Simultaneous Control of Multiple MEMS Microrobots (Supporting Material)

Bruce R. Donald[1,2,6], Christopher G. Levey[3], Igor Paprotny[1,4], and Daniela Rus[5]

[1] Department of Computer Science, Duke University, Durham, NC, USA.
[2] Department of Biochemistry, Duke University Medical Center, Durham, NC, USA.
[3] Thayer School of Engineering, Dartmouth College, Hanover, NH, USA.
[4] Department of Computer Science, Dartmouth College, Hanover, NH, USA.
[5] Department of EECS, MIT, Cambridge, MA, USA.

Below is the supplementary material for the following paper:

- B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus. "Simultaneous Control of Multiple MEMS Microrobots." Submitted to *WAFR 2008: The Eighth International Workshop on the Algorithmic Foundations of Robotics*, Guanajuato, Mexico. (2008)

Appendix

The following is an appendix which provides additional information to substantiate the claims of the paper [4]. **Appendix 1** describes the proof of Lemma 1. **Appendix 2** provides details for trajectory planning to reduce the parallel motion of n robots to parallel motion of two robots, followed by sequential control of single devices. Details regarding control strategies for microassembly are presented in **Appendix 3**.

1 Details of the Proof of Lemma 1

Lemma 1. *An n-robot STRING system has exactly* $n + 1$ *accessible control states.*

Proof. (By induction.)

The base case: a STRING system with $n = 1$, has two accessible control states, (0 - arm up) and (1 - arm down).

The inductive step: adding a single device, (changing the size of the system from n to $n + 1$) extends the number of accessible control state by exactly one, provided that both the n and $n + 1$ microrobotic systems remain STRING.

[6] Corresponding author: brd+wafr08@cs.duke.edu

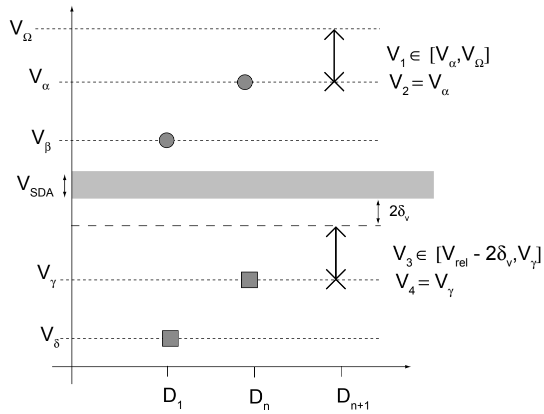


Fig. 1. Proof of Lemma 1.

Let n microrobots, labeled $D_1, \cdots, D_n$, be a STRING system sorted according to $V_{u,i}$ and $V_{d,i}$. Without loss of generality, $V_{d,n} \leq V_{d,n+1}$ and $V_{u,n} \leq V_{u,n+1}$ (If this is not the case, we can simply relabel the voltages and generate an equivalent system sorted as described above). Fig. 1 shows the ranges for the transition voltages of D_{n+1}, such that the new, $n+1$ robotic, system retains STRING. Let $V_\alpha, \cdots, V_\delta$ be significantly independent transition voltage levels, ordered such that $V_\delta < V_\gamma < V_u < V_\beta < V_\alpha < V_\Omega$. Let $V_{d,n} = V_\alpha$ and $V_{u,n} = V_\gamma$. It follows that the snap-down voltage $V_{d,n+1}$ can have a value V_1 in the range $[V_\alpha, V_\Omega]$, or voltage $V_2 = V_\alpha$. Similarly, the release voltage, $V_{u,n+1}$, can have the value V_3 in the range $[V_{rel} - 2\delta_v, V_\gamma]$, or voltage $V_4 = V_\gamma$ ($V_{u,n+1}$ can not exceed $V_{rel} - 2\delta_v$ without risking that V_{rel} might release the steering arm during the power delivery cycle). Consequently, for the $(n+1)$ robot system to remain STRING, one of the following combinations of the snap-down and release voltages for D_{n+1} must hold: (V_1, V_3), (V_1, V_4) and (V_2, V_3). We examine each case separately:

(V_1, V_3): Because the snap-down voltage of D_{n+1} is greater than the snap-down voltage of $D_1 \cdots D_n$, $V_{d,n+1} > V_{d,i}$, $i \in Z_n$ where $Z_n = \{1, \cdots, n\}$, we can only snap-down the arm of D_{n+1} after we snap-down the arms of all other devices. Since the release voltage of D_{n+1} is greater then the release voltage of $D_1, \cdots, D_n$, $V_{u,n+1} > V_{u,i}$, $i \in Z_n$, we can only release the arm of any other device *after* we have released the arm of D_{n+1}. Consequently, we can only change the state of D_{n+1} when $D_1, \cdots, D_n$ are in state 1. During all other states of the system, the state of D_{n+1} must remain 0. Consequently, the number of accessible control states increases by exactly one.

(V_1, V_4): This case is identical to (V_1, V_3), except that the arm of D_n is released at the same time as the arm of D_{n+1}. As long as $V_{d,n+1} > V_{d,i}$, we can snap down the arm of D_{n+1} only after all other devices $D_1, \cdots, D_n$ are in state 1. As a consequence, the number of accessible control states increases by one.

