On Stochastic Self-Assembly of Underwater Robots

Varadarajan Ganesan[1] and Mandar Chitre[1]

***Abstract*—We present a rule based approach which incorporates a state machine in a novel manner to carry out self-assembly in underwater environments. The rules formulated along with the state machine govern the manner in which individual robots combine to generate shapes like a straight line, a 'T' alphabet and a two sided pyramid. We analyze the factors that affect the time taken to complete self-assembly via simulations. Finally, we adopt a statistical dynamics approach to calculate the combination rates of the robots and show how it yields a Markov process model. We use the Markov model to predict the assembly process and show that it is similar to carrying out the full mechanics based simulation.**

***Index Terms*—Assembly, distributed robot systems, probability and statistical methods**

I. Introduction

SELF-assembling robots have gained a lot of popularity over the years owing to the compact nature of individual robots and their collective ability to form complex shapes [1]. These systems have the ability to assemble themselves into different shapes to execute different tasks [2] or help one another to perform a certain task [3]. The concept of self-assembly derives its inspiration from naturally occurring processes like DNA replication and protein folding [4]. For example, protein strings obtain their respective functionalities based on the structures they fold themselves into. Likewise, a group of robots have different capabilities based on the structures they assemble themselves into.

While self-assembling systems are common in land robotics [5], they are less popular in underwater environments [6], [7], [8]. The individual modules in [6] lack intelligence as they rely on directed fluid flow to assemble themselves into different shapes and have no control over the assembly process. The authors in [7] have proposed an underwater robot with a modular structure. Each module provides a specific functionality and the robot can assemble and/or reconfigure based on the requirements to carry out a desired task. In [8], parameters like attractive and repulsive forces are given as input to a control scheme. The control scheme then generates commands that results in the aggregation of individial modules which is in accordance with biological studies [9]. Likewise, the authors in [10] review works of self-assembly at all scales from a chemical reaction point of view. However, in both [8] and the works reviewed in [10], self-assembly occurs in a natural manner and assembling more complex structures require a programmable component.

We believe that self-assembly in fluid domains can be used as a novel technique to manufacture devices of different scales in three dimensions. Hence, the aim of the assembly process is to be able to program the individual parts such that they combine accordingly and form the desired structure reliably. Also, being able to predict the overall assembly process and possibly optimize it by studying the factors that affect self-assembly can be very useful.

Self-assembly can be broadly classified into two types: *passive* and *active* assembly. While passive assembly refers to the use of external stochastic forces to guide the assembly process [11], [12], [13], active assembly makes use of actuators instead [2], [14], [15], [16], [17]. We adopt the passive assembly strategy where we make use of stochastic forces to drive the assembly process.

We formulate a rule based approach in combination with a state machine in a novel fashion to control the assembly process in order to generate the desired structure. The proposed approach differs from our earlier work [18] in two ways. First, each robot executes only certain rules (for combination) that are associated with a particular state. This allows us to terminate the self-assembly process when individual robots reach a particular state. As a result, we can assemble many copies of the same structure with a larger number of individual modules. Second, the use of a state machine allows us to assemble more complex structures, an example of which is discussed in Section VII. Also, our work differs from [19] such that we assemble 3-D structures. Additionally, to create multiple copies of the same structure, individual robots in [19] need to transmit the entire program to other robots. In such a case, unsuccessful transmission can result in the failure of replicating structures. In contrast, we adopt a more distributed approach where individual robots are all pre-programmed and failure of an individual robot does not jeopardize the assembly process. While we present results from simulation studies, we also suggest ways in which the proposed approach can be practically implemented in Section III.

We study the various factors that affect the process of self-assembly. Additionally, we propose a statistical dynamics approach, which treats the combination of individual robots as a chemical reaction, to calculate specific "reaction" rates. Using these reaction rates, we show how the assembly process can be modelled as a discrete state, continuous time Markov process. The Markov model is then used to predict the states (see Section VI-A1) through which the system (collection of robots) traverses.

We choose Gillespie's method [20] which incorporates the

Manuscript received: September, 15, 2015; Revised November, 19, 2015; Accepted January, 7, 2016.

This paper was recommended for publication by John Wen upon evaluation of the Associate Editor and Reviewers' comments.

[1]Varadarajan Ganesan and Mandar Chitre is with Acoustic Research Laboratory, Tropical Marine Science Institute, National University of Singapore, Singapore. {varadarajan,mandar}@arl.nus.edu.sg

Digital Object Identifier xxxxxxxxxxxxxxxxxxx

calculated reaction rates with the Markov process model to predict the assembly process. Statistical estimation using Gillespie's method provides the following advantages. First, it can quickly estimate the assembly process as opposed to carrying out full mechanics based simulations which require considerable amount of time. Second, such a model can be used to estimate the self-assembly process for different initial conditions, especially when it is difficult to run the simulation. This happens when a large number of robots are used resulting in a very high computational load. Finally, we show that the full mechanics based simulation and the numerical estimate using Gillespie's method yield similar results.

II. Related Work

White *et al.* in [11] have demonstrated stochastic self-assembly in three dimensions using simulations where individual modules are modelled as cubes and are "immersed" in a tank (inverted hemisphere) filled with a fluid. The cubes are capable of attracting one another at close proximities. Stochastic forces are applied on the cubes to model the effect of agitating the fluid. They also built two such physical units (cubes) to demonstrate the ideas presented experimentally. The physical units have electromagnets on each face to attract one another and the fluid medium in which the cubes are immersed is agitated to drive the assembly process. However, in both simulations and experiments, the cubes rely on a central node which is the base plate of the tank for power, communication and instructions as to how the structure grows. As a result, failure of either power or communication at any level could seriously jeopardize the assembly process. In contrast, we adopt a distributed strategy and the robots are individually powered. We also formulate rules that account for events like dissociation between individual modules. Therefore, the failure of an individual robot does not jeopardize the assembly process.

