

This is a repository copy of *Self-assembly and Self-repair during Motion with Modular Robots*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/186796/>

Version: Accepted Version

Article:

Peck, Robert, Timmis, Jon orcid.org/0000-0003-1055-0471 and Tyrrell, Andy orcid.org/0000-0002-8533-2404 (2022) *Self-assembly and Self-repair during Motion with Modular Robots*. *Electronics*. 1595. pp. 1-33. ISSN 2079-9292



Reuse

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Self-assembly and Self-repair during Motion with Modular Robots

Robert H. Peck ¹ *, Jon Timmis ²  and Andy M. Tyrrell ³ *

¹ Department of Electronic Engineering, University of York; robert.h.peck@york.ac.uk ¹

² School of Computer Science, University of Sunderland; Jon.Timmis@sunderland.ac.uk

³ Department of Electronic Engineering, University of York; andy.tyrrell@york.ac.uk

* Correspondence: robert.h.peck@york.ac.uk, andy.tyrrell@york.ac.uk

Abstract: Self-reconfigurable modular robots consist of multiple modular elements and have the potential to enable future autonomous systems to adapt themselves to handle unstructured environments, novel tasks, or damage to their constituent elements. This paper considers methods of self-assembly, bringing together robotic modules to form larger organism-like structures, and self-repair, removing and replacing faulty modules damaged by internal events or environmental phenomena, which allow group tasks for the multi-robot organism to continue to progress while assembly and repair take place. We show that such “in motion” strategies can successfully assemble and repair a range of structures. Previously developed self-assembly and self-repair strategies have required group tasks to be halted before they could begin. This paper finds that self-assembly and self-repair methods able to operate during group tasks can enable faster completion of the task than previous strategies, and provide reliability benefits in some circumstances. The practicality of these new methods is shown with physical hardware demonstrations. These results show the feasibility of assembling and repairing modular robots whilst other tasks are in progress.

Keywords: dynamic self-assembly; dynamic self-repair; modular robots; self-assembly; self-repair; morphogenesis; modular

Citation: Peck, R. H.; Timmis, J.; Tyrrell, A.M. Self-assembly and Self-repair during Motion with Modular Robots. *Electronics* **2022**, *11*, 0. <https://doi.org/>

Received:

Accepted:

Published:

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Submitted to *Electronics* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Modular robots, first proposed in [1], can dock together to form “organisms” from a number of modules. Once formed, these organisms have abilities beyond those of a single lone module [2], allowing collective action for a period as a single, larger, organism [3]. Compared to current large, specialised, non-modular robots, modular robots will, once a matured technology, enjoy a variety of advantages stemming from the fact that extending the robot’s capabilities becomes a matter of adding additional modules rather than the significant redesign and rebuilding which would be necessary to add functionality to a large, specialised, “monolithic” robot.

Self-reconfigurable modular robots are able to reconfigure between morphologies for their group organism without human intervention, platforms such as SMORES [4], SYMBRION [5], HyMod [6] and Omni-Pi-tent [7] have modules with docking equipment which are individually mobile and can shift between discrete lattice positions or relative locations in continuous space to form different shapes.

NASA has taken great interest in the concept of modular robotics for space missions [8], with some modular robot studies [9] aiming towards this specific use case. Groups of modular robots could be sent to a region of interest in the solar system, reconfiguration would allow them to alter themselves for a range of terrains including unexpected obstacles. They could reconfigure for object manipulation as well as group motion tasks [9]. If some of these units fail, then others should still be able to accomplish many of a

¹ Now with the Institute for Safe Autonomy, University of York

36 mission's goals, this differs from present space robotics where a minor failure somewhere
37 can doom an entire mission. Using such reconfigurable modules would also provide a
38 useful reduction in payload mass and therefore propellant requirements.

39 Another field where modular robots may be of use is search-and-rescue in disaster
40 zones [10], in such circumstances transforming in morphology to, for example, a snake
41 configuration to allow them to enter narrow crevices under rubble before changing back
42 to object manipulation or open-space locomotion morphologies when space permitted.
43 This is, in many ways, a more environmentally chaotic and unstable equivalent to the
44 "pipe and tank" which [1] proposed as an ideal use case for modular robots.

45 Another field of use which could see demand for modular robots is that of infras-
46 tructure monitoring. With predictions of \$3407.7M being spent in 2022 on periodic
47 inspections of civil infrastructure, this is a field which already makes extensive use
48 of specialised robots to access hard-to-reach places [11]. Unlike static sensors, robot
49 inspectors can follow cracks to their source and seek out spots from which the most vital
50 information can be gleaned. Sending modular robots in, as described by [11], would
51 provide many advantages over using monolithic robots, the most obvious being that by
52 reconfiguring when necessary a single platform could perform a wide variety of tasks
53 while navigating across any infrastructure and that this single platform could also be
54 used on other items of infrastructure. By removing the need to specialise robots for
55 specific tasks on specific infrastructure, installation costs can be reduced and equipment
56 can be more readily available to perform inspections more often. A further feature
57 enabled by modularity is the ability to leave modules behind for a period of time to act
58 as temporary static sensors [11], allowing them to replicate the versatility of robots and
59 the wide observation timeframe of static sensors.

60 Work presented in this paper, builds on previous work [12], which aimed to develop
61 strategies to self-repair while the robot organism maintains collective actions. In that
62 paper a hypothesis for the Dynamic Self-repair project was stated that: *"Modular robots
63 using a self-repair strategy which can operate while the group maintains collective motion will be
64 able to complete their mission faster, and more effectively, than robots of the same hardware design
65 which are using a self-repair strategy which requires the system to stop and repair before resuming
66 motion"* [12]. It was suggested that this "Dynamic Self-repair" could be especially useful
67 in time critical situations, where robots must maintain the ability to take urgent group
68 actions in reaction to sudden stimuli, and in situations where mobility of modules is
69 restricted by environmental forces or obstacles.

70 The process of self-assembly, also referred to in the modular robotics field as mor-
71 phogenesis [13], allows a collection of independent modules to join together, without
72 needing to be directed by external systems, to form a connected organism which can
73 handle tasks beyond the capabilities of a single module. The process of self-repair allows
74 the removal of failed modules and the bringing in of spares to replace them [14], this
75 could prove to be a game changing feature of such systems in contrast to existing robots.

76 This paper focuses on the development of strategies for self-assembly and self-
77 repair with modular robots. The key novel aspect of the work here is the ability of our
78 new self-assembly and self-repair methods to function while the robots are in continuous
79 motion. We use strongly physically inspired V-REP [15] simulations (based around the
80 Omni-Pi-tent hardware design [7]) to compare the performance of new strategies, with
81 the ability to operate while in motion, to classical self-assembly and self-repair strategies
82 based on the work of Liu and Winfield [13] and Murray [10], these comparisons are
83 performed during a scenario in which robots must both perform the assembly or repair
84 operation and complete a locomotion task.

85 This paper's key contributions include:

- 86 • Demonstration that self-assembly and self-repair can be performed whilst a modular
87 robotic organism is simultaneously engaged in another task.
- 88 • A novel "Quadruplet" data structure for describing modular robotic structures.

- 89 • A series of algorithms designed to enable physically feasible self-repair and self-
90 assembly by processing these "Quadruplet" data structures.
- 91 • Demonstration of the use of onboard sensors to enable autonomous docking of
92 modular robotic hardware to moving seed modules, using only minimal external
93 (off-robot) infrastructure.

94 The rest of this paper is structured as follows: Sec.(2) discusses prior work on
95 self-assembly and self-repair; Sec.(3) discusses the software, hardware and simulation
96 aspects of the Dynamic Self-repair project in further detail and provides further context
97 for the work within this paper. In Sec.(4) the self-assembly strategies to be tested are
98 discussed in detail and the implementations explained both from a general viewpoint
99 and with reference to their interaction with the hardware capabilities of the Omni-Pi-tent
100 modules used in this work, the new Quadruplet format for describing modular robotic
101 structures is introduced here; Sec.(5) explains the simulated self-assembly experimental
102 scenarios, setup and results. Sec.(6) explains the new self-repair methods developed,
103 with Sec.(7) describing the experimental scenarios and the results. Sec.(8) provides a
104 hardware demonstration of one of the new self-assembly strategies to verify its feasibility,
105 and discusses reality gap effects found in the transition between simulation and the real
106 world. This section also investigates the challenges involved in performing the new
107 self-repair strategies using real robotic hardware. Sec.(9) discusses the results and Sec.
108 (10) concludes the paper whilst outlining future work.

109 2. Previous Work

110 Self-assembly is a key ability for self-reconfigurable modular robots and can be
111 divided in to systems [16] which use modules which are self-propelled along deliberate
112 paths to perform assembly, and those which rely on stochastic means [17] to "grow"
113 the structure out at desired points. For modular robots in near-future use cases in
114 unstructured environments, self-mobile self-assembly from scattered initial positions in
115 to specific defined structures is of greatest relevance. The numbers of robots would be
116 relatively low so high scalability, such as in [18], is not the key concern here.

117 Whilst [19] had considered inter-robot docking with heterogeneous hardware and
118 controllers earlier, [20] provided the first demonstration of autonomous docking between
119 separated modular robots. These self-assembly methods could not form arbitrary defined
120 structures. Later work [21] developed a self-assembly method for non-modular Kilobots,
121 gradient and area based methods have also been developed in other contexts, for example
122 [22], [23], [24] or [25]. The inaccuracies and variability of these mean that such methods
123 cannot be useful for smaller numbers of more complex robots needing to form precise
124 structures focused around hardware features such as joints within particular modules.
125 Self-assembly studies have also [26] worked on recognising similarities between existing
126 module arrangements and desired structures, this has typically required inter-module
127 negotiation between large numbers of units before any action can begin, and research
128 has often focused on the computing time required for decision making rather than the
129 practicalities of physically performing self-assembly.

130 Other simulated self-assembly work [27] has developed planning methods to break
131 larger structures in to tiled sections for simpler assembly, but may be slow to implement
132 with physical robots due to a reliance on edge following procedures for painstaking long
133 distance navigation, likely to be a highly non-trivial task with limited real-world sensors.

134 Yim's SAE work [28] considers how to reassemble a modular robotic structure after
135 a force has separated it into discrete modules and scattered them, it is therefore a form
136 of self-repair as well as self-assembly. The work used a centrally planned method of
137 assembly, with robots scanning the environment to find the shortest routes to come
138 together again. The SWARMORPH [29] script allows distributed formation of a structure
139 according to defined rules, robots use LEDs to recruit for randomly wandering units,
140 this required a set of rules specified rather than an explicit desired shape. [30] showed a

141 graph theory based representation of multi-module configurations, this worked with
142 explicitly defined docking connection lists.

143 [31], [32] and [13] all developed self-assembly methods based on lists of docked
144 connections. In particular [13] worked by allowing robots within the structure to, where
145 required by a data structure representing the desired morphology, recruit others to them.
146 This strategy was adapted for Murray's self-repair work [10] and provides a conceptually
147 convenient starting point for the new self-assembly strategies described in this paper.

148 Self-repair with modular robots is considered one of the grand challenges of mod-
149 ular robotics [2], with the earliest detailed discussions in [14], [33] and [34]. Despite
150 the usefulness, and the age of the concept, papers on self-repair with modular robotics
151 are remarkably rare and some of them describe themselves as morphogenesis and
152 self-assembly. While it is often described as a desirable capability of modular robots, self-
153 repair has not been achieved in many scenarios or with many of the platforms. Much
154 of this work also focuses on general reconfiguration but is either: non-autonomous
155 and without the use of sensor feedback; or is highly abstracted, for example much of
156 the purely-in-simulation 3D lattice work, with aims more focused to micro-scale pro-
157 grammable matter type "robots" rather than near-future industrial macro-scale systems.
158 Self-repair has also taken place with robots which are not self-reconfigurable modular
159 systems, such as [19] or [35], they consider how repairs of robot internals can occur
160 in encounters between robots rather than self-repair in structures. Other studies have
161 considered the concept of failed robots being dragged away [36] but not the mechanics
162 of structure reconfiguration as this happens.

163 Some pure simulation papers have, alongside self-reconfiguration and self-assembly,
164 considered self-repair processes, for example [37], [38], [39] and [40]. All of these how-
165 ever consider self-repair as a means of replacing modules in very large structures which
166 have been swept away by applied forces, rather than considering practical methods for
167 removing modules which have been individually damaged or suffered internal failures,
168 *and* bringing in others to replace them.

169 Work in [33], [14] and [34] provided some of the earliest demonstrations of the
170 self-repair concept, using a mixture of hardware and simulations designed to reflect the
171 actions that available hardware could undertake. Unlike in [28] these studies performed
172 removal of a damaged robot and a restoration of the initial structure, via an intermediate
173 structural configuration, rather than a total rebuild. The importance of having robots
174 around the failed unit to detect failure is noted by [14] and the concept of using a lack of
175 communications from a module as evidence of its failure is introduced.

176 As part of SYMBRION [5], Murray [10] developed, largely in simulation, ways to
177 split up, remove failed units from, and then reform, a group of modular robots. Their
178 strategy let a multi-robot structure break into substructures when a module failed, the
179 failed module could then be moved away. Substructures compared "repair potential"
180 scores to decide which would disassemble and which would start the rebuild. The
181 method was unable to handle structures containing loops. Experiments showed that
182 for large initial structures "repair potential"-based repair was much quicker than total
183 breakup strategies. A correlation was also found between larger "repair potentials"
184 and quicker times for repair. [10] noted potential for improvement if multi-module
185 groups could dock as connected units, considering that by "precisely coordinating the
186 movement of a structure" improved self-repair strategies would be possible.

187 The MNS project [41] provided the most significant follow-up to [10]'s work, in-
188 cluding showing a form of self-repair which could dock groups of robots with other
189 groups of robots. The system could perform group locomotion in response to a local
190 stimulus [41]. One robot acted as a brain unit, stimuli were received by other robots
191 and fed along towards it. Actuator commands from the brain unit were fed to other
192 units, each of which acted locally to produce its part of the group motion ordered by the
193 brain. Similarly to [29], messages were addressed to child robots based on the angle at
194 which the child was docked to its parent, messages from child robots were identified by

195 which angle they were coming from. The system achieved “sensorimotor co-ordination
196 equivalent to that observed in monolithic robots” [41]. MNS also had the ability for
197 self-repair. Robots produced messages according to a “heartbeat” protocol, parent robots
198 sent heartbeats to their children and the children acknowledged beats back. If these
199 messages ceased, a robot would be recognised as faulty by its children and/or parents.
200 The substructures around it could disconnect, then reform. The work was limited by,
201 amongst other things, the non-modular robots used.

202 With a few exceptions, [10] [28] [41] there is a lack of work in the modular robotics
203 field focused on practical self-repair methods, with even less work considering self-repair
204 with the kind of individually mobile modules likely to be used in a near-future system.
205 The existing self-repair methods which require a collective task to cease before a repair
206 occurs can, however, provide a basis from which to develop further self-repair strategies.

207 3. Developing a Modular Robotic Platform for Dynamic Self-repair

208 Previous work in [12] provides the foundation for our larger vision to develop
209 methods that allow modular robots to self-repair and to self-assemble during continuous
210 motion. A compelling justification for performing these procedures while maintaining
211 motion is that it may offer a speed advantage as the group of robots can continue moving
212 as it happens. This allows the time taken for a mission to be completed to be reduced
213 as compared to self-assembly and self-repair strategies which require the group to stop
214 before assembling or repairing. The importance of speed comes into play in many
215 scenarios such as:

- 216 • In a nuclear environment scenario operators will want to reduce the heat and
217 radiation exposure on robots. The less time spent in the hot radioactive environment
218 during each monitoring mission the longer a lifespan the robots will have before
219 they succumb to effects such as neutron embrittlement.
- 220 • In a disaster zone scenario unstable debris could collapse and crush a robot at any
221 moment. Self-assembling and self-repairing robots would benefit from retaining
222 the ability to rapidly move out of the way of falling objects, lest the operators
223 find themselves having to rescue their robots before they can resume searching for
224 survivors.
- 225 • Robots working on critical infrastructure may be required to perform repairs to
226 the infrastructure while it is sustaining damage in real time. This requires the
227 robot to maintain some elements of its group action while assembling or repairing
228 itself. Failure to act quickly in these kind of situations could lead to extensive
229 and irreparable damage to infrastructure should a robot be slowed by its own
230 self-assembly or self-repair procedures.