(V_2, V_3): The snap-down voltage of D_{n+1} is equal to the snap-down voltage of D_n, $V_{d,n+1} = V_{d,n}$. In this case, the arm of D_{n+1} is snapped down at the same time as the arm of D_n. Because the release voltage of D_{n+1} is greater than the release voltage of $D_1, \cdots, D_n$, $V_{u,n+1} > V_{u,i}$, where $i \in Z_n$, we can only release the arm of D_n (or any other devices) after we release the arm of D_{n+1}. As in the (V_1, V_3) case, the state of D_{n+1} must be 0 except when $D_1, \cdots, D_n$ are all snapped down, then D_{n+1} can be in either 0 or 1 by varying the release voltage. Consequently, the number of accessible control states increases by one.

We have now shown that adding a device to STRING system, such that the resulting system remains an STRING system, increases the number of accessible control states by exactly one. Combined with the base case ($n = 1$, two control states), it follows by induction that every n-robot STRING system has exactly $n + 1$ accessible control states. □

2 Motion Planning for *n*-Microrobot Assembly

We now describe how to plan the motion of n stress-engineered microrobots for microassembly, despite coupling of their motion through the global control signal. This section considers only nominal microrobot motion, and does not consider the accumulating control error.

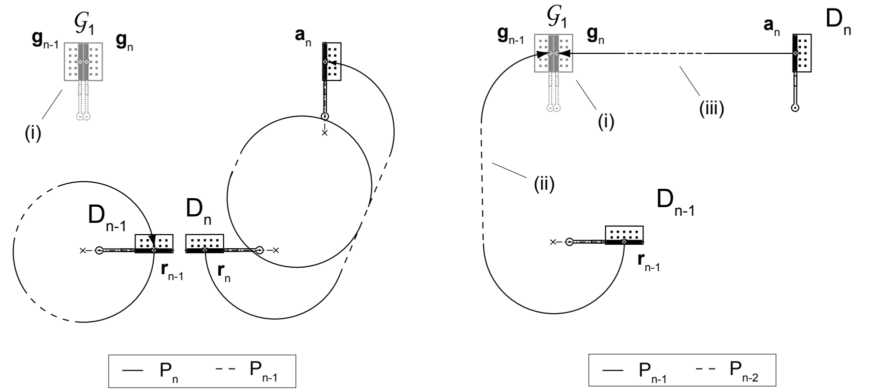
The configuration of the n microrobotic system is given by the vector $\mathbf{q} = (\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_n})$. We assume the robots start in an initial configurations $\mathbf{r} = (\mathbf{r_1}, \mathbf{r_2}, \cdots, \mathbf{r_n})$, and must be maneuvered to a goal configuration $\mathbf{g} = (\mathbf{g_1}, \mathbf{g_2}, \cdots, \mathbf{g_n})$ which corresponds to a desired target structure $\mathcal{G}_k$. We define the *gross motion planning problem* as the problem of finding a sequence of control primitives (called a *control sequence*) S that nominally (i.e. in absence of control error) maneuvers the robot from $\mathbf{r}$ to $\mathbf{g}$.

The structure of M allows us to reduce the parallel motion of n robots to parallel motion of two robots, followed by sequential motion for single devices. Without collisions, we cannot stop individual robots from moving, however we can confine them to move in circular orbits. Our heuristic planning algorithms consider the area swept by the orbiting robots (discs of radius r^* in the workspace) as obstacles. This approach is neither general nor complete, and requires a minimum separation between the orbiting microrobots. However, our approach works well in practice on small number of robots. We recognize that more general collision avoidance methods can be adopted from [1, 8], however we leave the implementation of such extensions for future work. Note that, given the sequence (order in which to assemble), which in our case is specified by the control matrix M, the sequencing of the motion of our robots is defined. Consequently our approach is analogous to [5], albeit using non-holonomic robots.

The assembly of a structure composed of n robots takes place in $n - 1$ steps.

2.1 Step 1. Assembly of $\mathcal{G}_1$:

$\mathcal{G}_1$ is always assembled through the simultaneous motion of the two robots with the highest indexes, i.e. D_n and D_{n-1}, as this allows the robots $D_1, \cdots, D_{n-2}$ to orbit in limit cycles without making progress towards the goal. The assembly of $\mathcal{G}_1$ is divided



a: Stage 1 **b: Stage 2**

Fig. 2. Assembling the initial stable shape, $\mathcal{G}_1$ using microrobots D_n and D_{n-1}. **a: Stage 1:** D_n is maneuvered to $\mathbf{a_n}$ while D_{n-1} orbits. **b: Stage 2:** D_{n-1} is maneuvered to $\mathbf{g_{n-1}}$ while D_n moves in straight line to $\mathbf{g_n}$.

into two stages, as shown on Figs. 2(a) and 2(b), respectively (the orbiting robots are not depicted in Fig. 2).

During stage 1 (Fig. 2(a)) microrobot D_n is maneuvered (using control sequence S_1) from an initial configuration, $\mathbf{r_n}$, to an intermediate goal configuration $\mathbf{a_n}$, using control primitives P_n and P_{n-1}. The reason for maneuvering robot D_n to $\mathbf{a_n}$ rather that directly to its goal $\mathbf{g_n}$ is that D_n will only move in a straight line during stage 2. Hence, in stage 1 D_n must be maneuvered to a configuration from which the robot can enter its goal, $\mathbf{g_n}$, during subsequent straight-line motion in stage 2.

