Group-Control Motion Planning Framework for Microrobot Swarms in a Global Field

Siyu Li[1], Afagh Mehri Shervedani[1], Miloš Žefran[1], and Igor Paprotny[1]

University of Illinois Chicago, Chicago IL 60607, USA
{sli230,amehri2,mzefran,paprotny}@uic.edu

Abstract. This paper investigates how group-control can be effectively used for motion planning for microrobot swarms in a global field. We prove that Small-Time Local Controllability (STLC) in robot positions is achievable through group-control, with the minimum number of groups required for STLC being $\log_2(n+2)+1$ for n robots. We then discuss the complexity trade-offs between control and motion planning. We show how motion planning can be simplified if appropriate primitives can be achieved through more complex control actions. We identify motion planning problems that balance the number of robot groups and motion primitives with planning complexity. Various instantiations of these motion planning problems are explored, with simulations to demonstrate the effectiveness of group-control.

Keywords: Micro Robots · Global Field · Control and Motion Planning.

1 Introduction

Microrobots have gained significant attention for their potential in medical, environmental, and industrial applications. Effective control mechanisms are crucial to enable their diverse functionalities. Various control modalities have been demonstrated for microrobots, including magnetic control [16, 22, 23], optical control [2, 9, 15, 19], and acoustic control [1, 10]. These systems are controlled by global fields, and as a result, they are massively underactuated.

Currently, parallel control of multiple microrobots in a global field is based on the design of robots with distinct physical characteristics, resulting in different responses of each robot to the same control signal. The Global Control Selective Response (GCSR) paradigm introduced in [7, 8] uses fabrication to make each robot respond appropriately to the control signal. Turning-rate Selective Control (TSC), introduced in [20], is a variation of GCSR and differentiates robots by explicitly designing them with different turning rates. These methods scale poorly to larger swarms due to fabrication complexities. Ensemble Control (EC), proposed by [3], leverages differences in the linear velocity and turning rate parameters among robots stemming from randomness in fabrication to control the position of individual robots. Unfortunately, the ability to control individual robots is inherently limited by noise, so this approach also scales poorly.

This paper focuses on electrostatic stress-engineered MEMS microrobots (MicroStressBot), originally developed in [6]. The swarm is powered by a uniform electrostatic field generated by a substrate, which means that all robots are controlled by a single global signal. Their size limits onboard control logic and power storage, making it essential to simultaneously coordinate large groups of microrobots for applications like micro-assembly or drug delivery. Our work explores the concept of on-board multi-stage Physical Finite-State Machines (PFSMs) introduced in [21]. PFSMs allow robots to be individually addressed and activated one by one. Our previous work [17] takes the idea of PFSMs further by introducing *group-based control*, where several robots can be activated together as a group. In this paper, we investigate the trade-off between the complexity of motion planning and control within the *group-control* framework. In particular, we introduce several instances of control and motion planning problems and discuss their scalability.

Motion planning for microrobots involves determining a sequence of movements to reach a given configuration while avoiding obstacles and collisions. Various methods exist, including graph-based methods like Dijkstra's and A^* search [13], sampling-based algorithms like Rapidly-exploring Random Trees (RRTs) [14] and optimization-based methods [11, 29]. Recent advances in machine learning, such as [24, 26, 27], leverage neural networks to speed up RRTs and control techniques for mobile robot navigation. However, despite a wealth of solutions for motion planning problems, motion planning for microrobot swarms in a global field remains a challenging problem because the system is high-dimensional yet massively underactuated, having a single control signal.

Fundamentally, when a microrobot swarm is controlled by a global control signal, the robots can be controlled individually only when each robot responds to the control signal differently. In this paper, we introduce the *group-control* framework to differentiate the motion of each microrobot in the swarm. We prove that Small-Time Local Controllability (STLC) is achievable in robot positions given an appropriate group allocation, with the minimum number of groups required for STLC being $log_2(n+2)+1$ for n robots. Additionally, we discuss the complexity trade-offs between control and motion planning. We identify motion planning problems that balance the number of robot groups and motion primitives with planning complexity. Various instantiations of these motion planning problems are explored, with simulations to demonstrate the scalability and effectiveness of the proposed algorithms.

2 Background

2.1 MicroStressBot

The MicroStressBot is a 120 μm $\times$ 60 μm $\times$ 10 μm mobile microrobot platform introduced in [6]. A MicroStressBot has two actuated internal degrees of freedom (DOF): an untethered scratch-drive actuator (USDA) that provides forward motion and a steering-arm actuator that determines whether the robot moves in a straight line or rotates. A single MicroStressBot can have its arm raised

or lowered, depending on the voltage applied across a substrate formed by an electrode array. When a sufficiently high voltage is applied to the substrate, the arm is pulled into contact with the substrate, and the robot rotates around the contact point. In contrast, when the voltage is reduced below a threshold, the arm is raised and the robot moves forward.

MicroStressBot control has been successfully implemented in [8]. It has been shown that if the pull-down and release voltages of the robots are different, they can be independently controlled. However, it is difficult to consistently fabricate robots so that they respond in the desired way. As a result, the approach scales poorly.

2.2 Physical FSM Robots

One solution to dramatically improve the scalability of the microrobot swarm control is to make the robots respond to a temporal sequence of (a small number of) voltage levels rather than to the voltage directly. Finite-State Machines (FSMs) can accept a set of input sequences [25] (sequences of control signal levels). Previously, in [21], we proposed the on-board MEMS Physical FSM (PFSM) that, upon acceptance of a unique control signal sequence, causes the arm of the MicroStressBot to be pulled up or down. PFSMs can be constructed from several basic modules that are combined together and thus fabricated efficiently. In this work, we build on this idea to propose *group-control*. In particular, we use the fact that several PFSM modules, each corresponding to one group, can be combined together, so that each robot can belong to several groups.

2.3 Group-Control

The core idea of *group-control* is that by equipping the robots with PFSMs, they can be assigned to (several) different groups. When the activation sequence corresponding to a group is sent to the swarm, all the robots that belong to the group are activated. In this paper, activating a MicroStressBot corresponds to raising its arm so that it can move forward (translate); otherwise, the robot rotates. By assigning each robot to a unique subset of groups, we can differentiate between the robots and make them respond differently to a sequence of inputs that activate all the groups one after the other, each time moving the robots belonging to the group for some distance.