231 The work involves both the development of modular robot control algorithms as
232 well as development of the Omni-Pi-tent hardware platform [7], see Fig.(1), on which to
233 test these algorithms. The design of the Omni-Pi-tent platform is inspired by estimates,
234 such as in [11], of what functionalities would be required for a generic modular robot
235 for infrastructure monitoring, planetary surface exploration or post-disaster search-and-
236 rescue applications.

237 Performing self-assembly and self-repair while in motion places unique demands
238 on the hardware platform, particularly in terms of mobility and docking sub-systems,
239 hence requiring a set of features not combined in any previous modular robot platform.
240 We now consider how the Omni-Pi-tent hardware platform meets these requirements.

241 3.1. Omni-Pi-tent

242 The Omni-Pi-tent modular robot platform [7] uniquely combines:

- 243 • Active genderless docking, such as previously used in [9] [6], this lets any port
244 connect with any other and ensures the possibility of single sided disconnection
245 if either side fails. Genderless docking vastly increases the variety of possible
246 configurations and reconfiguration methods.

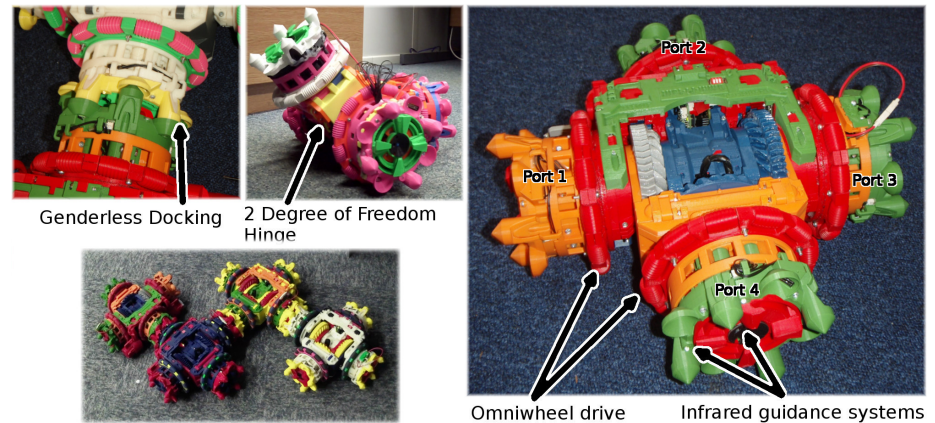


Figure 1. Photographs of an Omni-Pi-tent module.

- 247 • Omnidirectional locomotion, this allows for maintaining a compass orientation
248 decoupled from the driving direction, allowing for docking under a wider variety
249 of circumstances, and also increases the mobility of docked structures. [42] and [43]
250 were the only previous modular robots with omnidirectional drives.
- 251 • Self-contained sensor arrays to avoid reliance on external infrastructure, these limit
252 sensing to that locally available so scenarios more similar to real world deployment
253 of modular robots can be created. The robots have proximity and orientation sensors
254 as well as a 5KHz modulated IR system for docking guidance.
- 255 • Global and local communications using, respectively, Wi-Fi and line-of-sight 38KHz
256 IR LEDs. The Wi-Fi provides a global broadcast way in which to share data across
257 the whole swarm regardless of locations. The IR communication allows for local
258 communications which can make use of directionality and range to convey implicit
259 information beyond the data content of messages.

260 **Each module has 4 genderless docking ports arranged at the tips of a cross shape**
261 **and is designed with the necessary symmetry to dock in square grid arrangements.**
262 **Although not used in this work, it should also be noted that the hinging of the fourth**
263 **port (see Fig.(1)) of a module allows for rotations, for both pitch and roll, about the**
264 **module's centre, hence allowing the possibility of forming 3D cubic lattice structures.**

265 3.2. Simulating Omni-Pi-tent

266 While developing the Omni-Pi-tent hardware, and for testing algorithms in a more
267 debuggable environment, a simulation of the module hardware has been created using
268 the V-REP 3.5 [15] simulator. V-REP was chosen for its versatile options for defining
269 sensors and actuators and its inbuilt physics engine, enabling simulations to provide
270 reasonable approximations of how the real robots will behave. While the reality gap [44]
271 is present in any simulated work significant steps have been taken to reduce it:

- 272 • As detailed in [7] experiments using docking port hardware provided experimental
273 data on the analogue strengths of the docking guidance signal with both range and
274 angle from a recruiting port. An empirical polynomial model was fitted to this data
275 using R [45]. This model is called within the V-REP simulation to provide sensor
276 readings based on relative robot positions. Other measurements from real world
277 hardware tests were used to inform further simulation parameters.
- 278 • Simulated Omni-Pi-tent units run independent controllers. Each has access to
279 functions which can read sensor data and send actuator commands for that module,
280 much in the way that code running on the real robots does. The controllers run at 20
281 "ticks" per second in simulation, which matches the default rate at which controllers
282 on the real hardware run.
- 283 • Information sharing between the modules is handled using V-REP's "signal" and
284 "custom datablock" features. Line-of-sight (IR) messages are handled so they are

285 available to be read in later timesteps only by robots within the correct relative
 286 spatial regions to receive such a message in the real-world. Global (Wi-Fi) messages
 287 are passed to all modules and carry information which modules can act upon in
 288 later timesteps

289 With the simulation environment explained, strategies used by Omni-Pi-tent mod-
 290 ules for self-assembly can be now considered, which will then be used as a starting point
 291 for self-repair methods.

292 4. Self-assembly Strategies

293 The Omni-Pi-tent modular robot platform has a number of notable capabilities
 294 and features, these open up new possibilities when implementing self-assembly, even
 295 while sticking closely to prior strategies developed on previous hardware. A finite state
 296 machine for the initial self-assembly controller is outlined in Fig.(2) and explained in the
 297 next subsections, with comparisons made against [13], which inspired it and provides
 298 similar capabilities. **Pseudocode for the new self-assembly strategies is available in**
 299 **R.H.Peck's thesis [46] (etheses.whiterose.ac.uk/30288/ pp. 180–182).**

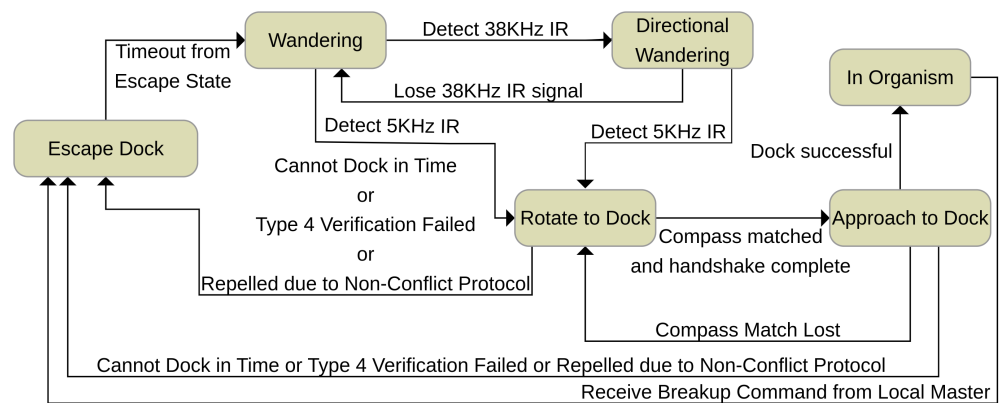


Figure 2. A finite state machine for the self-assembly controller. Robots start by default in the wandering state.

300 4.1. Wandering

301 Robots by default start in a “Wandering” state, as individual units performing
 302 obstacle avoidance and random wandering. Wandering robots have a Temporary ID
 303 value set to 0, this value can later be changed to reflect a module’s role in a structure. Seed
 304 modules transition out of the wandering state to enter the “In Organism” state (transition
 305 not shown in the FSM) upon making a decision, typically based on environmental
 306 information, to form a larger organism to complete a task which lone robots cannot.
 307 Modules becoming seeds take on a non-zero Temporary ID value and share, via Wi-Fi, a
 308 Recruitment List Quadruplet data structure defining the structure to be formed. Except
 309 in the case of a seed, modules only exit this “Wandering” state if they detect IR signals.
 310 Detecting 38KHz line of sight infrared messages causes robots to enter a “Directional
 311 Wandering” state which continues a wandering action but biased in the approximate
 312 direction from which the signal was received. These messaging signals are sent at the
 313 highest power level available and have the longest range of any IR signals used on the
 314 platform. The detection of slightly shorter ranged 5KHz analogue signal levels then
 315 triggers a transition of a robot from a “Wandering” or “Directional Wandering” state to
 316 begin the docking procedure by entering the “Rotate to Dock” state.

317 4.2. The Docking Procedure

318 On Omni-Pi-tent just two sensor systems are necessary to allow docking, a global
 319 angle reading supplied by a BNO 055 compass, and a system of infrared LEDs and photo-

320 transistors. For docking to occur a recruiting robot must supply data to an approaching
321 robot via local communications, such as the 38KHz IR. This data includes:

- 322 • A message type identifier
- 323 • Temporary and permanent ID numbers for the recruiting robot and in some cases
324 the approaching robot
- 325 • The global compass angle of the recruiting robot and details of any motion it is
326 currently performing
- 327 • Details of which docking ports are to be used by the recruiting and approaching
328 robots

329 For most communication types, the message does not contain an ID for the ap-
330 proaching robot but rather any robot within the area illuminated by the 38KHz message
331 will respond to such messages. A robot receiving the message matches its global orien-
332 tation to that of the recruiting robot, in many cases with offsets in multiples of 90° to
333 account for the choice of ports specified by the recruiting robot. This compass matching
334 behaviour is shown by the "Rotate to Dock" state in Fig.(2).

335 Once correctly rotated the approaching robot enters the "Approach to Dock" state
336 and drives in the direction of its recruited port towards the recruiting port, returning,
337 if it drifts out by more than a few degrees, to the "Rotate to Dock" state to correct its
338 compass rotation either by stopping and turning or by subtly changing the speeds on
339 each wheel so as to "add" together a turning motion and the driving vector. As it drives
340 in the "Approach to Dock" state it may find itself drifting away from the centre-line of
341 the illuminated 5KHz cone of light cast by an LED on the recruiting port, this is detected
342 by comparing the analogue IR strengths on phototransistors around the recruited port's
343 rim and the direction of driving is adjusted so as to bring the analogue values closer
344 together. Successful docking is identified on the physical robots by contact on a pair of
345 microswitches inside two of the spike accepting pits of the docking ports.

346 4.3. The "In Organism" State

347 Once connected to the port which recruited it, a robot can act as a sensor and
348 actuator slave to the robot it has connected to, referred to as its local master. This local
349 master can relay the docked robot's sensor readings towards the global master of the
350 organism, the seed robot, or to relay actuator commands from the global master down
351 to slave robots. Sec.(4.6) describes this messaging in greater detail. When a robot is
352 within the "In Organism" state it can also recruit further robots to it, to which it will then
353 become their local master. The concept for such recruitment is developed using Liu and
354 Winfield's work [13] as a foundation.

355 4.4. Defining Structures for Omni-Pi-tent

356 By omitting use of the [13]'s ordering array and making other alterations to their
357 SER strategy we find that it is possible to produce a new form of easily human readable
358 array data structure, the Quadruplet.

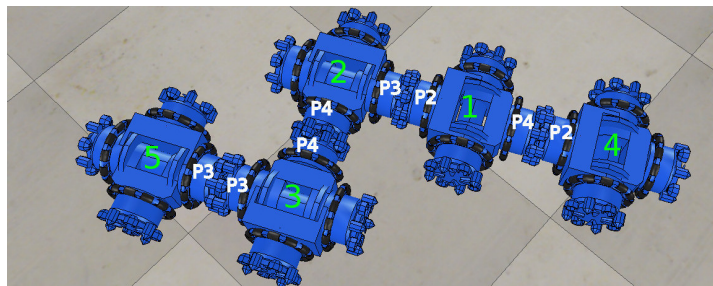
$$\begin{array}{c} \text{Liu and Winfield's Array Format} \\ \{\{X_1, Y_1\}, \{X_2, Y_2\}, \dots, \{X_n, Y_n\}\} \\ \text{Our Quadruplet Format} \\ \{\{A_1, B_1, C_1, D_1\}, \{A_2, B_2, C_2, D_2\}, \dots, \{A_n, B_n, C_n, D_n\}\} \end{array}$$

Figure 3. A comparison of the arrays used by Liu and Winfield [13]'s to define shapes, to the newly developed Quadruplet format. See main body text for explanation.

359 In [13]'s format, see Fig.(3), each pair of values defines the Temporary ID number
360 of a robot which is to perform a recruiting action, X , and the port it is to recruit on, Y .
361 An array of these pairs has as many pairs as there are docked connections within the
362 structure the array describes. [13] also needed a recruitment order list. Reading both
363 the pairs array and the order list was necessary to understand the structure's shape.

364 Recording the order of incoming robots was vital to making [13]’s SER strategy assign
 365 the correct Temporary IDs to new robots and hence vital to getting those new robots to
 366 then start the correct further recruitments.

367 In our new Quadruplet format, see Fig.(3), each Quadruplet of the Recruitment List
 368 contains: A, the temporary ID number of the robot which is to do recruiting; B, which
 369 port it is to recruit on; C, which port it wants an approaching robot to dock with; and D,
 370 what temporary ID the new robot should take once docked within the organism. C is
 371 used in the new system because Omni-Pi-tent is able to use any port to actively dock,
 372 hence any pair of ports can be recruit and recruiter. n Quadruplets are needed, one for
 373 each docked connection in the finished structure. Recruitment will occur on whichever
 374 ports it is required, at whatever times those ports are available to recruit robots to them.
 375 All the necessary information for describing the structure is contained in this single novel
 376 data structure, there is no requirement to refer to an additional list when interpreting the
 377 meaning. This data structure is shared across the global Wi-Fi communication system
 378 with updated versions transferred to all robots each time a module modifies its own
 379 local copy. Temporary ID numbers, used by both Liu and Winfield’s strategy and the
 380 improved strategy described in this paper, are only assigned to robots within organisms,
 381 free wandering robots do not have such a value specified until post-docking. Fig.(4)
 382 shows an example of how this new form of array translates to a structure.



$\{\{1,2,3,2\},\{2,4,4,3\},\{3,3,3,5\},\{1,4,2,4\}\}$

Figure 4. An example of how a Quadruplet array format converts to a robot structure. Robot Temporary ID numbers are shown in green, port numbers involved in docking are shown in white. The first Quadruplet states that the seed robot, with Temporary ID 1, is to recruit using its port 2 and call for port 3 of a robot to attach to it. This recruited robot is to identify itself with Temporary ID 2 once docked. Other Quadruplets specify other connections. These Quadruplets could be supplied in any order to define this structure.

383 The size of this Quadruplet data structure increases linearly with the number of
 384 robots involved in a structure, this is not a prohibitive requirement when considering
 385 macro-scale modular robots operating in groups of tens or even hundreds.

386 It should be noted here that forming structures containing loops is not attempted
 387 in this work due to the physical impossibility of a robot docking onto multiple ports
 388 arranged so as to form a concave space it must enter into.

389 4.5. Recruitment with Omni-Pi-tent

390 During each loop of any “In Organism” robot’s controller code, the robot checks
 391 how many Quadruplets are in the global Quadruplets array and checks what temporary
 392 ID it has, if there are no Quadruplets in the array the self-assembly is complete. If
 393 there are Quadruplets in the array then if the robot has a non-zero Temporary ID it
 394 searches through the array for Quadruplets where the A value, see Fig.(3), matches its
 395 Temporary ID. For any such Quadruplets which it finds it begins recruiting on the ports
 396 specified by the B value in the Quadruplet. A robot may find that there are multiple
 397 Quadruplets with its Temporary ID as the A value, in which case it will recruit on all the
 398 ports specified by the B values in those Quadruplets. When recruiting from each port
 399 the robot broadcasts from that port’s LED the local communication 38KHz IR message

400 detailing data such as compass orientations, speeds and port number requirements,
401 the port number requirements having been read directly from the Quadruplets. These
402 messages also include the Temporary ID number, from D in the Quadruplet, to be
403 taken by a recruited robot once it docks. The robot also uses the same LED for 5KHz
404 flashing. In simulation both the 5KHz cone and 38KHz message are represented as
405 being constantly emitted. On the physical robots sending both the 5KHz signal and the
406 38KHz message is not possible at the same time, however as the message is repeated
407 with an inter-message period at-least several times longer than the message length there
408 is sufficient time inbetween messages for 5KHz flashing to occur and for an approaching
409 robot to observe both the 5KHz analogue and 38KHz digital signals.

