Self-assembly in heterogeneous modular robots

Wenguo Liu and Alan F.T. Winfield

Abstract This paper describes the distributed self-assembly of a multi-robot 'organism' from a swarm of autonomous heterogeneous modular mobile robots. A distributed self-assembly strategy based on a symbol sequence representation is proposed. Constructed from a tree representation of an organism, the symbol sequence is presented as a well organised nested structure in a compact format. It includes not only information on the topology of the organism but also how the organism will be self-assembled. The proposed approach has been tested with real robot prototypes. Results show that robots can successfully self-assemble to required target body plans within certain time frames.

1 Introduction

Self-reconfigurable modular robots has became a popular research topic since late 1980's. From the CEBOT [1] to the SYMBRION project [2], more than 30 systems have been developed over 2 decades. Although earlier studies focused mainly on solving the mechanical engineering challenges for designing the hardware, recent research has paid more attention to high level algorithms toward greater controllability of self-assembly, self-reconfigurable and self-repair of the system. The complexity of the modules and the architecture of the system vary in different studies. Some use a lattice architecture where docking/undocking can only occur at points within some virtual cells. The lattice architecture requires a simpler mechanical design and simplifies the computational representation, thus the reconfiguration planning is more achievable and scalable. Some other systems do not use the virtual cell as the docking point for their units. Instead, a number of modules can form a chain to reach any point in the operating space; the so called chain architecture. To

Wenguo Liu and Alan F. T. Winfield
Bristol Robotics Laboratory, University of the West of England, Bristol, UK. e-mail: wenguo.liu@brl.ac.uk, alan.winfield@uwe.ac.uk

get to a specific point and carry out reconfiguration, the chain architecture requires a more complicated design with additional sensing. To overcome the limitation of both approaches, a hybrid architecture is favoured by many recent designs. Among these, a singular design of individual module is normally used, i.e. a homogeneous design. Apart from the essential functionality requirement of the modules, e.g. mechanical docking, some degree of freedom of rotation, robots have been provided with a very limited range of sensors as appropriate to the design challenge and research interests. Also, few designs have considered providing individual robots with autonomous motion capabilities. This typically requires that modules have to be manually attached to each other prior to any reconfiguration process. The lack of sensing and mobility limits the possibility of re-assembling after pre-assembled structures have fallen apart either purposely or accidentally.

To close the gap between the modular robotic systems and multiple/swarm robotic systems, more recently researchers are seeking new designs that are capable of autonomous self-assembly and self-reconfiguration. The SamBot [3] project consists of a group of identical robots. Each robot is equipped with two differential driven wheels and several IR sensors for autonomous self-assembly. As with many other modular robotic systems, a rotation arm is used in SamBot to provide one extra degree of rotation freedom thus enabling the self-reconfiguration capability. In contrast, the SYMBRION project (www.symbrion.eu) takes a heterogeneous approach, and robots with more sensing, computation and actuation. Various modules with complementary motion capabilities have been developed for this project. The main focus of SYMBRION is to investigate the controllability and evolvability of artificial multi-cellular organisms from both engineering and scientific perspectives. The SYMBRION robots can either work fully autonomously with their own sensing and locomotion capabilities to explore the environment, or physically dock with each other forming different organism structures to accomplish more complex tasks that a single robot is not capable of, for example, climbing a wall or moving over a gap.

One of the fundamental requirements of the SYMBRION project is that robots must be able to self-assemble into a given structure (body plan) starting from a single module. To achieve this, apart from the autonomous docking/undocking capabilities, we need the right morphology control mechanism. A bio-inspired gradient based process has been widely used to study the pattern growth problem in agent-based cell systems [5, 4, 6], and later adapted to the modular robotic systems [7]. This approach has however been tested only in simulation and for homogeneous systems. In addition, the uncertainty of the final generated body shapes imposes another challenge for controlling the macro-locomotion of the whole structure at later stage. An alternative is to build the target shapes, which are known to be controllable, from some stored pre-defined body shapes. A SWARMMORPH-script language has been proposed by Christensen for arbitrary morphology generation for a group of *s-bot* robots in a 2D environment [8]. The morphologies are pre-specified as sets of rules stored in scripts which can be communicated and subsequently executed on the newly connected robot. Note that unlike the *s-bot* robots, the SYMBRION robots will initially form a 2D planar structure and then lift itself from 2D planar

configuration to 3D configuration and, with respect to locomotion, will function as a macroscopic whole. The aggregated organism will also be able to disassemble and reassemble into different morphologies to fit the requirements of the task.