Table 1 shows how six robots can be assigned to different groups. In order to make the selection of groups unique to each robot, if we have n robots, we need $m = \log_2(n+2) + 1$ groups ($n+2$ rather than n because each robot needs to belong to at least one group so it can move forward, and no robot can belong to all groups so it can rotate). We add one more group where none of the robots translate, they all rotate in place. If we have m groups, this special group will be the group G_m.

As can be easily seen from Table 1, group assignment corresponds to assigning a unique bit pattern (a subset of groups) to each robot, where each bit

corresponds to a group (labeled G_i, $i = 1, 2, 3$ in Table 1). The robot then belongs to the groups where the corresponding bit equals 1. For example, robot R_1 belongs to the group G_3. Instead, robot R_3 belongs to the groups G_2 and G_3, etc. The group G_4 is the group with all 0's, representing the additional group where all robots are rotating. It is clear that this method guarantees that each robot has a distinct group allocation. Also, note that when $n = 2^{m-1} - 2$ (the maximum number of robots that can be controlled by m groups) each group (apart from the special group G_m) contains exactly half, that is, $\frac{n}{2} = 2^{m-2} - 1$, robots.

	Robot					
Group	R_1	R_2	R_3	R_4	R_5	R_6
G_1	0	0	0	1	1	1
G_2	0	1	1	0	0	1
G_3	1	0	1	0	1	0
G_4	0	0	0	0	0	0

Table 1: Allocating 6 robots to 3 groups.

The group allocation can be realized by the on-board PFSM. At each time step k, only one group of robots is activated, and the robots belonging to that group translate (move forward) while the remaining robots rotate. We call a sequence of group selections an *activation sequence*.

If m is the number of groups and n is the number of robots, we would need $k = O(\log_2 m) = O(\log_2(\log_2(n + 2) + 1))$ bits to select a group through a PFSM [21]. Thus, PFSMs for group-based control can be significantly simpler than in the case when each robot needs to be selected individually.

3 Modeling

3.1 Dynamics

To describe *group-control* mathematically, we start with a dynamic model of a MicroStressBot. The robot can move freely on a horizontal plane, so its configuration space is Euclidean group $SE(2)$. We will describe the state of the robot i with a vector $[x_i, y_i, \theta_i]^T$, where x_i and y_i are the Cartesian coordinates of the pivot point of the robot and θ_i is the robot orientation. The equations of motion are given by:

$$\frac{d}{dt}\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} = a_i \cdot \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{bmatrix} \cdot u + (1 - a_i) \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdot \omega_i, \tag{1}$$

where $a_i \in \{0, 1\}$ is the switching input that determines whether the robot is moving forward or turning in place, while $u \in \mathbb{R}^+$ and $\omega \in \mathbb{R}^+$ are the rates of forward motion and rotation, respectively, where $\omega_i = u/r_i$ with r_i being the

turning radius of each robot. If the control inputs are piecewise constant over each epoch ΔT, we obtain the following discrete-time model:

$$\begin{bmatrix} x_i(k+1) \\ y_i(k+1) \\ \theta_i(k+1) \end{bmatrix} = \begin{bmatrix} x_i(k) \\ y_i(k) \\ \theta_i(k) \end{bmatrix} + \begin{bmatrix} a_i(k)\cos\theta_i(k) \\ a_i(k)\sin\theta_i(k) \\ (1-a_i(k))1/r_i \end{bmatrix} u(k)\cdot\Delta T. \tag{2}$$

Note that u is a unilateral control input. In other words, MicroStressBot cannot go backwards or turn counterclockwise.

3.2 Switched and Embedded Systems

Group-control corresponds to setting the switching input a_i of the robot i to 1 if it belongs to the activated group or to 0 otherwise. *Group-control* with m groups thus turns the swarm into a switched system with m discrete states. Such *m-switched system* has a system state $q(t) \in \mathbb{R}^N$ that evolves according to the following dynamics:

$$\dot{q}(t) = f_{v(t)}(t, q(t), u(t)), \quad q(t_0) = q_0, \tag{3}$$

where at each $t > t_0$, the switching control $v(t) \in \{1, 2, \ldots, m\}$, and the control input $u(t) \in \mathbb{R}^M$ for some M (in our case $M = 1$). In the case of n robots that move in $SE(2)$, we have $N = 3n$. The vector fields $f_i : \mathbb{R} \times \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N, \quad i \in \{1, 2, \ldots, m\}$ are $\mathcal{C}^1$ and we assume that $u(t)$ is measurable. Note that the evolution of the state $q(t)$ governed by Eq. (3) does not experience any discontinuous jumps.

By introducing new variables $v_i(t) \in [0, 1]$ that satisfy the constraint $\sum_{i=1}^{m} v_i(t) = 1$, the switched system (3) can be converted to the embedded form [4, 28]:

$$\dot{q}(t) = \sum_{i=1}^{m} v_i(t) f_i(t, q(t), u_i(t)), \quad q(t_0) = q_0, \tag{4}$$

where u_i is the control input for each vector field f_i. It can be shown [4] that the set of trajectories of the switched system (3) is a dense subset of the set of trajectories of the embedded system (4). Consequently, any trajectory of the embedded system can be approximated by the switched system to any desired accuracy, which means that the controllability of the switched system is equivalent to the controllability of the embedded system.

4 Controllability Analysis

In order for the robots to navigate among obstacles, the system, in general, needs to be Small-Time Locally Controllable (STLC). Otherwise, there are configurations in the obstacle-free configuration space C_{free} that will always lead to a collision. First, we formally define the notion of STLC that will be of interest in this paper. Let $q(t)$ be the combined state of the robot swarm, where all robot

positions are first stacked together in a $2n$ vector, followed by a n vector of robot orientations. Let $p(t) = q(t)[1:2n]$ be the position states of the robots, and let $\Theta(t) = q(t)[2n+1:3n]$ be the orientations. Let $\mathcal{R}_{(p_0,\Theta_0)}(T)$ be the set of all positions reachable by the trajectories of the system starting at $q(0) = (p_0, \Theta_0)$ in time no greater than T.

