explained in section 4 there are only three types of contacts between polyhedral objects, namely *vf*, *fv* and *ee*. We say that there is a contact between two features if the distance between them is less than a threshold. The distance measures for the three types of contact are given below.

vf and fv contacts

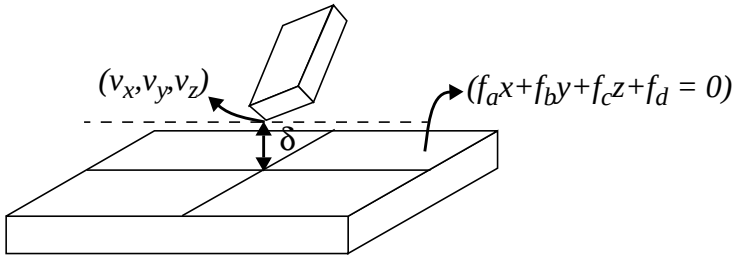


Figure 25 Contact condition for vf contact

If the coordinates of the vertex is (v_x,v_y,v_z) and the equation of the plane of the face is $(f_a x+f_b y+f_c z+f_d = 0)$, then the distance between the features is the perpendicular distance of the vertex to the face is given by (EQ61). If this distance is within a threshold then we say there is a contact.

$$e_a v_x + e_b v_y + e_b v_y + e_b v_y < \delta_{v_f} \qquad e_a^2 + e_b^2 + e_c^2 = 1 \tag{EQ61}$$

In addition to the distance threshold, we check if the projection of the vertex lies within the boundaries of the face. The contact condition for *fv* contact is the same as *vf* contacts. The only difference is the face is moving and the vertex is fixed.

ee contact

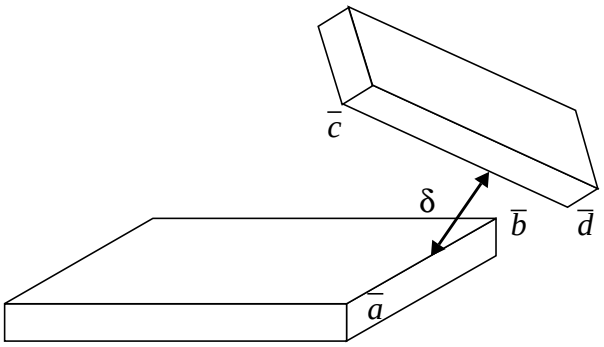


Figure 26 Projection for single ee contact

With edge on edge contacts the distance between two edges is the perpendicular distance between the two lines forming the edge. If the first edge has end points $\bar{a}$ and $\bar{b}$ and the second edge has end points $\bar{c}$ and $\bar{d}$, then the distance will be given as shown in (EQ62).

$$\begin{bmatrix}(\bar{b}-\bar{a}) & (\bar{c}-\bar{d})\end{bmatrix}\begin{bmatrix}d_1\\ d_2\end{bmatrix} = \begin{bmatrix}(\bar{c}-\bar{a})\end{bmatrix}$$
$$\delta = \left\|(\bar{a}+(\bar{b}-\bar{a})d_1)-(\bar{c}+(\bar{d}-\bar{c})d_2)\right\| \tag{EQ62}$$

In addition to the distance threshold, we check if the contact point lies within the bounding vertices of each edge.

Distance Threshold

The value of the distance threshold depends on the accuracy of the range data and the localization algorithm. It also depends on the scale of the objects used in the assembly. We use a two pass method for computing the contacts. In the first pass we use a 3mm threshold to compute contacts. Once we obtain the c-surface and project the observed configuration and correct it, we recompute the contacts using a threshold of 0.3mm.

In order to compute the contacts for each scene, we first instantiate the models of the assembled objects at the locations given by the 3DTM. We then compute the boundary representation of the objects in this model world. We then iterate through all the possible fea-

ture pairs between the assembled object and the objects in the environment and find the feature pairs satisfying the contact condition. Thus, for every observed scene and we end up with a set of feature pair sets in contact $\{c_{i1}, c_{i2},...\}$.

4.2.2 C-Surface Computation

For each observed scene, we are given the set of contacts between the moving object and the fixed objects. We choose a reference contact from this set of contacts. Then, we compute the three coefficient matrices $(E,D_1,D_2)_i$ for each of the other *(N-1)* contacts. The set of $(E,D_1,D_2)_i$ and the reference contact is the definition of the c-surface. Note that these coefficient matrices are dependent only on the geometry of the features in contact. So they can be pre-computed and stored as soon as the contacts are identified.

4.2.3 C-surface Projection

The projection of a point in c-space onto a c-surface depends on two functions. Every c-surface has two main methods associated with it. These are the space_to_surface method and the surface_to_space function.

Space_to_surface

The input to the function is a point in c-space in (x,y,z,θ,ϕ,ψ) coordinates. The output of space_to_surface is the values of the parameters defining the c-surface $(\bar{y},d_1,d_2)$.

Surface_to_space

The input to the surface_to_space function is a point on the c-surface and described by the $(\bar{y},d_1,d_2)$ parameters. The output of the function is the c-space point corresponding to these parameters in (x,y,z,θ,ϕ,ψ).