As D_n is maneuvered to the intermediate configuration $\mathbf{a_n}$, robot D_{n-1} orbits without making any progress towards the goal (i) (because control primitives P_n and P_{n-1} invoke only turning motion in D_{n-1}.) However, in order to calculate the length of the trajectory of robot D_{n-1} during stage 2 (Fig. 2(ii)), which determines the length of the straight trajectory of D_n (Fig. 2(iii)), which in turn determines the intermediate configuration $\mathbf{a_n}$, we must know the configuration of D_{n-1} at the beginning of stage 2. To achieve this, we ensure that robot D_{n-1} always orbits back to its initial configuration $\mathbf{r_{n-1}}$ at the end of stage 1 by adjusting the length of the trajectory for robot D_n (and correspondingly the lengths of the orbit of D_{n-1}). This allows us to use the initial configuration $\mathbf{r_{n-1}}$ as the starting configuration for planning the trajectory of robot D_{n-1} at the beginning of stage 2.

In stage 2, microrobot D_{n-1} is maneuvered from $\mathbf{r_{n-1}}$ to its target configuration, $\mathbf{g_{n-1}}$, using only primitives P_{n-1} and P_{n-2} (see Fig. 2(b)). Both these primitives are sufficient to maneuver robot D_{n-1} to an arbitrary configuration, but cause only straight-line motion in D_n. However, as we described above, we ensured that the intermediate configuration $\mathbf{a_n}$ is chosen such that D_n moves into its target configuration $\mathbf{g_n}$ during its straight-line motion stage 2.

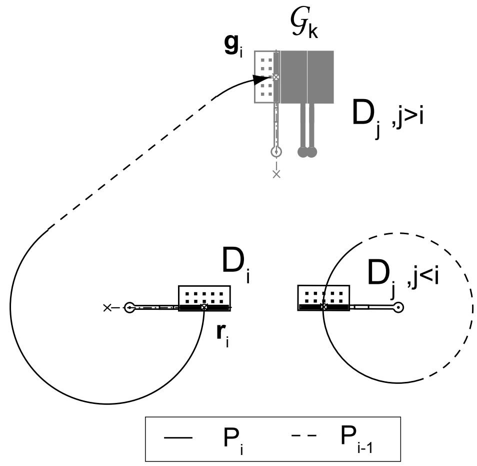


Fig. 3. Progressive docking of single microrobots with the assembling stable shape.

The planned trajectories are then examined for collisions by testing for intersections of the straight path with the swept area of the orbiting robots, as well as D_n and D_{n-1} . If necessary, we modify the trajectories of both D_n and D_{n-1} using geometric collision avoiding heuristics. The length of the trajectory for D_n during stage 1 might need to be readjusted to ensure that D_{n-1} starts in $\mathbf{r_{n-1}}$ at the end of stage 1.

2.2 Steps $2, \cdots, n-1$. Subsequent docking of single robots.

The concept of consecutively (in $n-2$ steps) adding single robot to the initial stable shape $\mathcal{G}_1$ to generate the stable goal-structure $\mathcal{G}_k$ is illustrated in Fig. 3. From this point on, only a single robot is maneuvered at any given time, while the remaining robots are either docked or orbiting. The structure of the control matrix M allows robot D_i to be maneuvered to its target configuration $\mathbf{g_i}$ using control primitives P_i and P_{i-1}, while robots D_j, $j < i$, orbit in place. Control primitives P_i and P_{i-1} cause straight-line motion in robots D_j, $j > i$, but, since our robots are assembled in decreasing order of i, they are already docked and immobilized as part of an intermediate stable structure shape.

As before, the paths are tested for intersection with the swept area of the orbiting devices, as well as the assembling shape. If necessary, we modify the trajectories of each of the robots.

3 Control Strategies for Microassembly

Any physical robotic system is subject to variability affecting its motion, called *control error*, which will perturb the motion of the robot away from its nominal trajectory. In contrast with the motion planning described above, in this section we solve the problem of maneuvering the robots to their target configurations in the presence of control error. We derive online control strategies that reduce the accumulating error through iterative execution and replanning of nominal microrobot trajectories. These control strategies are based on the theory of Error Detection and Recovery (EDR) [2], which allows us to *plan* for variability that occurs during microrobot motion, and to use compliance to further reduce the accumulating control error.

Because of inherent uncertainty in both the pose as well as control of the microrobots, we now use regions as opposed to point (exact) configurations. Let R_i be the starting region for robot D_i, typically a ball around the nominal initial configuration $\mathbf{r_i}$, signifying the pose uncertainty. Correspondingly, let G_i be the region of goal configurations for robot D_i, typically an open set around $\mathbf{g_i}$. Our objective becomes to maneuver the robots from their start region $R = R_1 \times R_2 \times \cdots R_n$, to their goal region $G = G_1 \times G_2 \times \cdots G_n$.

The control strategies are implemented through iterative execution and replanning of the initially planned control sequence S until we recognize that the robots have entered their goal regions G, or our assembly has failed. Iterative replanning of a gross motion plan is used to implement a *gross-motion control strategy*. However, our assembly scheme uses the gross-motion control strategy to maneuver the microrobots to the vicinity of their goal configuration. There, we switch to a *fine-motion control strategy* to complete docking. The fine-motion control strategy is based on interpolated turning and compliant motion, and allows precise control of the docking location of the microrobot.