410 When a previously wandering robot docks to a port that robot will take on the
411 Temporary ID numbers which it had been informed of via the IR messages. Come
412 the next loop of their controllers robots which have just docked can consult the global
413 Quadruplets array to see whether they should start recruiting.

414 Robots which are recruiting check their port microswitches, if they have been
415 pressed then a recruit has docked to them, hence they edit their copy of the Quadruplet
416 array to delete any Quadruplet which identifies this specific docking connection. They
417 then share the edited version with the rest of the organism-forming and wandering
418 robots via global Wi-Fi. Over time the Quadruplets array gets emptied as docking
419 connections are made. When no Quadruplets are left the self-assembly is complete.
420 A second globally accessibly copy of the Recruitment List, known as the Structure
421 Recruitment List, is also maintained, this copy does not get edited when connections are
422 formed and instead contains at all times a list of all desired connections to be made. This
423 second list can be used as a guide when repairing a structure. In this way a complete
424 structure can be formed, and at any point in time all the robots in the structure are able
425 to recruit on all of the ports which require a robot to be attached.

426 For the most part, a specific module involved in self-assembly only requires knowl-
427 edge of those Quadruplets in the data structure which include its own ID number.
428 Therefore, whilst all modules receive updated copies of the Recruitment List over Wi-Fi
429 when it is updated, delays in these updates reaching modules will not typically matter
430 as those parts of the Recruitment List most crucial to a certain module will be those parts
431 which it updates for itself when immediate neighbours are docked or disconnected.

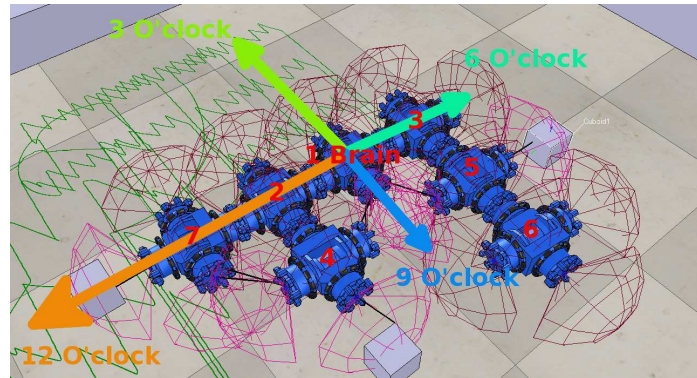
432 The strategy as described thus far is a novel development inspired by Liu and
433 Winfield's strategy and with similar capabilities, it is henceforth referred to as LW+. We
434 will now discuss features added to this base strategy so as to create two novel strategies
435 able to operate during motion, referred to as LW+MNS and MLR.

436 *4.6. Mergeable Nervous Systems for Omni-Pi-tent*

437 The Mergeable Nervous Systems concept [41] provides a method for modular robots
438 to exercise accurate control over the motion of an assembled organism. By combining
439 elements of it with LW+ it is possible to develop a strategy which can self-assemble
440 while moving.

441 The Mergeable Nervous Systems concept requires that sensor data can flow from
442 peripheral modules which detect surrounding stimuli, upward through robot to robot
443 links as far as the seed². Methods for this could either suffer from being non-scalable,
444 where every body robot tries to provide full sensor data to the brain and communication
445 becomes rapidly unreliable as the number of robots in the organism rises, or could tend
446 to loose large amounts of data via the compression required for communicating in a
447 scalable way. For the new implementation used in this work the sensor data which
448 needs transporting is proximity information, a method was chosen which was scalable
449 while still preserving useful features of the sensor data along the way. Each robot in
450 the structure checks uses knowledge about its location in the structure, derived from

² A robot in this position is also referred to as the global master and was described in [41] as the brain robot



Robot	Message Contents
Robot 4's	{255, 255, 255, 255, 255, 255, 255, 255, 255, 069, 255, 255}
Robot 7's	{255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 121}
Robot 2's (as seen)	{000, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255}
Robot 2's (including data from 4 and 7)	{000, 255, 255, 255, 255, 255, 255, 255, 255, 069, 255, 121}
Robot 6's	{255, 255, 255, 255, 255, 255, 255, 000, 000, 255, 255, 255}
Robot 5's (including data from 6)	{255, 255, 255, 255, 255, 255, 084, 000, 000, 255, 255, 255}
Robot 3's (including data from 5)	{255, 255, 255, 255, 000, 000, 084, 000, 000, 255, 255, 255}
Robot 1's (including data from 3 and 2)	{000, 255, 000, 255, 000, 000, 084, 000, 000, 069, 255, 121}

Figure 5. A visualisation of clock hand data flows. Temporary ID numbers are marked in red. Data structures start at 1 O'clock direction and run to 12 O'clock. 255 implies no known data in this direction, 0 implies no objects in range, numbers from 1 to 254 show objects at decreasing ranges. Note how, to avoid false positive detections, Robots 4 and 7 ignore the detections of each other in their own data structures.

451 the non-editable copy of the Recruitment List, to find what the angular position of that
 452 sensor is in terms of a clock face centred on the global master robot. The robot creates a
 453 12 byte array with each place corresponding to one of the hour positions on the clock face
 454 centred on the structure's highest level master. Sensor data is entered into any places
 455 in that array which represent angular regions of the clock face in which the robot has
 456 sensory information. Messages are passed up to the immediate master which uses a
 457 combination of its own sensor data and sensor data arrays from each of its slaves to fill
 458 in the 12 byte array which it relays up the hierarchy. Where a robot combining arrays
 459 has clock data from the same direction coming from multiple sources (multiple slaves
 460 or a slave and its own sensors) the closest value is placed in the array which will be
 461 passed on. This data flow continues until the seed receives a 12 byte array containing
 462 information about the closest obstacle in each direction, see Fig.(5). Recruitment List data
 463 structures let robots check whether any of their ports are docked or otherwise located
 464 such that parts of the same organism are within sensor range, data from these sensors is
 465 not relayed up the hierarchy so as not to swamp the brain robot with false detections
 466 where parts of the organism can see other modules.

467 Once the brain has processed sensor data it decides on the correct actuator response
 468 for the organism to make. The brain sets the relevant outputs from its actuators, usually
 469 the omniwheels but this could also include the 2DoF joint actuators in each module.
 470 It shares these via IR messages down the hierarchy to all slave robots immediately
 471 connected to it. Slave robots receiving such messages read their copies of the Recruitment
 472 List to find which of their ports is connected to which port on the master, then calculate
 473 the transform between the brain's reference frame and their own. The linear and angular
 474 velocities of the brain are calculated from its wheel speed information and used to
 475 identify an instantaneous centre of rotation about which the slave robot should move
 476 so as to follow the brain's motion. By calculating distances from its wheels to this
 477 instantaneous centre of rotation the necessary wheel velocities for the slave can be
 478 found. Slaves share these velocities via port to port IR messages to any further slaves

479 subservient to them. By these means the whole organism can rotate about a fixed centre
 480 of the master's command, drive linearly, or perform any combination of both actions.

481 If a problem should occur during self-assembly this implementation of Mergeable
 482 Nervous Systems also allows for port to port messages between robots signalling for
 483 them to undock from the structure, enabling a breakup of part or all of a structure. These
 484 messages cause transitions into the "Escape Dock" state.

485 4.7. Docking while in Motion

486 Unlike previous modular robot platforms Omni-Pi-tent is intended to be able to
 487 dock to moving robots. The ability of an omniwheeled robot to maintain orientation
 488 separately from direction of travel means this potentially difficult act becomes concep-
 489 tually simple. A moving recruiting robot, either the brain robot or any recruiting robot
 490 already in an organism, broadcasts its current wheel speeds as part of its 38KHz IR
 491 recruitment message. Robots responding to the recruitment message use the wheel speed
 492 and compass information contained in the 38KHz message to match motion so as to be
 493 stationary within the recruiting robot's reference frame. The motions of the approaching
 494 robot are added to this matched motion, allowing docking to proceed similarly to if the
 495 recruiting port was stationary, see Fig.(6).

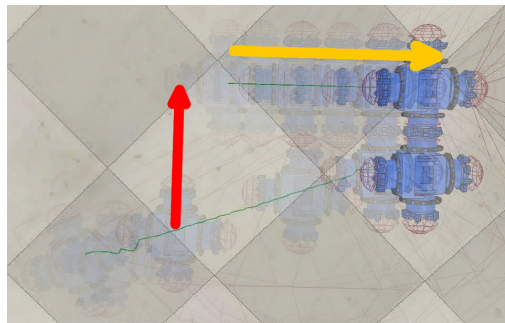


Figure 6. Docking of a robot to a moving seed, note how while in the global reference frame the recruited robot follows an angled path, from the reference frame of the recruiter it is simply approaching while keeping aligned to the cone's centre.

496 With the combining LW+ with Mergeable Nervous Systems results in a self-assembly
 497 strategy which, combined with the unique features of the Omni-Pi-tent hardware would
 498 appear to be one of the most capable and versatile self-assembly strategies yet devised.
 499 This strategy which combines features from Liu and Winfield's work with Mergeable
 500 Nervous Systems features is referred to in this paper as LW+MNS.

501 Enabling robots to gain a temporary ID number and recruit for others whilst they
 502 are still in the "Approach to Dock" state provides a possibility for a further novel strategy,
 503 that of Multi-Layered Recruitment (MLR). MLR should enable robots to recruit on
 504 multiple "concentric layers" outward from the seed robot, rather than recruiting being
 505 performed only by robots already docked. If a robot has recruits attach to it before it has
 506 itself docked to its local master then MNS allows it to maintain control of its substructure
 507 to move and dock. It was considered that MLR may enable faster self-assembly than
 508 LW+MNS by parallelising a robot's own recruitment with its recruitment of others.

509 5. Self-assembly Experiments

510 To demonstrate the effectiveness of self-assembly during motion it was necessary
 511 to devise a scenario in which strategies are compared not simply on the time to form an
 512 organism's structure but in which the completion of a concurrent locomotion task also
 513 has an important effect.

514 Consider a scenario where a swarm of modular robots, perhaps exploring a plane-
 515 tary surface, monitoring hard to reach infrastructure or penetrating the rubble beneath
 516 a collapsed building, have entered a space. At the far end of this space is some item

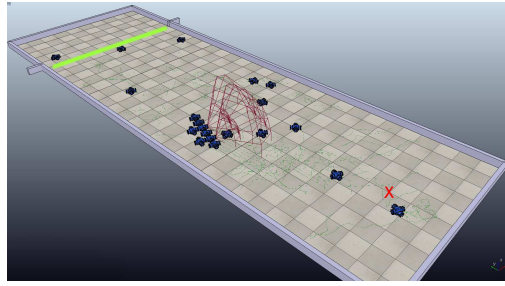







Figure 7. A screenshot from a simulation using the MLR strategy, note the scattering of robots throughout the space as the forming organism drives towards the finishing line, marked in green. The location where the seed initially starts is marked with a cross. The larger red wireframe shapes represent IR communication ranges on the highest power setting, the smaller ones show docking guidance signal ranges.

517 of interest, the handling of which requires the independent modules to dock together
 518 into some defined morphology. A robot near the entrance end of this space detects the
 519 item of interest at the far end and makes itself the seed robot, recruiting others to form
 520 the multi-robot structure, such self-promotion to seed robot due to circumstances has
 521 strong precedent from earlier research [5] [47] [48]. For the purpose of these experiments
 522 the robot which was to become the seed was instructed, immediately as the simulation
 523 began, of the correct structure to be formed.

Table 1. Table of structures formed, the seed robot in each structure being highlighted in pink. Comma separated numbers in columns indicate multiple values were tested.

Name	Image	Widths (m)	Lengths (m)	Robots in Struct.	Robots in Scene	Num. of Layers
S1		3,5	10	10	20	3
S2		3,5	10	5	20	2
S3		10	10	15	30	4
S4		3,5,10	10	10	20,30	5
S5		3,5	10,20	10	20	4

524 When using the strategy referred to as LW+ the robots cannot form a structure
 525 while moving so must assemble at the entrance end of the space before driving towards
 526 the item of interest at the opposite end. The LW+MNS and MLR strategies both have
 527 the ability to recruit and dock robots to them while in motion, hence strategies allow
 528 the structures to form as the robots drive along this “corridor”, see Fig.(7). For these
 529 experiments if robots using the LW+MNS or MLR strategies reach the end before forming
 530 the full structure they return back and forth through the space until the organism is
 531 complete at which point it heads for the item of interest again. In all scenarios the
 532 mission time is counted from the start of recruitment until the structure is both formed
 533 and is positioned at the far end where the item of interest was detected, in this way both
 534 the speed of assembly and the speed of locomotion are accounted for.

535 Using V-REP for simulations, tests were performed using five different structures.
 536 The number of robots present was varied for some scenarios, as were the length and
 537 width of the space, see Table.(1).

538 In each simulation scenario 40 runs of each of the three strategies were attempted
 539 with randomised starting positions and compass orientations for robots scattered through-
 540 out the space. The seed always began at a starting location, see Fig.(7), but had a ran-
 541 domised orientation. Each scenario was given a maximum simulated time after which
 542 the simulation would be ended if it had not succeeded in meeting both the assembly
 543 and locomotion goals by then, if many runs within a scenario timed out then it was
 544 re-run with a longer maximum allowable time specified. Maximum times ranged from
 545 15 minutes to an hour. As not all of the 40 runs completed within the time limit in all
 546 scenarios, some scenarios were analysed using the lower numbers of replicates which
 547 had managed to finish and write to files.

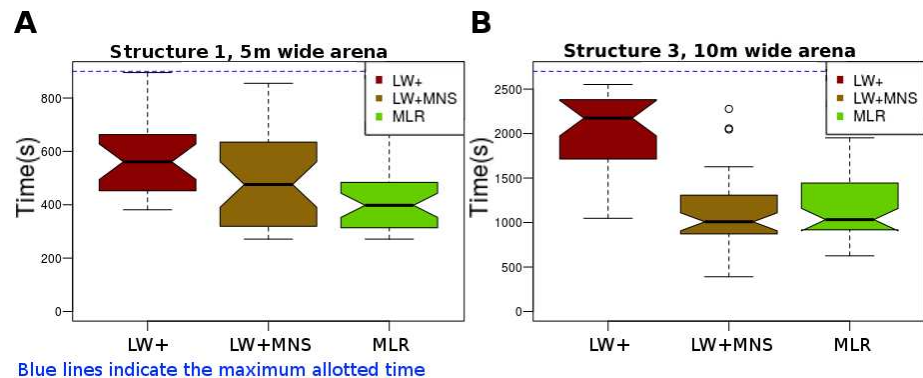
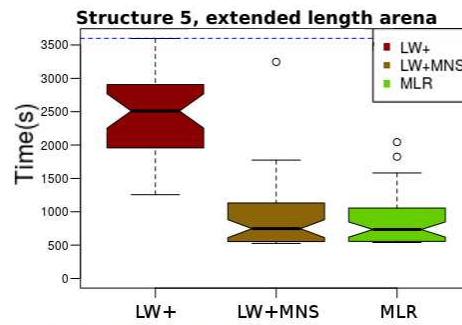


Figure 8. **A** Structure 1, run in a corridor of 5 metres width, with a null hypothesis that the different strategies would have the same performance. Amongst those simulations which did not time out LW+MNS and MLR have statistically significantly superior performance to LW+, p-values of 0.062 and 5.6e-3. Compared to the simulations in the 3m wide space all strategies see an improvement in performance, likely due to the unrecruited robots during later stages of assembly being able to more easily pass around the structure to reach whichever parts of it are still recruiting. The LW+, LW+MNS and MLR strategies had, respectively, time-out rates of 29%, 13% and 3%. **B** Structure 3, run in a corridor of 10 metres width and using a population of 30 robots. Earlier runs performed with all strategies for structure 3 in a 3 or 5 metre corridor mostly timed out, hence the cut-off time was increased to 45 minutes and the scenario re-run with a wider corridor and increased robot population. With these adjustments to the scenario, to ensure that a high proportion of simulations managed to complete within the time limit, a strong advantage can once again be seen for the LW+MNS and MLR strategies over LW+. Consideration across S1, S2 and S3 structures shows that the superiority of dynamic self-assembly strategies appears to be maintained regardless of structure size. p-values of 3.6e-10 and 2.0e-9 for LW+MNS and MLR, respectively, against LW+ show that “in motion” strategies give better performance in large structures and with higher densities of robots in the arena.