Given a point in c-space (x,y,z,θ,ϕ,ψ), we first send it to the space_to_surface function. The output of space_to_surface is then fed into the surface_to_space function. This is the projection of the observed configuration for the c-surface. This will be the corrected configuration of the assembled object in the scene.

Once we have the corrected configuration, we recompute the contacts. We repeat the contact computations and projections. The result of the second pass gives us the exact contacts and configuration of the assembled object in the scene. Because of the qualitative aspect involved in the computation of contacts these two passes are usually sufficient.

4.2.4 Segmenting the Observations

The previous step gives us the correct contact feature pair lists for each observed scene. We then use a simple set equivalence check to compute whether the contact features of two given observed scenes are equal. Using this, we segment the sequence of observations into contiguous segments such that all the contact feature pairs in all the scenes in a segment are the same.

4.2.5 Computing Path Segments

There is an important aspect of compliant motion paths lying on the c-space obstacle. If C_{i-1}, C_i and C_{i+1} are consecutive c-space obstacle features with finite path lengths. The start point of a path segment on C_i lies on the intersection of C_i and C_{i-1}. The end point of this path segment will lie on the intersection of C_i and C_{i+1}. These points are critical points in the path. We compute these start and end points of each path segment explicitly using a numerical technique.

The technique can be illustrated by considering a simple 2D case as shown in Figure 27. Let the last observation corresponding to the c-surface C_i be o_i, and let C_{i+1} be the c-surface corresponding to the next contact configuration. We want to compute the intersection q of the two c-surfaces C_i and C_{i+1} closest to o_i.

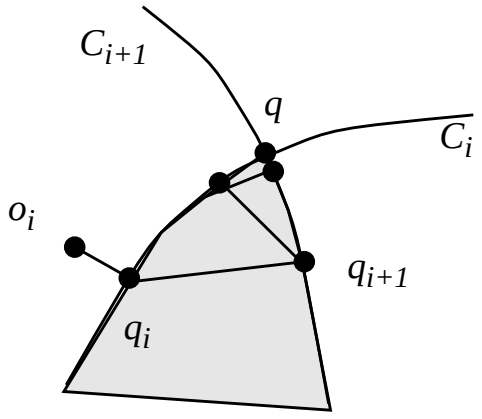


Figure 27 Intersection of two path segments

We find the intersection by successively projecting the observed point onto C_i and the C_{i+1} until the point converges. The procedure is as follows.

1. Project o_i onto C_i to obtain the projected point q_i.

2. Project q_i onto C_{i+1} to obtain a projected point q_{i+1}.

3. If $|q_i - q_{i+1}| < \varepsilon$ (threshold), return q_{i+1}; else set $o_i = q_{i+1}$ and go to step 1.

The procedure converges because the initial configuration o_i is usually close to the intersection point of two c-surfaces. An advantage of this technique is that it can find intersections of path segments lying on different classes of c-surfaces.

We compute the end points of all the path segments using this procedure. The start points of a path segment coincides with the end point of the previous path segment. The path segment is then obtained by interpolating the $(\bar{y},d_1,d_2)$ parameters from the start point $(\bar{y},d_1,d_2)_i$ to the end point $(\bar{y},d_1,d_2)_f$ through all the observed configurations with the same contacts $S_j = \{o_{j_1}, o_{j_2}, \ldots, o_{j_k}\}$.

Once we have the path segment on the c-space obstacle feature C_j, any point on this curve will maintain the contacts c_j. This will be the path segment PS_j.