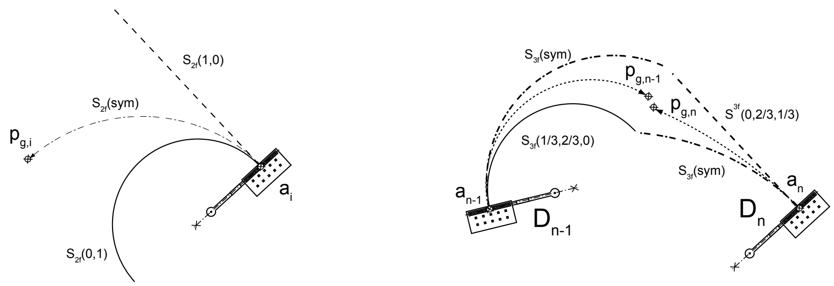
3.1 Fine-Motion Trajectories

Fine-motion trajectories use interpolated turning [3]. i.e. interleaving straight-line and curved trajectory segments, to approximate a turning radius $r'_i > r$, where r is the turning radius of the microrobot with its arm snapped down. Adjusting the ratio between the interleaved trajectory segments allows us to vary r', and construct a fine-motion trajectory between the intermediate configuration $\mathbf{a_i}$ and a target location $\mathbf{p_{g,i}}$, where $\mathbf{p_{g,i}} = (x_{g,i}, y_{g,i}) \in \mathbb{R}^2$ and $(\mathbf{g_i} = (\mathbf{p_{g,i}}, \theta_{g,i})^T = (x_{g,i}, y_{g,i}, \theta_{g,i})^T)$. The radius that allows a microrobot to reach $\mathbf{p_{g,i}}$ from $\mathbf{a_i}$ can be calculated as:

$$r' = \frac{\Delta x^2 + \Delta y^2}{2\,(\Delta x \cos\theta_{a,i} - \Delta y \sin\theta_{a,i})}, \tag{1}$$

where $\Delta x = x_{g,i} - x_{a,i}$ and $\Delta y = y_{g,i} - y_{g,i}$. The control error is compensated for by either reducing or increasing r' such that the fine-motion trajectory passes through $\mathbf{p_{g,i}}$. A change in r' will cause a deviation from the nominal approach angle of a docking microrobot. This deviation is subsequently corrected for using compliant interaction with the docking object.

Fine-motion trajectories are constructed using either three or two primitives, as shown in Figs. 4(a) and 4(b). Two-primitive fine motion trajectories are used to maneuver single robots, while three primitive fine-motion trajectories are are used to simultaneously maneuver two microrobots.



(a) (b)

Fig. 4. Cones of positions ($\mathbf{p_{g,i}}$) that can be be reached using two-primitive fine-motion trajectories (**a**) and three-primitive fine-motion trajectories (**b**).

Two-primitive fine-motion trajectory

Two-primitive fine motion trajectories for single microrobots are represented by the control sequence $S_{2f} = (P_{i,a}, P_{i-1,b}, \cdots, P_{i,a}, P_{i-1,b})$, where $a + b = \mathcal{T}$. Let $\rho_a = \frac{a}{\mathcal{T}}$ and $\rho_b = \frac{b}{\mathcal{T}}$. r' of S_{2f} are parameterized by ρ_a and ρ_b, $S_{2f}(\rho_a, \rho_b)$, is $r'_i = r\left(1 + \frac{\rho_a}{\rho_b}\right)$. A fine-motion trajectory $S_{2f}(\frac{1}{2}, \frac{1}{2})$ is referred to as *symmetric*, and is written $S_{2f}(sym)$. The positions that can be reached using S_{2f} are spanned by the cone of the trajectories of $S_{2f}(0, 1)$ (pure turning), and $S_{2f}(1, 0)$ (straight line motion) starting from $\mathbf{a_i}$ (Fig. 4(a).)

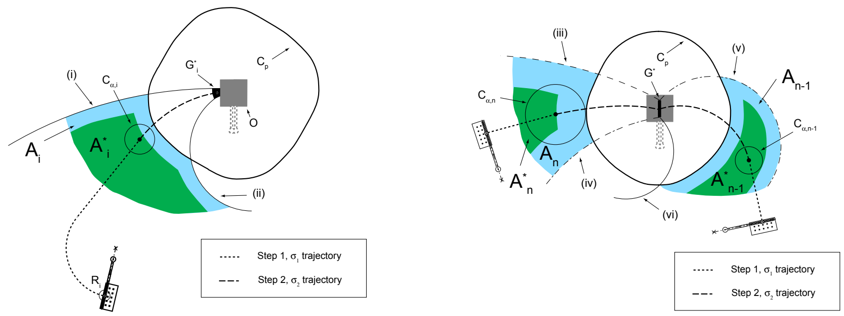
Three-primitive fine-motion trajectory

The three-primitive fine-motion trajectories are used to dock robots D_{n-1} and D_n during the assembly of the initial stable shape, and are represented by the control sequence $S_{3f} = (P_{n,a}, P_{n-1,b}, P_{n-2,c}, \cdots P_{n,a}, P_{n-1,b}, P_{n-2,c})$, where $a + b + c = \mathcal{T}$. As above, $\rho_a = \frac{a}{\mathcal{T}}$, $\rho_b = \frac{b}{\mathcal{T}}$, and $\rho_c = \frac{c}{\mathcal{T}}$, $S_{3f}(\rho_a, \rho_b, \rho_c)$. $S_{3f}(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ is referred to as symmetric, ($S_{3f}(sym)$). The trajectories for D_n and D_{n-1} will differ, because the motion of D_n and D_{n-1} is different during the application of control primitive P_{n-1}. The relationship between the radii r'_n and r'_{n-1} for D_n and D_{n-1} given ρ_a, ρ_b, and ρ_c is:

$$r'_n = r\left(1 + \frac{\rho_b + \rho_c}{\rho_a}\right) \tag{2}$$

$$r'_{n+1} = r\left(1 + \frac{\rho_c}{\rho_b + \rho_a}\right). \tag{3}$$

In order for D_{n+1} to be able to reach $\mathbf{p_{g,n+1}}$, $\mathbf{p_{g,n+1}}$ must lie within the cone spanned by $S_{3f}\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ and $S_{3f}\left(\frac{1}{3}, \frac{2}{3}, 0\right)$, as shown on Fig. 4(a). Correspondingly, $\mathbf{p_{g,n}}$ must be within the cone spanned by $S_{3f}\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ and $S_{3f}\left(0, \frac{2}{3}, \frac{1}{3}\right)$ (Fig. 4(b).)