While the authors in [12], [13], [21] have demonstrated self-assembly using individual robots moving randomly on an air table, Hara *et al.* in [17] demonstrate self-assembly on the water surface using actuated robots. Our work extends the works of the authors in [12], [13], [21] by implementing self-assembly in three dimensions to emulate an underwater environment. Additionally, White *et al.* in [12] briefly discuss various factors that affect the process of self-assembly without providing sufficient results or analysis. We analyze the various parameters that affect self-assembly in detail and provide results and insights into the same. A graph grammar based approach to self-assembly is proposed in [13], [21] and the authors in [22] extend the works in [13], [21], [23] by analyzing the performance of different grammars to assemble a particular shape by using statistical dynamics. However, they do not predict the assembly process itself, which is one of the foci of our work.

Finally, the graph grammar approach adopted in [13], [21], [22] associates an alphabet (state) with every edge of the physical module. An example would be the use of three alphabets for the three edges of a triangular module. The alphabets are updated when individual modules combine with one another. Considering the worst case scenario, each combination would then require three state updates and if all the edges combine, nine state updates would be required. In general, if N were the number of faces available for combination, the graph grammar approach has a worst case complexity of $O(N^2)$. It needs to be noted that moving to three dimensions increases the number of faces, N, which will eventually render the graph grammar approach to self-assembly intractable. However, our rule based approach in combination with a state machine reduces the complexity to $O(N)$ since we associate any given module with a single state as opposed to having multiple states to represent each face of the module. Also, the ease of formulating the rules can be seen in Section IV and Section VII.

III. Simulation Methodology

We simulate the process of self-assembly using Open Dynamics Engine [24] which is an open source physics engine that simulates rigid body dynamics, collision detection etc. Individual robots are modelled as cubes and placed in a tank (cuboid) filled with a fluid. The fluid in the tank is agitated randomly which is modelled as a random time varying velocity field. The mathematical model of the velocity field is as follows:

$$v_x(t) = \mathcal{N}(0, \sigma_x) \tag{1}$$

$$v_y(t) = \mathcal{N}(0, \sigma_y) \tag{2}$$

$$v_z(t) = \mathcal{N}(0, \sigma_z) \tag{3}$$

where the individual components of the velocity field are drawn from a Gaussian distribution with zero mean and standard deviation $\sigma_i, i \in \{x, y, z\}$ at every simulation time step for every point inside the tank. However, the random forces exerted on the cubes is a manifestation of the drag forces acting on the cubes. As a result, it is sufficient to calculate the velocity field in the vicinity of the cubes. To apply the drag forces on the cubes, we "divide" each cube into eight equal smaller cubes and apply the drag force on the centre of the "smaller" cubes. The drag force is calculated as follows:

$$\mathbf{f} = -\frac{1}{2} C_d \rho A |\mathbf{v}_{\text{rel}}| \mathbf{v}_{\text{rel}} \tag{4}$$

where ρ is the density of the fluid medium, C_d is the drag coefficient (= 1.05 for a cube in a laminar flow field as an approximation), A is the area of cross section and $\mathbf{v}_{\text{rel}}$ is the velocity of the smaller cube relative to the fluid medium. We adopt this approach of dividing cubes to apply drag forces primarily to account for rotational forces. Once the velocity field is generated, the cubes start experiencing random (drag) forces and exhibit Brownian motion. Also, during the assembly process, the drag forces are applied on each connected cube of the structure being assembled. Additionally, it needs to be noted that the external faces (faces that are not connected to other cubes) are only considered in calculating drag forces. The values of $\sigma_i, i \in \{x, y, z\}$ are chosen such that the cubes collide with each other and the walls of the tank. Also, this type of motion can be reproduced in experiments by using multiple directional pumps.

The faces of the cubes are treated uniquely and capable of attracting the faces of other cubes at close proximities based

on the rules formulated. Each face knows which other face it is colliding with. These conditions can be met in experiments by having electromagnets attached to each face, coloring the faces distinctly and having color sensors. When two faces are close to each other and can attach themselves, the faces turn on their electromagnets after a random delay. The face to turn on its electromagnet first gets detected by the electromagnet in the other face since a current will be induced in its coil due to the presence of a magnetic field. As a result, the second face decides not to turn on its electromagnet and gets attached to the first face by automatically assuming the opposite polarity. This approach can be adopted in experiments because if both the faces were to turn on their respective electromagnets, then the polarity of the faces need to be taken into account which constrains the self-assembly process.

We also include the possibility that the cubes can dissociate from one another when the distance between the faces is above a certain threshold due to collisions. Dissociation of cubes was modelled to emulate a real world scenario where objects can detach from one another when a large impulse (due to collision) is applied.

The cubes also have a state machine running and can relay their states to the other cubes in the vicinity. This can be accomplished in experiments by using different colored light-emitting diodes (LEDs) for different states and the corresponding color sensors on the faces of the cubes. The LEDs can be switched off to signal the completion of self-assembly. We now formulate the rules that are required for self-assembly.

IV. Rules

Each cube has a state machine and changes its state according to the rules formulated. We use identical state machines in each of the cubes. For each state machine, we denote the cube in which the state machine is running as C_g. The cube that is about to combine with the given cube C_g, is denoted as C_c and the cube that is already attached to C_g as C_n (its neighbor). Also, we denote the cube that is about to dissociate from a given cube C_g, as C_b. Fig. 1 shows an illustration of the different cubes combining and dissociating.

(a) Combination (b) Dissociation

Fig. 1: Two dimenstional illustration of cubes combining and dissociating along with their labels (C_g, C_n, C_c, C_b).