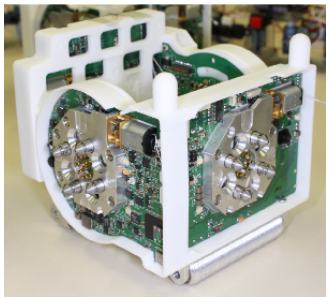
In our previous work, an ID-based strategy was proposed to self-assemble the SYMBRION robots into certain predefined 2D structures [9]. The ID-based approach assumed that all robots in the swarm are identical and each robot stores the same set of predefined structures information. In addition, this strategy allows only one robot to join the organism at a time, using a fixed docking face, more specifically, the Front side, which constraints the toplogy of the shapes the swarm can self-assemble to. This paper will extend the previous work by removing all these assumptions and limitations. A new strategy will be developed to enable the parallel recruiting and any-side-docking in order to improve the efficiency and robustness of the self-assembly strategy. In particular, the new propsed strategy will take the heterogeneity of the system into account.

The rest of the paper is organised as follows: Section 2 introduces the robots of the SYMBRION project and the hardware configuration for autonomous docking. Section 3 proposes the internal representation of organism body plan for the self-assembly process. Section 4 outlines the controller framework and discusses the morphology control strategy. The proposed approach is validated in Section 5 using simulation and real robot experiments. The paper ends by drawing some conclusions and further work in Section 6.

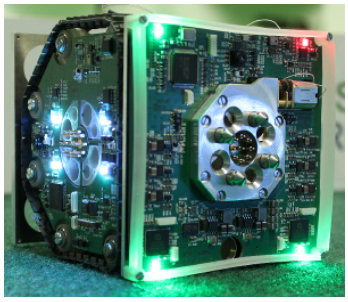
2 The Robots

Figure 1 shows three types of robots developed in the project. These robots have different shapes and are equipped with different locomotion actuators. The Backbone robot has two specially designed wheels which allow the robot to move forwards, backwards and sideways; this robot requires a flat surface. With its tracked locomotion the Scout robot can move across uneven surfaces and is suitable for exploration tasks. Like many other modular robots, both Backbone and Scout robots have cubical shapes with four docking faces and one degree of freedom of bending. The ActiveWheel robot is designed to carry and transport an organism consisting of several Scout or Backbone robots in the most energy-efficient way. It consists of two symmetrically arranged arms, connected via a 180° turning hinge, and 4 omni-wheels. Two docking elements are placed on the same axis of the hinge for docking with other robots.

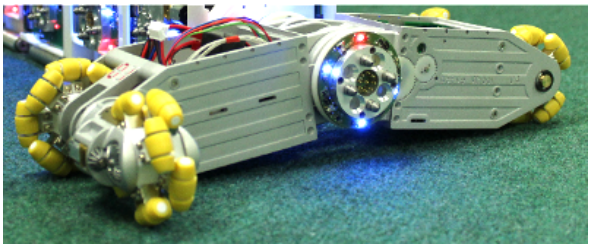
Some unified docking elements are placed on all three types of robots to allow stable physical connections between robots. In addition, electrical contacts next to the docking units can be coupled automatically to provide inter-robot communication and power sharing busses between two connected robots. As shown in Figure 2, two versions of docking elements are installed on robots: an active docking unit and a passive docking unit. By removing the locking mechanism, passive docking units can reduce the size of the assembly while still providing all other mechanical and



(a) BackBone robot

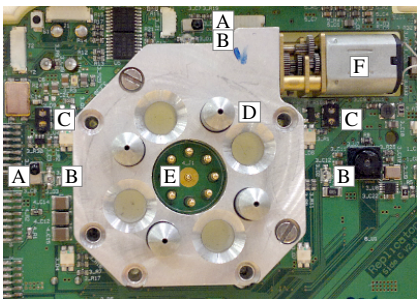


(b) Scout Robot

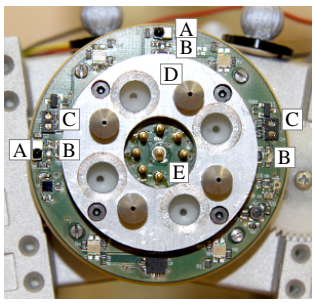


(c) ActiveWheel Robot

Fig. 1: Robot prototypes. The BackBone robot has 4 active docking faces. The Scout robot has 2 active docking faces and 2 passive docking faces, while the ActiveWheel has only 2 passive docking faces.



