

Assembly as a Noncooperative Game of its Pieces: The Case of Endogeneous Disk Assemblies

H. Işıl Bozma^{†*}, C. Serkan Karagöz[†] and Daniel E. Koditschek^{†**}

[†] Department of Electrical and Electronic Engineering
Boğaziçi University, Bebek, Istanbul 80815, Turkey

[‡] Artificial Intelligence Laboratory and Control Systems Laboratory
EECS Department, College of Engineering, University of Michigan
Ann Arbor, Michigan 48109 USA

Abstract

We propose an event-driven approach to planning and control of robot assembly problems using ideas from noncooperative game theory. We report on the results of an extensive simulation study for a very simple two degree of freedom case — the arrangement of disks on a plane by a disk shaped robot.

1 Introduction

This paper addresses autonomous robot assembly — automatic generation of robot actuator commands that cause it to move a collection of rigid-body parts from an arbitrary initial configuration to a final desired assembly. The traditional approach solves the path planning problem offline resulting in a sequence of motion trajectories that would be tracked by some local controller [7, 3, 10, 2]. Yet general motion planning problems may be solved by an alternative approach that employs feedback to achieve the desired goal via event driven reactions [6]. In contrast to open loop plans, if the vector field that directs these reactions is appropriately constructed and implemented, then robustness to small disturbances as well as convergence to the goal state may be guaranteed.

Recent work in extremely simplified problem settings suggests that such feedback techniques may be extended to the problem of Fig. 1 as well: the automatic generation of parts mating sequences and motions required to re-arrange the location of a multiplicity of parts with one robot [4, 11]. A feedback-based solution to the problem of sphere assemblies in one dimensional workspace where the robot moves in a line parallel to this workspace is offered in [4]. An extension of these ideas to the problem of 2D Sphere assemblies in a two-dimensional workspace where the

robot's movement is independent of the parts' movements is examined in [11]. Yet there is another complication arising in multiple parts assembly that has never before been addressed in the closed loop motion planning literature: the situation wherein the robot inhabits the same configuration space as the parts being manipulated — endogeneous assemblies¹. This paper addresses the problem of endogenous 2D sphere assembly.

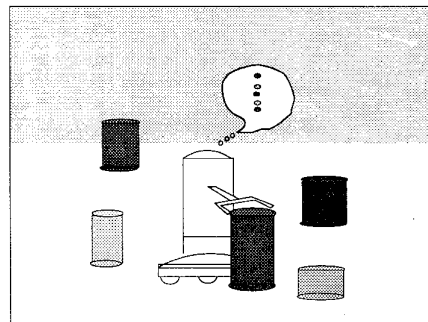


Figure 1: 2DOF Endogeneous assembly: the robot is to re-arrange the positions of the parts into the specified configuration. The robot must mate with each part in turn until the goal has been achieved, and collisions between parts along the way are to be avoided.

Consider a two-dimensional workspace in which a set of disk-shaped parts and a disk shaped robot are all located at arbitrary non-overlapping initial configurations. The parts are stationary by themselves while the robot can move freely, attach to a part and move it from one location to another. Moreover, it is assumed that the robot dynamics are known and the robot can sense the position of the parts. This paper describes

*Supported in part by the Turkish Scientific Research Agency MISAG grant 24-1992 and TÜBİTAK BAYG grant.

**Supported in part by the US National Science Foundation under grant IRI-9123266.

¹There does not seem to be too much attention paid even in the traditional motion planning literature to the distinction between exogenous and endogenous assembly situations. A notable exception is [1].

a composite algorithm that switches between different feedback controllers in a reactive manner. An extensive simulation study suggests that the algorithm succeeds in generating actuator commands that cause the robot manipulator to move the pieces from an arbitrary initial disassembled configuration to a specified final assembled configuration while avoiding collisions between them.