Definition 1. *The system is small-time locally controllable (STLC) in position states about an initial state (p_0, Θ_0), if p_0 is contained in the interior of the reachable sets $\mathcal{R}_{(p_0,\Theta_0)(T)}$ for all $T > 0$. [12]*

Note that each MicroStressBot in the swarm has $SE(2)$ symmetry (if the initial configuration of the robot is shifted, its trajectories will simply shift by the same amount). So, STLC at one initial configuration implies STLC everywhere.

In Eq. (1), we assumed that MicroStressBot i has the turning radius r_i. If each robot has a different turning radius, only two groups are needed to achieve STLC through group-based control: group G_1 that translates all robots and group G_2 that rotates all robots. To see this, consider the three-step primitive $P_1 = (G_1(d), G_2(\pi^{r_1}), G_1(d))$ where $G_1(d)$ moves all robots forward for d, $G_2(\pi^{r_1})$ rotates all robots so that robot 1 rotates exactly for π radians, and then $G_1(d)$ again moves all the robots forward for d, making robot 1 return to its starting position. Then the sequence $P_2 = (P_1, G_2((\pi - \Delta\theta_2)^{r_2}), P_1)$ moves robots 1 and 2 back to their initial position, where $\Delta\theta_2$ is the angular displacement of robot 2 due to P_1. Iteratively, P_{n-1} moves only one robot, while all others return to the starting position. By first rotating the robot so that the resulting movement due to P_{n-1} is in the desired direction, we can clearly individually move any robot in any direction, showing that the system is STLC. Note that a similar argument has been used in [20] to introduce TSC.

Assuming that all the robots have different turning rates again shifts the burden of control to fabrication and, in general, scales poorly. In the rest of the paper, we thus **assume that all robots have the same turning rate** r. We will show that *group-control* still guarantees STLC.

Remark 1. Given a collection of sets of robots $\{S_0, S_1, \ldots\}$ with each S_i containing robots with the same turning rate, and S_i and S_j having different rates if $i \neq j$, STLC can be achieved by combining the approach above with the methodology that will be described below within each set.

4.1 Unilateral Control to Bilateral Control

As stated above, in the rest of the paper, we only consider the case where all robots have the same turning radius r. In [17], we showed that a swarm of MicroStressBots under *group-control* is (globally) controllable (even without using the group where all robots rotate). However, we were unable to demonstrate that the system is STLC. In this paper, we show that by adding the group where all the robots rotate, STLC can be achieved.

One of the challenges in showing that our system is STLC is that the robots are controlled unilaterally. In other words, they can only move forward but not

backward. Similarly, they can rotate clockwise but not counter-clockwise. This is the result of the control input u in Eq. (4) being positive. We show that this restriction can be relaxed.

We return to the switched system in Eq. (1). Let

$$q(t) = [x_1, y_1, x_2, y_2, \ldots, x_n, y_n, \theta_1, \theta_2 \ldots, \theta_n]$$

be the state of the swarm, where $q(t) \in \mathbb{R}^{3n}$. We can see that $q(t)[1:2n]$ are the *position states*, with $q(t)[2j-1:2j]$ representing the position of robot j. Also, $q(t)[2n+1:3n]$ are the *orientation states*, where $x(t)[2n+j]$ is the orientation of the robot j. Next, let $\alpha_i = [\alpha_{i,1}, \ldots, \alpha_{i,n}]^T, i = 1, \ldots, m$ be the activation vector corresponding to the group i being activated. In other words, $\alpha_{i,j} = 1$ if robot j belongs to group i, and 0 otherwise. The overall swarm dynamics can then be written as follows:

$$\dot{q}(t) = f_{\nu(t)}(q(t)) \cdot u(t) \quad u(t) \in \mathbb{R}^+ \tag{5}$$

where $\nu(t) \in \{1, \ldots, m\}$, and for each i, f_i is obtained by choosing $a_j = \alpha_{i,j}$ in Eq. (1). This equation describes a switched driftless control-affine system [5,18].

We introduce the notion of induced vector fields g_i and h_i such that $f_i = g_i + h_i$. Here, g_i contains only position state information, and h_i contains only the orientation state information. Given that G_m is the group where all robots rotate and the rotation radius of all is equal, let $f_m(\pi)$ represent the rotation of all robots by π radians using the rotation-only vector field.

Proposition 1. *The control sequence $(f_i(d), f_m(\pi), f_i(d), f_m(\pi))$, where $f_i(d)$ corresponds to the activation of group G_i so the robots in the group translate for d, corresponds to a vector field $h_i(\frac{2d}{r})$ that rotates the robots that do not belong to the group i by $\frac{2d}{r}$ and keeps the rest of the robots where they were.*

Proof. Observe that $f_i(d)$ translates the robots that belong to G_i for d without changing their orientation and rotates the robots that do not belong to G_i for $\frac{d}{r}$. The application of $f_m(\pi)$ rotates all the robots for π, which means that the robots that belong to the group G_i now point in the opposite direction. The second application of $f_i(d)$ then returns the robots that belong to G_i to their initial position while adding another $\frac{d}{r}$ to the orientation of the robots that do not belong to G_i. Finally, $f_m(\pi)$ rotate the robots in G_i to their original orientation. The robots not in G_i have instead rotated in total for $\frac{2d}{r} + 2\pi = \frac{2d}{r}$. □

Remark 2. Observe that for any angle $\theta \in [0, 2\pi]$, $h_i(-\theta) = h_i(2\pi - \theta)$. The orientation vector field h_i is thus bilateral.

Once we have the orientation vector field h_i, we can easily obtain the translation vector field g_i:

Proposition 2. *The control sequence $(f_i(d), h_i(-\frac{d}{r}))$ corresponds to a vector field $g_i(d)$ that translates the robots in G_i for d and leaves all the other robots where they were.*

Remark 3. We can also easily see that $(f_m(\pi), g_i(d), h_i(\pi)) = g_i(-d)$. Thus, the translation vector field g_i is bilateral.

Note that while h_i rotates the robots that are not in G_i in place, the other robots travel back and forth and could potentially collide with an obstacle. However, we can restrict the motion of the robots in G_i to a small ball of radius ϵ and apply $h_i(\frac{\epsilon}{r})$ multiple times to obtain the desired rotation of the robots not in G_i without the robots in G_i leaving a small neighborhood of their initial location.