(a) Active docking unit



(b) Passive docking unit

Fig. 2: Unified docking mechanisms: A – IR receivers, B – IR LEDs, C – IR sensors, D – docking mechanics, E – electrical docking contacts, F – locking motor.

electrical features. Note that for any valid robot-robot connection, at least one active docking unit needs to be present so that two docking units can be locked securely.

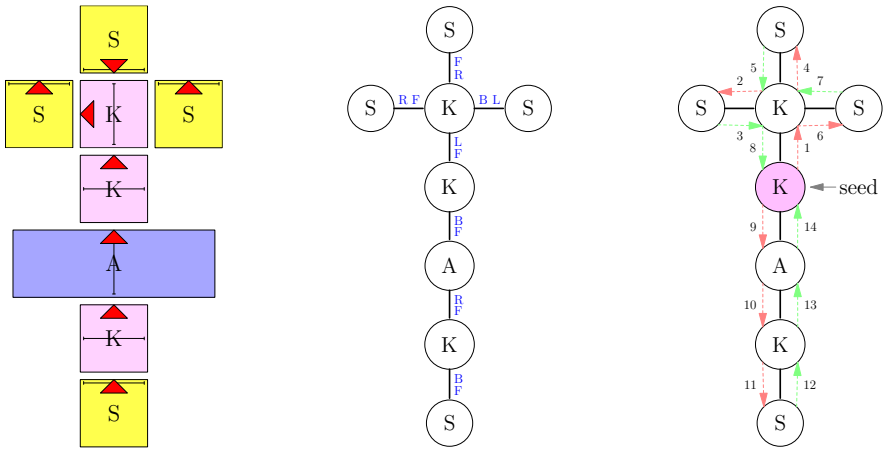
Infrared(IR)-based sensing - including proximity detection, docking alignment detection and local communications circuits - has been developed for the SYMBRION robot to achieve autonomous docking in a 2D planar environment [10]. These sensors have similar placement on each docking face of the robot. More specifically, two IR sensors have been placed symmetrically above and on either side of the docking unit; one IR LED is placed directly above the docking unit, while the other two LEDS are located on either side of the docking unit. These LEDs are used to emit different frequency signals for obstacle detection, docking alignment and communication. The IR sensors work for both obstacle detection and docking alignment detection. IR remote control receivers are placed next to the IR LED on each docking face for communications.

3 Internal representation of organism body plan

To investigate how specified organism shapes can be self-assembled from a swarm of freely mobile robots, a common representation of the target shape must first be defined. The representation is also essential for the topology exploration of the organism in the macro-locomotion and the self-reconfiguration tasks. This study assumes that the self-assembly process always occurs in a 2D planar environment between a partially formed organism and a group of freely moving robots where individual robots join the growing organism using their own locomotion. The hardware constraints of the robot platform indicate that at any time one robot can attach to the partially formed organism using only one docking port. Thus no circular paths can appear among several connected modules in the target organism shapes. The body plan of an organism in a 2D planar environment can therefore be represented as a tree structure T. If let $T = (V,C)$, then V is a set of nodes and C represents a set of connections. Each node $v \in \{r_1, r_2, ..., r_m\}$ corresponds to a robot module in the organism, where r_i is the type of robots and m is the total number of types in the system. Each connection $c = ((v_1, o_1), (v_2, o_2))$ includes a pair of nodes (v_1, v_2) and the orientation of corresponding connection ports, o_1 and o_2. Note that each type of robot r_i can have k_i connection ports. Clearly, unlike the common representation of a tree, there is no root node in the organism body plan and each node may have a different type.