As will be seen, the simulations suggest that the algorithm works successfully, but not particularly “intelligently:” parts are sometimes picked up and discarded prematurely from the view point of an impatiently watching human. We are reasonably confident that a variety of performance improvements can be made by tinkering with the vector field constructions reported in the body of the paper, and a new set of simulations is presently under way to test these ideas. In any case, we have been pleasantly surprised to find this simple algorithm working at all — we are aware of no game theoretic results in the literature that provide insight concerning the convergence of such a multi-objective descent on a non-convex space as our problem presents. Thus, the most pressing effort will be to prove that the algorithm is correct. This would establish that a plan for endogenous assembly can be generated in a completely reactive manner, providing a valuable guide for the performance oriented extensions required to make it useful.

1.1 Statement of Problem

Consider N disk shaped bodies that can be translated around on a plane. Denote the position of the center of each body, $i = 1, \dots, N$, as $b_i \in B_i = R^2$, its desired position $d_i \in B_i^*$, and its radius $\rho_i \in R^+$. Let $b \in B = \prod_{i=1}^N B_i$ denote the vector of all positions and $d \in B$ the vector of all the desired positions. The robot’s position is denoted by $r \in R = R^2$ and its radius is ρ_r . Let the robot be placed in the same two-dimensional workspace as the spherical bodies. Suppose that each body i is at position $b_i(0)$ and the robot is at position $r(0)$. We require a feedback control of a robot that results in the motion of these bodies from an arbitrary initial condition in the workspace to the desired goal or in termination of robot’s motion if the goal is not reachable. The bodies must never be allowed to touch each other along the way to their respective goal positions. For example, if all of the N -bodies could move at the same time, letting $\rho_{ij} = \rho_i + \rho_j$, then the scalar valued function $\beta \in C^\omega[B, R]$

$$\beta(b) = \prod_{i=1}^N \prod_{\substack{j=1 \\ j \neq i}}^N (b_i - b_j)^T (b_i - b_j) - \rho_{ij}^2$$

defines the configuration space obstacle \mathcal{O} to be avoided as $\mathcal{O} = \beta^{-1}[-\infty, 0]$.

1.2 Solution Approach

This paper develops an approach in which by sequentially switching among a family of feedback controllers, a plan is generated in a completely reactive

manner that we hope but have not yet proven to be assured of convergence — either to successful completion of the assembly or to termination in a spurious local minimum. This is achieved via the following steps:

Move part: We design a set of feedback controllers — one for moving each different part. Each of these controllers is defined by a navigation function [9] for the corresponding *part-mated-to-robot* pair that encodes the goal configuration for assembling that part along with the obstacle space presented by all the other parts when doing so.

Mate part: The robot is sent to mate with one designated part at a time and if the mating succeeds continues with the assembly of that part according to move-part until it becomes blocked. The mating is achieved by a controller again arising from a navigation function that encodes the allowed mating configurations and presents all the parts as obstacles.

Next part: If a mating fails because the robot encounters a local minimum of the mating function prior to reaching the designated part, then next-part is chosen and mate-part is re-invoked. Similarly, when move-part terminates at a local minimum of the active robot-moving-part function, then a next-part is chosen and mate-part is re-invoked. Thus, whenever blocked, the robot switches to the assembly of the next most “urgent” part.

The assembly plan is *implicitly* defined by which and in what order the individual parts’ controllers are selected during a given run. An assembly plan is correct if it implies the composition of controllers in a manner that ensures task achievement in case of a feasible goal and termination of the robot’s motion short of the finished assembly signalling infeasibility of the goal.

2 2 DOF Endogeneous Assemblies

2.1 Feedback Induces a Game

Let u be the means by which a robot can change the state of the environment according to the model

$$b(k+1) = f(b(k), u(k)). \quad (1)$$

Here, f is the transition map from one “mated-but-blocked” configuration of parts to the next: the ensemble of locations of all the parts after the robot has moved to the furthest possible extent the part, b_m , to which it is mated at stage k . The input, u , denotes a choice of index along with the motion of the single part indexed. We seek a means of assigning to the robot an index choice and a placement decision as a function of the present state

$$u(k) = (i(k), b_{i(k)}(k+1)) = \Phi(b(k)). \quad (2)$$

The induced closed loop system is governed by the iterates of the resulting closed loop map:

$$b(k+1) = f(b(k), \Phi(b(k))). \quad (3)$$

We hope to design Φ in such a fashion that a large set of initial conditions are eventually drawn into the desired goal set d after a number of moves or even more preferably, that *almost* all initial conditions can be guaranteed to eventually arrive at the goal. We now describe the construction of Φ .