The faces of each cube are represented as $f_j^{C_i}$ where $j \in \{1, \ldots, 6\}$ and $i \in \{g, n, c, b\}$. For the shapes being assembled, it is sufficient to represent the six faces of a cube using two values of j. The top and bottom faces are denoted by $j = 1$ and the side faces are denoted by $j = 2$. We denote the state of a cube as $S(C_i)$ where $i \in \{g, n, c, b\}$ and the rule that allows it to change from one state to another as Φ_k. The number of rules that are needed to assemble the desired structure is denoted by k. For each rule, there is a pre-condition which usually requires the cubes C_c and/or C_n to be in a particular state during combination (C_b and/or C_n during dissociation). The states of the cubes are updated after any successful combination (or dissociation) between 2 cubes. Initially, the states of all the cubes are set to 'a', i.e., $S(C_i) = a$ where $i \in \{g, n, c, b\}$.

While the rules are formulated with respect to cube C_g, the other cubes (C_c, C_n and C_b) will also run the same state machine. This is because the notations C_c, C_n and C_b are given with respect to a cube C_g, but C_g can be any cube. We now formulate the rules for a straight line and a 'T' Shape as follows:

A. Straight Line

For a straight line, it is sufficient if the top or bottom face of one cube combines with the top or bottom face of another cube. But the state machine of the cubes need to be updated at every stage of the assembly process. We restrict the number of cubes required to generate a straight line to four cubes. The rules for the overall assembly process is shown in Table I. The '$+$' operator in Table I indicates that the two cubes are about to combine (or dissociate) and ':' indicates that the two cubes have attached themselves while '$\rightarrow$' symbolizes a combination (or dissociation).Note that if a cube has two of its faces attached to two different cubes, the neighboring cubes are then denoted as C_{n_1} and C_{n_2}.

Once a cube reaches state 'b' and realizes that the state of its neighboring cube is 'd', it considers that the assembly process is complete and does not allow the addition of extra cubes (LEDs turned off). An illustration of the assembly process can be seen in Fig. 2 where the top and bottom faces are colored black. Note that the cube that is attached to the cube which is combining, also changes its state (Φ_5).

Rule	Rule type	Pre-condition	State change, $S(C_g)$
Φ_1	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = a$ or $S(C_c) = b$	a to b
Φ_2	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = b$	b to d
Φ_3	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = a$	b to c
Φ_4	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = a$ $S(C_n) = c$	b to d
Φ_5	-	$S(C_{n_1}) = d$ $S(C_{n_2}) = b$	c to d
Φ'_1	$f_1^{C_g} : f_1^{C_b} \rightarrow f_1^{C_g} + f_1^{C_b}$	$S(C_b) = b$	b to a
Φ'_2	$f_1^{C_g} : f_1^{C_b} \rightarrow f_1^{C_g} + f_1^{C_b}$	$S(C_b) = d$ $S(C_n) = b$	d to b
Φ'_3	$f_1^{C_g} : f_1^{C_b} \rightarrow f_1^{C_g} + f_1^{C_b}$	$S(C_b) = b$ $S(C_n) = b$	c to b
Φ'_4	$f_1^{C_g} : f_1^{C_b} \rightarrow f_1^{C_g} + f_1^{C_b}$	$S(C_b) = b$ $S(C_n) = d$	d to b
Φ'_5	-	$S(C_{n_1}) = b$ $S(C_{n_2}) = b$	d to c

TABLE I: Rules for self-assembly of a straight line.

B. T shape

The 'T' shape is generated by first assembling an 'L' shape following which an additional cube needs to be added at the junction. The rules are shown in Table II.

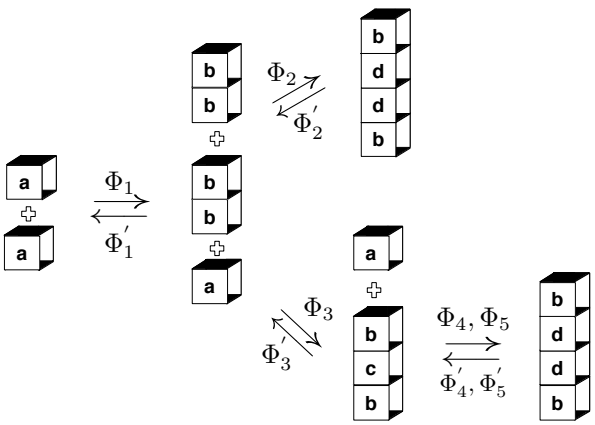


Fig. 2: Illustration of self-assembly of a straight line.

In this case, the cubes consider the self-assembly process as complete when they reach state 'b' and their neighbors are either in state 'd' or 'e'. An illustration of the assembly process can be seen in Fig. 3. We omit the dissociation rules in the interest of space.

Rule	Rule type	Pre-condition	State change, $S(C_g)$
Φ_1	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = a$	a to b
Φ_2	$f_2^{C_g} + f_1^{C_c} \rightarrow f_2^{C_g} : f_1^{C_c}$	$S(C_c) = b$	b to d
Φ_3	$f_1^{C_g} + f_2^{C_c} \rightarrow f_1^{C_g} : f_2^{C_c}$	$S(C_c) = b$	b to d
Φ_4	$f_1^{C_g} + f_1^{C_c} \rightarrow f_1^{C_g} : f_1^{C_c}$	$S(C_c) = a$	d to e

TABLE II: Rules for self-assembly of 'T' shape.

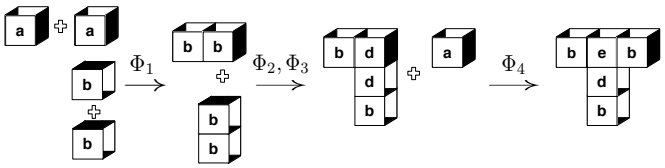


Fig. 3: Illustration of self-assembly of a 'T' shape. The top and bottom faces of the cubes are colored black.

V. Factors Affecting Self-Assembly

We next study the parameters that affect the process of self-assembly. In particular, we analyze the effect of the total number of cubes, the average velocity of the cubes, the size of cubes and the interactions between them on the time taken to finish assembly. The aim of the study is to provide insights into the choice of values for different parameters that will result in lesser time taken to complete assembly (first structure assembled) and hence optimize it. We present results on the effect of the above mentioned parameters on the formation of a 'T' shape. Similar trends were also observed for a straight line as well.