There are many different ways to represent trees in a computer system. The design of data structures depend on the algorithm used for the tree. As there is no central control module in the complete organism, information on the body shape will have to be transferred and stored across all modules. Besides a smaller memory requirement, a data structure which can be easily manipulated and transferred via common robot-robot communication means is preferred. To satisfy these requirements, this paper proposes a well formatted parenthesis symbol sequence to represent the body plan of the organism. Let T be a tree representation of an organism

with n robots, a symbol sequence S of tree T is defined as a sequence of $2(n-1)$ symbols s generated from a depth first traversal of T. There are two kinds of symbols in a sequence, s_c and s_p, where $s_c = f(c)$ is a "connection" symbol, which corresponds to a connection $c = ((v_1, o_1), (v_2, o_2))$ in T using a mapping function $f()$, while s_p is a special "pairing" symbol. The size of "connection" symbol $|s_c|$ equals $(\sum k_i)^2$ and "pairing" symbol $|s_p|$ is 1.



(a) Organism body plan (b) Tree represention (c) Order of traversal

Fig. 3: An example of organism body plan and its tree representation. 'K' represents BackBone robot, 'A' for ActiveWheel robot and 'S' for Scout robot. 'F', 'R', 'B', 'L' denotes the Front, Right, Back and Left docking ports of a robot respectively.

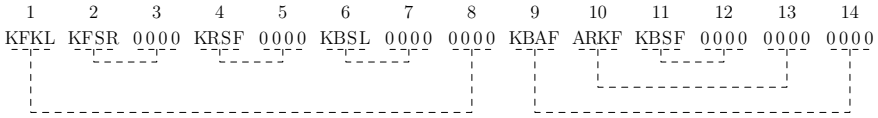
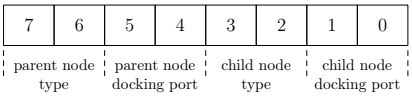


Fig. 4: An example symbol sequence of the organism. Each pair of s_c and s_p symbols, annotated and connected with dashed lines, represent one edge in its tree representation.

To obtain a symbol sequence from a tree representation, a node, denoted as v_o, must first be chosen as the starting point of the traversal. Depending on the starting node, different symbol sequences can be generated for the same tree. Compared to the general tree representation, a symbol sequence representation is encoded with the information where the organism starts to grow, which is indeed desirable for the morphology control mechanism introduced in the next section. Take the organism

body plan shown in Fig.3(a) as an example. There are 3 Backbone, 4 Scout and 1 ActiveWheel robots in the organism where the headings are indicated with triangles and the rotation axes are marked with a bar. Its corresponding tree representation is shown in Fig.3(b) and all connected docking ports are labelled along the edges. Starting from a node marked with dark colour, Fig.3(c) depicts the order of each edge being visited in a depth first traversal, which follows the travel order of "Front $\rightarrow$ Right $\rightarrow$ Back $\rightarrow$ Left" for each node. Through the traversal, a s_c symbol is added to a symbol sequence if an edge is first traversed and a s_p symbol is added each time an edge is traversed in the opposite direction. For simplification, let s_c to be a string in a format of "parent node type | parent connection side | child node type | child connection side", where each node type or connection side is denoted with one character, and s_p to be a string of "0000", then the generated symbol sequence can be presented as shown in Fig. 4. Here the node types are abbreviated as 'K', 'S' and 'A' for Backbone, Scout and Activewheel robot respectively, while connection sides 'F', 'R', 'B' and 'L' stand for Front, Right, Back and Left respectively. For instance, a symbol of "KFKL" can be read as a traversal along the edge from a Backbone robot's front side to another Backbone robot's left side. Also shown in 4, the symbol sequence can be annotated as a nested structure. For each s_c symbol, there is a corresponding pairing symbol s_p in the sequence. Each pair of s_c and s_p symbols corresponds one edge in the tree representation. Their relative positions then depict the topology of the tree structure.

A symbol sequence can be obtained recursively from a depth first traversal of the tree. By carefully choosing the mapping function $f(c)$ and pairing symbol s_p, the generated symbol sequence requires only a small amount of memory to be stored and transferred. For example, in this study each s_c symbol is constructed using only 1 byte, with every two bits storing the robot type or the orientation of the connected docking port as shown below. If let 1 - 3 denote the robot type 'K', 'S' and 'A'



respectively and 0 - 3 correspond to the connection ports 'F', 'R', 'B' and 'L' respectively, then a symbol of "KFKL" can be depicted as 0b01000111 = 0x47, while symbol s_p can be identified using 0.