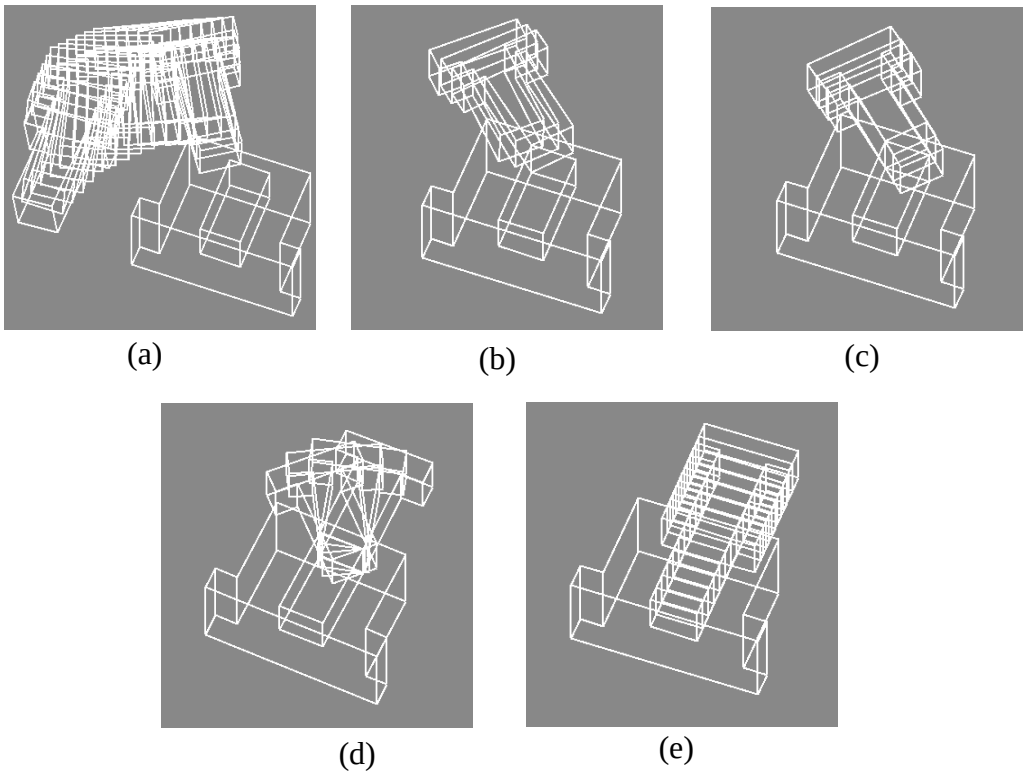


Figure 28 Path segments in task

The main path segments of the assembly task shown in Figure 22 is shown in Figure 28.

The path segment shown in Figure 28 (a) corresponds to the path of the peg in free space. This is considered special because no contacts exist. This path usually avoids obstacles. Hence unlike other path segments, this path is interpolated between all observed configuration points. The path segment shown in Figure 28 (b) corresponds to the a double *vf* contact that is made initially. The segment following that in Figure 28 (c) corresponds to the multiple *vf* and *fv* contacts once the peg enters the hole. The remaining path segments complete the rest of the assembly. The concatenated assembly path corresponds to the motion of the assembled peg as shown in Figure 29.

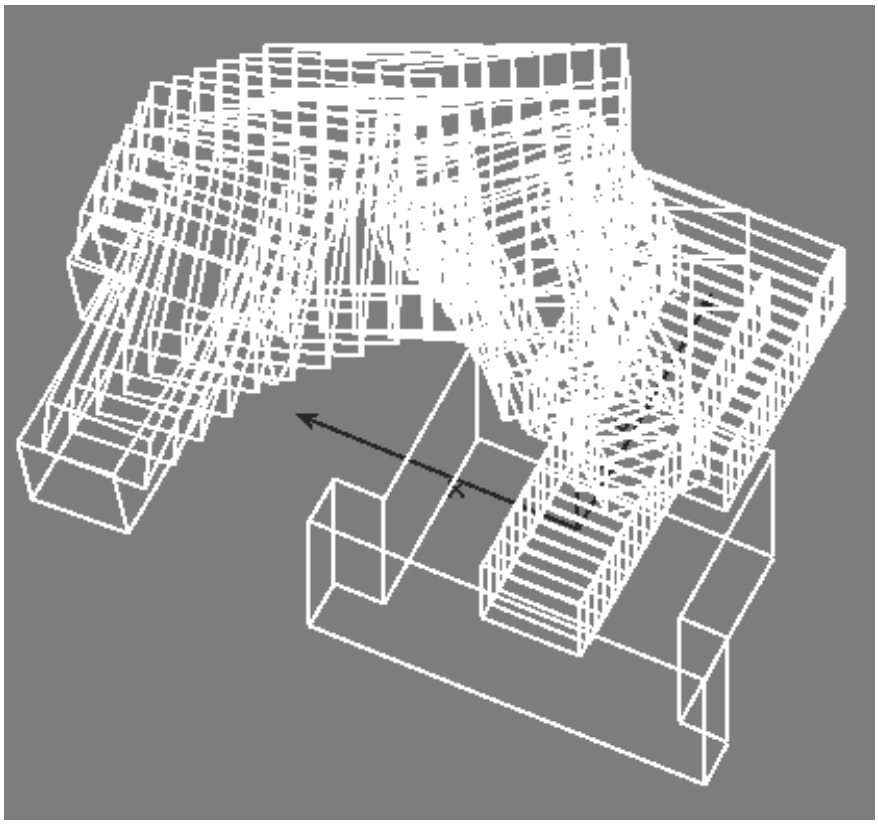


Figure 29 Path of the peg in Vantage

4.3 Robot Execution

The compliant path reconstructed from the observation is a nominal assembly path. In order to robustly execute the assembly task using a robot, we need a force-position control strategy. But, if we have a very accurate robot and are given the position of the objects in the robot space, then we can still use the motion plan to generate a trajectory for the robot.

The transformations involved in converting the modeled path to a robot trajectory is shown in Figure 30.