The simulations were carried out in a tank with dimensions 2 units $\times$ 2 units $\times$ 3 units. The cubes are neutrally buoyant and were placed at random locations inside the tank having random orientations at the start of the simulation. The faces of the cubes attract one another (based on the rules, Φ_k, formulated) when they collide with each other and the angle between their surface normals is greater that 135°. Cubes can also dissociate as mentioned earlier.

A. Effect of Varying Average Velocity

Increasing the agitation level of the fluid increases the average velocity of the cubes. This is because the force exerted by the fluid on the cube is greater when the fluid is further agitated. We increase the agitation level of the fluid by increasing the values of $\sigma_i, i \in \{x, y, z\}$ (Section III). We used 60 cubes and each with a length of 0.15 units for this study. Simulations were carried out for 50 times, each for five different settings of σ_i $(3.0, 3.5, 4.0, 4.5$ and 5.0 units/time step). The average velocities of the cubes obtained for the five different settings of σ_i were 0.8, 1.0, 1.2, 1.35 and 1.5 units/time step respectively. Fig. 4(a) shows the distribution of the time taken to complete self-assembly (first 'T' shape assembled) for the different values of the average velocity.

Fig. 4(a) shows that an initial increase in the average velocity reduces the time taken to finish assembly. This can be attributed to the fact that by increasing the velocities of the cubes, we can expect the number of collisions between the cubes to increase as well. As a result, the cubes combine faster resulting in a lesser time taken to complete the assembly process. However, a further increase in the average velocity increases the number of collisions which lead to dissociation and the rate of decrease in time taken to finish assembly becomes smaller. Hence, we can see that the decrease in time taken to finish assembly begins to saturate after any increase in average velocity beyond 1.35 units/time step.

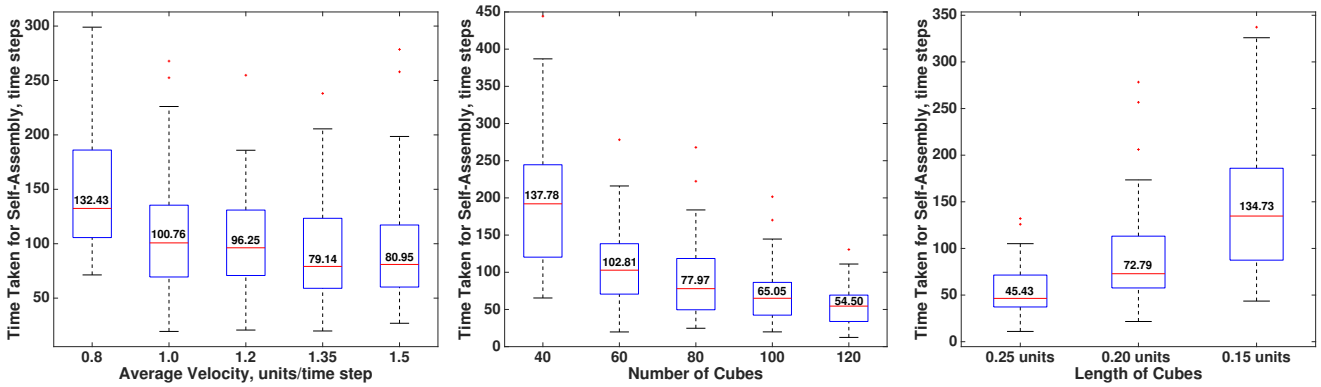
B. Effect of Varying Number of Cubes

In this study, we vary the number of cubes to see how the time taken to complete self-assembly varies. The average velocity of the cubes is set to 1.2 units/time step (by setting $\sigma_i = 4.0$ units/time step) and the cube length is kept constant at 0.15 units. We carry out simulations for five different settings of number of cubes (40, 60, 80, 100 and 120 cubes). We repeat the simulation for 50 times for the different settings.

It can be seen from Fig. 4(b) that as the number of cubes increase, the time taken to complete self-assembly reduces. The number of effective collisions (leading to combination between the cubes) increases when the number of cubes increases. This explains the reduction in time taken to finish assembly. An interesting observation is that the ratio of the number of cubes is approximately equal to the inverse of the ratio of the respective time taken (median time) to complete assembly. For example, consider the cases where 80 and 60 cubes were used. The time taken for these settings were 77.97 and 102.82 time steps. The inverse ratio of the time taken, 1.31, is approximately equal to the ratio of the cubes used in the simulations (80/60 = 1.33). The same applies to any two combination of settings. This indicates that there is an inverse relationship between the time taken and the number of cubes.

C. Effect of Varying Cube Size

Additionally, we discuss the effect of varying the size of the cubes. We keep the number of cubes constant (40 cubes) and set the average velocity of the cubes to be 1.2 units/time



(a) Average Velocity (b) Number of Cubes (c) Length of Cubes

Fig. 4: Distribution of time taken for self-assembly with varying parameters.

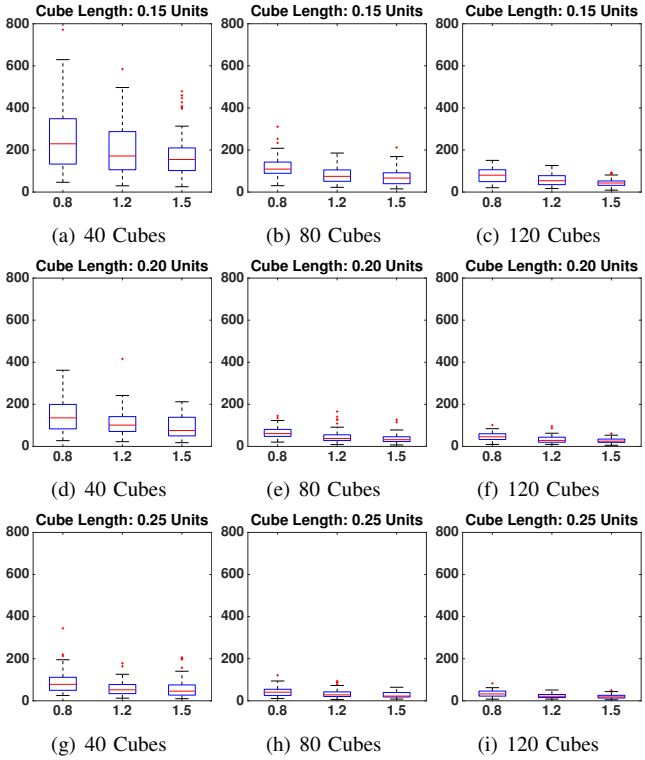
step and vary the cube lengths (0.25, 0.20 and 0.15 units). We consider cubes of all different sizes to be neutrally buoyant. We ran 50 simulations for each setting and the results are shown in Fig. 4(c).

