IEEE Copyright Notice

A Cognitive Architecture for Modular and Self-Reconfigurable Robots

P. Levi*, E. Meister*, A.C. van Rossum†, T. Krajník‡, V. Vonásek‡, P. Stepan‡, W. Liu§, F. Caparrelli¶

*Institute of Parallel and Distributed Systems, University of Stuttgart, Universitätstr. 38, 70569 Stuttgart, Germany
† Almende B.V., Westerstraat 50, 3016 DJ Rotterdam, Netherlands
‡ Czech Technical University in Prague, Fac. of electrical eng., Technicka 2, 166 27, Prague, Czech Republic
§ University of the West England, Bristol, Coldharbour Lane BS16 1QY Bristol, UK
¶ Sheffield Hallam University, Sheffield, UK

***Abstract*—The field of reconfigurable swarms of modular robots has achieved a current status of performance that allows applications in diverse fields that are characterized by human support (e.g. exploratory and rescue tasks) or even in human-less environments. The main goal of the EC project REPLICATOR [1] is the development and deployment of a heterogeneous swarm of modular robots that are able to switch autonomously from a swarm of robots, into different organism forms, to reconfigure these forms, and finally to revert to the original swarm mode [2]. To achieve these goals three different types of robot modules have been developed and an extensive suite of embodied distributed cognition methods implemented [3]. Hereby the methodological key aspects address principles of self-organization. In order to tackle our ambitious approach a Grand Challenge has been proposed of autonomous operation of 100 robots for 100 days (100 days, 100 robots). Moreover, a framework coined the SOS-cycle (SOS: Swarm-Organism-Swarm) is developed. It controls the transitions between internal phases that enable the whole system to alternate between different modes mentioned above. This paper describes the vision of the Grand Challenge and the implementation and the results of the different phases of the SOS-cycle.**

I. Introduction

Modular, reconfigurable robotics is a special branch of robotics, that has been studied for over 25 years [4]. This field has been continuously refined [5], [6]. Two different approaches have been investigated. One direction of development has been characterized by the extended use of technical and physical phenomena. An excellent example of this kind of efforts is the self-assembling M-cube robot developed by Rus et. al. [7]. The other direction of development has been geared towards behavioural aspects, such as distributed cognition and sustainable survivability. This contribution addresses the behavioral, cognitive approach. Until now most of these behavioural and cognitive aspects have been studied in isolation. Only nowadays becomes it possible to cross-disciplinary solve the wide diversity of problems at hand, such as, for example, energy balancing in both body and swarm mode, a range of controllers for different gaits distributed over robotic modules, cognitive fusion of various sensor data in swarm mode, etc. Adaptation of autonomous robots for industrial purposes hinges on the level of their autonomy. Robots that survive for 100 days (performing a diversity of tasks) would demonstrate that an autonomous robotic system can operate for a long time without any human intervention indeed. The envisioned scenario of our Grand Challenge starts with a swarm of 100 heterogeneous robot modules that are placed in a previously unknown arena of which certain environmental structures are slowly changing. Figure 1 demonstrates the initial situation where all the different robot modules are in swarm or previously assembled organism mode.

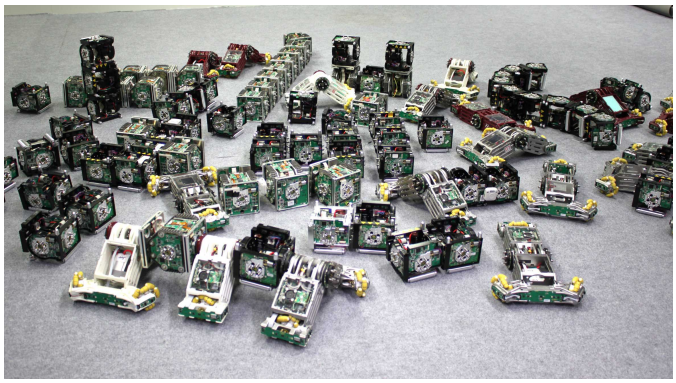


Figure 1: Start-up situation of modular, reconfigurable robots to achieve different tasks of the Grand Challenge

The environmental restrictions are varying in an unpredictable way. Thus, energy resources can change their position and height; therefore if the power sockets are mounted too high (e.g. for an organism to reach it), then the organism must be reconfigured in order to reach the power source (transfer from swarm mode to organism mode) or first to overcome an obstacle. One additional big goal of this challenge is given by the fact that the external conditions are so strong that the robots can only survive when they are aggregated in an organism or disaggregated in a swarm and newly shaped organisms. In this way, we can achieve an enhanced distributed recognition, an extended available affordance and actuation capabilities [8].

To achieve this challenge, robots must be capable to perform diverse important functionalities such as self-aggregation, self-regulation, reconfiguration, energy regulation, self-repair and autonomous locomotion capabilities. The realization of this Grand Challenge has been manifested in a demonstration in an arena with real robots. Therefore, the first task was to

design and produce such a high number of modular robots.