(a) (b)

Fig. 5. Two-stage EDR control strategies. (**a:**) docking a single robot to a stable shape and (**b:**) assembly of the initial stable shape. Error-bounds (i) – (vi) are defined in Sec. 3.4.

3.2 A Two-step EDR Control Strategy

Having defined fine-motion trajectories, we use the theory of Error Detection and Recovery (EDR) [2] to construct a robust control strategy for implementing microassembly in the presence of uncertainty and compliance.

Our robots are maneuvered to dock with another rigid body, which we denote as O. In the case of docking a single robot to a stable shape, O is the stable shape. In the case of the assembly of the initial stable shape, O is the other robot. In either case, we define C_O as the *proximity space* of O, which is the region from which the microrobot may not be able to avoid colliding with O. The region C_O can be obtained geometrically by expanding the boundary of O in C-space by $2r^*$ (r^* is defined in Fig. 2(b) in [4]). Such an expansion is a simplified geometric approximation to the obstacle transformation method for non-holonomic robots used by for example Papadopoulos and Poulakakis [8].

We construct an EDR strategy to reliably maneuver the robots between their individual start, R_i, and goal, G_i, regions. We assume that the start regions R_i are outside C_O. To simplify the construction of the complete control strategy from R to G we use backward chaining to create a two-step EDR control strategy [2]. During step one, the robots are maneuvered to an intermediate goal region A_i outside C_O, using the gross-motion control strategy. The region A_i is defined as the region outside C_O from which there exists a fine-motion control strategy such that the robot is guaranteed to enter G_i. Consequently, during the second step, the robots are maneuvered from A_i to G_i using the fine-motion control strategy. Figure 5 shows a conceptual illustration of the two-step control strategy for the case of the assembly of the initial stable shape (a), and the sequential docking of single robots (b).

Let Σ_1 be the set of all gross motion control strategies, and Σ_2 be the set of all fine-motion control strategies. Let σ_1 be a strategy in Σ_1 and σ_2 be a strategy in Σ_2. The reachability diagram for the two-step EDR strategy is $R_i \xrightarrow{\sigma_1} A_i \xrightarrow{\sigma_2} G_i$.

We use the preimage terminology and notation from [7, 2]. The strong preimage of Y, $P_{X,\theta}(Y)$, is the region of C-space from which the robot is guaranteed to recognizably enter the region Y when starting in region X and applying control strategy θ. The weak preimage of Y, $\widehat{P}_{X,\theta}(Y)$, is the region of C-space from which the robot might recognizably enter Y, given fortuitous sensing and control events, when starting in X and applying control strategy θ. The forward projection, $F_\theta(X)$, of X under θ is the region of C-space which the robot might reach after the execution of the control strategy θ when starting in region X. Note that our control strategies are based on progressive re-planning and execution of a trajectory towards a nominal target configuration, e.g. $\mathbf{y}$, and consequently, the control strategy θ includes a specific nominal goal $\mathbf{y}$. This also implies that $F_\theta(X)$ does not grow much over time, because the control error is continuously reduced through re-planning of the robot trajectory.

Let C_F be the region outside C_O, $C_F = C - C_O$. We define A_i to be the intersection of C_F, the strong preimage of G_i under σ_2, and the forward projections of R_i under σ_1; $A_i = P_{A_i,\sigma_2}(G_i) \cap C_F \cap F_{\sigma_1}(R_i)$. In addition, for A_i to be guaranteed reachable from R_i, it must hold that $R_i \subset P_{R_i,\sigma_1}(A_i)$. A_i must contain the forward projection of the gross-motion control strategy, σ_1, from R_i, $F_{\sigma_1}(R_i)$. We can bound $F_{\sigma_1}(R_i)$ by a cylinder around a target configuration $\mathbf{a_i}$, $C_{a,i} = B_{r_{a,i}}(\mathbf{a_i}) \times [\theta_{a,i} - h_{a,i}, \theta_{a,i} + h_{a,i}] \subset \mathbb{R}^2 \times S^1$, where $B_{r_{a,i}}(\mathbf{a_i})$ is a ball of radius $r_{a,i}$ around $\mathbf{a_i}$. The size of $C_{a,i}$ is derived in Appendix 3.4, and depends on the control algorithm that implements the control strategy.

We construct the strategies σ_1 and σ_2 using $C_{\alpha,i}$ as the goal and start region, respectively. As we describe in Appendix 3.5, we can replace goal region G_i with a larger region G_i^*, from which the robot can achieve goal G_i using compliance. In order for the cylinder $C_{\alpha,i}$ to be completely contained in $P_{A_i,\sigma_2}(G_i^*)$ while outside C_O, the intermediate target configuration $\mathbf{a_i}$ must be at least $r_{\alpha,i}$ away from the boundaries of $P_{A_i,\sigma_2}(G_i^*)$ and C_O in $\mathbb{R}^2$, and $h_{\alpha,i}$ in S^1. The region A_i^*, defined as the set of all intermediate configurations $\mathbf{a_i}$, can now be obtained geometrically. If $A_i^* = \emptyset$ we can not guarantee that the robot will reach G_i.