548 5.1. Self-assembly Results

549 Completion times for the runs were recorded, as were the times at which finish
 550 line crossings and dockings involved in the self-assembly procedure occurred. Some
 551 timeouts occurred in most of the scenarios, where timeouts did occur the organisms
 552 had usually recruited most of the desired robots with only one or two remaining to be
 553 recruited, however attempts to recruit these final modules had typically been ongoing
 554 for some time, suggesting that timeouts usually occurred because the reduced number
 555 of free robots still wandering in the arena were in positions such that they rarely came
 556 within range of those ports which were still recruiting. Analysis was performed using
 557 R [45], the completion time distributions of those runs which did not timeout were
 558 subjected to Mann-Whitney tests to find p-values and Vargha-Delaney A-tests for effect
 559 size calculations. The statistical tests across all experimental scenarios are summarised



Blue lines indicate the maximum allotted time

Figure 9. Structure 5 had good performance in the 3m and 5m wide 10m long arenas for both of the dynamic self-assembly strategies, their superiority over the LW+ strategy being more pronounced than for the Structure 1 scenarios. The extension of the corridor length to 20m in this scenario clearly gives more benefit to the strategies which can assemble as they move, with A-test measures rising to 0.96 and 0.98 for the LW+MNS and MLR strategies, respectively, over LW+. An aspect to this success may be that as the space increases in length the density of wandering robots decreases and therefore it becomes more difficult for a robot staying stationary with the LW+ strategy to have anything better than a very slow rate of recruits wandering close enough to it.

Table 2. Vargha-Delaney A-test scores comparing the various strategies. In each column the strategy named on the left of the *vs* is compared to the strategy named on the right, values above 0.5 indicate slower performance by the strategy on the left, scores below 0.5 indicate the right hand strategy was worse. Results for scenarios in which statistical significance p-values were below the 5% threshold are marked in green.

Scenario	LW+ vs LW+MNS	LW+ vs MLR	LW+MNS vs MLR
S1 R20 W3	0.81	0.85	0.45
S1 R20 W5	0.64	0.76	0.58
S2 R20 W3	0.94	0.96	0.27
S2 R20 W5	0.89	0.93	0.27
S3 R30 W10	0.94	0.92	0.41
S4 R20 W3	0.73	0.74	0.61
S4 R20 W5	0.97	0.94	0.48
S4 R30 W10	0.75	0.63	0.35
S4 R20 W5 Seed Changed	Inconclusive	Inconclusive	Inconclusive
S5 R20 W3	0.91	0.91	0.41
S5 R20 W5	0.79	0.82	0.50
S5 R20 W5 L20	0.96	0.98	0.50
S5 R20 W5 Seed Changed	0.72	0.70	0.51

560 in Table (2) while several scenarios of particular interest are discussed with the graphs in
561 Figs.(8A), (8B) and (9).

562 6. Self-repair Strategies

563 The concept underlying self-repair is based around detecting a failed module and
564 effecting its removal, then replacing it and reforming around the replacement. Any
565 failed module will either have modules lower in the hierarchy than it (local slaves) or
566 higher in the hierarchy than it (a local master), or very often both. A module without
567 a local master will be the master of the structure it is in, recovering from these failures
568 is was handled by reprocessing the Recruitment List to assign a new robot to become
569 master of the structure. The failure case on which this work is mostly focused is of a
570 failed module with both a master and slave(s) attached.

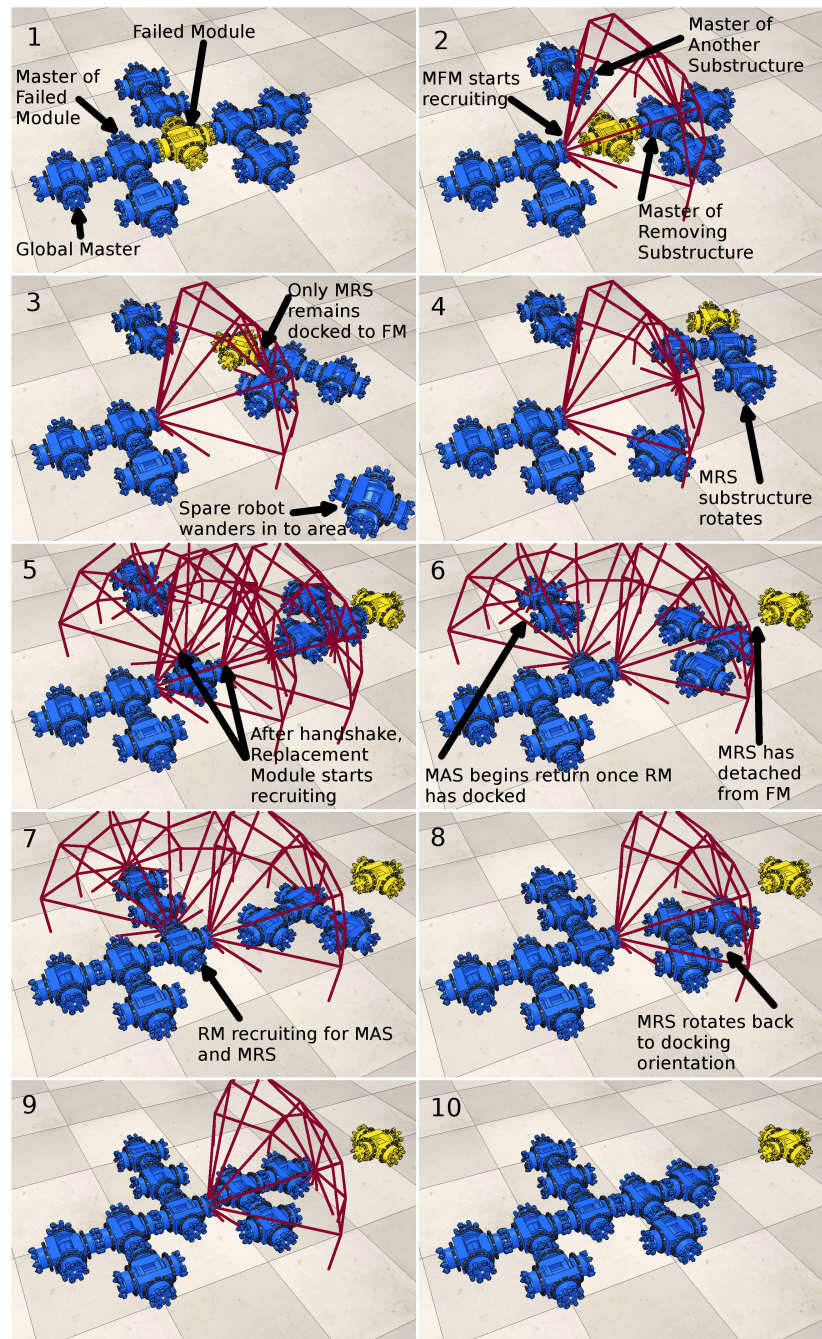


Figure 10. The concept of self-repair. Note how only the MFM, MRS, MAS and RM are involved in self-repair, robots not immediately neighbouring the failed module remain oblivious to the self-repair procedure and continue to act normally as slaves and/or masters.

571 The process of repair can be split in to those activities which must be performed by
572 the immediate local slaves of the failed module, and those which must be performed
573 by the immediate local master. The slave side of the repair process is responsible for
574 removing the failed module, and the master side for the recruitment of a replacement,
575 then the slave side continues in the repair procedure until it has re-docked to the
576 replacement. This parallelises aspects of the repair procedure, enabling re-recruitment to
577 occur without needing to wait for the disposal of the failed module. Fig.(10) shows the
578 key stages of the self-repair procedure within a static scenario.

579 During the moment at which a self-repair operation begins the modules neighbour-
580 ing a failed module take one of the following roles:

- 581 • **Master to the Failed Module (MFM)**, The robot which is specified in the Quadru-
582 plet list as the failed module's immediate master handles the re-recruitment side of
583 the self-repair procedure.
- 584 • **Master of the Removing Substructure (MRS)**, One of the failed module's slaves
585 will be promoted to act as the MRS. The MRS will control the substructure responsi-
586 ble for dragging away the failed module, will safely dispose of the failed module,
587 and will then guide itself back to the replacement module such that it, and its
588 attached substructure, can reconnect to the main structure.
- 589 • **Master of Another Substructure (MAS)**, Robots which were slaves to the failed
590 module and have slaves of their own also form substructures. These substructures
591 retreat and lurk at distance from the main structure until the failed module is
592 replaced before returning to rebuild it. The use of MAS substructures means
593 multiple substructures can be preserved.
- 594 • **Lone Module (LM)**, Modules which served as slaves to the failed module and
595 which have no slaves themselves enter the "Escape Dock" state, detach from the
596 structure and become "Wandering" modules.

597 During self-repair a further role is also used:

- 598 • **Replacement Module (RM)**, The RM starts as a free wandering module. However
599 once it completes the IR handshaking procedure involved in the transition from "Ro-
600 tate to Dock" to "Approach to Dock" it behaves differently to an ordinary recruited
601 module. For any connections which the Recruitment List requires it to recruit for, in-
602 stead of sending a general IR recruitment message it sends a specialised addressed
603 message only readable by the MRS and MAS modules.

604 Roles are assigned using Recruitment Lists as they are when a failure occurs, not
605 just the Structure Recruitment List of a completed organism. As the Recruitment List
606 is shared during self-assembly, and updated copies shared each time a robot docks or
607 undocks, there is no requirement for robots to communicate descriptions of the structure
608 to each other during self-repair. Each robot assigns itself the appropriate role without
609 requiring any further communication.

610 The use of genderless active docking on all ports and the use of an omnidirectional
611 drive mean a much wider array of structures can be handled, and connections and
612 breaks can occur as determined solely by the requirements of the hierarchy. Dynamic
613 Self-repair is achieved by applying velocity transforms to this repair procedure, much as
614 done for self-assembly. The use of mobile substructures within this self-repair procedure
615 is a novel development, made possible by Omni-Pi-tent's omnidirectional drive and
616 MNS [41] co-ordinated control.

617 Fig.(11) shows the finite state machine for MLR self-assembly modified to include
618 extra states required to enable self-repair. The new states added include those which
619 a module enters upon detecting a neighbouring module has failed, a series of states
620 involved in the repair procedure, and an extra state within the docking routines which
621 replacement robots enter upon docking. **Pseudocode for the new self-repair strategies**
622 **is available in R.H.Peck's thesis [46] ([etheses.whiterose.ac.uk/30288/](https://theses.whiterose.ac.uk/30288/) pp. 222–223, 253–**
623 **256).**

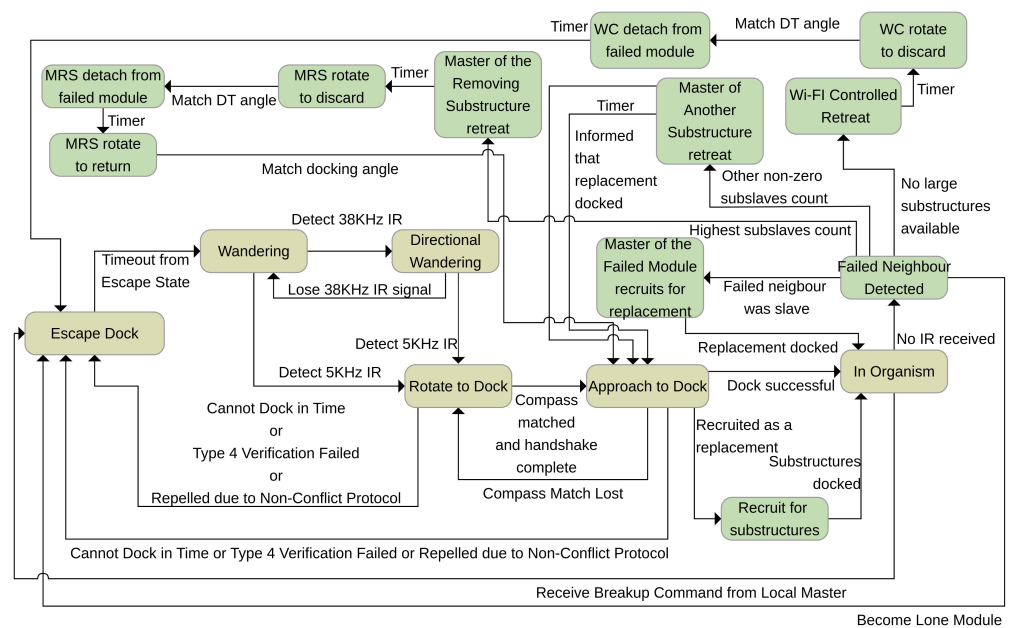


Figure 11. A simplified diagram of the FSM underlying the self-repair controller, states not present in the self-assembly controller are highlighted in green.

6.1. Detecting Failed Modules

624 Within a structure IR port-to-port links transfer MNS data constantly. As failed
 625 modules will either be turned off as a direct effect of the damage they receive, or shut
 626 themselves down upon endogenous detection of an internal fault, a failed module
 627 can be detected by a prolonged period without outward IR messages from its ports.
 628 Robots enter the “Failed Neighbour Detected” state if they see a neighbouring robot
 629 not communicating and process Recruitment Lists to decide how to react. **A robot is
 630 identified as failed after a period of 6 seconds in which it does not output any port to
 631 port communications, this time is short compared to the experimental run times, yet is
 632 long enough to minimise the risk of a functioning module being falsely diagnosed as
 633 failed due to temporary communications interruption to the infrared links.**
 634

6.2. Processing Recruitment Lists

636 Removing a failed module requires collective action by modules to provide the
 637 necessary forces. Decisions in the “Failed Neighbour Detected” state must be reached in
 638 such a way as to ensure that there will be one, and only one, substructure attached to the
 639 failed module serving as the Removing Substructure. Decisions must be reached without
 640 relying on any communication passed through the failed module, ideally needing no
 641 communication at all between the neighbours of the failed module.

642 By selecting the largest substructure to serve as the Removal Substructure it is
 643 ensured that in any scenario where the failed module has a slave attached which in
 644 turn has slaves the system will be able to use this substructure of ≥ 2 modules as
 645 the Removal Substructure. When the decision making process is complete modules
 646 transition to the “Master of Another Substructure retreat”, “Master of The Removal
 647 Substructure retreat”, “Wi-Fi Controlled Retreat”, “Master of the Failed Module recruits
 648 for replacement” or “Escape Dock” state.

6.3. Removing the Failed Module

650 The MRS module retreats for 20 seconds acting as master of its substructure and
 651 using MNS principles to prevent the substructure snagging on obstacles, it drives in
 652 such a direction as to pull the failed module directly away from the MFM and maintains

653 its compass orientation throughout. Next the MRS module enters the "MRS rotate to
654 discard" state to turn until it is facing the failed module away from the initial retreat
655 direction. The angle at which to dump the failed module, and the compass direction
656 along which to perform the later stages of retreat away from the main substructure are
657 calculated from the long term averaged motion of the master side structure, as read from
658 the MFM or RM's wheel speed messages relayed over Wi-Fi. Once the failed module is
659 released the MRS module begins its return to dock to the RM.