Because of the well organised format and nested structure of a symbol sequence, its tree representation can be reconstructed using Algorithm 1. This is important as the topology information can be more easily accessed from a tree representation for macro-locomotion control of a complete organism, while the symbol sequence representation is more convenient to be shared among the robots. Meanwhile, it provides a trival way to check any geometrical confliction of the body shape that a symbol sequence stands for. Fox eample, if the geometrical distance between any two non-neighbouring robots in the reconstructed tree is less than certain value, the

body plan can not be self-assembled because of the physical interference between robots.

Algorithm 1 symbol sequence to graph representation

Precedure: OgSequenceTraversal (symbol_sequence S, tree T, node v)
1: **for** each branch sequence S_b in S **do**
2: create new node w from S_b.symbols[0];
3: insert w to T
4: **if** v is not empty node **then**
5: connect w with v;
6: **end if**
7: create a new symbol sequence S_1 by removing the first and last symbol of S_b;
8: OgSequenceTravelsal(S_1, T, w);
9: **end for**

4 Self-assembly strategy

In SYMBRION scenarios, depending on whether physically connected to other robots, a robot works in one of two modes: swarm mode and organism mode, and switches between them accordingly. The transition from swarm mode to organism mode is determined by the morphology control strategy. In general, successful docking requires the collaboration between two robots: one remains stationary, called the recruiter, which emits some guiding signals; the other moves and aligns along these signals and then docks to the recruiter. To build an organism shape, the process needs to be initialised by a robot, called the seed. During the self-assembly process, one freely moving robot can dock and join to the partially formed organism only when it is signalled. To grow the correct body shape, it is critical that the right robots in the organism mode become recruiters at the right time, and the right type of freely moving robots respond to the recruiters.

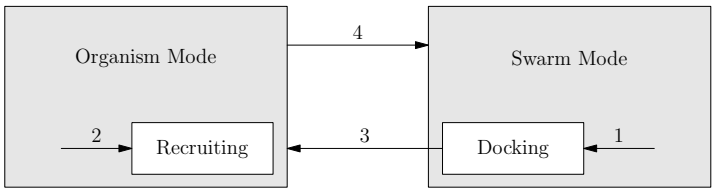


Fig. 5: Controller framework for the SYMBRION robots. Transition conditions: 1 – recruiting signals are received and request type matches; 2 – recruiting is required as per morphology control strategy; 3 – docking is accomplished; 4 – disassembly is required.

This paper adopts a finite state machine as the controller framework for all robots. Figure 5 shows part of the controller which focuses on the topic of this paper, a complete version can be found in [9]. Once the self-assembly process starts, the seed robot becomes the first recruiter in the organism and thus moves to state *Recruiting*. Assuming that the target organism shape has been already stored as the symbol sequence in its memory, the seed robot extracts the branch symbol sequences from the complete symbol sequence. A branch symbol sequence S_b of S is defined as a sub symbol sequences between one of the starting node v_o associated s_c symbol and corresponding paired s_p symbol. The number of branch symbol sequences equals the number of associated s_c symbols of S. For example, the symbol sequence shown in Fig.4 has two branch symbol sequences: "KFKL KFSR 0000 KRSF 0000 KBSL 0000 0000" and "KBAF ARKF KBSF 0000 0000 0000". Each branch sequence is in fact endowed with connection information to its descendants in the target organism body shape. The seed robot can therefore use this information to decide which types of robot need to be recruited from which docking port. The behaviours in the *Recruiting* state can be grouped into different stages:

stage 1 broadcast recruiting message from corresponding docking ports, indicating the types and connection side of the robot recruited;
stage 2 emit guiding signals to help docking robots to align to them;
stage 3 synchronise the locking process if required, as both active docking unit and passive docking unit may be involved;
stage 4 send the branch symbol sequences to newly docked robots.

Any robots in swarm mode may become docking robots when recruiting messages are received and their type matches with that requested. The docking robot then moves close to the recruiter guided by the signals and docks to it using the correct docking port. Upon receiving the branch symbol sequence S_b from the recruiter, it extracts a new symbol sequence S_1 by removing the first s_c symbol and the last s_p symbol in S_b. This new symbol sequence represents the sub tree which is rooted from itself. If there are branch sequences in the symbol sequence S_1, the newly joined robot becomes a recruiter and moves into state *Recruiting* to recruit more robots into the organism. Otherwise it changes state to organism mode and hence stops any further growth of the organism from itself. A recruiter exits the *Recruiting* state once all its required docking ports are joined by other robots. Algorithm 2 outlines the behaviours for the recruiter and docking robots. Note that the exact behaviours in each state can have different implementations because of the heterogeneity of robots.