Technically, the rotations we used in the constructions above cannot be performed in zero time. However, for the purpose of position control, they can be assumed to take place instantaneously. Therefore, in the rest of the paper, we can safely assume that the control input can be bilateral.

With the construction above, the original embedded system in Eq. (4) with unilateral vector fields $\{f_i, \ldots, f_m\}$ becomes an embedded system with **bilateral** vector fields

$$\mathcal{F} = \{g_1, \ldots, g_{m-1}, h_1, \ldots, h_{m-1}, f_m\}. \tag{6}$$

4.2 Control of Robot Orientation

We have shown that each group G_i induces a vector field g_i that only translates the robots in the group and a vector field h_i that just rotates the robots that are not in G_i. What we show next is that we can independently control the orientation of at least 3 robots:

Proposition 3. *The orientation of any 3 robots can be controlled to any desired orientation $(\theta_1, \theta_2, \theta_3)$. Moreover, there are at most m robots that can be controlled to any desired orientation $(\theta_1, \ldots, \theta_m)$.*

Note that the rest of the swarm that cannot be directly controlled will rotate in place to some orientation.

To illustrate this proposition, consider the rotation vector fields $g_1, \ldots, g_{m-1}$ in (6) and the additional original rotation-only vector field f_m. All of these vector fields can rotate the robots without translating them. The proposition can then be proved by observing that, using these vector fields, the orientations of the robots linearly depend on the arguments (rotation angles) of these vector fields. For example, for the 4-group case in Table 1, we get the equation:

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T = q[2n+1:3n] - q[2n+1:3n](0) \tag{7}$$

where $q[2n+1:3n]$ are the orientations of the robot after the application of the vector fields, and $q[2n+1:3n](0)$ are their initial orientations.

It is evident that for m groups, the rank of the matrix in the equation above is at most m. This implies that at most m of the n equations can be solved exactly for the desired inputs u_i, indicating that the orientation can be controlled for at most m robots.

Next, we illustrate that the orientation of any 3 robots can be controlled. To see this, note that if one just considers the first $m-1$ columns of the matrix, any two rows are distinct (since no two robots belong to the same set of groups), which means that at least two of them are linearly independent. Adding the last column with all the ones, it is clear that any additional row will be linearly independent from these two rows, which means that the matrix will have a rank of at least 3 and that any 3 of the n equations can be solved exactly for the desired inputs u_i.

4.3 Small-Time Local Controllability (STLC)

After we have identified the control vector fields in Eq. (6), we can proceed to show that the swarm is STLC in positions. Traditionally, we can do this by analyzing the closure of $\mathcal{F}$ under the Lie bracket operation. However, we will show that we can independently translate any robot in arbitrary direction without moving any other robot. This shows that the system is STLC in positions.

Suppose that we want to independently translate some robot $k \in \{1, \ldots, n\}$. Let $G_{\hat{k}}$ be any group that contains the robot k. Let $\mathrm{Rot}_{i_1,i_2,i_3}(\theta_1, \theta_2, \theta_3)$ correspond to setting the orientation of robots i_1, i_2, i_3 to $\theta_1, \theta_2, \theta_3$ as described above. Also, assume that the members of the group $G_{\hat{k}}$ are robots $l_1, \ldots, l_j, k$.

We can then state the following:

Proposition 4. *The control sequence* $P_1(d) = (g_{\hat{k}}(d), \mathrm{Rot}_{l_1,l_2,k}(\pi, \pi, 0), g_{\hat{k}}(d))$ *only translates robots* $l_3, \ldots, l_j, k$.

Proof. Observe that under the control sequence above, the first 2 robots in $G_{\hat{k}}$ will travel back and forth while the robot k will undergo translation for $2d$. The rest of the robots in the group will undergo some unspecified motion.

Once we have the control sequence P_1, it is easy to see that a control sequence $P_2 = (P_1(d), \mathrm{Rot}_{l_3,l_4,k}(\pi - \Delta\theta_{l_3}, \pi - \Delta\theta_{l_4}, 0), P_1(d))$ will only translate robots $l_5, \ldots, l_j, k$, where $\Delta\theta_{l_3}, \Delta\theta_{l_4}$ are rotations of robots l_3, l_4 caused by $P_1(d)$. By repeating this process at most $l_j/2$ times allows us to eliminate the translation of all robots $l_1, \ldots, l_j$ except for the robot k. Of course, we can always initally rotate the robot k so that the final translation is in the desired direction, showing that each robot can be translated independently and that the system is STLC in positions. □

5 Motion Planning

After we have shown that the MicroStressBot swarm under *group-control* is STLC, we consider the motion planning problem for the swarm. Given our proof of STLC, the problem is not how to move the swarm to a particular configuration,

but how to move it **efficiently**. In other words, ideally, we would like the robots to move to their desired configuration in parallel and not one by one. However, planning such parallel motion directly is complex due to the high dimension of the configuration space and the fact that we have a single control input. In this paper, we argue that effective motion planning for global *group-control* in high-dimensional state spaces should involve the design of **motion primitives**. In particular, motion primitives will be associated with **subgroups**, trading off the complexity of implementing additional groups in hardware (using PFSMs) with the complexity of control.

The idea of motion primitives is to divide a robot group into smaller subgroups. These primitives simplify the motion planning task by reducing the coupling between robots in the group. We present primitives inspired by Lie bracket motions, following the intuition that, in general, the higher the order of the Lie bracket, the fewer robots are moved by it. It is important to note that, in this paper, these subgroups are logical constructs as opposed to $\log(n+2)+1$ physical groups. However, these logical groups could be implemented in hardware, increasing the number of physical groups and simplifying motion planning and control for the swarm. Of course, this would come with increased fabrication complexity and cost.

One trade-off of interest in this paper is between planning complexity and path length. A primitive corresponding to a smaller subgroup will, in general, involve the recursive application of primitives associated with larger subgroups, as can be observed in the proof of Proposition 4. Because of this, such primitives will involve lots of back-and-forth motions of the robots, resulting in longer path lengths. However, motion plans that involve such primitives will be simpler. On the other hand, when the primitives used for motion planning involve larger subgroups (not much back-and-forth), the motion of the robots is coupled, which means that the motion plan needs to be more complex, resulting in a longer overall path because of this. Fig.1 illustrates these trade-offs. Point A represents the case where a primitive moves a single robot, while point C corresponds to the case where no primitives are used, only the original hardware groups. The blue region indicates that primitives with fewer robots result in more complex paths, while the red region shows that subgroups that are more coupled will increase the complexity of the plan. We hypothesize that there is a good medium (point B) where the motion plans are not that complex, and the primitives are relatively efficient in terms of the path length.