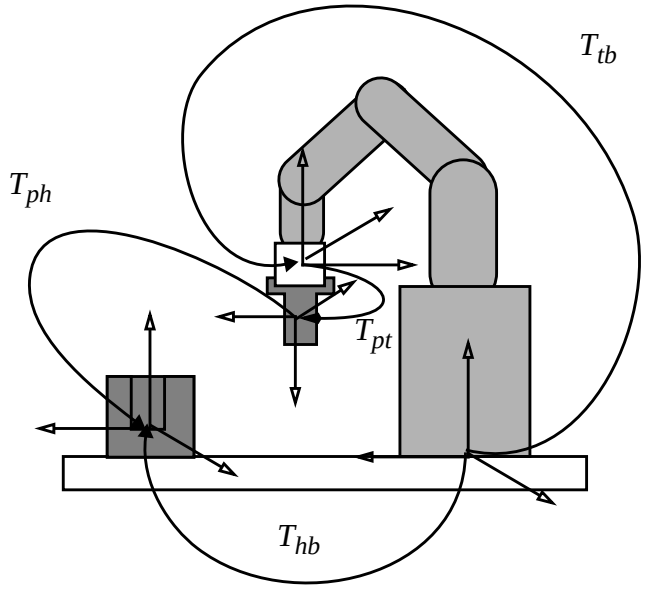


Figure 30 Transformations in the Robot Space

If T_{hb} is the transformation matrix representing the hole with respect to the base of the robot. T_{ph} is the transformation matrix obtained from the modeled path. The matrix T_{pt} is the transformation between peg to the tool of the robot. Finally T_{tb} is the transformation of the tool with respect to the robot base. Then the robot transformation matrix T_{tb} is given by (EQ63).

$$T_{tb} = T_{tp} T_{ph} T_{hb} \qquad \text{(EQ63)}$$

We are given a series of T_{ph} matrices defining the assembly path of the peg into the hole. We can compute the series of transformation matrices T_{tb}. Each of these T_{tb} can be converted to the coordinates of the robot and will form the trajectory of the robot. The exe-

cution stages of the robot for the assembly task is shown in Figure 31.

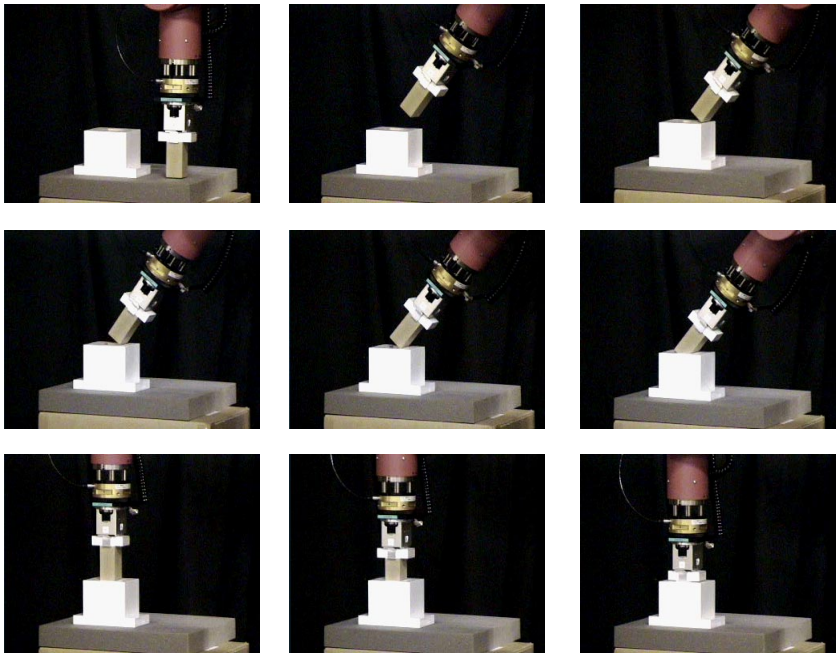


Figure 31 Fujitsu Arm Execution

5 Conclusions

Our theory took shape from its motivation, namely the APO system. We believe that the basic methods we proposed are general enough to apply in other domains. In this section we summarize our results, discuss the limitations and suggest future directions of research.

5.1 Summary

The method we proposed for representing c-surfaces is general. The generality means that it can represent any given set of contacts. It is also general in the sense that it works for both polygonal objects in planar space and polyhedral objects in space. Using this representation, we showed how we can compute the projection of a point in c-space onto the c-surface. The interpolation of a smooth path between two points on a c-surface is then an extension of the projection technique. The motivation for our theory was the assembly plan from observation system. We demonstrated our theory by implementing the APO system.

5.2 Discussion

The biggest limitation of our projection method is the lack of an analytical solution for the system of non homogenous polynomial equations which form the constraints. We showed that for special cases the constraints reduce to a linear system of equations. But an analytical solution would be the perfect answer to the problem. The iterative solution to find the projection is not only less elegant but could take more time if the projections on the c-surfaces are needed for real time task execution.

The 3 mm distance threshold we used for finding contacts between objects in our implementation is a “magic number” which depended on the dimensions and errors in our vision system. One can easily find objects which can be shaped such that the contacts concluded from this threshold will result in a infeasible set of contacts. The remedy is to replace the

current simplistic contact finding function with a better framework which can reason about the contacts.

