Modeling Planar Assembly Paths from Observation

George V Paul and Katsushi Ikeuchi
The Robotics Institute,
Carnegie Mellon University,
Pittsburgh, PA 15213.

Abstract
This paper describes a system for obtaining the motion plan for a planar assembly task, given a sequence of observations of a human performing it. The motion plan in configuration space is a series of connected path segments lying outside and on the configuration space obstacle. We use the observed configurations of the assembled objects to selectively compute the features of the c-space obstacle on which the path lies. We project the observed configurations onto these features and reconstruct the path segments. The connected path segments form the model of the observed task and can be used to program a robot to repeat the task. We demonstrate the system using the planar peg in hole task.

1 Introduction

Automatic planning of robots for assembly tasks has been attempted by a number of researchers[7][8][3]. This approach is computationally expensive. Another approach is to program robots for assembly tasks by demonstrating the task [5][10][12]. Our assembly plan from observation (APO) system is an example of the latter approach. Automatic planning typically involves the combinatronics of computing all possible contacts to build the configuration space (c-space) obstacle, and then searching for a feasible path in c-space. Unlike that approach, our system computes only the features of the c-space obstacle which are relevant to the observed assembly task. In addition, our system reconstructs a feasible path, instead of searching for one.

We use a real-time vision system to record the assembly task. We then recognize and track the parts in each observed scene. Using the position and geometry of the objects we find the contacts between the objects in each observed scene. We identify the distinct contact configurations involved in the assembly task. Next, we analytically compute the feature on the c-space obstacle corresponding to each distinct contact configuration. We then use the observed configurations to reconstruct the path segments lying on each computed c-space obstacle feature. These connected path segments constitute the model of the assembly task. Finally, we use the modeled path to program a robot to repeat the assembly task.

Figure 1 illustrates the theory behind this work. Consider a series of observed 2D scenes of the dark rectangle coming into contact with a large rectangle as shown in Figure 1(a). If we consider only 2D translation, the c-space obstacle will be the shaded rectangle as shown in Figure 1(b). Instead of computing this complete c-space obstacle, we find the contacts made in each observed scene and compute only the necessary features of the c-space obstacle, C1 and C2, which correspond to the contacts. The observations can then be corrected by projecting onto their corresponding c-obstacle feature. We also find the intersections between adjacent path segments explicitly. The motion path is then a series of path segments s0, s1, and s2 as shown in Figure 1(c).

Figure 1 Path in c-space for 2D (translation only).

The compliant motion plan obtained can be directly used to program a relatively accurate robot system. In the presence of uncertainties a fine motion plan can be derived from it using techniques such as in [4][14].

This work demonstrates a working system which can reconstruct the motion plan of a planar assembly task from observations of the task. We also propose a numerical technique to find the local intersection of path segments lying on adjacent c-space obstacle features.

2 Related Work

Previous work on the APO system[5][10] used sparse observations of the assembly task. The system then computed the contact state of the object at the beginning and end of the task. The motion plan was then obtained by searching a transition graph of contact states. This approach is sufficient for most simple assembly tasks, but it cannot recover the compliant motion in the case of assembly tasks such as peg in hole insertion. Our new approach tries to extract the compliant motion of assembly tasks by observing the task continuously. We accomplish this by adapting some of the results of the configuration space approach to automatic planning.

The configuration space approach for automatically planning the fine motion paths for assembly tasks was proposed by Lozano-Perez[7][8], Mason[9] and others. The idea is to compute the c-space obstacle and plan the path in c-space. Computing the c-space obstacle is crucial for assembly tasks where there is contact between the assembled
object and the environment. Work such as by Bajaj et al [2]
deal with algebraically computing the c-space obstacle, but
is limited to translational motion only. Other work such as
by Avnaim et al [1], and Brost [3] deal with computing the
c-space obstacles in the planar case with both rotation and
translation. We use Brost’s results in our work.