In the following section, we investigate how the complexity of motion planning can be simplified by using motion primitives. Subsequently, we study different instantiations of the motion planning problem through simulations to explore the trade-offs in Fig. 1.

5.1 Motion Planning Approximation Scheme

We start by giving some assumptions and definitions that are fundamental to the *group-control* framework. We start with the following **assumptions**:

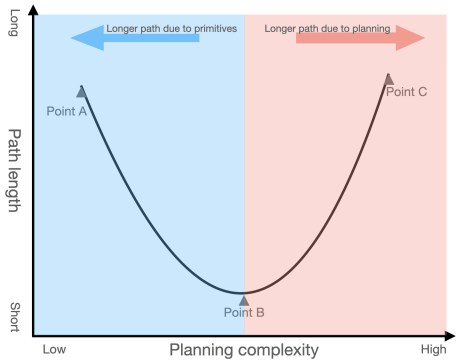


Fig. 1: Trade-off between complexity of motion planning and path length.

- Robots are controlled by a global signal that operates them in groups, with one group activated at a time.
- A predefined map of the area is available and known.
- Each robot is assigned a starting point and a goal configuration. The starting point includes the robot's position and orientation, whereas the goal configuration only involves position, the orientation can be arbitrary.

Unlike physical groups (encoded through PFSMs), a subgroup and the associated motion primitive are a logical construct. A subgroup contains fewer robots than physical groups, and when a subgroup is activated, only those robots move forward. This abstraction will allow us to formulate the motion planning problem in layers of increasing difficulty. Physically, the motion induced by a given subgroup p corresponds to a particular Lie bracket and can thus be physically realized. We will refer to such motion as a *motion primitive associated with the subgroup p*.

Definition 2. ***[Primitive Order]*** *The order of a primitive corresponding to the subgroup p is the order of the Lie bracket used to realize it. The order of the Lie bracket is the number of generators from the set of vector fields $\mathcal{F}$ in Eq.* (6) *in it.*

In general, the higher the order of a Lie bracket, the fewer robots will move by that bracket. However, there is no one-to-one correspondence between the order of the Lie bracket and the number of robots it moves, as this also depends on the vector fields involved in the Lie bracket. Fundamentally, the order of the Lie bracket indicates the difficulty of its implementation; the higher order is more difficult to implement. On the other hand, the number of robots affected by a Lie bracket reflects the complexity of motion planning; the more robots move at the same time, the more difficult it is to move each of them to the desired configuration.

Definition 3. ***Motion Planning Problem Abstraction*** *The motion planning problem for n robots of level r, denoted as $\mathcal{M}_s^g(n, r)$, corresponds to the problem of finding a collision-free trajectory (sequence of motion primitives) for*

n robots controlled by a global field from some initial state s to a goal state g, where the motion primitives that are used have the primitive order at most r.

Proposition 5. *Consider the motion planning problem $\mathcal{M}_s^g(n, r_k)$ for n robots and $r_k \leq r_s$, where r_s is the primitive order required to move each individual robot. If $r_1 < r_2 \leq r_s$, the complexity of the motion planning problem $\mathcal{M}_s^g(n, r_2)$ is less than $\mathcal{M}_s^g(n, r_1)$. Initially, when $r_k = r_s$, the complexity of the motion planning problem $\mathcal{M}_s^g(n, r_s)$ is linear in the number of robots n. Moreover, for any other r_k, the complexity of the motion planning problem $\mathcal{M}_s^g(n, r_k)$ has a upper bound $o(nL^{r_s - r_k})$, assuming that moving along the first-order Lie bracket has a constant complexity L.*

Proof. Consider $r_k = r_s$. In this case, there is only one robot moved by each Lie bracket. The complexity in this case is linear in the number of robots, since moving one robot to the goal takes a constant time. Secondly, when $r_k = r_s - c$, each Lie bracket moves multiple robots. Moving only one robot is equivalent to using the c-th order Lie bracket involving vector fields that are Lie brackets of order r_k. Thus, the complexity is $o(nL^c) = o(nL^{r_s - r_k})$. In reality, multiple robots could move in parallel, so we consider this complexity to be the upper bound. □

Example 1. Continuing with the example of 6 robots with 4 groups, consider the motion planning problem $\mathcal{M}(n = 6, r_k = 3)$ where the minimum order to move each robot is $r_s = r_k = 3$. Further, we select six primitives involving the highest-order Lie brackets that move each robot individually. The problem $\mathcal{M}(6, 3)$ reduces to rotating each robot to the goal and then moving it using its corresponding primitive. Since this plan for each robot takes constant time, the complexity of the motion planning problem $\mathcal{M}(6, 3)$ is linear in the number of robots.

For the motion planning problem $\mathcal{M}(n = 6, r_k = 2)$, the first-order Lie brackets are chosen as the primitives. Each primitive moves two robots simultaneously. For example, $[h_2, g_1]$ moves robots 4 and 5 sideways and $[h_2, g_3]$ moves robots 1 and 5 sideways. To move each robot individually, the sequence of primitives must mimic the higher-order Lie brackets. For instance, robot 1 is moved in the direction of the Lie bracket $[h_1, [h_2, g_3]]$ using a sequence of primitives $[h_2, g_3]$ and h_1. Thus, moving one robot requires a complexity of $o(L^{3-2}) = o(L)$ and the overall complexity of $\mathcal{M}(6, 2)$ is $o(nL)$.

In short, motion planning is simplified if each robot moves sequentially; however, this approach sacrifices path optimality. The complexity of motion planning increases exponentially as robots get coupled in groups, but this trade-off for parallelism results in more optimal paths.

5.2 Instantiations of the motion planning problem

The motion planning problem $\mathcal{M}(n, r_k)$ involves moving n robots with primitives of order r_k. In this section, we investigate some instantiations of the motion planning problem $\mathcal{M}(n, r_k)$.