The morphogenesis process completes when all recruiters exit the *Recruiting* state. As the self-assembly process is fully distributed and no single module acts the coordinator for the growth of the organism, once the process has been initialised by a seed robot, it is important that all robots are aware of completion of the morphogenesis process and then behave accordingly to transform the organism from 2D to 3D then initialise the macro-locomotion controller. One solution to check the self-assembly process is to pass a message token across the developing organism periodically to request all robots to register if they are recruiters or not. This message

Algorithm 2 Self-assembly strategy for recruiter and docking robots

```
Behaviour: in state Docking
if docking is NOT accomplished then
    locate and align to the recruiter;
else if new symbol sequence information received then
    extract the branches from received symbol sequence;
    if branches exisit then
        enable the corresponding docking ports;
        start to emit beacon signals;
        move to state Recruitment;
    else
        issue a message token to check progress of self-assembly;
        if message token is returned then
            if No recruiters are presented then
                notify the completion of self-assembly;
            end if
            move to organism mode;
        end if
    end if
end if
Behaviour: in state Recruitment
if All required docking port is docked then
    move to organism mode;
else
    for Each recruiting docking port do
        if new robot is docked then
            send branch symbol sequence;
            stop emitting guiding signals;
        else if currenttime % RECRUITMENT_SIGNAL_INTERVAL then
            broadcast recruiting message;
        end if
    end for
end if
```

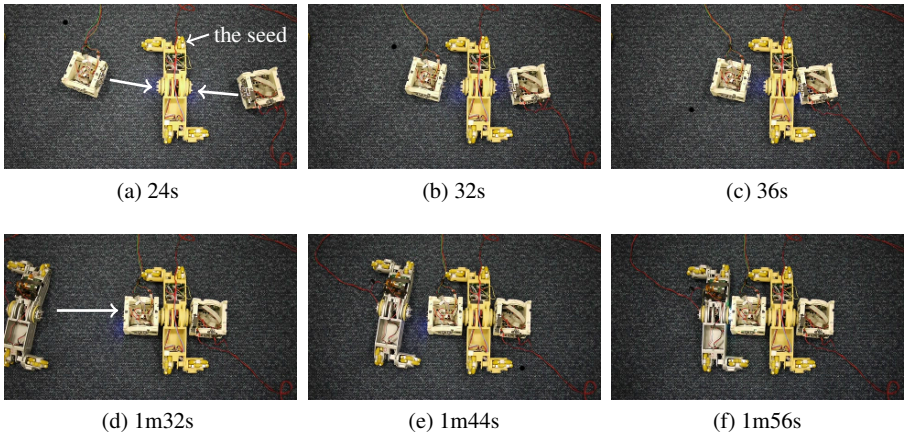
token travels across the organism following some rules, for example in the same way a symbol sequence is generated, and back to the original sender. Once it shows no recruiters are present in the organism, the self-assembly process is assumed finished and the sender will issue another message to notify the success of morphogenesis to all robots. The selection of such a sender remains open. It can be any robot in the organism. A good candidate is however the most recently joined robot that has no other robots to be recruited, i.e. a leaf node robot in the tree representation. In other words, whenever a leaf node robot joins the organism, a progress check of self-assembly will be performed.

Communications, both wireless (IR) and wired (Ethernet), are very important in synchronising the high level self-assembly strategy and also the low level autonomous docking process between two robots. Due to the interference of IR signals and low bandwidth, messages are more likely to get lost when transferred wirelessly. To address this issue, IR messages should normally be kept as simple (short) as possible. Meanwhile, each IR message may need to be repeated several times until it

been acknowledged. A full analysis of effect of communication failures on the algorithm is presented in [11].