3 Observation of the Assembly Task

The compliant motion used by the human performing
the assembly, consists of relatively small but significant motion
when objects are in contact. The APO system captures this
information using a real time stereo system[6]. The stereo
system gives us a sequence of dense and accurate 3D images
of the scene. The assembled object is then identified and
tracked throughout the range image sequence using the
3D template matching algorithm[13]. A sample of nine images
from a sequence sixty four images of a peg in hole as-
sembled task is shown in Figure 2. The models of the
localized objects are overlaid on each image. The output
of the observation system is a list of poses (x,y,z,θ,ϕ,ψ) of
the assembled parts in each observed scene.

Figure 2 Object tracking (only 9 of 64 observed scenes
are shown here)

The APO system uses the geometrical modeler called
Vantage to create scenes of the task by instantiating models
of the assembly objects at their observed poses. Since we are
considering only planar assembly tasks, we project these
models onto the xy plane.

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

4 Assembly Contacts and C-Surfaces

The key information in an observed scene of an assembly
task is the contacts that are made between the assembled
object and the environment. This involves identifying the
feature pairs which are in contact.

4.1 Assembly Contacts

Each feature pair in contact involves a feature of the as-
sembled object (vertex and edge for polygonal objects) and
a feature of the objects in the world as shown in Figure 3.

Figure 3 Types of contact pairs

Given the pose of the assembled object and the objects
in the world, we can obtain the geometry of each feature of
the assembled object and that of the other objects in the en-
vironment from the geometrical modeler, Vantage. The ex-
istence of contact between each feature pair is then
computed using one of the following contact conditions.

For vertex-on-edge (ve) and edge-on-vertex (ev) contacts,
if the vertex coordinates are (vₓ,vᵧ) and the edge equation
is eₓx + eᵧy + eₖ = 0, then the primary condition
for the pair to be in contact is given by (1).

\[ eₓvₓ + eᵧvᵧ + eₖ < δ_{ve} \]

In addition, we check if the projection of the vertex lies
within the end points of the edge.

For vertex-on-vertex (vv) contacts if the two vertex co-
ordinates are (vₓ,vᵧ) and (vₓ,vᵧ), then the condition for
the pair to be in contact will be given by (2).

\[ \sqrt{(vₓ - vₓ)^2 + (vᵧ - vᵧ)^2} < δ_{vv} \]

For edge-on-edge (ee) contacts, if the edge equations
are eₓₓx + eᵧᵧy + eₖ = 0 and eₓₓx + eᵧᵧy + eₖ = 0, then the
primary condition for the pair to be in contact will be given
by (3).

\[ 1 - (eₓₓx^2 + eᵧᵧy^2) < δ_{ee1} \land (eₓₓx - eᵧᵧy < δ_{ee2}) \]

In addition, for ee contacts we check if the edge seg-
ments overlap.

4.2 Configuration Space Obstacle Features

By definition, the set of configurations of the assembled
object which keeps a set of its features in contact with a set
of features of another object will be a feature of a c-space
obstacle in the configuration space. The order of this feature
in c-space depends on the degree of contact. For example, a
single ve contact results in a two dimensional c-surface in
(x,y,0) space, whereas two simultaneous ve contacts corre-
spond to a one dimensional edge in c-space. Given the set of
feature pairs in contact and their geometry, we can algebra-
ically compute the infinite surface on which the feature of
the c-space obstacle will lie.

For polygonal objects in a plane, the features of the c-
space obstacles can be two dimensional facets, one dimen-

1. (eₓₓx, eᵧᵧy) is a 2d unit vector.
sional edges and zero dimensional vertices. There are two types of facets of configuration obstacles corresponding to single ve and single ev contacts. There are four classes of edges which correspond to contact configurations with more than one ve or ev contacts or when there are ee or vv contacts. The vertices of the c-space obstacle are defined by their coordinates.

We briefly introduce the representation of a two dimensional surface in c-space corresponding to a single vertex-on-edge (ve) contact here. The representations of the all other possible contact configurations can be found in Brost’s work [3].