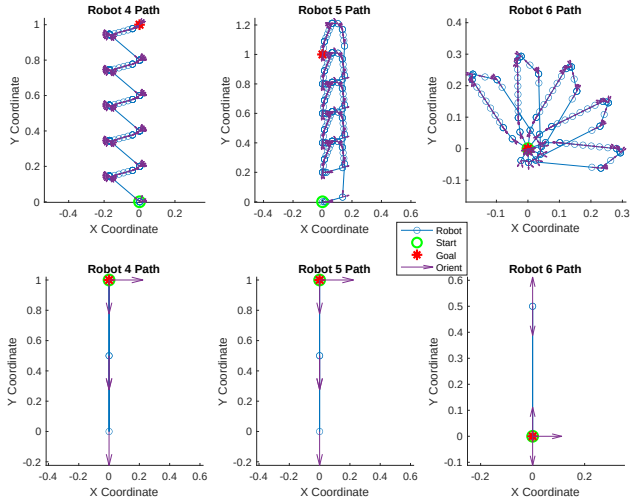


Fig. 2: Motion planning problem $M(6,2)$ with the primitive corresponding to $[h_2, g_1]$. Top: numerical optimization; Bottom: hand-designed primitive.

For a given set of n robots, we use m groups to control them. Depending on the chosen group allocation, the set of all primitives up to the order r_k is defined as $\mathcal{P}^{r_k} = \{f_j, S^{r_i}(\mathcal{F})\}$, where $j = 1, ..., m$ and $i \leq r_k$. This set includes the original m groups and any additional subgroups that correspond to Lie brackets up to the order r_k. By choosing different subsets of $\mathcal{P}^{r_k}$, we will obtain different instantiations of the motion planning problem. Next, we will first discuss the feasible implementation of primitives and then address solving several motion planning instances.

Implementation of Primitives We next compare two approaches for implementing the primitives: (a) numerical optimization; and (b) primitives designed by hand. In numerical optimization, we choose a large enough number of control steps and randomly assign which group is active during each control step. We then use numerical optimization to compute the time for which each group is activated (the distance of motion) so that the total path length is minimized. Tjis approach is similar to the approach described in [17]. In contrast, hand-designed primitives follow the procedure described in the proof of Proposition 4, where to obtain a motion primitive corresponding to a subgroup, robots are eliminated one by one from the original group.

Figure 2, shows the results in the case of the motion planning problem $M(6,2)$, where we have 6 robots and 4 groups, and the particular motion primitive that we are interested in is the Lie bracket $[h_2, g_1]$. This primitive moves robots 4 and 5. Therefore, the subgroup has 2 robots ($\{4,5\}$), as opposed to the original group that contains 3 robots ($\{4,5,6\}$). The results of numerical optimization are shown in the top panel. The number of control steps was chosen to be $k = 35$. A group is randomly selected for each control step, and then we use numerical optimization to compute the activation time for each group (distance of motion) so that the total path length is minimized and that the robots

move a fraction of the way towards the target configuration. The distance to the target configuration was chosen to be 1, and the fraction was chosen to be 0.2; this means that the computed primitive was repeated 5 times to reach the desired goal configuration. The bottom panel shows the hand-designed primitive using the control vector field $f_1 = [\mathbf{0}_6, \cos_4, \sin_4, \cos_5, \sin_5, \cos_6, \sin_6, \mathbf{0}_3, \frac{1}{r}, \frac{1}{r}, \frac{1}{r}]$. The primitive is defined as $P_2^1 = (f_1, \mathrm{Rot}\, r_4, r_5, r_6(0, 0, \pi), f_1)$. The operation $\mathrm{Rot}\, r_4, r_5, r_6(0, 0, \pi)$ rotates robot 6 by π while keeping robots 4 and 5 in place. This rotation involves the rotational vector fields h_i as described in Section 4.2. All robots start from the origin, oriented along the positive x-axis. It can be seen that the hand-designed nested primitives in the bottom panel yield a much simpler path. Moreover, solving optimization problems in high dimensions is challenging. Thus, we will use the hand-designed primitives in the remainder of the paper.

Motion planning Cases Next, we investigate time complexity and optimality trade-offs for motion planning in our system by applying Rapidly-Exploring Random Trees (RRT) [14]. RRT is a widely used sampling-based algorithm that constructs a tree of feasible trajectories by incrementally extending it from an initial state toward randomly sampled points in the configuration space. The tree grows by iteratively sampling the points (nodes) and connecting them to the nearest node in the existing tree using a local planner. In our formulation, the local planner uses subgoups. It first uses a rotation to align the robots in a subgroup so they are directed toward the sampled points and then moving them using the motion primitive corresponding to the subgroup.

Table 2: Average Runtime and Path length for Intermediate Cases 3 and 4

Case	**Ave RRT Runtime**	**Ave Tree Nodes**	**Ave path Length**
2 robots parallel	292.23 seconds	2140.5 nodes	268.65
2 robots sequential	23.71 seconds	116.40 nodes	192.61
robot 13	3.50 seconds	32.90 nodes	58.38
robot 26	4.19 seconds	39.30 nodes	67.41
robot 45	16.02 seconds	44.20 nodes	66.82

Extreme Case 1: Pure planning problem In this scenario, there are no additional subgroups, only the original groups are used. Thus, each robot belongs to more than one group, which means that the robot motions are coupled. Figure 3(a) shows a random run path for 6 robots in 4 groups. Initially, the robots are deployed in the range $y = [0, 10]$ along $x = 0$, and the final goal positions are along $x = 15$, with the y coordinates permuted randomly. Specifically, robots have start positions $[(0, 2.0), (0, 5.2), (0, 8.4), (0, 11.6), (0, 14.8), (0, 18.0)]^T$ and goal positions $[(12, 11.6), (12, 18.0), (12, 5.2), (12, 2.0), (12, 8.4), (12, 14.8)]^T$. The environment is the square $[0, 20] \times [0, 20]$ that contains two circular obstacles in black.

Extreme Case 2: Pure control problem In this scenario, each subgroup contains a single robot. The planning problem involves guiding individual robots to the goal. The primitives are of order $r_s = 3$, which move one robot at a time.