The displacement of objects can be also represented using the transformation matrix *(T)*. The *T* matrix possess the group properties of dual vectors which we used to great advantage. But the major limitation is the number of elements in a *4x4 T* matrix is *12* compared to *8* in the dual quaternion. (*6* in the *3x3* matrix compared to *4* in the planar quaternion). Unlike the single constraint on the rotation part of the dual quaternion, the constraints on the elements of *T* matrices are six in the 3D case (*2* in the planar case). Solving for the c-surface analytically among these myriad constraints will require complex symbolic mathematics software such as the one used in Donald's work [9]. On balance we think that the dual vector representation is less complex.

5.3 Directions for Future Research

The application of a general and computationally manageable representation of c-surfaces is enormous in the field of motion planning. The future directions of research lie not only in the application of our theory, but also in the investigation of the possibility of direct solutions of the non-linear equations which form the constraints.

The obvious application of the theory of c-surfaces would be to use it on-line during the execution phase of the robot. The nominal path reconstructed from the observations is not sufficient for guaranteeing success during execution because of uncertainty during robot execution. The actual c-surface can be computed from the position of the objects obtained from vision sensors in the robot workspace during execution. The differentials of our error measure in our projection technique provides a good estimate of the error control vector on the robot trajectory.

Mason et al. [36] work details the compliant-guarded motion strategy to reliably execute given an assembly plan. Further work by Erdmann et al. [10] incorporated additional parameters like friction into this strategy. Future work on the execution module of the APO could incorporate our c-surface representation into this approach to guarantee the success of the assembly task even in the presence of uncertainty.

As we explained in the representation sections of this thesis, the set of three coefficient matrices (E,D_1,D_2) decide the independence of the rotation and translation parameters of the c-surface. The skew symmetry of all D_1 matrices makes the d_1 parameter independent. There might be a good chance that there are individual or aggregate properties of the matri-

ces which can decide the characteristics of the c-surface. Canny proposed an analytical solution of a system of homogenous polynomial equations. A similar solution may be possible for the non-homogenous equations in our case.

The basic contacts involved pairs of primitive geometrical features of the polygonal and polyhedral objects such as face, edge, and vertex. If we can define a set of such features for curved objects, it should be possible to derive the representation in dual quaternions. Admittedly they will be much more complex that the simple parametric and implicit equations for the case we dealt with in this thesis. Solving this problem will probably be a big step towards analyzing real world objects used in assembly.

Appendix A : Projection Formula

The minimization problem we are concerned with is given by (EQ64).

$$\underbrace{Min\left(Q = \sum_{2}^{N-1}\left(\bar{y}^T\left[M(d_1, d_2)\right]_i\bar{y}\right)^2 + \lambda(\bar{y}^T\bar{y} - 1)\right)}$$

$$\begin{aligned}
\frac{\partial}{\partial \bar{y}}Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial d_1}Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial d_2}Q(\bar{y}, d_1, d_2, \lambda) &= 0 \\
\frac{\partial}{\partial x}Q(\bar{y}, d_1, d_2, \lambda) &= 0
\end{aligned} \qquad \text{(EQ64)}$$

The solution to (EQ64) in (y,d_1,d_2,λ) space can be obtained by setting the partial differentials to zero and solving the resulting equations. Since the problem is non-linear we solve it iteratively by using the Newton-Raphson's method as shown in (EQ65).

$$\begin{bmatrix} \bar{y} \\ d_1 \\ d_2 \\ \lambda \end{bmatrix}_{k+1} = \begin{bmatrix} \bar{y} \\ d_1 \\ d_2 \\ \lambda \end{bmatrix}_k - \left(Q''\big|_k\right)^{-1} Q'\big|_k$$

$$\dot{Q}\big|_k = \left[\begin{matrix} \frac{\partial Q}{\partial \bar{y}} \\ \frac{\partial Q}{\partial d_1} \\ \frac{\partial Q}{\partial d_2} \\ \frac{\partial Q}{\partial \lambda} \end{matrix}\right]_{(\bar{y}_k,\, d_{1_k},\, d_{2_k},\, \lambda_k)} \qquad \ddot{Q}\big|_k = \left[\begin{matrix} \frac{\partial^2 Q}{\partial \bar{y}^2} & \frac{\partial^2 Q}{\partial \bar{y}\partial d_1} & \frac{\partial^2 Q}{\partial \bar{y}\partial d_2} & \frac{\partial^2 Q}{\partial \bar{y}\partial \lambda} \\ \frac{\partial^2 Q}{\partial \bar{y}\partial d_1} & \frac{\partial^2 Q}{\partial d_1^2} & \frac{\partial^2 Q}{\partial d_1\partial d_2} & \frac{\partial^2 Q}{\partial d_1\partial \lambda} \\ \frac{\partial^2 Q}{\partial \bar{y}\partial d_2} & \frac{\partial^2 Q}{\partial d_1\partial d_2} & \frac{\partial^2 Q}{\partial d_1^2} & \frac{\partial^2 Q}{\partial d_2\partial \lambda} \\ \frac{\partial^2 Q}{\partial \bar{y}\partial \lambda} & \frac{\partial^2 Q}{\partial d_1\partial \lambda} & \frac{\partial^2 Q}{\partial d_2\partial \lambda} & 0 \end{matrix}\right]_{(\bar{y}_{k+1},\, d_{1_{k+1}},\, d_{2_{k+1}},\, \lambda_{k+1})} \qquad \text{(EQ65)}$$