![Figure 4](image)

**Figure 4** A single ve contact [3]

Consider two polygons in contact in a plane as shown in Figure 4. The vertex of the moving object is in contact with the edge of the fixed object. The configuration of the moving triangle is (x,y,θ). The configuration obstacle feature corresponding to this contact will be a subset of an infinite two dimensional surface embedded in (x,y,θ) space. This surface (c-surface) can be parameterized by two variables (p, θ). Brost defines p as a perimeter variable which is the distance of the contact vertex from the clockwise vertex of the contact edge as shown in Figure 5. The parameter θ is identical to the angle of the moving object.

![Figure 5](image)

**Figure 5** The parameter p [3]

The infinite c-surface can then be described in parametric form by (4), (5) and (6).

\[
x = f_x(p, \theta)
\]

\[
y = f_y(p, \theta)
\]

\[
\theta = f_\theta(p, \theta) = \theta
\]

The x and y functions for the ve contact are given by the equations (7) and (8). A example of the 2D c-surface corresponding to a ve contact is shown in Figure 8.

\[
f_{ve}(p, \theta) = k_{1x} + k_{2x}p + k_{3x}\cos(\theta + \kappa_{3x})
\]

\[
k_{1x} = x_f \quad k_{2x} = \cos(\beta_f) \quad k_{3x} = -r_m \quad \kappa_{3x} = v_m
\]

\[
\begin{align*}
  f_{ve}(p, \theta) &= k_{1y} + k_{2y}p + k_{3y}\sin(\theta + \kappa_{3y}) \\
  k_{1y} = y_f \quad k_{2y} = \sin(\beta_f) \quad k_{3y} = -r_m \quad \kappa_{3y} = v_m
\end{align*}
\]

\[
(8)
\]

There exists an inverse mapping of a configuration space point in (x,y,θ) space to a point on the c-surface defined by the (p, θ) variables. The form of these equations are given by (9) and (10).

\[
p = f_p(x, y, \theta)
\]

\[
\theta = f_\theta(x, y, \theta) = \theta
\]

The equation (9) for the ve contact is given by (11).

\[
e_{cw}^v = \begin{bmatrix} x + r_m\cos(\theta + v_m) - x_f \\ y + r_m\sin(\theta + v_m) - y_f \end{bmatrix}
\]

\[
p = f_p(x, y, \theta) = e_{cw}^v \cdot \hat{e}_f
\]

(11)

Given any configuration space point o (x,y,θ) and a c-surface C_p, the equations (9) and (10) will find the (p, θ) of the closest point t_i which lies on C_p. We use this property to correct the raw configurations obtained from the observation system. The coordinates of t_i (p, θ) can be converted back to (x,y,θ) using (4)-(6). The error in the observation will be the distance between o_i and t_i given by (12).

\[
E(o, C_p) = \|o - o_i\|
\]

(12)

The counterparts of the equations (4)-(6) and (9)-(10) exist for the ev facets and the four classes of edges. These are derived in [3]. The form of these equations are listed in the Appendix for reference.

### 4.3 C-surface Selection

For every observed configuration o_i, we obtain a set of feature pairs FP_i which satisfy the contact conditions (1)-(3). From the set FP_i we need to find the most likely contact configuration and c-surface corresponding to it. This c-surface can easily be obtained when the contact pairs are well defined as shown in Figure 6(a). But in certain configurations such as the one shown in Figure 6(b), due to the proximity of the object features, not all the contact pairs in the set FP_i will correspond to real contacts. The true contact and the c-surface in this case is obtained as follows.

![Figure 6](image)

**Figure 6** Ambiguous contacts

Given the feature pair set FP_i, we generate all combinations of contacts which can correspond to a c-surface. We then project the observed configuration onto these c-surfaces and find the corrected configuration in each case. The c-surfaces are mathematical constructs, hence there will al-
ways be a projected configuration. But these projected configurations may not be legal as it might cause collision of other features of the object with the environment. All contact pair combinations causing such collision are then discarded. Among those that remain we find the c-surface which has minimum error using equation (10). This is the c-surface which corresponds to the observed configuration.