5 Experiments and discussion

The proposed controller and self-assembly strategy has been implemented and tested with a small number of real prototype SYMBRION robots. Each robot runs the same controller framework as described in the paper and utilises the IR signals for recruiting and docking alignment. Autonomous docking between the recruiter and a docking robot is synchronised via local sensing and wireless communication. The locomotion control for each type of robots has been carefully designed according to their unique hardware specification. As the initialisation of the self-assembly process is out the scope of this study, each time a seed robot is chosen manually with a predefined symbol sequence to start the self-assembly process. Fig. 6 shows the progress of autonomous self-assembly of an organism with 2 ActiveWheel and 2 BackBone robots. The approach was successfully and repeatably demonstrated for a number of configurations and seed robot positions. In all cases we see that the target organism shape is formed within a certain time frame and all robots in the organism are notified of the success of the process using the progress checking mechanism. Videos of different experiments are available online at http://goo.gl/VPGGm.



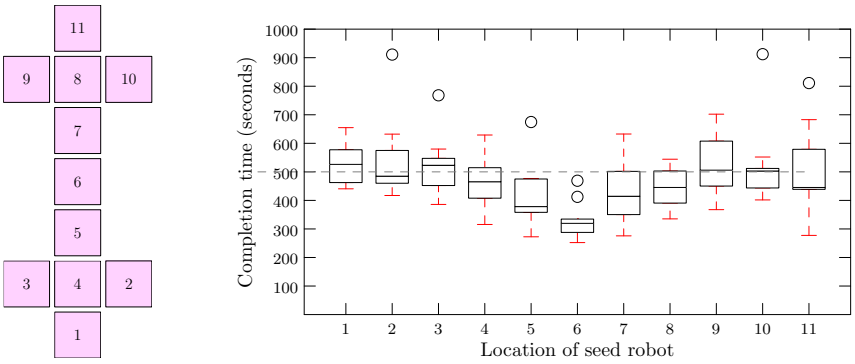
(a) 24s (b) 32s (c) 36s

(d) 1m32s (e) 1m44s (f) 1m56s

Fig. 6: Self-assembly experiment with 2 Backbone and 2 ActiveWheel robots. The experiment is initialised by the ActiveWheel in the middle with a symbol sequence of “AFKF0000ABKFKBAF00000000”.

Since a symbol sequence representation includes also the location information of the seed robot in the organism, a selection of different locations for the seed, thus

different symbol sequences, do indeed result in building the same organism. Although robots have to join the organism one after another, the potential for multiple recruiters in the self-assembly process shows that the growth of the organism is in fact a parallel process. It follows that the position of the seed robot (in the organism) can affect the completion time of the self-assembly process. To evaluate this, experiments are performed in Robot3D [12] simulator, instead of real robots as only 4 SYMBRION robots were availabe for testing at the time of writing, for growing the same body plan as shown in Fig.7a. In all cases, 25 robots are initially deployed uniformly in an arena sized 4 m $\times$ 4 m, and the pre-selected seed is always located in the centre of the arena with varying headings in 10 repeated runs.



(a) (b)

Fig. 7: Comparison of completion time for self-assembly initialised by seed robots in different positions in the organism. Each box in the plot represents the first to the third quartile of the data from 10 experimental runs.

Fig. 7b compares the completion time for self-assembly with seed robots in different positions. We see clearly that the completion time varies with different seed robot positions. It is obvious that the more parallel recruitment in the developing organism, the quicker the growth process. As expected, a seed located in the centre of the organism, i.e. postion 6, leads to the fastest growth with an average completion time of 330 seconds. The further the seed from the centre of the organism, the longer the completion time; for example, a seed from postion 1 has an average completion time of 539 seconds. Since the tested target organism shape has a symmetrical topology, the completion time is very close when the position of the seed robots are similar. For example, the seeds next to the centre of the organism, position 5 and 7, give average completion times of 415 and 425 seconds respectively. In order to optimise the efficiency of the self-assembly process, we therefore require that the process needs to be initialised by a seed robot located as close as possible to the centre of the target body shape. To check whether a seed robot with a particular symbol sequence could result in the most efficient self-assembly process, we can count the number of edges in each branch sequence for this symbol sequence, if all

of them are less than half of totel number of edges of the parent symbol sequence, then the associated seed robot is located in the center of the body shape that this symbol sequence represents.