We execute the gross-motion control strategy σ_1 for parallel motion of D_n and D_{n-1} and sequential motion of D_i, $i \in Z_{n-2}$ between R_i and $C_{\alpha,i}$, $i \in Z_n$. Once the robots enter A_i, the fine-motion control strategy, σ_2, is executed between the D_i's measured configuration, $\mathbf{a_i^*}$, and G_i^*. The assembly terminates when the robots enter the goal G_i or the failure region H. The failure region $H = F_{\sigma_2}(A) - \widehat{P}_{A_i\sigma_2}(G^*)$ (See [2]), can be approximated as the boundary of O that is not in G_i^* (this is where the devices may get stuck.)

Details regarding the algorithms that implement σ_1 and σ_2 can be found in Appendix 3.3, while the derivations of the error bounds, A_i and $C_{\alpha,i}$ can be found in Appendix 3.4.

3.3 The Control Algorithms

The σ_1 and σ_2 control strategies are implemented through iterative execution and replanning of control sequence S towards a nominal target configuration until we recognize that the robots have entered their goal or failure regions (G_i^*, A_i, or H). This iteration begins with the generation of the control sequence S using the planning algorithms described in Sec. 2, and the current position of the robot. A portion

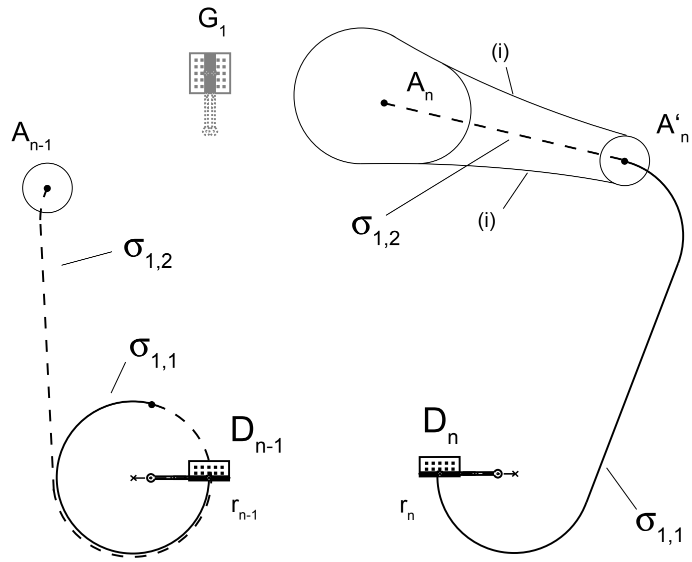


Fig. 6. The trajectories and the goal-regions for D_n and D_{n-1} during the assembly of $\mathcal{G}_1$.

of S, up to the re-planning time interval t_x, is then executed, i.e. the waveforms corresponding to each control primitive $P_i(t)$ are sequentially applied through the operating environment. Following this partial execution of S, the new pose of the robot is measured, and the control cycle is repeated. The growth of $F_{\sigma_1}(R_i)$ over time is proportional to t_x.

However, an extension of this basic re-planning algorithm is required to implement σ_1 on D_n and D_{n-1} during the assembly of the initial stable shape $\mathcal{G}_1$. Even though both D_n and D_{n-1} are controlled simultaneously, error correction can be performed on the trajectory of a single robot only. The assembly is still performed in two stages, however the trajectory of D_n is not corrected during its straight-line motion in stage 2.

The trajectories and goal-regions for the control scheme of assembling $\mathcal{G}_1$ is shown in Fig. 6. Let $\sigma_{1,1}$ be part of the gross-motion control strategy implemented in stage 1, and $\sigma_{1,2}$ be the part of the gross-motion control strategy implemented in stage 2. The region A'_n is analogous to the intermediate configuration $\mathbf{a_n}$ in trajectory planning (See Appendix 2). D_n is controlled to A'_n during stage 1, while D_{n-1} orbit a limit cycle. In stage 2, D_{n-1} is controlled to A_{n-1}, while D_n moves to A_n along a straight-line trajectory. The region A_n is the forward projection of region A'_n after the execution of $\sigma_{1,2}$, $F_{A'_n,\sigma_{1,2}}(A'_n)$. The regions A_n and A_{n-1} are obtained geometrically.

3.4 Error Bounds

Error bounds are used to bound regions $C_{a,i}$ and $C'_{a,i}$, as well as the size of forward-projections and preimages for our control strategies. We derive error bounds from the kinematic model of the microrobot by substituting $\omega = \frac{vah}{r}$ and adding error components v_e and ω_e to the microrobot turning rate (ω) and linear velocity (v):

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \sin\theta \\ \cos\theta \\ 0 \end{pmatrix} (v + v_e) + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} (\omega + \omega_e) \tag{4}$$

.