Let $\psi(r, b_1, \dots, b_N)_{i=1, N}$ denote a collection of smooth scalar valued maps on the state space and $\bar{b}_i \equiv (b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_N)$ be a vector in the subspace $\bar{B}_i = R^{(N-1)n}$ remaining after B_i has been factored out. Define the vector field v_i to be the negative gradient of the map ψ_i with respect to the vector b_i

$$v_i(r, b_i; \bar{b}_i) \equiv -[D_{b_i} \psi_i(r, b_1, \dots, b_N)]$$

Here, the semicolon notation is intended to call attention to the parametric role that the other pieces $(b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_N)$ will play in the motion of piece b_i . Motion on this subspace of the the state space will be governed by the limit properties of the gradient dynamical system

$$\dot{b}_i = v_i(r, b_i; \bar{b}_i)$$

whose integral curve through the initial condition $b_i(0)$ will be denoted by $v_i^t(r, b_i; \bar{b}_i)$.

The second component of Φ in (2) is now specified by the limit set

$$\begin{aligned} b_{i(k)}(k+1) &= v_{i(k)}^\infty(r, b_{i(k)}; \bar{b}_{i(k)}) \\ &\equiv \lim_{t \rightarrow \infty} v_{i(k)}^t(r, b_{i(k)}; \bar{b}_{i(k)}) \end{aligned}$$

When $v_i(r, b_i^*; \bar{b}_i) = 0$ implies that $D_{b_i} v_i(r, b_i^*; \bar{b}_i)$ has full rank, it can be guaranteed that the limit set of every trajectory through any possible initial condition is some isolated singularity $\Phi(i(k), b(k)) = \{b_i^*\}$. However, this is not a generic property and the vector field v_i passes through bifurcation points as the parameters $(b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_N)$ vary over the state space. In order to proceed, the same limiting properties must persist even at bifurcation. With this notation and assumptions in force, each function ψ_i gives rise to a (generally discontinuous) map $v_i^\infty(r, b_i; \bar{b}_i)$ of B_i into itself.

We use the term *game* to describe the resulting discrete dynamical system (3)

$$f(b(k), \Phi)_j = \begin{cases} \Phi(j, b(k)) & \text{if } j = I(b(k)) \\ b_j(k) & \text{otherwise} \end{cases} \quad (4)$$

since each of the *players*, $\{b_i\}_{i=1, N}$ — the bodies to be assembled — tries at each stage (through the “agency” of the robot that tows it) to minimize its distinct cost function, ψ_i . The fixed points of the discrete system are the solutions of the game and determine whether the assembly is to be successfully completed or terminated.

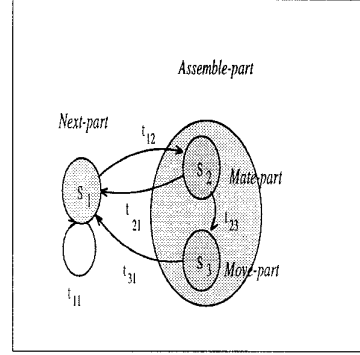


Figure 2: Assembly Game

2.2 Implementation of the Game

We propose a simple, but rather general two-level feedback control scheme as shown in fig 2. In the *next-part state* S_1 , a very simple switching logic dictates the manner in which several “assemble part” subplans are switched in and out of operation. During *assemble part*, there are two possible states: *mate-part state* S_2 followed possibly by *move-part state* S_3 . A class of “mate part m” algorithms move the robot through its workspace following the gradient system defined by φ_m in order to bring the robot as closely as possible to the m th part, while all the remaining parts are standing in their places. A second class of “move part i” algorithms move the robot-part m pair following the gradient system defined by ψ_m in order to position to the m th part. Transitions between states will depend on a number of inverse images of critical points of φ_m and ψ_m . Let us denote each with C_{φ_m} and C_{ψ_m} respectively² and the ϵ -neighborhood around these sets by N_ϵ . Let us refer to the ϵ -neighborhood around these sets by N_ϵ . In this framework, the whole assembly can be viewed as the robot refereeing a non-cooperative game being played between subassemblies [5].