6 Conclusion and future work

This paper presents a fully distributed algorithm for morphology control in a group of heterogeneous modular self-assembling robots. Each robot in the system is fully autonomous with its own sensing and locomotion, and able to physically join with others using a unified docking mechanism in a 2D planar environment. This study focuses on how specific organism body shapes can be built, starting with a seed robot. A well organised symbol sequence structure has been proposed to represent the target organism body plan and also to form the basis of the self-assembly strategy. The symbol sequence representation encodes not only the topology of the target body shape but also information on how the shape should grow from one robot. Its compact nested structure allows the target body shape to be stored and transferred among robots with minimum memory and low bandwidth communication, which is a requirement of robots with limited resources. Depending on the topology of the body plan, multiple robots may join the partially formed organism from different positions simultaneously. Each time a robot joins the partially formed 2D organism, it receives a branch symbol sequence from its recruiter. Thus, during self-assembly, only the seed robot stores the entire body plan of the target organism; other robots store only parts of the body shape, including itself and its branch. There is a big advantage compared to the ID-based single entry self-assembly strategy proposed in [9]. In case there are failures in the self-assembly process, for example, a newly joined robot malfuctions, only that branch of body shape will be affected, and organism growth from other points will still go on. If proper fault detection and recovery mechanisms are introduced to detect the faults and remove the malfuction robot, the self-assembly from that point can continue as the shape information can be retrieved directly from the parent recruiter robot.

Since no complex communication protocols and conflict resolution mechanisms are present in the recruitment-docking process, more than one robot can be attracted to the same docking port of a recruiter at the same time. Competition among these robots can inevitably increase self-assembly completion time. We have tested a simple “expelling” message, broadcast by the docking robot as a way of reducing the possible competition; however this has only been tested in simulation with idealised local communication. How efficient this conflict resolution mechanism is and how robustly it works with a large swarm of real robots need to be further investigated. Note that there are many ways in which faults on hardware might disrupt the self-assembly process including, for instance, mechanical failure of the docking mechanism or failure of the power or communications buses across the docking mechanism. Extending and adapting the algorithm to compensate for such faults during self-assembly is ongoing work as presented in [11].

Acknowledgements The SYMBRION project is funded by the European Commission within the work programme Future Emerging Technologies Proactive under grant agreement No. 216342.

References

1. Fukuda, T., Nakagawa, S.: Dynamicaly reconfigurable robotic system. In: Proceeding IEEE international conference on Robotics and Automation, vol. 3, pp. 1581 – 1586 (1988).
2. Levi, P., Kernbach, S. (eds.): Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution. Springer (2010)
3. Wei, H., Chen, Y., Tan, J., Wang, T.: Sambot: A self-assembly modular robot systems. IEEE/ASME Transactions on Mechatronics **16**(4), 745–757 (2011).
4. Doursat, R.: Organically Grown Architectures: Creating Decentralized, Autonomous Systems by Embryomorphic Engineering, *Understanding Complex Systems*, vol. 21, pp. 167–199. Springer (2008).
5. Nagpal, R.: Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics. Ph.D. thesis, Massachusetts Institute of Technology (2001)
6. Werfel, J.: Biologically realistic primitives for engineered morphogenesis. In: Proceedings of the 7th international conference on Swarm intelligence ANTS'10, pp. 131–142. Springer, Belgium (2010).
7. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems **54**, 135–141 (2006).
8. Christensen, A., O'Grady, R., Dorigo, M.: SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence **2**(2), 143–165 (2008).
9. Liu, W., Winfield, A.F.: Autonomous morphogenesis in self-assembling robots using IR-based sensing and local communications. In: Proceedings of the 7th international conference on Swarm intelligence ANTS'10, pp. 107–118. Springer-Verlag (2010).
10. Liu, W., Winfield, A.: Implementation of an IR approach for autonomous docking in a self-configurable robotics system. In: Proceedings of Towards Autonomous Robotic Systems, pp. 251 – 258 (2009)
11. Murray, L, Liu, W, Winfield, A, Timmis, J, and Tyrrell, A: Analysing the Reliability of a Self-reconfigurable Modular Robotic System, In Proceedings of International ICST Conference on Bio-Inspired Models of Network, Information and Computing Systems (BIONETICS 2011), York, (2011).
12. Winkler, L., Wörn, H.: Symbricator3D – a distributed simulation environment for modular robots. In: Proceedings of the 2nd International Conference on Intelligent Robotics and Applications, pp. 1266–1277 (2009).