The first order partial differential terms are given in (EQ66).

$$\frac{\partial Q}{\partial \bar{y}} = \sum_{2}^{N} 2\left(\bar{y}^T \left[M(d_1, d_2)\right]_i \bar{y}\right)\left(\left[M(d_1, d_2)\right]_i + \left[M(d_1, d_2)\right]_i^T\right)\bar{y} + \lambda \bar{y}''$$

$$\frac{\partial Q}{\partial d_1} = \sum_{2}^{N} 2\left(\bar{y}^T \left[M(d_1, d_2)\right]_i \bar{y}\right)\left(\bar{y}^T \left[\frac{\partial}{\partial d_1} M_i(d_1, d_2)\right]\bar{y}\right) \quad \text{(EQ66)}$$

$$\frac{\partial Q}{\partial d_2} = \sum_{2}^{N} 2\left(\bar{y}^T \left[M(d_1, d_2)\right]_i \bar{y}\right)\left(\bar{y}^T \left[\frac{\partial}{\partial d_2} M_i(d_1, d_2)\right]\bar{y}\right)$$

$$\frac{\partial Q}{\partial \lambda} = \bar{y}^T \bar{y} - 1$$

The second order partial differential equations are given in (EQ67).

$$\frac{\partial^2 Q}{\partial \bar{y}^2} = \sum_2^N 2\left(\bar{y}^T\left[M(d_1,d_2)\right]_i\bar{y}\right)\left(\left[P(d_1,d_2)\right]_i\right)+\sum_2^N 2\left(\left[P(d_1,d_2)\right]_i\bar{y}\right)\left(\bar{y}^T\left[P(d_1,d_2)\right]_i\right)+2\lambda I$$

$$\frac{\partial^2 Q}{\partial \bar{y}\partial d_1} = \sum_2^N 2\left(\bar{y}^T\left[M(d_1,d_2)\right]_i\bar{y}\right)\left(\left[P_x(d_1,d_2)\right]_i\bar{y}\right)+\sum_2^N 2\left(\bar{y}^T\left[P_x(d_1,d_2)\right]_i\bar{y}\right)\left(\left[P(d_1,d_2)\right]_i\bar{y}\right)$$

$$\frac{\partial^2 Q}{\partial \bar{y}\partial d_2} = \sum_2^N 2\left(\bar{y}^T\left[M(d_1,d_2)\right]_i\bar{y}\right)\left(\left[P_z(d_1,d_2)\right]_i\bar{y}\right)+\sum_2^N 2\left(\bar{y}^T\left[P_z(d_1,d_2)\right]_i\bar{y}\right)\left(\left[P(d_1,d_2)\right]_i\bar{y}\right)$$

$$\frac{\partial^2 Q}{\partial \bar{y}\partial\lambda} = 2\bar{y}$$

$$\frac{\partial^2 Q}{\partial d_1^2} = \sum_2^N 2\left(\bar{y}^T\left[P_x(d_1,d_2)\right]_i\bar{y}\right)^2$$

$$\frac{\partial^2 Q}{\partial d_1\partial d_2} = \sum_2^N 2\left(\bar{y}^T\left[P_x(d_1,d_2)\right]_i\bar{y}\right)\left(\bar{y}^T\left[P_z(d_1,d_2)\right]_i\bar{y}\right) \quad \text{(EQ67)}$$

$$\frac{\partial^2 Q}{\partial d_2^2} = \sum_2^N 2\left(\bar{y}^T\left[P_z(d_1,d_2)\right]_i\bar{y}\right)^2$$

$$\frac{\partial^2 Q}{\partial d_1\partial\lambda} = 0$$

$$\frac{\partial^2 Q}{\partial d_2\partial\lambda} = 0$$

$$\left[P(d_1,d_2)\right]_i = \left[M(d_1,d_2)\right]_i + \left[M(d_1,d_2)\right]_i^T$$

$$\left[P_x(d_1,d_2)\right]_i = \left[D_x(d_1,d_2)\right]_i + \left[D_x(d_1,d_2)\right]_i^T$$

$$\left[P_z(d_1,d_2)\right]_i = \left[D_z(d_1,d_2)\right]_i + \left[D_z(d_1,d_2)\right]_i^T$$

Bibliography

[1] Asada, H, By, A. B. “Kinematic Analysis of Workpart Fixturing for Flexible Assembly with Automatically Reconfigurable Fixtures”, *IEEE Journal of Robotics and Automation*. Vol. RA-1, No. 2, pages 86-94, June 1985.

[2] Avnaim, F. Boissonnat, J. D, and Faverjon, B. “A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles”. *IEEE International Conference. on Robotics and Automation,* Vol. 3, pages 1656-1661, 1988.