660 *6.4. Mergeable Nervous Systems over Wi-Fi*

661 If a failed module has slaves attached but none of these slaves have subslaves then
662 multiple single modules attached to different ports on the failed module must cooperate
663 without having a direct physical connection. MNS actions can therefore be performed
664 using Wi-Fi communication links in these circumstances. Temporary ID numbers, unique
665 within an organism, are used to select which slave of the failed module will act as MNS
666 master and which will become slaves to it. Unlike port-to-port IR communications Wi-Fi
667 communication also requires the permanent ID numbers of modules by which they will
668 be addressed, modules therefore send broadcast Wi-Fi messages to all Wi-Fi addresses
669 which act as requests for robots with the correct Temporary ID to respond with their
670 Permanent ID. With this complete MNS control can proceed ordinarily, except that the
671 instantaneous centre of rotation calculation is modified to use to relative positions of
672 modules separated by a failed module located inbetween them. The master of a Wi-Fi
673 controlled retreat substructure uses the same behaviours as an MRS module.

674 Once the failed module is discarded the Wi-Fi controlled retreat structure breaks up
675 in to Lone Modules, each in the "Wandering" state. As the this structure does not involve
676 any connections between multiple operational robots there is no point in attempting to
677 recover it in the way that multi-robot substructures are re-recruited.

678 *6.5. Recruiting a Replacement*

679 We now consider the MFM Module's role throughout the self-repair process. Upon
680 detaching from the failed module it begins recruiting for a replacement module by
681 entering the "Master of the Failed Module recruits for replacement" state. In this state
682 initial recruitment is identical to that used during self-assembly, however once the
683 recruited module, which will become the RM, has rotated to dock an alternative form of
684 recruitment is used at the time when the RM is in the "Approach to Dock" state.

685 Receipt of this specialised recruitment message type triggers the recruited robot
686 to begin another new type of recruitment, Type 10, on its docking ports. Type 10
687 recruitment messages are emitted by the Replacement Module only on ports which
688 require still existing multi-module structures to connect to them, and are readable only
689 by MAS and MRS modules. The special roles performed by the MFM module cease to
690 operate as soon as it is no longer recruiting on any ports, the RM module then takes over
691 these tasks.

692 *6.6. Guiding the Substructures*

693 The features so far discussed are useful to dynamic self-repair, and many are newly
694 implemented in the field of modular robotic self-repair, but it is the ability to operate
695 while group motion is maintained which crucially distinguishes Dynamic Self-repair
696 from earlier strategies.

697 Upon detaching from the failed module the MFM begins transmitting a combination
698 of wheel speed and compass data over Wi-Fi broadcast. These messages are read by the
699 MAS and MRS modules and used to set reference frames from which their retreat and
700 return motions are structured. Robots acting as MAS and MRS modules record the sum
701 of the wheel speeds received over Wi-Fi and record the sums of their own motions to
702 perform odometry calculations and calculate which driving direction will be necessary

703 to return them to a region in which they will find the recruitment IR cone of the relevant
704 docking port on the RM module.

705 Even in the idealised environment of simulation this odometry data gives relatively
706 poor readings with returning modules often overshooting or undershooting the point
707 at which they cross the trajectory of the main structure, hence missing the recruitment
708 cones. Hence the IR line of sight messaging is also used to guide MRS and MAS
709 substructures back. The Replacement Module within the main structure commands
710 all undocked ports on the structure to, for a fraction of a second every few seconds,
711 emit messages at full power. These messages identify which module, by Temporary
712 ID, within the substructure is emitting the message. Any robot in an MAS or MRS
713 controlled substructure may receive such messages and forward them on to the MRS
714 of MAS module of its substructure. Using a combination of compass readings, for the
715 emitting and receiving robots, and of Structure Recruitment List derived knowledge of
716 distances between robots with a given Temporary ID and the location of the RM within
717 the main structure, an MRS or MAS robot can switch to a behaviour in which it attempts
718 to enter the recruiting cone by "orbiting" around the main substructure in the direction,
719 calculated from the Structure Recruitment List, which allows it to travel the shorter
720 distance to reach the recruiting cone. Once within range substructures manoeuvre under
721 MNS control to dock to the RM and return the organism to its original form.

722 *6.7. Dynamic Master Switching*

723 A further version of the self-repair controller was created with additional features
724 able to change the location of the master robot. Unlike the other strategies described
725 this one can replace a failed global master, exhibiting a form of neuroplasticity. In a
726 Quadruplet list each docked connection is specified in terms of which robot will be
727 master within that connection, which will be the slave, and which port each will use.
728 Whilst this Quadruplet array usually acts as a form of directed graph, it also contains
729 all the information necessary to plot out alternative hierarchies of mastery able to
730 represent the same structure. It is therefore possible to remap the location of the master
731 within the organism, dynamically switching it between different robots while preserving
732 connections and Temporary ID numbers.

733 Remapping occurs immediately after a failure is detected in a neighbouring module,
734 and as described with the earlier self-repair process, is handled independently by each
735 module neighbouring the failed module, with the expectation that all will, as they have
736 the same understanding of the structure and the failed module's position within it, reach
737 common conclusions on how to proceed with repair. The process takes a copy of the
738 current Structure Recruitment List, which represents the structural plan regardless of the
739 current physical state, and the Recruitment List, modified in real-time to reflect docking
740 and undocking events and stores them in other variables. The Structure Recruitment List
741 is then remapped to be centred, temporarily, on the failed robot. The Recruitment List is
742 updated by remapping as necessary to match the new Structure Recruitment List. With
743 this temporary remapping in place all other modules in the organism can be treated as
744 slaves to the failed module, and sub-slave counting methods can compare the size of the
745 structures connected to each of the failed module's ports. Decisions can then be made on
746 whether to transfer global mastery away from the current global master, and in to part
747 of the largest structure attached to the failed module, or revert to the copies taken of the
748 unedited Quadruplet arrays. Such a transfer of global mastery is ofcourse essential if the
749 global master is the failed module. A key point to note about the algorithms presented
750 here is that, when used correctly, they cannot convert a physically feasible structure in to
751 an impossible one during the process of remapping it.

752 After the remapping process is complete modified Structure Recruitment Lists and
753 Recruitment Lists are shared over Wi-Fi in a specialised "atomic" operation. This is
754 necessary so that other modules within the organism, aside from those directly connected
755 to the failed module which calculated the remapping for themselves, have the correct

756 information necessary to supply Mergeable Nervous Systems sensor data in the reference
757 frame of their new master. An "atomic" operation is used to prevent chaos which could
758 ensue if modules received updates to the Structure Recruitment List and Recruitment
759 List at different times and spent the intervening time with an updated copy of one but
760 an obsolete copy of the other. As the remapping simply changes the order of terms
761 inside specific Quadruplets within the Structure Recruitment List and Recruitment List
762 arrays no robot will have experienced any disruptive change such as a modification of
763 Temporary ID value, and no undocking or redocking is required for the master switched
764 structure to resume MNS activity.

765 As well as being able to recover from a failure of the global master the Dynamic
766 Master Switching strategy also provides an opportunity to allocate roles differently to
767 the substructures around the failed module. Unlike with the earlier DSR strategy, if the
768 largest group of modules connected to a failed module is a slave of the failed module
769 the largest substructure can be placed in-charge of the re-recruitment part of self-repair
770 and can contain the MFM module. Mergeable Nervous Systems over Wi-Fi methods
771 can be prioritised where possible to allow the disposal of failed modules to be handled
772 by single modules which can be replaced individually, decreasing the need for large
773 substructures to be guided back to dock after retreating and turning to dispose of a failed
774 module. Where possible, if sufficient smaller substructures exist, larger substructures
775 can be prioritised for MAS roles rather than MRS, meaning they will typically spend
776 less time wandering away from the main structure and should be easier to guide back
777 to dock once a Replacement Module is in place. However if there are not sufficient
778 individual modules connected to a failed module for a Wi-Fi controlled retreat, and no
779 smaller substructures available to act as MRS either, then the Dynamic Self-repair with
780 Dynamic Master Switching strategy will allocate any available substructure to play the
781 vital role of removing the failed module.

782 7. Self-repair Experiments

783 Five forms of self-repair were compared. Dynamic Self-repair (DSR), Static Self-
784 repair (SSR), naive Dynamic Self-repair (nDSR), naive Static Self-repair (nSSR) and
785 Dynamic Self-repair with Dynamic Master Switching (DMS). Static Self-repair provides
786 a close analogue to the strategies developed by Murray [10], acting like DSR but pausing
787 the group motion at the moment of failure and remaining static until the structure is
788 repaired. The two "naive" strategies are based on Murray's full break-up strategy, which
789 they labelled "naive". The Omni-Pi-tent platform makes a complete breakup utterly
790 superfluous, so in nDSR and nSSR a complete breakup is performed of everything *below*
791 the failed module in the Recruitment List hierarchy, recursively transitioning all slaves
792 and subslaves of the failed module in to the "Wandering" state via "Escape Dock", while
793 leaving other sections of the structure unchanged. On the master side naive self-repair
794 involves only an undocking from the failed module and a brief retreat away from it so
795 as to make space for replacements to be recruited, this re-recruitment is done exactly
796 like ordinary self-assembly. nSSR and nDSR differ in that in nDSR Omni-Pi-tent's ability
797 to dock during motion means that the main structure can keep driving whilst this
798 recruitment of a replacement and of replacement subslaves for it takes place.




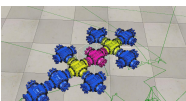
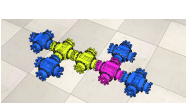
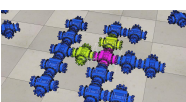
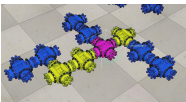
799 Example videos of all five strategies are available in the supplementary material
800 (S1, S2, S3, S4, S5).

801 In the experiments performed to compare DMS, DSR, nDSR, SSR and nSSR self-
802 repair strategies a structure was initially formed at a starting point by self-assembly
803 with the seed not moving. This initial self-assembly was necessary only because of the
804 nature of the V-REP simulation and the difficulties V-REP would cause if one wished to
805 initiate a simulation with modules already in a docked state, the time taken here did not
806 count towards the measurements used in the analysis. Once self-assembly was complete
807 the structure began a drive towards the opposite end of a long arena, and the timer
808 clock began. After starting motion a failure was injected in to a selected module within

809 the structure and the failed module's port-to-port communication ceased. The task
 810 for the other modules in the structure was to identify this failure and self-repair, with
 811 static strategies stopping group motion to perform the repair and dynamic strategies
 812 continuing along the arena. The finishing time for each run was recorded as the moment
 813 when a completely repaired structure crossed a finishing line at the far end of the arena.
 814 In all scenarios 30 robots were present in the scene at the start, an arena of 7 m width
 815 was used with the finish line placed 10 m away from the starting point, and 20 minutes
 816 of simulated time were given as a maximum time after which a run was considered
 817 timed-out.

818 This scenario was run for 7 different structures, see Table.(3). For each structure
 819 runs were performed involving the failure of a variety of different modules, as specified
 820 by Temporary ID number, within it. 100 runs of each of the nSSR, SSR, nDSR DSR, and
 821 DMS strategies were performed for each combination of structure and failed robot.

Table 3. Structures used in the self-repair experiments, each structure is named with a Recruitment List and an image displayed. The seed robot (pink) and those which were injected with faults (yellow) are highlighted. 10B, 12A and Rand are sourced from [10], the rest are derived from the self-assembly experiments with S2 enlarged and modified to make the repair task more interesting.

Name	Image	Temp. IDs of fault injected modules	Recruitment List
10B		5,7	{ {1,4,2,2},{1,3,3,3},{3,4,2,4}, {3,1,1,5},{5,4,2,6},{5,3,3,7}, {7,4,2,8},{7,1,1,9},{9,4,2,10} }
12A		2,4	{ {1,4,2,2},{1,1,4,6},{1,3,4,5}, {5,2,4,8},{6,2,4,7},{2,4,4,3}, {3,2,4,4},{4,1,4,11},{4,3,4,10}, {11,2,4,12},{10,2,4,9} }
Rand		6,7,9	{ {1,2,2,2},{1,3,2,3},{1,4,2,4}, {4,4,3,5},{1,1,2,6},{6,4,2,7}, {7,1,4,8},{7,3,2,9},{9,4,3,10}, {10,1,1,11},{11,3,3,12} }
S1		2,5	{ {1,1,3,5},{1,3,1,2},{2,4,4,9}, {2,2,4,10},{2,3,2,3},{3,4,4,4}, {5,4,2,8},{5,1,2,6},{5,2,2,7} }
S2		4,5	{ {1,1,1,4},{1,2,2,2},{1,3,2,3}, {4,3,2,5},{5,3,2,6},{5,4,4,7} }
S3		2,10	{ {1,1,4,3},{1,2,4,10},{1,3,4,2}, {1,4,4,4},{4,2,2,5},{2,2,4,7}, {3,2,4,6},{7,3,3,8},{6,1,1,9}, {10,2,4,11},{11,1,2,12},{12,4,4,14}, {11,3,2,13},{13,4,4,15} }
S5		4,7,8	{ {1,1,4,4},{1,3,4,7},{1,4,1,2}, {2,3,3,3},{4,3,4,5},{5,2,4,6}, {7,2,4,8},{8,3,3,9},{9,4,1,10} }

822 7.1. Self-repair Results

823 Table (4) shows proportions of repair operations which completed within the time
 824 limit while Table (5) summarises the A-test scores when comparing task completion
 825 times among the self-repair runs. Two example scenarios are shown with the graphs in
 826 Figs.(12) and (13).

Table 4. Proportions of runs completed without timing out for the self-repair strategies under comparison. Yellow cells mark situations where, for a given structure and failed robot, a particular strategy achieved $\geq 60\%$ success, green cells indicate completion rates $\geq 80\%$. Averages across all structures and failures tested give: nSSR 37%, nDSR 90%, SSR 72%, DSR 68% and DMS 76%. This indicates that both the motion involved in Dynamic strategies and the ability to self-repair both provide reliability advantages over static self-assembly based nSSR. SSR's higher proportion of completed runs within the time limit suggests that DSR's speed advantage is traded against a reduced reliability. As well as a higher averaged completion percentage than the standard DSR strategy, DSR with DMS manages a completion rate above 80% in 43% of scenarios to DSR's 25%.

Structure	ID of Failed Module	nSSR	nDSR	SSR	DSR	DMS
10B	5	3%	95%	45%	72%	81%
10B	7	15%	99%	58%	72%	87%
12A	2	15%	79%	81%	62%	72%
12A	4	78%	99%	56%	61%	54%
Rand	6	15%	77%	69%	62%	52%
Rand	7	40%	71%	61%	62%	51%
Rand	9	12%	75%	62%	60%	64%
S1	2	53%	97%	63%	82%	94%
S1	5	45%	97%	99%	98%	97%
S2	4	61%	98%	98%	91%	82%
S2	5	67%	100%	99%	100%	99%
S3	2	3%	81%	32%	35%	68%
S3	10	5%	69%	60%	49%	69%
S5	4	90%	99%	91%	53%	78%
S5	7	18%	98%	74%	56%	81%
S5	8	68%	98%	99%	74%	85%

Table 5. A-test results showing effect sizes of the differences between distributions of total task completion times. For each column of Strategy A vs Strategy B results show the chance that A will take longer than B. Green cells mark those for which the p-value indicated a statistically significant difference in strategy performance. In all statistically significant examples nSSR is inferior, slower, than all other strategies. DSR and nDSR are often faster than SSR, with some exceptions where SSR outpaces nDSR. nDSR and DSR usually have similar performance, but where they are statistically significantly different DSR proves faster. DSR with DMS proves statistically significantly different to nSSR in almost all scenarios, whilst differing from DSR and nDSR in fewer scenarios. DMS is faster than nDSR for a majority of scenarios and slower than standard DSR for more than half of the statistically different scenarios.