It can be seen that the time taken to complete assembly increases as the cube length decreases (Fig. 4(c)). Interestingly, it can be observed that the time taken to complete the assembly process is inversely proportional to the square of the cube length for any two combination of settings. For example, consider the case where the cube lengths are 0.20 units and 0.15 units. The ratio of the square of the cube lengths is equal to 1.778 and the inverse ratio of the time taken (median value) is equal to 1.848 which is approximately equal to 1.778. This can be explained by the fact that increasing the cube length provides additional surface area (square of the cube length) for effective collisions between cubes and hence the corresponding reduction in time taken to complete assembly. We also found that by keeping the ratio of the tank dimensions to the cube dimensions constant, the time taken to complete self-assembly does not vary. This verifies that scale does not affect the assembly process.

D. Interaction Effects Between the Parameters

Finally, we study the interaction effects between the three parameters by running simulations for all possible combinations of settings for average velocity (0.8, 1.2, 1.5 units/time step), number of cubes (40, 80, 120) and cube length (0.15, 0.20 and 0.25 units): total of 27 different settings and 50 simulations for each setting. The results for all the settings are shown in Fig. 5.

It can be seen from Fig. 5 that an increase in the average velocity of the cubes reduces the time taken to complete self-assembly for different settings of cube length and number of cubes. Likewise, by keeping any two parameters constant, the trend resulting from varying the third parameter is the same for different combinations of the first two parameters. This indicates that varying any one parameter does not change the relationship between the other two parameters.



(a) 40 Cubes (b) 80 Cubes (c) 120 Cubes
(d) 40 Cubes (e) 80 Cubes (f) 120 Cubes
(g) 40 Cubes (h) 80 Cubes (i) 120 Cubes

Fig. 5: Distribution of time taken for self-assembly with different combination of parameters. x axis of the subfigures denotes the average velocity (units/time step) and y axis denotes the time taken for self-assembly (time steps).

VI. Statistical Dynamics of Self-Assembly

Self-assembly is usually a slow process when the robots rely on stochastic forces to drive the assembly process. Hence, carrying out large number of experiments and/or simulations becomes tedious. In this section, we address this problem by adopting a statistical dynamics approach to predict the self-assembly process. We restrict the mathematical formulation to the self-assembly of a straight line. However, it can be easily extended to other structures.

A. Key Concepts

We define the following concepts that are necessary to study the statistical dynamics of the system:

1) System State: System state refers to the state in which the system (the collection of robots) is, *viz.*, the number of monomers (individual cubes), the number of dimers (two cubes combined), the number of trimers (three cubes combined) and the number of quadmers (four connected cubes) at any given point in time during the assembly process. We denote the system state as $\mathbf{s}$. For example, if the system has ten monomers, three dimers, five trimers and zero quadmers, we write the system state using a vector notation as follows:

$$\mathbf{s} = (10, 3, 5, 0)^T \tag{5}$$

The system state can change whenever monomers, dimers, trimers or quadmers combine with one another or dissociate themselves. The combination and dissociation are called "reactions". For example, a monomer can combine with a trimer to form a straight line. This can be written as follows:

$$\mathbf{s} = (10, 3, 5, 0)^T \tag{6}$$

$$1 + 3 \rightarrow 4 \tag{7}$$

$$\mathbf{s}' = (9, 3, 4, 1)^T \tag{8}$$

where $\mathbf{s}(1)$ = 10, $\mathbf{s}(2)$ = 3, etc., and $\mathbf{s}'$ is the new state of the system. We denote the addition reaction rate between by $k_{i+j\rightarrow l}$ where i, j and l denote the type of the cube (monomers, dimers, trimers or quadmers). Reaction rates refer to the rates at which different types of cubes combine during the self-assembly process. For the reaction above, the rate of the reaction would be denoted as $k_{1+3\rightarrow 4}$. Furthermore, the reaction can happen in $\mathbf{s}(1)\mathbf{s}(3)$ ways when the system is in state $\mathbf{s}$. This is called the multiplicity of the reaction and is denoted by $M_{1+3\rightarrow 4}$ $(= \mathbf{s}(1)\mathbf{s}(3))$ for this reaction. Finally, reaction (7), irrespective of which state the system is in, can happen in $N_{1+3\rightarrow 4}$ ways. But this value varies with the rules, Φ_k, formulated. In this case, $N_{1+3\rightarrow 4} = 4$ is in accordance with the rules formulated in Section IV-A. This is because a monomer can combine in two ways (top/bottom face) at the two ends of a trimer. Putting all this together, the overall rate at which system state $\mathbf{s}$ changes to $\mathbf{s}'$ can be written as follows:

$$k(\mathbf{s}, \mathbf{s}') = N_{i+j\rightarrow l} M_{i+j\rightarrow l} k_{i+j\rightarrow l} \tag{9}$$

and for reaction (7), it is as follows:

$$k(\mathbf{s}, \mathbf{s}') = N_{1+3\rightarrow 4} M_{1+3\rightarrow 4} k_{1+3\rightarrow 4} \tag{10}$$

These rates are analogous to the state transition matrix in a Markov process and is in fact used to calculate the state transition probabilities in Section VI-B. Hence, the system states and the rates between them can be interpreted as a discrete state, continuous time Markov process.