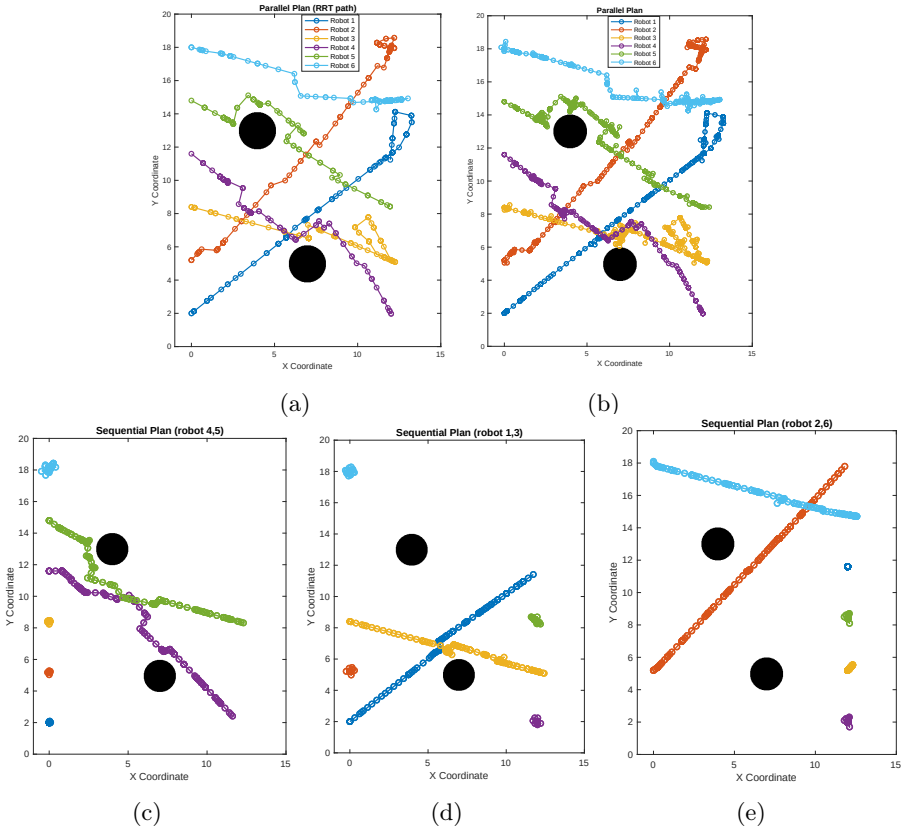
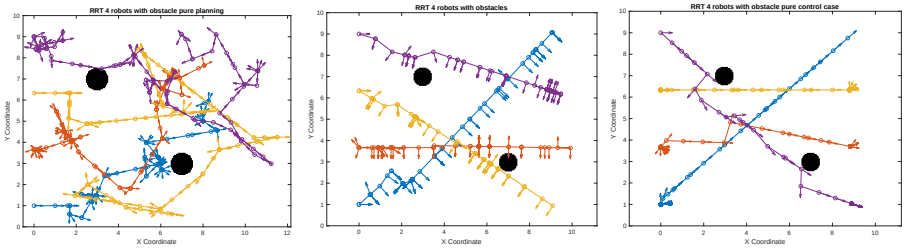


Fig. 3: (a) Intermediate Case 3: RRT path (each point is an RRT node); (b) Intermediate Case 3: Physical path (each point is a switch between groups); (c),(d) and (e) Intermediate Case 4: Physical path

Intermediate Case 3: Parallel planning In this case, we explore the complexity trade-off between primitive implementation and RRT planning. Fig. 3(a) presents an example in which we use three disjoint subgroups, each of them containing 2 different robots. These subgroups are realized by the Lie brackets $[h_2, g_1]$, $[h_3, g_2]$ and $[h_1, g_3]$. The RRT stops once a neighborhood of the goal with a radius of 0.5 is reached. This experiment was repeated ten times, with an average planning time of 292.23 seconds. Fig. 3(b) shows the physical paths of the robots, including the motion due to implementation of the primitives where some robots need to move back and forth.

Intermediate Case 4: Sequential planning In contrast to Intermediate Case 3, here we plan the motion of each subgroup individually. This effectively reduces the dimensionality of planning problems. However, the subgoups are moved sequentially, which means that the time of motion is about 3 times the time it takes for the robots to complete the motion in Intermediate Case 3. Figures 3(c), (e), and (f) show the motions where 6 robots were divided into 3 subgroups using Lie brackets $[h_2, g_1]$, $[h_1, g_3]$ and $[h_3, g_2]$. Compared with Fig. 3(b), the average



(a) Pure planning (b) Intermediate case (c) Pure control

Fig. 4: Panels (a), (b) and (c) show RRT planned path. Panels (a) and (c) are Extreme Cases 1 and 2, respectively. Panel (b) uses two second-order Lie brackets as subgroups.

planning time and average path length diminish to 23.71 seconds and 192.61 under the same RRT parameters. Table 2 shows running times. It is clear that for large swarms, sequential planning is computationally dramatically better than parallel planning.

Finally, we demonstrate Extreme Cases 1 and 2, and Intermediate Case 3 for 4 robots and compare their runtime in Fig. 4. Observing the path and runtime data in Table 3, the running time decreases as we introduce more complex primitives. In particular, the pure planning RRT stops when arriving within 2.5 of the goal, which is pretty far away. Still, it takes a really long time to find a path compared to using primitives. This shows that subgroups are essential to make RRT work as the size of the swarm increases.

Table 3: Average runtimes and numbers of tree nodes for paths in Fig. 4

Case	**Ave Runtime**	**Ave Tree Nodes**	**Neighborhood Radius**
Fig. 4(a)	35.33 minutes	10583 nodes	2.5
Fig. 4(b)	23.46 seconds	148.20 nodes	0.5
Fig. 4(c)	4.10 seconds	20.75 nodes	0.5

6 Conclusion

This paper introduces a novel group-control framework for microrobot swarms controlled by a global field. We focus on a swarm of MicroStressBots, electrostatically actuated MEMS stress microrobots. We show that the system is Small-Time Locally Controllable (STLC) in positions even though the control inputs are unilateral. We also demonstrate that the minimum number of groups to achieve STLC for n robots is $log_2(n+2)+1$. We study the complexity of the motion planning problem for a MicroStressBot swarm under group-based control. In particular, we compare the trade-off between the complexity of control with the complexity of motion planning. We introduce the notion of motion primitives and subgroups, which allows us to balance the two. We show that subgroups significantly simplify the motion planning problem and make the approach applicable to larger swarm sizes. Simulations confirm the framework's effectiveness, suggesting significant potential for applications in various fields requiring precise microrobot coordination.