[3] Bajaj, C. Kim, M-S. “Generation of Configuration Space Obstacles: Moving Algebraic Surfaces”. *The International Journal of Robotics Research*, Vol. 9. No. 1, pages 92-112. Feb 1990.

[4] Ball, T. S. “A Treatise on the Theory of Screws”. Cambridge University Press (1900).

[5] Balakumar, J. C. Robert, R. H. Ikeuchi, K. and Kanade, T. “Vantage: A Frame-based Geometric/Sensor Modeling System - Programmer/User’s Manual v1.0”. Technical Report, Carnegie Mellon University, Robotics Institute, 1989.

[6] Brost, R. C. “Analysis and Planning of Planar Manipulation Tasks”. *Ph.D. Thesis*, CMU-CS-91-149, pages 96-103, 264-268, 1991.

[7] Canny, J. F. “The Complexity of Robot Motion Planning”. *Ph.D. Thesis*, MIT Press, ACM Doctoral Dissertation Award, 1987.

[8] Craig, J. J. “Introduction to Robotics: Mechanics and Control”. Addison-Wesley, Reading, Mass. 1986.

[9] Donald, R. B. “On Motion Planning with Six Degrees of Freedom: Solving the Inter-

section Problems in Configuration Space". *IEEE International Conference. on Robotics and Automation*, pages 536-541, 1985.

[10] Erdmann, M. A. Mason, M. T. "An Exploration of Sensorless Manipulation". *IEEE Journal of Robotics and Automation*. Vol. 4, pages 369-379, Aug 1988.

[11] Finkel, R. Taylor, R. Bolles, R. P. and Feldman, J. "AL, A Programming System for Automation". Technical Report AIM-177, Stanford University, Artificial Intelligence Laboratory, Stanford, CA, 1974.

[12] Ge, Q. and McCarthy, J. M. "Equations for Boundaries of Joint Obstacles for Planar Robots". *IEEE International Conference on Robotics and Automation*, Vol. 1, pages 164-169, 1989.

[13] Ge, Q. and McCarthy, J. M. "An Algebraic Formulation of Configuration-Space Obstacles for Spatial Robots". *IEEE International Conference on Robotics and Automation*, Vol. 3, pages 1542-1547, 1990.

[14] Hirai, S. and Sato, T. "Motion Understanding for World Model Management of Telerobot". In *IEEE International Conference on Intelligent Robots and Systems*, Vol. 1, pages 124-131, 1989.

[15] Hirai, S, Asada, H. "Kinematic and Statics of Manipulation Using the Theory of Polyhedral Convex Cones on Networks". *The International Journal of the Robotics Research*, Vol. 12, no 5, pages 434-447, 1993.

[16] Hirukawa, H. Matsui, T. Takase, K. "A General Algorithm for Derivation and Analysis of Constraint for Motion of Polyhedra in Contact". *IEEE International Conference on Intelligent Robots and Systems,* Vol. 1, pages 38-43, 1991.

[17] Hopcroft and Wilfong. "Reducing Multiple Objects Motion Plan to Graph Search". Technical Report TR-84-616, CS Cornell University, 1984.

[18] Hwang, Y. K. "Boundary Equations of Configuration Obstacles for Manipulators". *IEEE International Conference on Robotics and Automation*, Vol. 1, pages 298-303, 1990.

[19] Hunt, K. H. "Kinematic Geometry of Mechanisms". Clarendon, Oxford, 1978.

[20] Inaba, M. and Inoue, H. “Visual-based Robot Programming”. *International Symposium on Robotics Research*, pages 129-136, 1989.

[21] Jiar, Y. Wheeler, M.D. and Ikeuchi, K. “Hand Action Perception for Robot Programming”. *IEEE International Conference on Intelligent Robots & Systems*, Vol. 3, pages 1586-1593, Nov 1996.

[22] Kang, S. B, Webb, J. A, Zitnick, C. L and Kanade, T, “An active multibaseline stereo system with real-time image acquisition”, Technical Report CMU-CS-94-167, 1994.

[23] Kang S. B. and Ikeuchi, K. “Towards Automatic Robot Instruction from Perception: Recognizing a Grasp from Observation”. *IEEE Transactions on Robotics and Automation*, 1993, Vol. 9, no 4, pages 432-443.

[24] Koutsou A. “A Method for Motion of a Moving Object in Contact with Fixed Objects” *Proceedings of the first ACM Symposium on Computational Geometry*, June 1985.

[25] Kuhn, H. W. and Tucker, A. W. editors, “Linear Inequalities and Related Systems”. *Annals of Math Studies*, Vol. 39, Princeton, 1956.

[26] Kuniyoshi, Y. Inoue, H, Inaba, M. “Design and Implementation of a System that Generates Assembly Programs from Visual Recognition of Human Action Sequences”. *IEEE International Workshop on Intelligent Robots and Systems*. pages 567-574, 1990.

[27] Ikeuchi, K., Kawade, M.; Suehiro, T. “Toward Assembly Plan from Observation - Task recognition with Planar, Curved and Mechanical contacts”. *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Yokohama, Japan, Vol. 3, pages 2294-2301, July 1993.