2.2.1 Subgoal: Next-part S_1

Given a particular instance of an assembly, the first step is to decide which is the best piece to move. For this, we use a function φ that encodes the goal configuration and the obstacle space globally. In particular, we construct the function $\varphi(b) = \frac{\gamma^{k_1}(b)}{\beta(b)}$ where the function γ has the form

$$\gamma(b) = \sum_{m=1}^N (b_m - d_m)^T (b_m - d_m), \quad (5)$$

²Let it be noted that C_{φ_m} is the limit set of the system $\dot{r} = -D_r \varphi_m(r, b(k))$ and C_{ψ_m} is the limit set of the system $\dot{r} = -D_r \psi_m(r, b_m; b_m)$.

β represents the obstacle space boundary as:

$$\beta(b) = \prod_{i=1}^N \prod_{\substack{j=1 \\ j \neq i}}^N (b_i - b_j)^T (b_i - b_j) - \rho_{ij}^2$$

and $k_1 \in Z^+$ is an appropriately selected positive integer. Letting $I(b)$ be an index-valued function with the property

$$I(b) = \arg \max_{m \leq N} |D_m \varphi(b)|$$

Hence, the switching logic picks out the part $i(k)$ at time k whose direction of descent with respect to φ is the greatest. Once $i(k)$ selected, a state transition t_{12} from next-part state to mate-part state occurs. Let it be remarked that at this state, the robot-mating-part- m is not blocked $r \notin N_\epsilon(C_{\varphi_m})$ as well as robot-moving-part- m $r \notin N_\epsilon(C_{\psi_m})$.

2.2.2 Subgoal: Mate part S_2

Assume that a finite number of smooth scalar valued functions $\{\varphi_m\}$, $m = 1, \dots, N$ on regions of $\mathcal{R} \times \mathcal{B}$ have been constructed, that measure the distance of the robot to the corresponding indexed part's position in such a way that for each $r \in \mathcal{R}$, each of the functions is a navigation function. This is a straightforward navigation problem of a disk-shaped robot [9], where the desired final point is some neighborhood of the m th part. The function φ_m has the form $\varphi_m(r, b) = \frac{\gamma_m^{k_2}(r, b)}{\beta_r(r, b)}$ where the function γ_m is given by :

$$\gamma_m(r, b) = ((r - b_m)^2 - \rho_{rm}^2)^2,$$

where $\rho_{rm} = \rho_r + \rho_m$, the function β_r is :

$$\beta_r(r, b) = \prod_{i=1}^N (r - b_i)^2 - \rho_{ri}^2$$

and $k_2 \in Z^+$ is a positive appropriately selected integer. It remains to specify a mating policy that is ensured of robot-part- m mating. This is accomplished by adopting a control law:

$$u_{\varphi_m} = -D_r \varphi_m(r, b)$$

The robot stays in the mating state until its progress is blocked due to a critical point $r \in N_\epsilon(C_{\varphi_m})$. If the critical point is within some *a priori* specified proximity of the contact space of part m — $r \in N_\epsilon(b_m)$, and if can move that part $r \notin N_\epsilon(C_{\psi_m})$, a state transition t_{23} to move-part state occurs. Otherwise, the driving automaton goes back to the find-next state with a state transition t_{21} .

2.2.3 Subgoal: Move part S_3

Once the robot is mated to a part m , the two coupled bodies act as a body in the extended space $\mathcal{R} \times SO(2)$.

Let us denote the augmented state vector as $\begin{bmatrix} r \\ \theta \end{bmatrix}$.