Structure	ID of Failed Module	nSSR vs nDSR	nSSR vs SSR	nSSR vs DSR	nDSR vs SSR	nDSR vs DSR	SSR vs DSR	nSSR vs DMS	nDSR vs DMS	SSR vs DMS	DSR vs DMS
10B	5	0.77	0.88	0.85	0.52	0.56	0.60	0.80	0.47	0.41	0.37
10B	7	0.88	0.89	0.90	0.31	0.45	0.60	0.82	0.34	0.50	0.40
12A	2	0.69	0.96	0.94	0.7	0.82	0.78	0.82	0.34	0.50	0.40
12A	4	0.84	0.81	0.89	0.29	0.52	0.80	0.83	0.47	0.74	0.45
Rand	6	0.67	0.94	0.98	0.72	0.89	0.90	0.89	0.79	0.74	0.49
Rand	7	0.55	0.75	0.74	0.57	0.58	0.62	0.70	0.58	0.59	0.51
Rand	9	0.82	0.92	0.91	0.57	0.67	0.68	0.91	0.67	0.73	0.49
S1	2	0.89	0.83	0.91	0.36	0.60	0.77	0.88	0.57	0.72	0.48
S1	5	0.93	0.79	0.93	0.16	0.45	0.84	0.95	0.47	0.89	0.52
S2	4	0.88	0.92	0.99	0.23	0.67	0.92	0.92	0.27	0.72	0.13
S2	5	0.95	0.86	0.95	0.13	0.52	0.87	0.95	0.74	0.91	0.67
S3	2	0.86	0.97	0.90	0.58	0.67	0.71	0.95	0.52	0.42	0.33
S3	10	0.65	0.95	0.96	0.71	0.85	0.85	0.93	0.85	0.85	0.52
S5	4	0.73	0.82	0.84	0.36	0.47	0.73	0.92	0.66	0.89	0.74
S5	7	0.91	0.96	0.94	0.42	0.55	0.70	0.94	0.66	0.83	0.64
S5	8	0.80	0.89	0.90	0.41	0.55	0.77	0.92	0.52	0.84	0.48

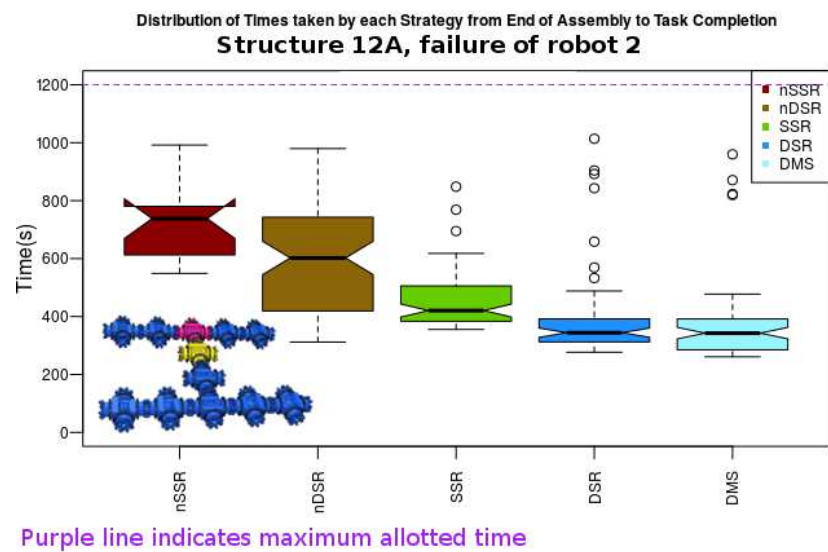


Figure 12. A boxplot showing the task completion times when Robot 2 fails in structure 12A. Robot 2 has a single large substructure of 6 modules attached below port 4. DSR out-competes SSR, but has quite a few outliers where substructures became lost and struggled to find the Replacement Module's recruitment cone. Dynamic Master Switching allows the larger part of the structure to remain and has the row of 5 robots, containing the original master, act as the Removal substructure. This alteration has negligible effect on task completion times, the challenge here still remains that of navigating a large substructure back in to place. The structure is shown for reference with the master robot marked in pink and the failed robot in yellow.

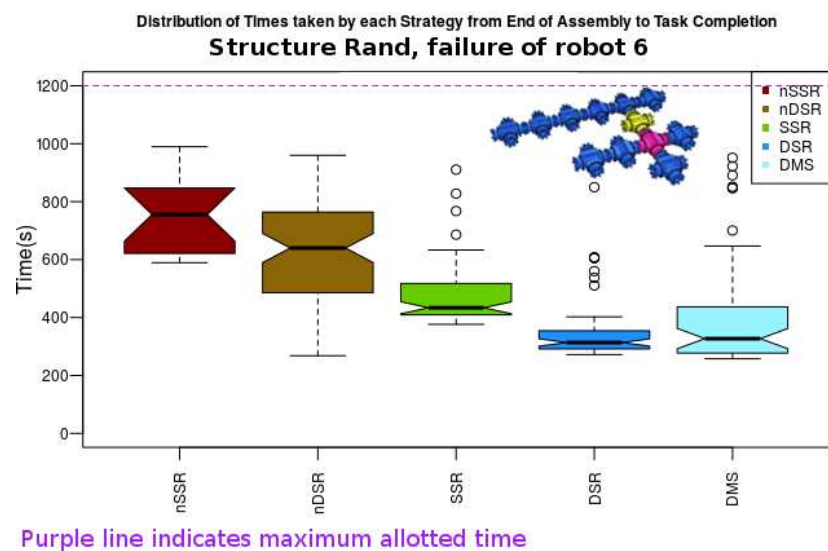


Figure 13. When robot 6 fails in structure Rand dynamic self-repair methods allow faster task completion than comparable static strategies. Substructure based self-repair also provides a speed advantage over naive breakup methods.

827 The size of a structure below the failed module can be considered either as an
 828 absolute count of subslaves or as a measure of how large specific substructures below it
 829 are. Fig.(14) shows task completion times plotted against the number of robots below the
 830 failed robot in the hierarchy of a structure. Lines of best fit indicate how each strategy's
 831 time requirements scale with the amount of robots below the failure site. Fig.(15) explores

832 the same trend but in terms of the largest single substructure below the failed module,
 833 rather than the total number of subslaves.

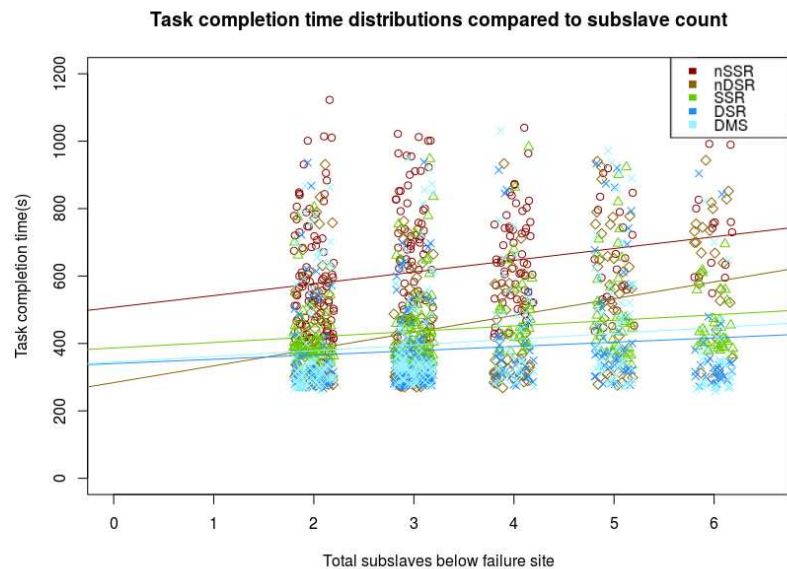


Figure 14. Datapoints from all structure and failure scenarios, excepting those which timed out before completion, showing how the performance of each of the four strategies varies with the number of subslaves docked below the failed module in the structure's hierarchy. The naive breakup based strategies exhibit stronger Pearson correlation coefficients than the self-repairing strategies, for nSSR, nDSR, SSR, DSR and DMS respectively these are 0.25, 0.34, 0.19, 0.11 and 0.13. For graphical clarity a random sample of datapoints have been hidden and left-right jitter has been applied, lines of best fit are still derived from full unjittered datasets.

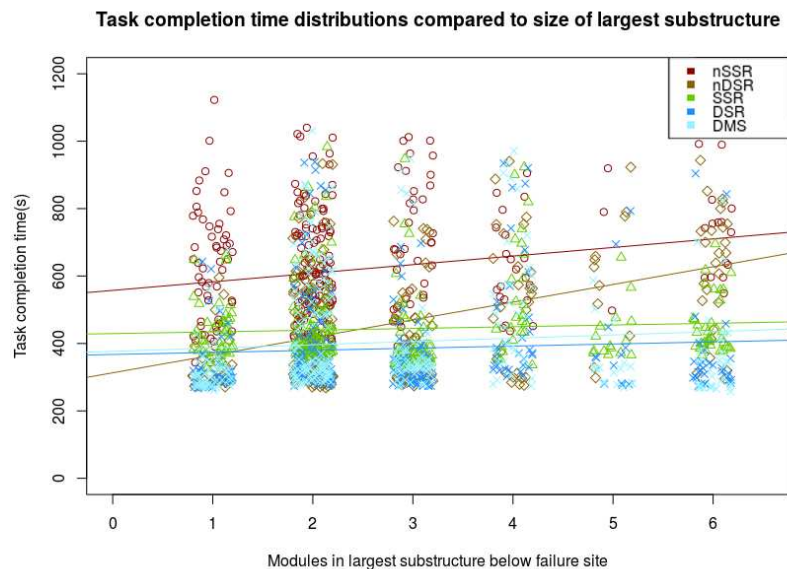


Figure 15. Comparing completion times for the locomotion task, during which the self-repair operation occurred, to the size of the largest single substructure below the failed module shows stronger correlations, 0.18 and 0.4 for nSSR and nDSR respectively, for naive strategies than for self-repair based strategies. The self-repair strategies plausibly appear uncorrelated to the size of the largest substructure involved in the self-repair operation, suggesting them to be highly scalable.

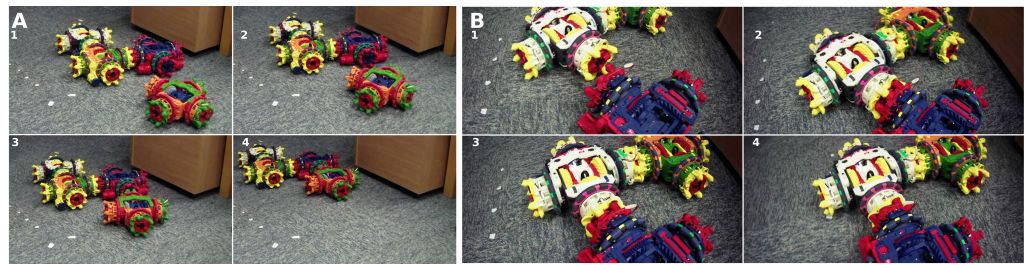
834 **8. Hardware Demonstration of Principles**

Figure 16. **A** The yellow robot (right) navigates to dock with port 3 of the white robot, meanwhile a blue robot (left) in the “wandering” state avoids the forming structure. **B** The seed robot (blue) recruits for a robot to take a Temporary ID of 3 (the white robot), the robots were positioned and held so that a further robot (green) could dock to this recruit before the recruit docked with the seed. This image sequence shows the use of co-ordinated MNS motion to enable a docking between the two robot group and the seed. See S6 in the supplementary material for video footage.

835 To demonstrate the feasibility of MLR self-assembly in hardware robots were
 836 programmed with a C implementation of the MLR controller. It was known that the IR
 837 38KHz communications could prove unreliable, and features were added to the controller
 838 which broadcast copies of some of the IR message types over Wi-Fi, particularly the
 839 later parts of the recruitment handshaking procedure. The hardware implementation
 840 also added an extra state to the Finite State Machine, see Fig.(2), this state, referred to as
 841 “Ramming Speed”, was placed between “Approach to Dock” and “In Organism”. This
 842 state was entered as soon as both docking switches on the appropriate port of a recruited
 843 robot were pressed indicating reaching the docking position, and lasts for 3 seconds
 844 during which the recruited robot drives at full speed against the port to maintain good
 845 contact while its hooks lock (a 0.3 second procedure), after which the robot transitions
 846 to the “In Organism” state and acts equivalently to in simulation. This is in contrast to
 847 simulations where docking actuation is instant.

848 A robot within the swarm was manually instructed to become seed for one of two
 849 different structures, a T shaped structure ($\{\{1,1,1,2\},\{1,3,1,3\},\{1,4,1,4\}\}$) or an S shaped
 850 structure ($\{\{1,2,2,2\},\{2,3,4,3\},\{3,3,1,4\}\}$), and began moving along a northward direction
 851 calculated from its compass readings. Due to the limited number of robots available
 852 it was not feasible for random wandering to bring robots within recruitment range
 853 regularly, hence other modules were placed in regions where they could receive its
 854 recruitment communications over IR and allowed to dock autonomously from there.
 855 Robots were allowed to dock to form the structures and navigated themselves for
 856 both positioning and compass alignment, some of the placed robots were put at 90° or
 857 180° away from the orientations but nonetheless still aligned and docked. Fig.(16A)
 858 shows an example of these docking operations. Further robots docked until the structure
 859 was formed. The multi-robot structures were then allowed to move around and avoid
 860 obstacles to provide proof of the MNS capabilities. Videos of some of these recruitment
 861 events and of coordinated group motion are provided in the supplementary materials
 862 (S6).

863 During docking it was found that real world non-uniformity in magnetic fields
 864 meant that in the distance between two connected robots the field often turned by up
 865 to 10° , so robots coming together would often be 10° off compass alignment. Random
 866 noise of $\pm 5^\circ$, varying every 0.6 seconds, was added to the compass reading during
 867 final approach for docking ensuring that sooner or later the two modules would align
 868 their hooks to within the angle accuracy needed to dock successfully. This magnetic
 869 field non-uniformity did not affect robots once docked within an organism, as our MNS
 870 implementation was carefully designed to avoid explicit reference to local compass
 871 readings and handle all data in the reference frame of the master robot with inter-robot
 872 angles calculated by knowledge of port to port links and not by compass readings.

873 A notable problem encountered was establishing whether a robot was docked on
874 any given port. Whilst a simple check to perform in simulation, the physical hardware's
875 tolerances are such that when two robots are docked there is sufficient play within the
876 mechanism to allow for the contact switches to break contact even whilst a dock exists.
877 When determining whether a port is docked there are two considerations of importance,
878 the first is simple, knowing whether a robot's own docking hooks on a particular port are
879 locked or unlocked, the second is knowing whether the other robot involved in the dock
880 is present and has its hooks locked. Methods based on low pass filtering the docking
881 switch states to remove brief fluctuations struggled to tell the repeated impacts on each
882 switch involved in pre-docking manoeuvres from a successful dock, and also incorrectly
883 assumed a dock no longer existed when robots were driving in such a direction as to
884 relieve the pressure on the switches for significant time periods. Wi-Fi communications
885 were used to let robots agree that a dock had formed between them, with any robot
886 undocking sending a Wi-Fi message to this effect. But reliance on communications to
887 signal the making and breaking of docked connections allowed for occasional incidents
888 where a robot wrongly believed itself to be docked, and was believed by its immediate
889 master to have docked, despite having become disconnected in the 0.3 seconds between
890 beginning actuating the docking hooks and having them reach the locked position due
891 to the master robot moving at that instant in ways which "Ramming Speed" was not able
892 to correct for. Almost all activities with modular robots would appear to require reliable
893 ways for a robot to check at any given moment whether any given one of its ports is
894 docked, so it may not be possible to develop controllers which can achieve desirable
895 levels of reliability without accurate "dock presence sensing".

896 Difficulties were also encountered in some runs due to the replacement of line-of-
897 sight IR communications in some roles with universally available Wi-Fi messages. This
898 made the non-conflict docking procedures more difficult, as only the initial recruitment
899 messages were now sent over the line-of-sight IR system. With short range verification
900 messaging not able to work reliably this too had to be replaced with Wi-Fi, therefore in a
901 small number of cases robots were able to dock to ports other than the one which they
902 thought they were approaching. Unable to rely on the readings of the contact switches,
903 robots were not able to easily detect such incorrect dockings. Further Wi-Fi handshaking
904 could not entirely mitigate this as they were not position dependent in the way that the IR
905 was designed to be. These problems show the benefit which line-of-sight communication
906 can bring if it is reliable, particularly that approximate positional information can be
907 inferred and that receipt of certain message types guarantees relative positions, and the
908 hazards of trying to work around line-of-sight to use global communications. It is worth
909 repeating that the difficulties caused by a lack of line-of-sight communication here show
910 that line-of-sight communication is not inferior to global communication in all modular
911 robotics applications, despite the popularity of broadcast methods [49].