4.4 Path Segments

The compliant motion of an assembly in c-space is composed of path segments lying on the c-space obstacle. All configurations in a path segment will maintain the same contacts and correspond to the same c-surface. The motion in a path segment will correspond to a smooth curve lying on this c-surface.

There is an important aspect of fine motion paths lying on the c-space obstacle, where \( C_{i+1}, C_{i}, \text{ and } C_{i+1} \) 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 \( C_{j} \) and \( C_{i+1} \). The end point of the same 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 7. Let the last observation corresponding to the c-surface \( C_{i} \) be \( q_{i} \), 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 \( q_{i} \).

![Figure 7 Intersection of two path segments](image)

**Figure 7 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 \( q_{i} \) onto \( C_{i} \) using equations (9)-(10) and obtain \( (p_{i}, \theta_{i}) \). Convert \( (p_{i}, \theta_{i}) \) back to \( (x_{p}, y_{p}, \theta_{i}) \) using equations (4)-(6). This is the projected point \( q_{i} \).
2. Project \( q_{i} \) onto \( C_{i+1} \) using equations (9)-(10) and obtain \( (p_{i+1}, \theta_{i+1}) \). Convert \( (p_{i+1}, \theta_{i+1}) \) back to \( (x_{i+1}, y_{i+1}, \theta_{i+1}) \) using equations (4)-(6). This is the projected point \( q_{i+1} \).
3. If \(|q_{i} - q_{i+1}| < \delta\) (threshold), return \( q_{i+1} \); else set \( q_{i} = q_{i+1} \) and go to step 1.

The procedure converges because the initial configuration \( q_{i} \) is close to the intersection point of two c-surfaces whose intersection exists. 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 \((p, \theta)\) parameters from the start point \((p_{i}, \theta_{i})\) to the end point \((p_{i}, \theta_{i})\) through all the observed configurations with the same contacts \( S_{j} = \{ o_{j1}, o_{j2}, \ldots, o_{jk} \} \).

\[
PS_{i} \rightarrow \left( f_{x_{C_{i}}} (p, \theta), f_{x_{C_{i}}} (p, \theta), \theta \right)_{(p_{i}, \theta_{i})}
\]

(13)

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 \( PS_{j} \). The continuous path segment for a single ve contact in the \((p, \theta)\) space as shown in Figure 8.

![Figure 8 Path segment on the c-surface corresponding to a ve contact](image)

**Figure 8 Path segment on the c-surface corresponding to a ve contact**

5 Implementation

The main steps of the algorithm which converts the observed poses of the assembled object to the fine motion path shown in Figure 9.

\[
\{ o_{1}, o_{2}, \ldots, o_{k} \}
\]

Find Contacts

Compute C-obstacle features

Compute Path Segments

\( (PS_{1}, PS_{2}, \ldots, PS_{n}) \)

**Figure 9 Main steps of the Algorithm**

The Algorithm is as follows:
• Instantiate the models of the assembled objects at each observation, \( o_j \). Find the feature pairs which are in contact for each \( o_j \) of the assembled object with the environment as explained in Section 4.1. Let the contact feature pair set for the observed configuration \( o_j \) be \( FP_j \).

• Compute the unique c-space obstacle feature \( C_j \) using equations (4), (5) and (6) corresponding to each observation \( o_j \) using the feature pair set \( FP_j \) as explained in section 4.3.

• Segment the observed configurations \( \{o_1, o_2, \ldots, o_k\} \) into contiguous segments \( \{S_j, S_{j2}, \ldots, S_{jk}\} \) such that all observations in a segment \( S_j \) maintain the same contact pair set \( c_j \) and have the same c-surface.

\[
S_j = \{o_{j1}, o_{j2}, \ldots, o_{jk}\}
\]

\[
FP_{j1} \cap FP_{j2} \cap \ldots \cap FP_{jk} = c_j
\]

• Compute the path segment \( PS_j \) lying on the c-space obstacle feature \( C_j \) through the observed configurations in the segment \( S_j = \{o_{j1}, o_{j2}, \ldots, o_{jk}\} \) as explained in section 4.4.