The state of the robot is:

$$b_m = r + c_g \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$$

where $c_g \in R$ is the mating distance. Assume that a finite number of smooth scalar valued functions ψ_m , $m = 1, \dots, N$ have been constructed, that measure the distance of the robot-mated-part to this part's destination in such a way that for each $\mathcal{R} \times S$, this is a set of navigation functions. This is again a navigation problem — a spider-like robot moving in a world of spherical obstacles [8]. Each function ψ_m has the form $\psi(r, b) = \frac{\gamma^{k_3}(b)}{\beta_m(r, b)}$, where γ is defined as in eq. 5, the function β_m is as follows:

$$\beta_m(r, b) = \prod_{i=1}^N ((b_m - b_i)^T (b_m - b_i) - \rho_{mi}^2) \times ((r - b_i)^T (b_m - b_i) - \rho_{ri}^2)$$

where $k_3 \in Z^+$ is an appropriately selected positive integer. The moving is accomplished by adopting a particular control law that acts as follows:

$$u_{\psi_m} = -D_r \psi_m(r, b)$$

The automaton stays in the move-part state until the progress of the robot-part m pair gets blocked — $r \in N_\epsilon(C_{\varphi_m})$, in which case a state transition t_{31} takes the robot back to the next-part state where the next part to be assembled is determined.

2.3 Summary of Switching Logic

A (very simple) switching logic can now be defined with respect to a set of partitions \mathcal{P}_m — one for each part — imposed upon the robot's configuration space \mathcal{R} or its mated extension, $\mathcal{R} \times SO(2)$, when piece m is active. We will delineate the algorithmic consequences of this partition by reference to the symbolic variable

$$s \in \{\text{next-part}, \text{mate-part}, \text{move-part}\}$$

Each partition $\mathcal{P}_m, m = 1, \dots, N$ depends on the inverse images of critical points of φ_m and ψ_m as:³

$$s = \begin{cases} \text{next-part} & r \in N_\epsilon(C_{\varphi_m}) \text{ and } r \notin N_\epsilon(b_m) \\ \text{or} & \\ & r \in N_\epsilon(C_{\psi_m}) \\ \text{mate-part} & r \notin N_\epsilon(C_{\varphi_m}) \text{ and } r \notin N_\epsilon(C_{\psi_m}) \\ \text{move-part} & r \in N_\epsilon(C_{\varphi_m}) \text{ and } r \notin N_\epsilon(C_{\psi_m}) \\ & \text{and } r \in N_\epsilon(b_m) \end{cases}$$

Partition \mathcal{P}_m now governs the choice of continuous controllers as:

$$u = \begin{cases} u_{\varphi_m} & s = \text{mate-part} \\ u_{\psi_m} & s = \text{move-part} \\ 0 & s = \text{next-part} \end{cases}$$

³Note that the two missing cases in the effective decision table — where $r \notin N_\epsilon(C_{\varphi_m})$ and $r \in N_\epsilon(C_{\psi_m})$ — are inconsistent with the *a priori* assumption that part m is presently active.

Note that each time a new part m' , is chosen following a transition to $s = \text{next-part}$, the next partition, \mathcal{P}_m governs the control decisions.

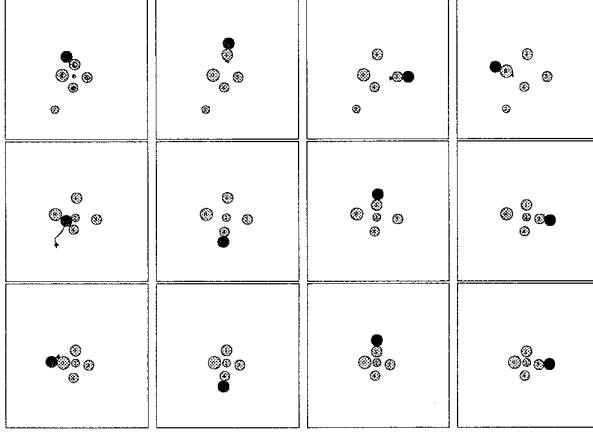


Figure 3: A 5 sphere assembly sequence with frames sequenced top-bottom. Frame 1 shows the initial configuration. The last frame is the assembled configuration. The intermediate frames show the sequence of moves of the robot.