912 The Mergeable Nervous Systems [41] inspired co-ordinated group control methods
913 worked extremely well, giving reliable control of the multi-robot organism. Tests were
914 performed by placing an IR reflecting obstacle in to various positions around a completed,
915 or partially completed, organism and watching its reactions. Obstacles placed beside any
916 docking port, whether on the seed robot, on a robot connected to the seed, or on a robot
917 two layers away from the seed, were detected and the structure retreated away from
918 them according to the clock direction in which they were observed. Ports facing towards
919 other parts of the structure were, as intended, not sensitive to obstacles. Rotation of the
920 robotic group was also demonstrated.

921 Out of 26 dockings attempted between wandering modules and moving recruiters,
922 20 of these actions succeeded, those which failed were mostly due to the approaching
923 robot timing out of the docking attempt after taking too long.

924 Tests were also performed to demonstrate MLR's use of co-ordinated docking be-
925 tween robotic groups. For the S shaped structure a robot entering the position equivalent
926 to a Temporary ID of 3 was allowed to complete the handshaking procedure and then

927 temporarily manually restrained. An extra robot was placed close by this robot's Port 3
 928 and allowed to dock to it. The robot with a Temporary ID of 3 was then released, to act as
 929 master of a two robot group and attempt docking to the seed robot. Fig.(16B) shows this.
 930 It was difficult to create the circumstances for this event and the robot with a Temporary
 931 ID of 3 usually completed a docking to the seed before its subslave could be introduced
 932 to the area. This proves the principle of MLR to be feasible with hardware, and matches
 933 up with the simulations, which show a lack of significant difference between MLR and
 934 LW+MNS performance, in that most of the time robots did not typically form such
 935 substructures before docking, due to speed considerations involved.

936 8.1. Self-repair, and the Reality Gap

937 For self-repair demonstrations modules were manually placed for quick initial
 938 self-assembly, then once they were moving a switch was pressed on one module to
 939 trigger a failure mode. In this failure mode the module's wheels stopped, all IR and
 940 Wi-Fi communications ceased, and proximity sensing was disabled. Connected modules
 941 were required to detect the failure, then remap their structure and take on appropriate
 942 roles for self-repairing. Once the failed module had been removed it was manually
 943 released from its failure mode and the software reinitialised such that it acted as a free
 944 wandering robot. The limited number of modules available meant that providing a spare
 945 module to become the Replacement Module would too greatly limit the structure sizes
 946 possible, so the failure mode was made reversible, with the failed module transitioning
 947 to "Wandering" after being dumped. It was then allowed to dock again either as the
 948 Replacement Module or as a recruit to replace other parts of the broken up structure
 949 when Mergeable Nervous Systems over Wi-Fi was in use.

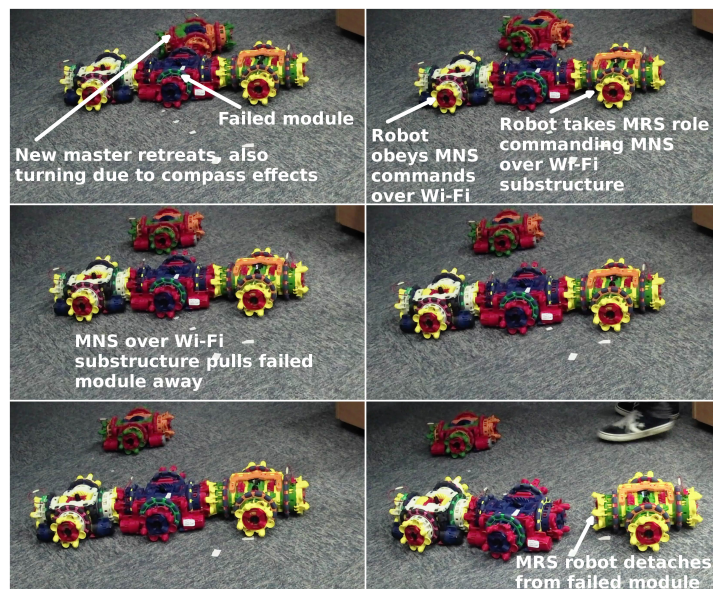


Figure 17. Neighbouring robots attempt to manoeuvre a failed module, whilst the new master retreats.

950 With only 4 hardware modules available scenarios involving large substructures
 951 could not be created, a T structure was used with a failure induced in the global master.
 952 After Dynamic Master Switching processes had occurred, see the supplementary material
 953 for video footage (S7), Mergeable Nervous Systems over Wi-Fi was automatically selected
 954 by the robots as the method for removing the failed module. Unfortunately the two
 955 modules available to remove the failed module had insufficient torque to drag it when
 956 its wheels were stationary. The controller was modified to allow a failed module to still
 957 listen to MNS actuation commands over Wi-Fi from the robot commanding its removal,
 958 however these communications were not always acted upon correctly by the failed robot,

959 likely due to other effects created by the failure state induced. Fig.(17) shows the attempt
960 in progress, the failed module followed the Wi-Fi commands at some points but at others
961 the whole retreating structure was stalled when it did not. The master side section of the
962 robotic organism did nonetheless successfully escape and begin re-recruitment.

963 This demonstration showed the ability of modules to detect a failed neighbour, and
964 showed Dynamic Master Switching working over Wi-Fi. Although coding errors made it
965 difficult to return a module to correct operation after a failure was induced, one incident
966 was observed in which modules which had attempted an MNS over Wi-Fi retreat were
967 re-recruited to partially rebuild the structure, see the supplementary material (S7).

968 Another demonstration was attempted where Dynamic Master Switching features
969 were disabled to allow ordinary Dynamic Self-repair to occur, this was intended to cause
970 a situation where the docking of a multi-module substructure to the newly recruited
971 Replacement Module could be observed. An S structure was used with the fault injected
972 such that a two module MRS substructure was formed. This demonstration however
973 suffered from a number of reality gap issues. The difficulties inherent in dock detec-
974 tion meant robots struggled to recognise undocking events, Recruitment Lists shared
975 between them were therefore often inaccurate descriptions of the structure's current
976 state. Compared to self-assembly, the requirements which self-repair places upon the
977 reliability of docking detection equipment are far more exacting. Whilst in self-assembly
978 the difficulties could be overcome with software work-arounds, self-repair's more fre-
979 quent and repetitive docking and undocking events overwhelmed the ability of software
980 work-arounds to accurately keep track of the port states. This provides an example of
981 how self-repair proves to be a much more complex task than self-assembly, not only
982 at the theoretical level and in the design of robotic controller software, but also at the
983 practical level in ways which do not become apparent until real hardware is involved.

984 As successful removal of a failed module could not be performed it was not fea-
985 sible to test in hardware the guidance methods to return substructures to the main
986 organism. However by providing proof that it is possible for neighbouring hardware
987 modules to detect failed modules, and given that the Mergeable Nervous Systems and
988 Multi-Layered Recruitment hardware demonstrations showed that the collaborative
989 driving necessary for removing failed modules and for returning substructures to dock is
990 possible, achieving Dynamic Self-repair with hardware would appear to be primarily a
991 matter of improving wheel torques, docking detection and line-of-sight communication
992 reliability.

993 **9. Discussion**

994 *9.1. Self-assembly*

995 In the scenarios where a low p-value shows a strong statistical difference between
996 strategy performance the dynamic self-assembly strategies produce A-test measures
997 indicating their superiority, in many of these scenarios dynamic self-assembly methods
998 perform better in over 90% of randomised runs. The LW+MNS and MLR scenarios rarely
999 see a statistically significant performance difference.

1000 Significantly low p-values (<0.05) from the comparison, for most scenarios, of the
1001 LW+ distribution to the LW+MNS or MLR distributions allow a null hypothesis that
1002 being able to assemble in motion has no effect on task completion times to be rejected.
1003 Effect sizes for the comparison of these strategies all favour the self-assembly strategies
1004 which can assemble during motion, for many of the scenarios these effect sizes are
1005 extremely strong indeed.

1006 MLR and LW+MNS do not substantially differ in performance, this may be due
1007 to the existence of a maximum possible wheel speed for robots. A robot undergoing
1008 recruitment, therefore in the "Rotate to Dock" or "Approach to Dock" phase of the finite
1009 state machine, must move so as to match the motions of the recruiter's reference frame
1010 and travel towards the recruiting port. It would appear sensible to use, once speed
1011 matching to the recruiter is achieved, whatever reserves of possible motor speed are left

1012 to perform this approach. Robots therefore attempt this approach at somewhere at, or
1013 above 70% of their maximum wheel speed. Any robot recruited to a moving module will
1014 have only up to 30% of wheel speed with which to approach towards it. MLR therefore
1015 often does not in practice result in multi-module structures forming around a robot as it
1016 is still in its "Approach to Dock" state. The only way that such formation would become
1017 more common would be if robots used lower speeds for their approach to recruiting
1018 ports. It appears unlikely that slowing the speeds, as a percentage of the maximum
1019 wheel speed, used for the approach would hasten assembly overall and it would make
1020 timing-out, entering the "Escape Dock" state, more common. Optimising the wheel
1021 speeds used during docking to see whether it is more beneficial to overall self-assembly
1022 time to use a slow approach, with more scope for multiple layers of recruitment, or
1023 a fast approach, despite the reduction of opportunities for multi-layered recruitment
1024 that this would cause, could be an interesting topic for further research. It is perhaps
1025 therefore best considered that while MLR provides an interesting idea, and a useful
1026 stepping-stone from which further capabilities could be built, it is not in itself able
1027 to, typically, provide significantly better performance, when self-assembling while in
1028 motion, than the LW+MNS strategy.

1029 While the simulated experiments showed the performance benefits of self-assembly
1030 during motion, hardware experiments proved its practical feasibility, with docking
1031 to a moving target and coordinated MNS control of a structure shown to be reliable,
1032 once workarounds had been implemented to account for communication and sensing
1033 limitations, in the real world. This is the first demonstration of self-assembly during
1034 motion with modular robots. **It is also of interest to note that the improved co-ordination
1035 of modules demonstrated here may in future allow simulated work which requires
1036 complex robot navigation [27] to become feasible to physically implement.**

1037 9.2. Self-repair

1038 Comparison of the proportion of runs which completed without timing out gives
1039 an indication of the reliability of a particular self-repair strategy. nDSR's high proportion
1040 of successful runs across all scenarios indicates reliability. The two substructure based
1041 self-repair strategies displayed levels of reliability which varied across scenarios. nSSR
1042 performed very poorly in this regard. nDSR and DSR both have greater chances of en-
1043 counteracting wandering modules to recruit as replacements than static strategies, with the
1044 main limitation of DSR being the difficulty of guiding the removal handling substructure
1045 back to dock. The main lesson here appears to be that for industrially useful self-repair,
1046 with a near 100% completion rate within some timeframe, more work must be done on
1047 extending the range of guidance and finding solutions to aid in navigation over distance,
1048 when subject to the condition of not relying on explicit GPS style location data this is an
1049 interesting problem for sensor development and search pattern driving algorithms

1050 The ability to repair while in motion, both for nDSR and DSR, gave statistically
1051 significant performance differences, with p-values below the 5% threshold, in most
1052 scenarios. A-test results largely above 0.8 in statistically significant scenarios show nDSR
1053 outperformed nSSR, and A-test results above 0.7 for a vast majority of scenarios also
1054 show DSR's speed advantage over SSR. A null hypothesis that being able to repair
1055 during motion has no effect on task completion times can be confidently rejected, dy-
1056 namic strategies regularly outperform their equivalent static strategies. Performing both
1057 the repair and motion tasks in parallel rather than serial can be considered as one of
1058 the causes behind this effect. This provides a good indication of the assistance that
1059 Dynamic Self-repair methods can provide for scenarios where overall task time is the
1060 key measurement, such as for reducing accumulated radiation doses.

1061 The advantage provided by strategies able to operate in motion does little to ac-
1062 celerate the repair procedure itself, despite the speed advantages which it provides for
1063 the overall task being attempted. Whilst DSR and nDSR mostly outperformed nSSR
1064 the SSR strategy proved most effective in terms of time for the repair itself to complete.

1065 Differences, measured in seconds, between different strategies times to perform the
1066 repair operation itself tend to be considerably smaller than the differences in task com-
1067 pletion times, showing that the differences in repair times are not the main cause for
1068 the differences in task completion times. The repair procedure itself is observed to be
1069 significantly faster when substructure rather than breakup strategies are in use.

1070 Naive breakup based strategies were outperformed in terms of task completion
1071 times by substructure based self-repair in some of the scenarios. Across static strategies
1072 SSR performed faster than nSSR by a statistically significant amount for all combinations
1073 of structure and failed module, A-test scores for the effect size were all high with many
1074 above 0.90. This provides a replication of some of Murray's findings [10] with a new
1075 platform and code base. When comparing between dynamic strategies the situation is
1076 less clear with statistically significant differences only occurring in 7 of the 16 scenarios.
1077 In those scenarios where statistically significant differences were found all favoured
1078 the substructure based DSR over the breakup based nDSR, with A-test scores ranging
1079 from 0.6 to 0.89. It is possible here that the extremely high reliability of nDSR makes
1080 it difficult for DSR, a less reliable strategy, to substantially outperform it in many of
1081 the statistically similar scenarios, improving sensors or search patterns may change
1082 this. DSR may also display less advantage over nDSR than SSR does over nSSR due
1083 to the increased probability of substructures involved in the repair getting lost whilst
1084 attempting to relocate the main structure, a scenario less likely to happen when the main
1085 structure remains static.

1086 The experiments found, for SSR and DSR, no strong evidence of correlations be-
1087 tween either the total number of subslaves below a failed module in a hierarchy, or the
1088 number of modules in the largest single substructure formed during self-repair by the
1089 failed module's slaves. Both static and dynamic self-repair methods allow for task com-
1090 pletion times which do not appear to scale up with structure size by either the subslave
1091 count or largest substructure measure. Task completion times do however rise, with
1092 Pearson correlation coefficients in the range of 0.18 to 0.40 for breakup based strategies.
1093 nDSR has the strongest correlation coefficient, being strongest when compared against
1094 the size of the largest substructure rather than the total subslave count. This suggests that
1095 naive breakup based strategies are less scalable than substructure preserving self-repair,
1096 that the size of the largest substructure has a stronger effect than the total subslave count
1097 also indicates that an increased number of "layers" on which the re-assembly must take
1098 place may be of importance here. Breakup based strategies lose performance, relative to
1099 substructure preserving ones, as the number of modules and size of substructures below
1100 the repair site rises. The relative performance of DSR against SSR stays constant with
1101 structure sizing, showing that the beneficial effect of being able to self-repair during
1102 motion remains roughly constant with structure size.

1103 The DMS strategy largely performs similarly to DSR, with, overall, no statistically
1104 significant speed benefit above DSR. However the ability to switch master location
1105 enables repairs in scenarios which the other strategies cannot handle, such as the failure
1106 of the global master, and provides reliability improvements due to the use of smaller
1107 removal substructures and therefore reduced opportunities for substructures to become
1108 lost while guiding themselves back to dock.

1109 Hardware demonstrations found self-repair to be a much more difficult task than
1110 self-assembly, with sensor, communication and wheel torque limitations proving particu-
1111 larly problematic. The practicality of detecting a module failure via a lack of port-to-port
1112 communications from it was verified, and the ability of the master side neighbour of
1113 a failed module to retreat was shown. Dynamic master switching was demonstrated
1114 to work on real robots with Recruitment Lists correctly updated and transferred. The
1115 feasibility of the underlying principles for dynamic self-repair has therefore been demon-
1116 strated, but fully proving its practicality was not possible with this hardware. The
1117 importance of implicit information in communications is highlighted by the difficulties
1118 encountered. Such difficulties encountered also suggest that successful completion of

1119 dynamic self-repair strategies may be a worthwhile benchmarking test to compare the
1120 performance of modular robotic platforms.