2) Reaction Rates, $k_{i+j\rightarrow l}$: In order to build a Markov process model, we need the reaction rates, $k_{i+j\rightarrow l}$, for the various reactions occurring in the full mechanics based simulation. It is sufficient if the rates are calculated just once. We use the help of an example to describe how the reaction rate is calculated. We use 60 cubes $(\mathbf{s}(1) + 2\mathbf{s}(2) + 3\mathbf{s}(3) + 4\mathbf{s}(4) = 60)$ to calculate the rates of different reactions. The cube length was set to 0.15 units and the average velocity of the cubes were 1.2 units/time step. Let us consider the reaction between a monomer and a dimer to yield a trimer. The reaction rate to be calculated is $k_{1+2\rightarrow 3}$. Also it is possible that in the process a dimer splits into two monomers, and we denote that reaction rate as $k_{2\rightarrow 1+1}$. We start the simulation with the initial system state, $\mathbf{s}_0$, as:

$$\mathbf{s}_0 = (S_1, S_2, 0, 0)^T \tag{11}$$

and run the simulation for n times and each time with random initial conditions (location and orientation of the cubes). As soon as the reaction of interest occurs, in this case either $1 + 2 \rightarrow 3$ or $2 \rightarrow 1+1$, we note the time of that particular reaction and restart the simulation but do not restart the time. Then the instances when the first reaction occurred can be written as:

$$\delta_{1+2\rightarrow 3} = (t^1_{1+2\rightarrow 3}, t^2_{1+2\rightarrow 3}, \ldots, t^r_{1+2\rightarrow 3}) \tag{12}$$

$$\delta_{2\rightarrow 1+1} = (t^1_{2\rightarrow 1+1}, t^2_{2\rightarrow 1+1}, \ldots, t^s_{2\rightarrow 1+1}) \tag{13}$$

We hypothesize that the interval $\Delta t^i_{1+2\rightarrow 3} = t^{i+1}_{1+2\rightarrow 3} - t^i_{1+2\rightarrow 3}$ is distributed according to the Poisson's waiting time with the mean $\lambda = 1/k_{1+2\rightarrow 3}$. Now the rates can be calculated as follows:

$$k_{1+2\rightarrow 3} \approx \frac{1}{S_1 S_2 \overline{\Delta t_{1+2\rightarrow 3}}} \tag{14}$$

$$k_{2\rightarrow 1+1} \approx \frac{1}{S_2 \overline{\Delta t_{2\rightarrow 1+1}}} \tag{15}$$

where $\overline{\Delta t_{1+2\rightarrow 3}}$ and $\overline{\Delta t_{2\rightarrow 1+1}}$ are the average waiting times. Note that we divide the rates by the multiplicity of the reaction. However, we do not divide by the number of ways the reaction can occur, i.e. $N_{1+2\rightarrow 3}$, since it is dependent on the rules generated. Also, it does not matter if we divide by $N_{1+2\rightarrow 3}$ since while calculating the rate $k(\mathbf{s}, \mathbf{s}')$ we multiply by this number again. But the multiplicity, $M_{1+2\rightarrow 3}$, of the reaction can vary over the course of the assembly process. Hence, we divide the rate by the multiplicity of the reaction.

The method suggested above is similar to the method suggested in [22]. However, there is a difference since we now repeat the same procedure and calculate the rates over different initial system states. We then take the average of all the rates calculated over different initial system states. This method captures all possible system states during the course of an assembly process and hence a gives a more accurate estimate of the reaction rates. Examples of other initial system states are $\mathbf{s}_0 = (S_1, S_2, S_3, 0)^T$, $\mathbf{s}_0 = (S_1, S_2, S_3, S_4)^T$, etc. We choose the initial system states such that they are the states through which the system traverses during the self-assembly process. The rates thus calculated for the eight possible reactions are shown in Table III.

TABLE III

Reaction Type	Reaction Rate	Reaction Type	Reaction Rate
1+1→ 2	7.88×10^{-5}	2→ 1+1	4.53×10^{-5}
1+2→ 3	9.29×10^{-5}	3→ 2+1	3.89×10^{-5}
1+3→ 4	5.32×10^{-5}	4→ 3+1	0.77×10^{-5}
2+2→ 4	6.29×10^{-5}	4→ 2+2	1.2×10^{-5}

B. Gillespie's Method

After obtaining all the reaction rates, the system can be summarized using a *rate vector*, $\mathbf{k}$ which holds the rates at which

the system can change from state $\mathbf{s}_p$ to $\mathbf{s}_q^r, r \in \{1, \ldots, v\}$ where v is the number of possible reactions. It can be written as follows:

$$\mathbf{k} = (k(\mathbf{s}_p, \mathbf{s}_q^1), k(\mathbf{s}_p, \mathbf{s}_q^2), \ldots, k(\mathbf{s}_p, \mathbf{s}_q^v)) \tag{16}$$

where $v = 8$ in the case of assembling a straight line and the system state changes from $\mathbf{s}_p$ to $\mathbf{s}_q^1$ or $\mathbf{s}_q^2$ through reactions $1 + 1 \rightarrow 2$ or $1 + 2 \rightarrow 3$ respectively and so on.