3 Simulations

In this section, we present the computer simulations of the closed loop behavior of first order system. A typical anecdotal run of such a simulation study is presented in fig. 3. In the top left frame of figure 3, a random configuration as represented by shaded circles is shown superimposed on the goal assembly configuration as represented by the diamonds. The robot is represented by the black circle. Let it be observed that in this particular case, all the parts except part 2 are near their goal configurations and furthermore that the robot cannot move part 2 to its goal position unless some of the parts are moved away from their goal positions. The rest of the frames show sequentially sampled moves of the robot. In the top center frame, the robot moves part 1 away from its goal position and it moves part 3 and 0 away from their goal positions in the next two frames. The robot is then able to move part 2 to its goal position as shown in the bottom left frame. It then visit and re-visits the outer parts and either moves each back to near its goal position or improves positioning accuracy.

3.1 Statistics

In this section, we present results based on our extensive simulations. Our assemblies contain six disk-like objects of varying radii. We consider six different randomly chosen final assembly configurations of increasing difficulty - decreasing log of destination β - as shown in fig. 4. The initial position of the robot is the left upper corner of the workspace. In the graphs, each data point represents the mean and standard deviation of 25 runs with random initial configurations. In this study, we use four measures of performance: 1.)

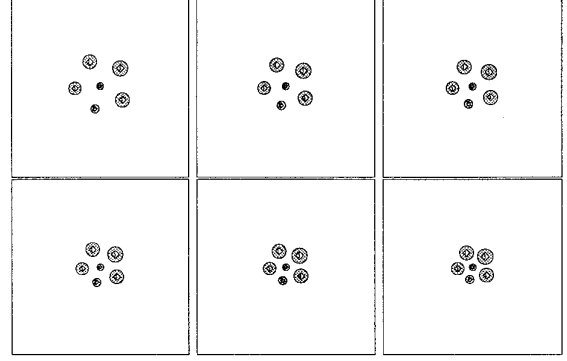


Figure 4: Assemblies of increasing difficulty (1 to r, t to b): (a) $\beta = 1.3 \times 10^{57}$, (b) $\beta = 7.5 \times 10^{55}$, (c) $\beta = 3.0 \times 10^{54}$, (d) $\beta = 7.6 \times 10^{52}$, (e) $\beta = 9.3 \times 10^{50}$, (f) $\beta = 8.8 \times 10^{48}$.

Normalized assembly path length (npl), 2.) Normalized robot path length (rpl), 3.) Number of switches and 4.) Positioning inaccuracy (pi). Assembly path length is the distance travelled in R^{2N} by the disk-like parts from an initial configuration to a final “assembled” configuration. In order to account for the variations in the initial conditions, it is normalized by the Euclidean distance from the initial configuration to the goal configuration. If the time of the assembly is from t_i to t_f , its formula is as follows:

$$\text{npl} = \frac{\int_{t_i}^{t_f} b \, dt}{|b(0) - d|}$$

Similarly, the distance travelled by the robot including for both mating to and moving the parts is normalized by the Euclidean distance from the initial configuration to the goal configuration. Its formula is as follows:

$$\text{rpl} = \frac{\int_{t_i}^{t_f} r \, dt}{|b(0) - d|}$$

Furthermore, as the parts are sometimes sloppily placed, the Euclidean distance from actually realized final assembled configuration to goal configuration is used to assess positioning accuracy. Its formula is as follows:

$$\text{pi} = |b(t_f) - d|$$

A sample run is shown in figure 5. Let it be observed that in this particular case, all parts except part 1 are not near their goal configurations. The rest of the frames show sequentially and non-uniformly sampled moves of the robot. In the top center frame, the robot moves part 1 away from its goal position and it moves part 4 closer to its goal position. It then moves part 5 closer to its goal position. In the next frame, we observe part 2 being moved to a closer neighborhood of

its assembled position. Similarly, part 3 is moved to a closer neighborhood of its assembled position in frame 6. Part 0 is then moved to its goal position in the next frame. The robot then improves the positional accuracy of the parts.