1121 10. Conclusion

1122 This paper has proposed and demonstrated new strategies for self-assembly and
1123 self-repair with modular robots during motion. It has been explained how the unique
1124 features of the Omni-Pi-tent platform make such strategies feasible and discussion of
1125 implementation details has been presented. The new self-assembly strategies were com-
1126 pared against a “classical” self-assembly to a static seed, inspired by Liu and Winfield’s
1127 work. It was found that, in all scenarios for which conclusive data could be gathered,
1128 one or both dynamic self-assembly strategies had statistically significantly better per-
1129 formance. Hardware tests verified the feasibility of docking and Mergeable Nervous
1130 Systems group control and showed self-assembly during motion as well as Multi-layered
1131 scenarios. No prior self-assembly work has managed to operate at all with co-ordinated
1132 group motion occurring while assembly progresses. The new self-repair strategies were
1133 compared against methods inspired by Murray’s work, strategies able to repair during
1134 motion were found to lead to faster task completion than strategies requiring motion
1135 to halt for repair to be undertaken, strategies able to use substructures were found to
1136 be faster to complete and more scalable than strategies requiring breakup of structures.
1137 Hardware demonstrations proved many of the fundamental activities required for dy-
1138 namic self-repair, but could not achieve the full procedure due to reality gap issues. The
1139 challenges encountered indicate the exacting demands self-repair places on hardware
1140 and connector interface reliability above those required to allow self-assembly.

1141 Future work could focus on modifying self-assembly and self-repair methods to
1142 handle loops, potentially by forming multiple non-loop-containing structures then bring-
1143 ing them together under MNS control, and rules regarding specialised heterogeneous
1144 modules within an organism, as well as 3 dimensional self-assembly and self-repair via
1145 the use of module hinge joints and improved long range robot guidance methods and
1146 post-disassembly search patterns to aid in relocating modules during self-repair. Such
1147 search patterns may be able to draw inspiration from those used at sea to locate drifting
1148 objects. There are also possibilities for performing self-repair where detached sections of
1149 the structure may re-assemble to replace parts of a morphology which are not the same
1150 sections which they previously acted as, **potentially adapting some of [26]’s findings for
1151 use in this context.** Due to the hardware findings of the importance of reliable docking,
1152 dock detection, and port to port communication equipment for modular robots, another
1153 major aspect of future work would be refining those systems and developing standards
1154 for use in future modular robotic designs.

1155 **Supplementary Materials:** The following videos are available in the supplementary materials,

1156 S1: Example_DMS_Scenario.mp4,

1157 S2: Example_DSR_scenario.mp4,

1158 S3: Example_nDSR_scenario.mp4,

1159 S4: Example_SSR_scenario.mp4,

1160 S5: Example_nSSR_scenario.mp4,

1161 S6: Multi-Layered-Recruitment_Self-assembly_hardware_demonstration.mp4,

1162 S7: Self-repair_and_Dynamic-Master-Switching_hardware_demonstration.mp4,

1163 S8: Self-assembly_Highlights.mp4,

1164 S9: Self-repair_Highlights.mp4

1165 **Author Contributions:** Conceptualisation, All; Hardware design and development, R.H.P; Strat-
1166 egy design and development , R.H.P; Methodology, All; Investigation and validation, R.H.P;
1167 Writing R.H.P; Supervision, J.T and A.M.T; Funding acquisition, A.M.T. All authors have read and
1168 agreed to the published version of the manuscript.

1169 **Funding:** This research was funded by EPSRC via R.H.Peck’s PhD studentship.

1170 **Data Availability Statement:** Detailed pseudocode for the self-assembly, self-repair and dynamic
 1171 master switching algorithms, as well as further results and hardware details, are available in
 1172 R.H.Peck's thesis [46] (etheses.whiterose.ac.uk/30288/).

1173 **Acknowledgments:** The authors wish to thank: Mark Hough, Andy White, Mike Angus, Dave
 1174 Hunter and the rest of the Fourth Floor technicians for advice on hardware fabrication, and the
 1175 University of York Research Computing Team for Viking Cluster support.

1176 **Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the
 1177 design of the study; in the collection, analyses, or interpretation of data; in the writing of the
 1178 manuscript, or in the decision to publish results.

1179 Abbreviations

1180 The following abbreviations are used in this manuscript:

1181	MNS	Mergeable Nervous Systems
	LW+	Liu and Winfield inspired self-assembly
	LW+MNS	LW+ with MNS capabilities
	MLR	Multi-Layered Recruitment
	MFM	Master to the Failed Module
	MRS	Master of the Removing Substructure
	MAS	Master of Another Substructure
1182	LM	Lone Module
	RM	Replacement Module
	nSSR	Naive Static Self-Repair
	nDSR	Naive Dynamic Self-Repair
	SSR	Static Self-Repair
	DSR	Dynamic Self-Repair
	DMS	Dynamic Master Switching

References

1. Fukuda, T.; Nakagawa, S. Dynamically reconfigurable robotic system. In Proceedings of the 1988 IEEE international conference on robotics and automation; IEEE, , 1988; pp. 1581–1586. doi:10.1109/ROBOT.1988.12291.
2. Yim, M.; Shen, W.M.; Salemi, B.; Rus, D.; Moll, M.; Lipson, H.; Klavins, E.; Chirikjian, G.S. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robotics & Automation Magazine* **2007**, *14*, 43–52. doi:10.1109/MRA.2007.339623.
3. Levi, P.; Kernbach, S. *Symbiotic multi-robot organisms: reliability, adaptability, evolution*; Vol. 7, Springer Science & Business Media: Berlin, Germany, 2010. doi:10.1007/978-3-642-11692-6.
4. Davey, J.; Kwok, N.; Yim, M. Emulating self-reconfigurable robots-design of the SMORES system. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE, , 2012; pp. 4464–4469. doi:10.1109/IROS.2012.6385845.
5. Kernbach, S.; Meister, E.; Scholz, O.; Humza, R.; Liedke, J.; Ricotti, L.; Jemai, J.; Havlik, J.; Liu, W. Evolutionary robotics: The next-generation-platform for on-line and on-board artificial evolution. 2009 IEEE congress on evolutionary computation; IEEE, , 2009; pp. 1079–1086. doi:10.1109/CEC.2009.4983066.
6. Parrott, C.; Dodd, T.J.; Groß, R. HyMod: A 3-DOF hybrid mobile and self-reconfigurable modular robot and its extensions. In *Distributed Autonomous Robotic Systems*; Springer, 2018; pp. 401–414. doi:10.1007/978-3-319-73008-0_28.
7. Peck, R.H.; Timmis, J.; Tyrrell, A.M. Omni-pi-tent: An omnidirectional modular robot with genderless docking. Annual Conference Towards Autonomous Robotic Systems; Springer, Cham, , 2019; pp. 307–318. doi:10.1007/978-3-030-25332-5_27.
8. Hancher, M.D.; Hornby, G.S. A modular robotic system with applications to space exploration. 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06); IEEE, , 2006; pp. 1–8. doi:10.1109/SMC-IT.2006.9.
9. Baca, J.; Hossain, S.; Dasgupta, P.; Nelson, C.A.; Dutta, A. Modred: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. *Robotics and Autonomous Systems* **2014**, *62*, 1002–1015. doi:10.1016/j.robot.2013.08.008.
10. Murray, L. Fault tolerant morphogenesis in self-reconfigurable modular robotic systems. PhD thesis, University of York, 2013.
11. Jahanshahi, M.R.; Shen, W.M.; Mondal, T.G.; Abdelbarr, M.; Masri, S.F.; Qidwai, U.A. Reconfigurable swarm robots for structural health monitoring: a brief review. *International Journal of Intelligent Robotics and Applications* **2017**, *1*, 287–305. doi:10.1007/s41315-017-0024-8.
12. Peck, R.H.; Timmis, J.; Tyrrell, A.M. Towards Self-repair with Modular Robots During Continuous Motion. Towards Autonomous Robotic Systems: 19th Annual Conference, TAROS 2018, Bristol, UK July 25–27, 2018, Proceedings; Springer, , 2018; Vol. 10965, pp. 457–458.

13. Liu, W.; Winfield, A.F. Autonomous morphogenesis in self-assembling robots using IR-based sensing and local communications. *International Conference on Swarm Intelligence*. Springer, 2010, pp. 107–118. doi:10.1007/978-3-642-15461-4_10.
14. Tomita, K.; Murata, S.; Kurokawa, H.; Yoshida, E.; Kokaji, S. Self-assembly and self-repair method for a distributed mechanical system. *IEEE Transactions on Robotics and Automation* **1999**, *15*, 1035–1045. doi:10.1109/70.817668.
15. Rohmer, E.; Singh, S.P.; Freese, M. V-REP: A versatile and scalable robot simulation framework. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE, , 2013; pp. 1321–1326. doi:10.1109/IROS.2013.6696520.
16. Groß, R.; Dorigo, M. Self-assembly at the macroscopic scale. *Proceedings of the IEEE* **2008**, *96*, 1490–1508. doi:10.1109/JPROC.2008.927352.
17. Rus, D.L.; Gilpin, K.W. Modular robot systems. *IEEE Robotics & Automation Magazine* **2010**, *17*, 38–55. doi:10.1109/MRA.2010.937859.
18. Tucci, T.K.; Piranda, B.; Bourgeois, J. A distributed self-assembly planning algorithm for modular robots. *International Conference on Autonomous Agents and Multiagent Systems*; , 2018. doi:10.5555/3237383.3237465.
19. Bererton, C.; Khosla, P.K. Towards a team of robots with repair capabilities: a visual docking system. In *Experimental Robotics VII*; Springer: Berlin, Germany, 2001; pp. 333–342. doi:10.1007/3-540-45118-8_34.
20. Rubenstein, M.; Payne, K.; Will, P.; Shen, W.M. Docking among independent and autonomous CONRO self-reconfigurable robots. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004; IEEE, , 2004; Vol. 3*, pp. 2877–2882. doi:10.1109/ROBOT.2004.1307497.
21. Rubenstein, M.; Cornejo, A.; Nagpal, R. Programmable self-assembly in a thousand-robot swarm. *Science* **2014**, *345*, 795–799. doi:10.1126/science.1254295.
22. Doursat, R. Organically grown architectures: Creating decentralized, autonomous systems by embryomorph engineering. In *Organic computing*; Springer: Berlin, Germany, 2009; pp. 167–199. doi:10.1007/978-3-540-77657-4_8.
23. Nagpal, R. Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics. PhD thesis, Massachusetts Institute of Technology, 2001.
24. Werfel, J. Biologically realistic primitives for engineered morphogenesis. *International Conference on Swarm Intelligence*; Springer, , 2010; pp. 131–142. doi:10.1007/978-3-642-15461-4_12.
25. Stoy, K. Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems* **2006**, *54*, 135–141. doi:10.1016/j.robot.2005.09.017.
26. Dutta, A.; Dasgupta, P.; Nelson, C. Distributed configuration formation with modular robots using (sub) graph isomorphism-based approach. *Autonomous Robots* **2019**, *43*, 837–857. doi:10.1007/s10514-018-9759-9.
27. Li, H.; Wang, T.; Chirikjian, G.S. Self-assembly Planning of a Shape by Regular Modular Robots. *Advances in Reconfigurable Mechanisms and Robots II*. Springer, Cham., 2016, Vol. 36, pp. 867–877. doi:10.1007/978-3-319-23327-7_74.
28. Yim, M.; Shirmohammadi, B.; Sastra, J.; Park, M.; Dugan, M.; Taylor, C.J. Towards robotic self-reassembly after explosion. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE, , 2007; pp. 2767–2772. doi:10.1109/IROS.2007.4399594.
29. Christensen, A.L.; O’Grady, R.; Dorigo, M. SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence* **2008**, *2*, 143–165. doi:10.1007/s11721-008-0012-6.
30. Baca, J.; Yerpes, A.; Ferre, M.; Escalera, J.A.; Aracil, R. Modelling of modular robot configurations using graph theory. *International Workshop on Hybrid Artificial Intelligence Systems*. Springer, 2008, pp. 649–656. doi:10.1007/978-3-540-87656-4_80.
31. Liu, C.; Lin, Q.; Kim, H.; Yim, M. SMORES-EP, a Modular Robot with Parallel Self-assembly. *arXiv preprint arXiv:2104.00800* **2021**. doi:10.48550/arXiv.2104.00800.
32. Wei, H.; Li, D.; Tan, J.; Wang, T. The distributed control and experiments of directional self-assembly for modular swarm robots. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE, , 2010; pp. 4169–4174. doi:10.1109/IROS.2010.5650244.
33. Murata, S.; Yoshida, E.; Kurokawa, H.; Tomita, K.; Kokaji, S. Self-repairing mechanical systems. *Autonomous Robots* **2001**, *10*, 7–21.
34. Fitch, R.; Rus, D.; Vona, M. A basis for self-repair robots using self-reconfiguring crystal modules. *Intelligent Autonomous Systems*. Citeseer, 2000, Vol. 6, pp. 903–910.
35. Ackerman, M.K.; Chirikjian, G.S. Hex-DMR: a modular robotic test-bed for demonstrating team repair. 2012 IEEE International Conference on Robotics and Automation; IEEE, , 2012; pp. 4148–4153. doi:10.1109/ICRA.2012.6225214.
36. O’Grady, R.; Pinciroli, C.; Groß, R.; Christensen, A.L.; Mondada, F.; Bonani, M.; Dorigo, M. Swarm-bots to the rescue. *European Conference on Artificial Life*. Springer, 2009, pp. 165–172. doi:10.1007/978-3-642-21283-3_21.
37. Arbuckle, D.; Requicha, A.A. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. *Autonomous Robots* **2010**, *28*, 197–211. doi:10.1007/s10514-009-9162-7.
38. Stoy, K.; Nagpal, R. Self-repair through scale independent self-reconfiguration. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566); IEEE, , 2004; Vol. 2, pp. 2062–2067. doi:10.1109/IROS.2004.1389701.
39. Rubenstein, M.; Shen, W.M. Scalable self-assembly and self-repair in a collective of robots. 2009 IEEE/RSJ international conference on Intelligent robots and systems; IEEE, , 2009; pp. 1484–1489. doi:10.1109/IROS.2009.5354716.
40. Christensen, D.J. Experiments on fault-tolerant self-reconfiguration and emergent self-repair. 2007 IEEE Symposium on Artificial Life; IEEE, , 2007; pp. 355–361. doi:10.1109/ALIFE.2007.367817.
41. Mathews, N.; Christensen, A.L.; O’Grady, R.; Mondada, F.; Dorigo, M. Mergeable nervous systems for robots. *Nature communications* **2017**, *8*, 1–7. doi:10.1038/s41467-017-00109-2.

42. Zhang, Y.; Song, G.; Liu, S.; Qiao, G.; Zhang, J.; Sun, H. A modular self-reconfigurable robot with enhanced locomotion performances: design, modeling, simulations, and experiments. *Journal of Intelligent & Robotic Systems* **2016**, *81*, 377–393. doi:10.1007/s10846-015-0228-9.
43. Popesku, S.; Meister, E.; Schlachter, F.; Levi, P. Active wheel-An autonomous modular robot. 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM); IEEE, , 2013; pp. 97–102. doi:10.1109/RAM.2013.6758566.
44. Jakobi, N.; Husbands, P.; Harvey, I. Noise and the reality gap: The use of simulation in evolutionary robotics. *European Conference on Artificial Life*. Springer, 1995, Vol. 929, pp. 704–720. doi:10.1007/3-540-59496-5_337.
45. Team, R.C.; others. R: A language and environment for statistical computing **2013**.
46. Peck, R. Self-repair during Continuous Motion with Modular Robots. PhD thesis, University of York, 2021.
47. Eiben, A.E.; Bredeche, N.; Hoogendoorn, M.; Stradner, J.; Timmis, J.; Tyrrell, A.; Winfield, A. The triangle of life: Evolving robots in real-time and real-space. *Proceedings of the European Conference on Artificial Life (ECAL-2013)*, 2013, pp. 1–8. doi:10.7551/978-0-262-31709-2-ch157.
48. Groß, R.; Bonani, M.; Mondada, F.; Dorigo, M. Autonomous self-assembly in swarm-bots. *IEEE transactions on robotics* **2006**, *22*, 1115–1130. doi:10.1109/TRO.2006.882919.
49. Seo, J.; Paik, J.; Yim, M. Modular reconfigurable robotics. *Annual Review of Control, Robotics, and Autonomous Systems* **2019**, *2*, 63–88. doi:10.1146/annurev-control-053018-023834.