[28] Ikeuchi, K. and Suehiro, T. “Towards an Assembly Plan from Observation, part I: Task Recognition with Polyhedral Objects”. In *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 3, pages 368-385, June 1994.

[29] Lammineur, P. and Cornillie, O. “Industrial Robots”. Pergammon Press, pages 43-54, 1984.

[30] Laugier, C. “Planning fine motion strategies by reasoning in the contact space”.

IEEE International Conference on Robotics and Automation, Vol. 1, pages 653-659, 1989.

[31] Lieberman, L. L. and Wesley, M. A. “Autopass: an Automatic Programming System for Computer Controlled Mechanical Assembly”. *IBM Journal of Research and Develop*ment. 21(4):321-333, 1977.

[32] Lozano-Perez, T. “Automatic Planning of Manipulator Transfer Movements”. *IEEE Transactions on System Man and Cybernetics*, SMC-11(10): pages 681-689, 1981.

[33] Lozano-Perez, T. Mason, M. T. and Taylor, R. H. “Automatic Synthesis of Fine Motion Strategies for Robots”, *International Journal of Robotics Research*, Vol.3, No 1, pages 3-24, 1984.

[34] Lozano-Perez, T. Automatic Planning of Manipulator Transfer Movements. *IEEE Transactions on System Man and Cybernetics*, SMC-11(10):681-689, 1981.

[35] McCarthy, J. M. “An Introduction to Theoretical Kinematics”, *The MIT Press*, Chapter 4, pages 53-65, 1990.

[36] Mason, M. T. “Compliance and force control for computer controlled manipulators” *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 11, no. 6, pages 418-432, 1981.

[37] Mattikalli, R. S. and Khosla, P. K. “Motion Constraints from Contact Geometry: Representation and Analysis”. *IEEE International Conference on Robotics and Automation*, Vol. 3, pages 2178-2185, 1992.

[38] Miura, J. Ikeuchi, K. “Task Oriented Generation of Visual Sensing Strategies”. *IEEE International Conference on Computer Vision*, pages 1106-1113, June 1995.

[39] Montana, D. J. “The Kinematics of Contact and Grasp”, *The International Journal of Robotics Research*. Vol. 7, No. 3, pages 17-32, June 1988.

[40] Ohwovoriole, M. S. and Roth, B. “An Extension of Screw Theory”. *Journal of Mechanical Design*, Oct 1981.

[41] Paul, G. V. and Ikeuchi, K. “Partitioning Contact Space using the Theory of Polyhe-

dral Convex Cones". *IEEE International Conference on Robotics & Automation*, Vol. 1, pages 421-426, May 1995.

[42] Paul, G. V. and Ikeuchi, K. "Modeling Planar Assembly Paths from Observation". *IEEE International Conference on Intelligent Robots & Systems*, Vol. 2, pages 520-525, Nov 1996.

[43] Paul, G. V. and Ikeuchi, K. "A Quasi-Linear Method for Computing and Projecting onto C-Surfaces: Planar Case". *IEEE International Conference on Robotics & Automation*, Vol. 3, pages 2032-2027, 1997.

[44] Popplestone, R. J. Ambler, A. R and Bellos, I. "An Interpreter for a Language for Describing Assemblies". *Artificial Intelligence*, 14(1), pages 79-107, 1980.

[45] Press, W. H. Flannery, B. P. Teukolsky, S. A. and Vetterling, W. T. "Numerical Recipes in C". *Cambridge University Press*, pages 60-71, 1990.

[46] Silwa, N. O. and Will, R. W. "A Flexible Telerobotic System for Space Operations". In *Proceedings of Space Telerobotics Workshop*, pages 285-292, Pasadena, 1987.

[47] Strang, G. "Introduction to Applied Mathematics". *Wellesley-Cambridge Press*, pages 373-380, 1990.

[48] Suehiro, T. and Ikeuchi, K. "Towards an Assembly Plan from Observation, part II. Correction of Motion Parameters Based on Face Contact Constraints". *IEEE International Conference on Intelligent Robots & Systems*, Vol. 3, pages 2095-2102, July 1992.

[49] Suehiro, T. and Takase, R. "Skill Based Manipulation System". *Journal of the Robotics Society of Japan*, 8(5), pages 47-58, October 1990. (in Japanese).

[50] Shoemake, K. "Animating Rotations with Quaternion Curves". Computer Graphics, *Proceedings of the SIGGRAPH*, 19(3), pages 245-254, July 1985.

[51] Wheeler, M. D and Ikeuchi, K. Sensor Modeling, "Probabilistic Hypothesis Generation, and Robust Localization for Object Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol. 17, no 3, pages 252-265, Mar, 1995.

[52] Wilson, R. H. and Matsui, T. “Partitioning an Assembly for Infinitesimal Motions in Translation and Rotation”. Proceedings of the 1992 *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2, pages 1311-1318, Raleigh, NC, July 1992.

[53] Whitney, D. E. “State Space Models of Remote Manipulation Tasks”. In *Proceedings of 1st International Conference on Artificial Intelligence*, pages 495-507, 1969.