References

1. Aghakhani, A., Pena-Francesch, A., Bozuyuk, U., Cetin, H., Wrede, P., Sitti, M.: High Shear Rate Propulsion of Acoustic Microrobots in Complex Biological Fluids. Science Advances **8**(10), eabm5126 (2022)
2. Banerjee, A.G., Chowdhury, S., Losert, W., Gupta, S.K.: Real-time Path Planning for Coordinated Transport of Multiple Particles using Optical Tweezers. IEEE Transactions on Automation Science and Engineering **9**(4), 669–678 (2012)
3. Becker, A., Onyuksel, C., Bretl, T., McLurkin, J.: Controlling Many Differential-drive Robots with Uniform Control Inputs. The International Journal of Robotics Research **33**(13), 1626–1644 (2014)
4. Bengea, S.C., DeCarlo, R.A.: Optimal Control of Switching Systems. Automatica **41**(1), 11–27 (2005)
5. Bullo, F., Lewis, A.D.: Geometric Control of Mechanical Systems, Texts in Applied Mathematics, vol. 49. Springer (2004)
6. Donald, B.R., Levey, C.G., McGray, C.D., Paprotny, I., Rus, D.: An Untethered, Electrostatic, Globally Controllable MEMS Micro-Robot. Journal of Microelectromechanical Systems **15**(1), 1–15 (Feb 2006)
7. Donald, B.R., Levey, C.G., Paprotny, I.: Planar Microassembly by Parallel Actuation of MEMS Microrobots. Journal of Microelectromechanical Systems **17**(4), 789–808 (Aug 2008)
8. Donald, B.R., Levey, C.G., Paprotny, I., Rus, D.: Planning and Control for Microassembly of Structures Composed of Stress-engineered MEMS Microrobots. The International Journal of Robotics Research **32**(2), 218–246 (2013)
9. Hu, W., Ishii, K.S., Ohta, A.T.: Micro-assembly Using Optically Controlled Bubble Microrobots. Applied Physics Letters **99**(9) (2011)
10. Jeong, J., Jang, D., Kim, D., Lee, D., Chung, S.K.: Acoustic Bubble-based Drug Manipulation: Carrying, Releasing and Penetrating for Targeted Drug Delivery Using an Electromagnetically Actuated Microrobot. Sensors and Actuators A: Physical **306**, 111973 (2020)
11. Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. The international journal of robotics research **30**(7), 846–894 (2011)
12. Kawski, M.: The combinatorics of nonlinear controllability and noncommuting flows. In: Mathematical Control Theory, ICTP Lecture Notes, vol. VIII, pp. 224–311. Abdus Salam Int. Center for Theoretical Physics, Trieste (2002)
13. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
14. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. The International Journal of Robotics Research **20**(5), 378–400 (May 2001)
15. Li, J., Liu, W., Li, T., Rozen, I., Zhao, J., Bahari, B., Kante, B., Wang, J.: Swimming Microrobot Optical Nanoscopy. Nano Letters **16**(10), 6604–6609 (2016)
16. Li, J., Li, X., Luo, T., Wang, R., Liu, C., Chen, S., Li, D., Yue, J., Cheng, S.h., Sun, D.: Development of a Magnetic Microrobot for Carrying and Delivering Targeted Cells. Science Robotics **3**(19) (Jun 2018)
17. Li, S., Žefran, M., Paprotny, I.: Group-based control of large-scale micro-robot swarms with on-board physical finite-state machines. In: 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE). pp. 524–530. IEEE (2022)
18. Liberzon, D.: Switching in systems and control. Springer Science & Business Media (2003)

19. Palagi, S., Singh, D.P., Fischer, P.: Light-controlled Micromotors and Soft Microrobots. Advanced Optical Materials **7**(16), 1900370 (2019)
20. Paprotny, I., Levey, C., Wright, P., Donald, B.: Turning-rate selective control : A new method for independent control of stress-engineered mems microrobots. In: Robotics: Science and Systems. vol. 8, pp. 321–328 (2013)
21. Paprotny, I., Zefran, M.: Finite state machine (FSM) addressable MEMS microrobots: a new paradigm for controlling large numbers of mems microrobots. In: 2017 International Conference on Manipulation, Automation and Robotics at Small Scales (MARSS). pp. 1–6. IEEE (2017)
22. Pawashe, C., Floyd, S., Sitti, M.: Modeling and Experimental Characterization of an Untethered Magnetic Micro-robot. The International Journal of Robotics Research **28**(8), 1077–1094 (2009)
23. Qiu, F., Nelson, B.J.: Magnetic Helical Micro-and Nanorobots: Toward Their Biomedical Applications. Engineering **1**(1), 021–026 (2015)
24. Qureshi, A.H., Miao, Y., Simeonov, A., Yip, M.C.: Motion Planning Networks: Bridging the Gap between Learning-based and Classical Motion Planners. IEEE Transactions on Robotics **37**(1), 48–66 (2020)
25. Sipser, M.: Introduction to the Theory of Computation. ACM Sigact News **27**(1), 27–29 (1996)
26. Strudel, R., Pinel, R.G., Carpentier, J., Laumond, J.P., Laptev, I., Schmid, C.: Learning obstacle representations for neural motion planning. In: Conference on Robot Learning. pp. 355–364. PMLR (2021)
27. Wang, J., Chi, W., Li, C., Wang, C., Meng, M.Q.H.: Neural RRT*: Learning-based Optimal Path Planning. IEEE Transactions on Automation Science and Engineering **17**(4), 1748–1758 (2020)
28. Wei, S., Uthaichana, K., Žefran, M., DeCarlo, R.A., Bengea, S.: Applications of Numerical Optimal Control to Nonlinear Hybrid Systems. Nonlinear Analysis: Hybrid Systems **1**(2), 264–279 (2007)
29. Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: Chomp: Covariant Hamiltonian Optimization for Motion Planning. The International journal of robotics research **32**(9-10), 1164–1193 (2013)