Let $\hat{v} = v + v_e$, and $\hat{\omega} = \omega + \omega_e$. The displacement with error, $\Delta q_e(t)$, is:

$$\Delta q_e(t) = \begin{pmatrix} \frac{\hat{v}}{\hat{\omega}} \left(\cos\left(\theta - \frac{\pi}{2}\right) + \cos\left(\theta - t\hat{\omega} + \frac{\pi}{2}\right)\right) \\ \frac{\hat{v}}{\hat{\omega}} \left(\sin\left(\theta - \frac{\pi}{2}\right) + \sin\left(\theta - t\hat{\omega} + \frac{\pi}{2}\right)\right) \\ t\hat{\omega} \end{pmatrix} \tag{5}$$

Eq. (5) represents the error bound for a single control primitive. Let $\delta(t) = \int_S \Delta q_e(t) - \Delta q(t) dt$ be the error integrated over the control sequence S, where $\delta(t) = (\delta_x(t), \delta_y(t), \delta_\theta(t))^T$, and $\delta_{xy}(t) = \sqrt{\delta_x(t)^2 + \delta_y(t)^2}$ (error in $\mathbb{R}^2$).

Bounding $F_{\sigma_1}(R_i)$

The size of the forward-projection of R with the gross motion control strategies σ_1, $F_{\sigma_1}(R_i)$, is bounded by the maximum error ($\delta(t)$) that can occur during the execution of the control algorithms described in Appendix 3.3. We start by deriving $C_{a,i}$ as the bound for $F_{\sigma_1}(R_i)$ for D_i, where $i \in Z_{n-2}$ using σ_1 for single robots. We then use these results to derive the bound for $F_{\sigma_1}(R_{n-1})$ and $F_{\sigma_1}(R_n)$ for D_{n-1} and D_n during step 1 of microassembly.

Let $t_\theta = \theta/\hat{\omega}$ be the time it takes the robot to rotate by angle θ while in control state 1. The forward-project of R using σ_1 for the control of single robots from R_i to A_i, $F_{\sigma_1}(R_i)$ for $i \in Z_{n-2}$, is equal to $\delta(t_{2\pi})$. The reason for this is that our microrobots can only turn one way, and correcting a small error may require up to a $2\pi r$ long trajectory, since the robot may have to complete a full circle. Consequently, σ_1 for single robots may not be able to reduce the control error to below $\delta(t_{2\pi})$, thus $F_{\sigma_1}(R_i)$ for D_i with $i \in Z_{n-2}$ is bounded by cylinder $C_{\alpha,i}$ with $r_{\alpha,i} = \delta_{xy}(t_{2\pi})$ and $h_{\alpha,i} = \delta_\theta(t_{2\pi})$.

$F_{\sigma_1}(R_n)$ for D_n is different from $F_{\sigma_1}(R_i)$ for D_i with $i \in Z_{n-2}$ because control error in D_n is not corrected during its straight-line motion in stage 2 of the gross-motion control strategy for D_n and D_{n-1}. Because accumulating control error in D_n during stage 2 is not reduced, the bound $C_{\alpha,n}$ depends on the length of the trajectory for D_{n-1} during stage 2, and is the forward projection of A'_n with $\sigma_{1,2}$ during stage 2, $F_{\sigma_{1,2}}(A'_n)$. Region A'_n can be bounded by $F_{\sigma_{1,1}}(R_n)$ of D_n, which is $C'_{a,n-1}$ with $r'_{a,n-1} = \delta_{xy}(t_{2\pi})$ and $h'_{a,n-1} = \delta_\theta(t_{2\pi})$.

However, $F_{\sigma_{1,2}}(A'_n)$ may be prohibitively large, because of large uncertainly in the length of the trajectory for $\sigma_{1,2}$. For example, the robot D_{n-1} might drive in a straight line to the goal, or the control error might force it to turn 2π during every replanning interval. We can reduce the size of $F_{\sigma_{1,2}}(A'_n)$ through an iterative approach, by assuming a nominal trajectory for D_{n-1} during $\sigma_{1,2}$. When D_{n-1} fails to enter A_{n-1} as D_n enters A_n (meaning that the executed trajectory of D_{n-1} was longer than nominal due to error correction), or D_n fails to enter A_n as D_{n-1} enters A_{n-1} (meaning that the executed trajectory of D_{n-1} was shorter than nominal due to error correction), we simply implement a new gross-motion control strategy σ_1 from current (recorded) configurations of D_n and D_{n-1}. D_{n-1} will now be in a configuration closer to A_{n-1}, reducing the uncertainty in the length of trajectory during $\sigma_{1,2}$. Iterative replanning of σ_1 ensures that the length of of the trajectory along which D_{n-1} travels to A_{n-1} using σ_1 approaches an upper bound of $s_{4\pi}$, where $s_{4\pi}$ is the length of the trajectory for D_{n-1} to turn 4π (complete two full turns). Consequently, the size of A_n approaches the *timed forward projection* [6], $\mathrm{F}_{\sigma_{1,2}}(A'_n, t_{4\pi})$, where $t_{4\pi}$ is the time it takes for D_{n-1} to turn 4π. $\mathrm{F}_{\sigma_{1,}}(A'_n, t_{4\pi})$ can be derived geometrically, using error bounds obtained by integrating Eq. (5) over straight-line motion of D_n with duration $t_{4\pi}$.

Because D_{n-1} is controlled last, i.e. during stage 2, $F_{\sigma_1}(R_{n-1})$ for D_{n-1} is identical in size to the region $F_{\sigma_1}(R_i)$, $i \in Z_{n-2}$, as any control error is removed by replanning the trajectory of D_{n-1}.