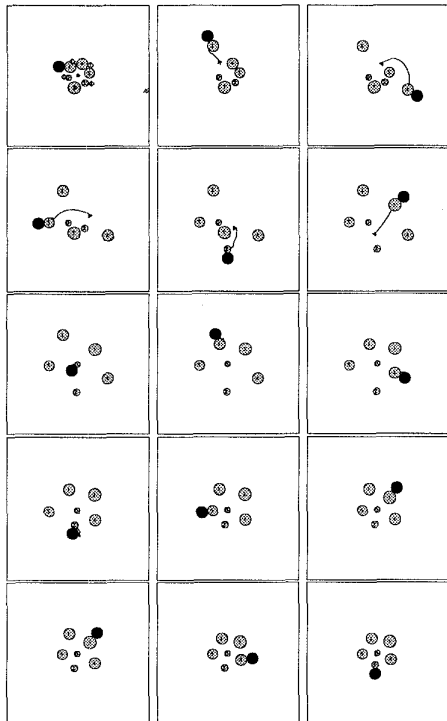


Figure 5: A 6 sphere assembly sequence with destination $\beta = 8.8 \times 10^{48}$ with frames sequenced top-bottom, left-right. Frame 1 shows the initial configuration. The last frame is the assembled configuration. The rest of the frames show the sequence of moves of the robot.

Normalized Path Length vs. Assembly Difficulty

Figure 6 shows that normalized path length varies in a manner that matches our intuitive expectation - the closer the parts need to be packed together, a greater distance they need to be moved. The path-length performance correlates inversely with the assembly difficulty - that closely packed desired assembly of figure 4(f) are more difficult to assemble than a loosely packed assembly of figure 4(a). It is also noted that path length is on average about five times longer than the euclidean distance between the initial and final configurations. Two factors account for this: First, the parameter k_3 of the moving function ψ_m is chosen such that the obstacle avoiding term dominates unless the part is close to its destination which means that in general parts move away from their assembled positions before moving towards them. Secondly, in

some of the randomly generated initial assembly configurations, some parts - although at their assembled positions - may need to be moved away before other parts can be assembled.

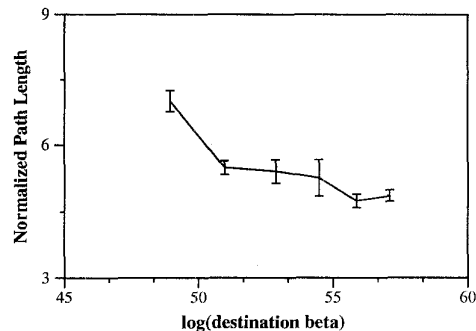


Figure 6: Normalized path length statistics.

Robot Path Length vs. Assembly Difficulty

The normalized path travelled by the robot matches also our intuitive notions of assembly difficulty. Again, the tightly packed assembly of figure 4(f) cause the robot to travel a longer path length than that of a more loosely packed assembly. It is observed that the path travelled by the robot is of magnitude about 30 times that of the Euclidean distance between the initial and final assembly configurations. Three factors contribute to this: First, as explained earlier on, the robot is initially located on the upper left corner of the workspace - far from the parts to be assembled and this fact is not accounted for in our normalization. Secondly, the k_2 parameter of the mating function φ_m is chosen such that the obstacle avoidance terms dominates which means that the robot travels in a path distant from all the parts. Finally, in some of the randomly generated initial configurations where some of the parts are located close to their assembled positions, the robot may move these parts away from their locations before moving them back.

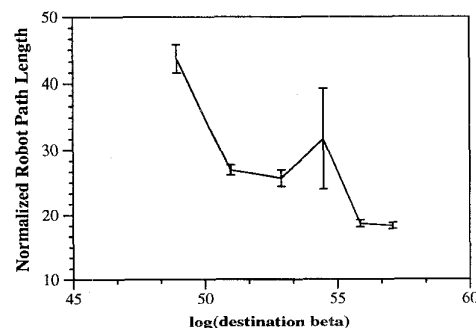


Figure 7: Normalized robot path length statistics.