We define a new vector $\mathbf{p}$ as follows:

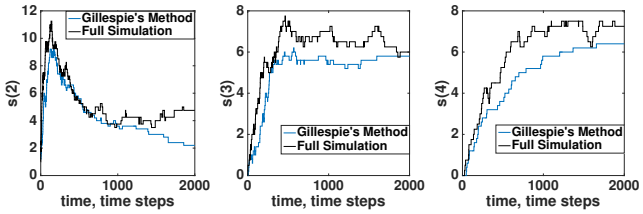
$$p_r = \frac{k_r}{\sum_{r=1}^{v} k_r} \tag{17}$$

This vector $\mathbf{p}$ is analogous to the state transition matrix (vector) associated with a Markov process. Finally, we generate distinct trajectories that the system takes during the self-assembly process using Gillespie's method [20]. Let the initial state of the system be $\mathbf{s}_0$ at time $t_0 = 0$. We generate a random trajectory $\{(\mathbf{s}_i\ ,\ t_i)\}_{i \in \mathbb{N}}$ as follows:

1) For a given system state $\mathbf{s}_i$, we calculate the $\mathbf{p}$ vector and choose the system state $\mathbf{s}_{i+1}$ randomly according to the $\mathbf{p}$ vector.
2) In order to estimate the time at which the reaction happened, we choose $\tau > 0$ randomly according to the exponential distribution, $p(\tau) = \lambda \exp(-\lambda\tau)$ where $\lambda = 1/\sum_{r=1}^{v} k_r$, which is the mean time for any possible reaction to occur. We then set the time when the reaction happened to as, $t_{i+1} = t_i + \tau$.

C. Results

We ran the full mechanics based simulation ten times for 2000 time steps each. The initial state of the system was set to $\mathbf{s}_0 = (60, 0, 0, 0)^T$. We also estimate the average trajectory of the number of dimers, trimers and quadmers using Gillespie's method over ten runs with the same initial system state, $\mathbf{s}_0$. Fig. 6 shows the average trajectory obtained from the full mechanics based simulation and Gillespie's method.



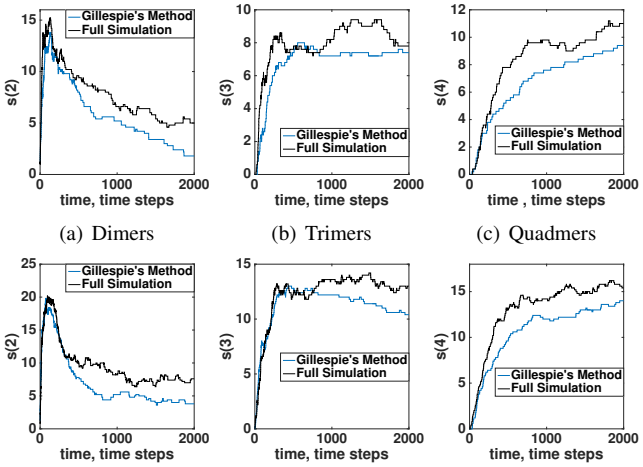
(a) Dimers (b) Trimers (c) Quadmers

Fig. 6: Average trajectory of the self-assembly process for initial condition, $\mathbf{s}_0 = (60, 0, 0, 0)^T$.

From Fig. 6, it can be clearly seen that the Gillespie's method is able to predict the trends associated with the assembly process very well. Additionally, the average time taken to form the first straight straight line using simulations and Gillespie' method were 89.42 time steps and 93.84 time steps respectively. Finally, the Gillespie's method took approximately 0.037 seconds on a 3.4 GHz Intel Core i7, Macintosh desktop to generate ten distinct trajectories while the full fledged simulations took close to five hours (for ten simulations).

We also carry out full simulations with 80 and 120 cubes and see if Gillespie's method can still predict the assembly process using the rates obtained from 60 cubes (Table III). Note that when we try to estimate the trajectory for the assembly process of 80 or 120 cubes, the multiplicity ($M_{i+j \rightarrow l}$) for different reactions is different and the components of $\mathbf{k}$ are updated in accordance with equation (9). The underlying assumption is that the components of the rate vector, $\mathbf{k}$, vary linearly with the concentration (multiplicity) of the reactants (monomers, dimers, trimers and quadmers). In other words, $k_{i+j \rightarrow l}$ is fairly constant over different concentrations. For this to hold true, we require the system to be "well mixed" as suggested in [20], [22] and has been verified in this case.

Again, from Fig. 7, it can be seen that Gillespie's method accurately predicts the trends associated with the assembly process. Hence, it can be used to predict the assembly process for different initial conditions (as long as the system is "well mixed"), especially when it is difficult to carry out a full mechanics based simulation because the computational load is very high (large number of cubes).



(a) Dimers (b) Trimers (c) Quadmers

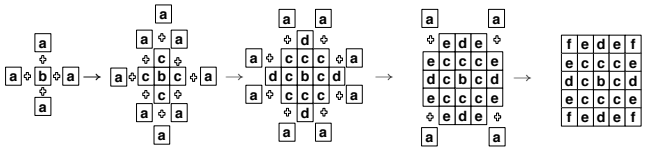
(d) Dimers (e) Trimers (f) Quadmers

Fig. 7: Average trajectory (ten runs) of the self-assembly process for initial conditions, $\mathbf{s}_0 = (80, 0, 0, 0)^T$ (Fig. 7(a), 7(b) and 7(c)) and $\mathbf{s}_0 = (120, 0, 0, 0)^T$(Fig. 7(d), 7(e) and 7(f)).

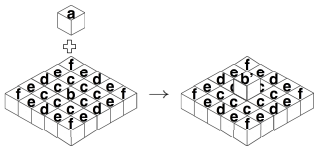
VII. Complex Structures

As mentioned in Section II, the use of a rule based approach in combination with a state machine reduces the difficulty associated with assembling complex structures. We demonstrate this by assembling a two sided pyramid. In the interest of space, we demonstrate the assembly process using an illustration (Fig. 8). Also, we omit the possibility of dissociation between the cubes for this case.