The complete assembly motion path is then the concatenation of the path segments \( PS_j \). The reconstructed path in c-space of the peg in the task observed in Figure 2 is shown in Figure 10.

![Figure 10 Path of the peg in configuration-space](image)

6 Assembly Task Execution

Once we have modeled the exact assembly path used in the task, we can program a robot to execute the trajectory. Since we have all the contact information at every instant of the task, we can compute both the positional and force references for the assembly path. For our implementation we used only the positional information and a relatively accurate robot (RobotWorld) to execute the modeled assembly path. The snapshots of the execution are shown in Figure 12.

![Figure 12 Assembly Task Execution on RobotWorld](image)

7 Conclusions

We have successfully modeled the motion plan of planar assembly tasks from observation involving polygonal objects. We reconstructed the path by first computing the relevant features of the c-space obstacle. Then we computed the path segments lying on these features. The connected
path segments constitute the model of the observed assembly task.

We plan to extend the system to model assembly paths involving 3D polyhedral objects in the 3D space.

Appendix

The features of the configuration space obstacle involving polyhedral objects in a plane will be facets, edges, and vertices in the 3D space \((x, y, \theta)\). The analytical equations describing these features are parametrized by \(p\) or \(\theta\), or both, or neither. (Only the \(x\) component of the features are given here for brevity). The complete derivation and details are in [3].

There are two types of facets, one due to a single vertex on edge (ve) contact, and the other due to a single edge on vertex (ev) contact. The analytical functions describing these facets are given by equations (15) and (16).

\[
f_{x_{ve}}(p, \theta) = k_{1_x} + k_{2_x} p + k_{3_x} \cos(\theta + \kappa_{3_x})
\]

\[
f_{x_{ev}}(p, \theta) = k_{1_y} + k_{2_y} p \cos(\theta + \kappa_{2_y}) + k_{3_y} \cos(\theta + \kappa_{3_y})
\]

There are four classes of edges of the c-obstacle which result from multiple ve or ev contacts or ee contacts or vv contacts. The \(x\) coordinate of a class-0 edge is parametrized by \(p\) and is given by (17).

\[
f_{x_{class-0}}(p) = k_{0_x} + k_{1_x} p
\]

The \(x\) coordinate of class-1, class-2 and class-3 edges are parameterized by \(\theta\) as given by (18), (19) and (20).

\[
f_{x_{class-1}}(\theta) = k_{0_x} + k_{1_x} \sin(\theta) + k_{2_x} \cos(\theta)
\]

\[
f_{x_{class-2}}(\theta) = k_{0_x} + \frac{k_{1_x} \sin(\theta) + k_{2_x} \cos(\theta)}{k_{3_x} \sin(\theta) + k_{4_x} \sin(\theta)}
\]

\[
f_{x_{class-3}}(\theta) = k_{0_x} + k_{1_x} \sin(\theta) + k_{2_x} \cos(\theta) + k_{3_x} + k_{4_x} \sin(\theta) + k_{5_x} \cos(\theta)
\]

The vertex on a c-space obstacle can be defined by its coordinates.

The inverse functions which map from the \((x, y, \theta)\) space to the parameter space \((p, \theta)\) exist for each class of feature given above. \(\theta\) is identical in both parameter spaces. The \(f_d(x, y, \theta)\) function is defined for the single ve, ev contacts and class-0 edges. The \(f_d(x, y, \theta)\) function can be computed using the dot product given by (21), where \(e_{cw}\) and \(e\) can be computed from the geometry of the contacts.

\[
p = f_d(x, y, \theta) = e_{cw} \cdot \hat{e}
\]

Acknowledgments

This research was funded in part by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), US Air Force, Wright Patterson AFB, Ohio 45433-6543 under Contract F33615-90-C-1465 EPA Order No 7597 and in part by NSF under Contract CDA 31211797.

References