Switches vs. Assembly Difficulty

Figure 8 shows the mean standard deviations for the number of switches. Here we observe that the number of switches required to complete an assembly rises as a function of the assembly difficulty. The easy assemblies require on average each part to be switched three times while the more difficult assemblies have both a greater mean of the number of switches as well as higher variance.

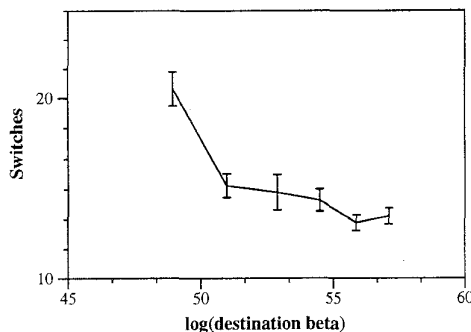


Figure 8: Switching statistics.

Positional Inaccuracy vs. Assembly Difficulty

We observe that the positional accuracy of the assembled parts decreases with the difficulty of the assembly. The more closely the parts need to be assembled together, the more crucial it is that the robot places a part precisely at its first attempt since chances of that part being blocked by other assembled parts increases once the parts are assembled - even sloppily. However, the construction of moving function is such that a part needs to be re-visited more than once before it is accurately positioned.

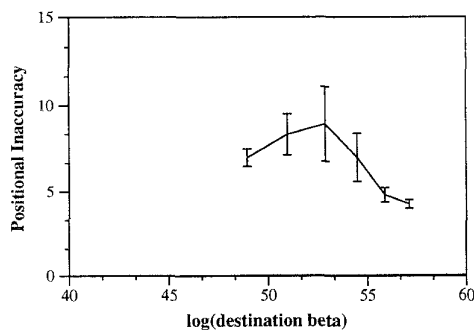


Figure 9: Positional inaccuracy statistics.

4 Conclusion

In this paper, we propose an event-driven approach to the control of robot engaged in endogeneous assembly - an assembly situation that arises when the robot inhabits the same workspace as the parts themselves.

We have explored the performance of a concrete instance of this approach in a simple problem setting - a mobile robot moving a set of cylindrical objects from an initial arbitrary configuration to a final desired configuration. We show with our simulations that a working implementation of feedback policies can lead to successful assemblies without requiring an abstract description of all the mating sequences. In this feedback-based approach, the assembly plan is specified implicitly as a sequence of control laws for bringing unactuated degrees of freedom into a final configuration with a single actuated robot possessing fewer degrees of freedom.

References

- [1] R. Alami, T. Simeon, and J. Laumond. A geometrical approach to planning manipulation tasks: The case of discrete placements and grasps. In *Preprint of 5th International Symposium of Robotics Research*, pages 113-123, 1989.
- [2] Thomas L. De Fazio and Daniel E. Whitney. Simplified generation of all mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 1987.
- [3] L.S. Homem de Mello and A.C. Sanderson. A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 1991.
- [4] D.E. Koditschek. An approach to autonomous robot assembly. *Robotica*, 12:137-155, 1994.
- [5] D.E. Koditschek and H.I. Bozma. Assembly as a noncooperative game of its pieces: Analysis of 1d disk assemblies. *paper in preparation*.
- [6] D.E. Koditschek and E. Rimon. Robot navigation functions on manifolds with boundary. *Advance in Applied Mathematics*, 11:412-442, 1992.
- [7] t. et. al Lozano-Peréz. Handey: A robot system that recognizes, plans and manipulates. In *Proceedings of IEEE Int. Conference on Robotics and Aut.*, pages 843-849, 1987.
- [8] E. Rimon. A navigation function for a simple rigid body. Technical Report 9014, Yale University, 1990.
- [9] E. Rimon and D.E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501-518, 1992.
- [10] R.P. Paul and H. Zong. Robot motion trajectory specification and generation. In *Second International Symposium On Robotics Research, Kyoto, Japan*, 1984.
- [11] L.L. Whitcomb, D.E. Koditschek, and J.B.D. Cabrera. Toward the automatic control of robot assembly tasks via potential functions: The case of 2D sphere assemblies. In *Proceedings of IEEE Int. Conference on Robotics and Aut.*, 1992.