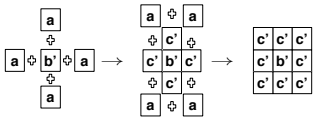
The assembly process uses a seed robot (state set to 'b') to initiate assembly and is split into three stages. The first stage involves the assembly of a square base plate (5×5 cubes). This is achieved by formulating rules that allow the side faces of the cubes to combine with the side faces of the seed robot and growing it further (Fig. 8(a)). After the first stage is complete, the second stage begins by first adding two cubes (top/bottom face) to the top and bottom faces of



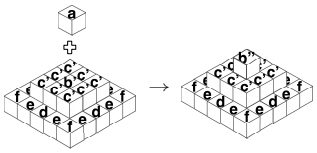
(a) First stage, top view



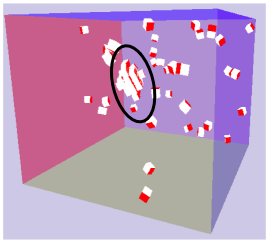
(b) Second stage, isometric view



(c) Second stage, top view



(d) Final stage, isometric view



(e) Two sided pyramid from simulation. The top and bottom faces of the cubes are colored red

Fig. 8: Illustration of self-assembly of a two sided pyramid along with the state updates.

the cube located at the centre of the base plate (Fig. 8(b)). Following this, a 3×3 plate on either side of the base plate is assembled (Fig. 8(c)). Finally, the last stage involves adding two more cubes (top/bottom face) to the centre of the two 3×3 plates completing the assembly process (Fig. 8(d)). The two sided pyramid assembled in simulation can be seen in Fig. 8(e) (inside the black oval).

VIII. Conclusions and Future Work

We have combined a rule based model with a state machine in a novel manner to carry out self-assembly of robots in underwater environments. A passive assembly strategy which makes use of stochastic forces to drive the assembly process was adopted. We make use of the proposed approach to assemble simple and complex structures. We also carried out simulations to analyze the effect of various parameters on the time taken to complete self-assembly. We also formulated a statistical dynamics approach to predict the assembly process. We are in the process of building the robots (cubes) with the necessary sensors and magnets to demonstrate the concepts experimentally.

The rules formulated consider the top/bottom and all the side faces as the same. This limits the number of shapes that can be assembled. We intend to explore the use of all the faces in a distinct manner to generate more complex arbitrary structures as a part of our future work. Finally, we intend to develop an algorithm which generates the rules after receiving the desired structure as input from the user. This provides flexibility in terms of being able to assemble more complex structures and completely automates the process of self-assembly in underwater environments.

References

[1] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Autonomous self-assembly in swarm-bots," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1115–1130, Dec 2006.

[2] E. Tuci, R. Groß, V. Trianni, F. Mondada, M. Bonani, and M. Dorigo, "Cooperation through self-assembly in multi-robot systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 2, pp. 115–150, Dec. 2006.

[3] R. O'Grady, R. Gross, A. Christensen, F. Mondada, M. Bonani, and M. Dorigo, "Performance benefits of self-assembly in a swarm-bot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 2381–2387.

[4] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell, 4th edition*. Garland Science, 2001.

[5] K. Stoy, D. Brandt, and D. J. Christensen, *Self-Reconfigurable Robots—An Introduction*. MIT Press, 2010.

[6] M. T. Tolley and H. Lipson, "Programmable 3D Stochastic Fluidic Assembly of cm-scale Modules," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011.

[7] I. Vasilescu, P. Varshavskaya, K. Kotay, and D. Rus, "Autonomous modular optical underwater robot (AMOUR) design, prototype and feasibility study," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 1603–1609.

[8] E. H. Østergaard, D. J. Christensen, P. Eggenberger, T. Taylor, P. Ottery, and H. H. Lund, "HYDRA: From Cellular Biology to Shape-Changing Artefacts," in *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, vol. 3696, pp. 275–281.

[9] M. S. Steinberg, "Reconstruction of tissues by dissociated cells," *Science*, vol. 141, no. 3579, pp. 401–408, 1963.

[10] G. M. Whitesides and B. Grzybowski, "Self-Assembly at All Scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002.

[11] P. White, V. Zykov, J. C. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots." in *Robotics: Science and Systems*. Cambridge, 2005, pp. 161–168.

[12] P. White, K. Kopanski, and H. Lipson, "Stochastic self-reconfigurable cellular robotics," in *IEEE International Conference on Robotics and Automation*, vol. 3, April 2004, pp. 2888–2893.

[13] J. Bishop, S. Burden, E. Klavins, R. Kreisberg, W. Malone, N. Napp, and T. Nguyen, "Programmable parts: a demonstration of the grammatical approach to self-organization," in *IEEE International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 3684–3691.

[14] M. Yim, D. Duff, and K. Roufas, "Polybot: a modular reconfigurable robot," in *IEEE International Conference on Robotics and Automation*, vol. 1, 2000, pp. 514–520.

[15] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: self-reconfigurable modular robotic system," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 431–441, Dec 2002.

[16] E. H. Østergaard, K. Kassow, R. Beck, and H. Lund, "Design of the ATRON lattice-based self-reconfigurable robotatron lattice-based self-reconfigurable robotatron lattice-based self-reconfigurable robot," *Autonomous Robots*, vol. 21, no. 2, pp. 165–183, 2006.

[17] I. O'Hara, J. Paulos, J. Davey, N. Eckenstein, N. Doshi, T. Tosun, J. Greco, J. Seo, M. Turpin, V. Kumar, and M. Yim, "Self-Assembly of a swarm of autonomous boats into floating structures," in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 1234–1240.

[18] V. Ganesan and M. Chitre, "Self-Assembling Robots in an Underwater environment," in *Proceedings of MTS/IEEE OCEANS*, 2015.

[19] A. L. Christensen, R. O'Grady, and M. Dorigo, "SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 143–165, 2008.

[20] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.

[21] E. Klavins, R. Ghrist, and D. Lipsky, "A grammatical approach to self-organizing robotic systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 6, pp. 949–962, June 2006.

[22] N. Napp, S. Burden, and E. Klavins, "The statistical dynamics of programmed self-assembly," in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 1469–1476.

[23] K. Hosokawa, I. Shimoyama, and H. Miura, "Dynamics of self-assembling systems: Analogy with chemical kinetics," *Artificial Life*, vol. 1, no. 4, pp. 413–427, 1994.

[24] Open dynamics engine. [Online]. Available: http://www.ode.org