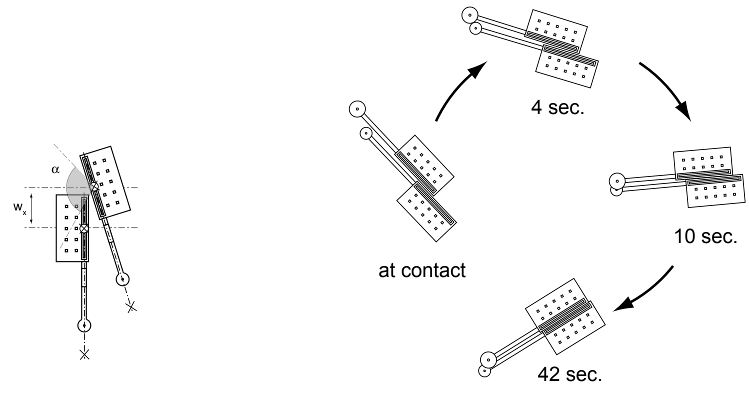
Error Bounds for Fine-Motion Control Strategies (σ_2)

The forward-projection of the fine-motion control strategies is smaller that of the gross-motion control strategies ($F_{\sigma_1}(A_i)$), and its size approaches the uncertainty in configuration of the robot D_i.

The error bound for two-primitive fine-motion trajectories are obtained by integrating Eq. (5) over the control sequence for either all turning or all straight-line trajectories (i.e. successively applying the control primitives), defined through control sequences $S_{2f}(0,1)$ and $S_{2f}(1,0)$. With respect to three-primitive fine-motion trajectories, the error bounds for D_n are derived by integrating Eq. (5) over the range of control sequence that can be used to vary its trajectory without changing the trajectory of D_{n-1}; namely $S_{3f}(\frac{1}{3},\frac{2}{3},0)$ and $S_{3f}(sym)$. The error bounds for D_{n-1} are derived in a similar fashion by integrating Eq. (5) over $S_{3f}(sym)$ and $S_{3f}(0,\frac{2}{3},\frac{1}{3})$.

3.5 Compliance

The use of fine-motion control strategies relies on the ability to reduce the accumulating error in rotation of the docking microrobot. We use compliance between the docking robots to reduce this error. We plan for compliance by considering a larger goal-region G^*, from which the robots are guaranteed to enter G using compliance. We observed two distinct types of compliance; self-aligning compliance between the two robots that dock to form the initial stable shape, and docking compliance in the case of a microrobot docking with a stable shape.



(a) (b)

Fig. 7. Self-aligning compliance. (a) Limits for self-aligning compliance; $w_y < 80\mu$m and $\alpha < 60°$. (b) An example of self-aligning compliance during the assembly of $\mathcal{G}_1$. Outlines of D_n and D_{n-1} recorded four times during a self-aligning experiment. The initial shape rotates by 79° overall.

Self-aligning Compliance

Two microrobots that dock to form the initial stable shape $\mathcal{G}_1$ self-align during the application of a power-delivery waveform. The opposing robots slide relative to one another until the front edge of both robots is aligned, and they reach a stable configuration. Self-alignment is a form of local, pairwise self-assembly, however the underlying alignment mechanics are not fully understood. Empirical data indicate that self-alignment occurs if the incident angle at which both robots dock (α) is within ± 60°, and the position misalignment is bounded by ± 80 μm (for simplification we only measure position, and not angular, misalignment, see Fig. 7(a)). The goal regions G_n and G_{n-1} for both robots can be enlarged correspondingly, resulting in the expanded goal regions G_n^* and G_{n-1}^*. Fig. 7(b) shows an example of self-alignment between two docking robots. Outlines of the two devices measured four times during a self-aligning experiment are shown, illustrating the reduction in relative error. Note that the initial shape rotates while the two robots self-align.

Docking Compliance

Compliance between a single robot and a stable shape occurs when the robot is commanded to move forward with the steering arm in the elevated position, and one of its corners makes contact with the stable shape. When the corner of the robot contacts the edge of the stable shape (at point c_i), the robot rotates around this point (the steering arm is always elevated during docking), and aligns with the flat edge of the stable shape. The alignment occurs only if the incident angle (α) of the robot approaching the object is within its sticking cone (the range of incident angles at which the corner

c_o will stick to the object), which we have conservatively bounded by ± 45°. Similar to self-aligning compliance, docking compliance allows us to enlarge the preimage of the goal, effectively enlarging the target region G_i by the sticking cone, resulting in the expanded target region G_i^*. Adjusting r_i' in the fine-motion strategy allows us to precisely control the location of c_i.

References

1. S. A. Cameron. Collision detection by four-dimensional intersection testing. In *in Proceeedings of the IEEE International Conference on Robotics and Automation, (ICRA 1990*, pages 291–302, 1990.
2. B. R. Donald. *Error Detection and Recovery in Robotics. Lecture Notes in Computer Science*, volume 336. Springer-Verlag, 1987.
3. B. R. Donald, C. G. Levey, C. McGray, I. Paprotny, and D. Rus. An untethered, electrostatic, globally-controllable MEMS micro-robot. *Journal of Microelectromechanical Systems*, 15(1):1–15, January 2006.
4. B. R. Donald, C. G. Levey, I. Paprotny, and D. Rus. Simultaneous control of multiple mems microrobots. In *submitted to WAFR 2008: The Eighth International Workshop on the Algorithmic Foundations of Robotics*, Guanajuato, Mexico, 2008.
5. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.
6. M. A. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1984.
7. T. Lozano-Perez, M. T. Mason, and R. H. Taylor. Authomatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1), 1984.
8. E. Papadopoulos and I. Poulakakis. Planning and obstacle avoidance for mobile robots. In *the proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 3967–3972, Seoul, Korea, May 21th - 26th 2001.