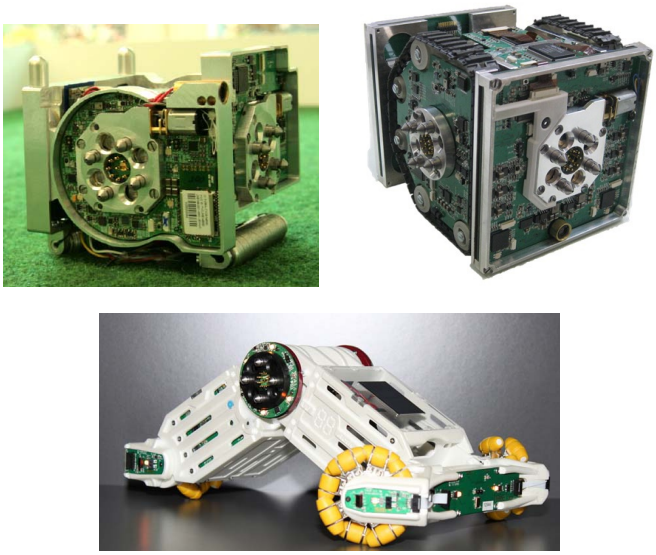


Figure 2: Modular robots: (a) Backbone [9]; (b) Scout [10]; (c) Active Wheel [11].

To achieve our goals, an heterogeneous swarm of robots has been built. Figure 2 shows three different robotic platforms: the Backbone, the Scout, and the Active Wheel, which are on the one hand heterogeneous and differ in their mechanical and electrical properties, however on the other hand are fully compatible through the common docking element, which enables the robots to aggregate into different robot configurations.

The best way to integrate a continuous following of a recurrent goal is to run a cyclic program. For this reason, a survival cycle framework, named S-O-S (swarm - organism -swarm) has been integrated, which controls the behavior of individual robots in a swarm and also in an organism. Robots run through different phases of the cycle and make decisions either to target or to neglect the corresponding state. The SOS-cycle is illustrated in Figure 3. Phases do not necessarily need to be run in a sequential way but rather can dynamically be changed by an active action selection mechanism explained in Section III. This framework allows reactive and adaptive behavior of robots dependent on environmental changes.

The objectives of this paper are first of all to present the developed architecture, which reflects the survival strategy for modular and self-reconfigurable robots, and secondly to explain used technologies and the achieved results. The paper is organized in the following way: In Section II, we introduce the SOS cycle with its individual phases. In Section III, we introduce the individual methodologies and the implementation strategies with results, which helps to achieve the desired degree of autonomy. Finally, Section IV concludes the work.

II. The SOS Cycle

The key difference between reconfigurable and self-reconfigurable robots is the ability to operate fully autonomously without human intervention. A modular robot should be able to operate as a stand-alone robot in a swarm of

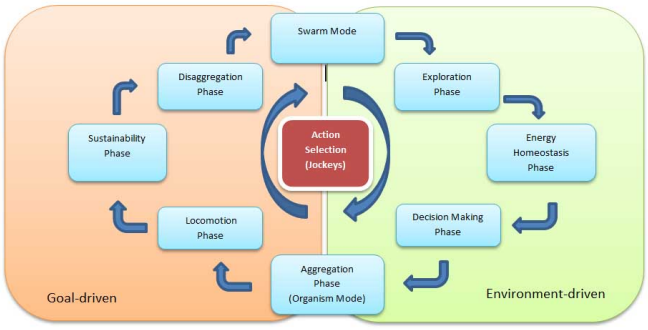


Figure 3: Swarm-Organisms-Swarm (SOS): Phase Transition Diagram of Cognitive Framework in Autonomy Cycle.

homogeneous or heterogeneous robots and should be capable to operate in a collective manner when assembled into a robot organism. The complexity for such a framework rely mainly in the software and the electronics design of a robot. Cognition can be achieved when data from different sensors are fused together enabling transitions from one state to another. In this paper, the realization of such framework has been done using two different software and control modes, namely the swarm and the organism mode. When robots act in a swarm, they use random walk, collision avoidance and are collecting information by exploring the environment. In organism mode, robots start to cooperate and need a guiding framework, which can handle the complexity of the state transitions. In this paper, we present a cycle-based cognitive framework, which enables to switch between swarm and the organism mode in alternating manner.

We distinguish between intrinsic and extrinsic goals of the robots. The major intrinsic task robots are trying to fulfill is to stay alive, e.g. to find power sources and to share this information among the swarm. Only when intrinsic goals are guaranteed, robots should focus on extrinsic tasks, which depend on the applied scenario. One typical scenario to illustrate both modes is the foraging scenario where swarm robots try to detect power sources, however can reach them only when assembling into a three dimensional robot organism. This scenario becomes a part of the Grand Challenge.

In the rest of this section, the main objectives of different phases in the SOS cycle are introduced. The implementation details of those phases are then explained in the following section.

Swarm Phase
In the Swarm Phase modular robots perform random walk with collision avoidance.

Exploration Phase
In the Exploration Phase, robots explore the environment using different sensors such as IR distance sensors, ambient light sensors, laser and camera. These sensors allow to recognize objects in the arena, to distinguish between robots and obstacles and, most importantly, to detect power sources. Robots also perform mapping and navigation tasks. Distributed sensor information is collected and merged by the robots to get the whole map of the operating environment. Navigation system

helps to store the positions of the recognized objects, which are shared between the robots.

Energy Homeostasis Phase
In the Energy Homeostasis Phase of the SOS cycle, the robots are searching for available power sources in the environment. Here, robots develop strategies how to share available power sources between the robots in order to get maximum runtimes of the robots.

Decision Making Phase
After power sources are localized, robots try to balance between intristic and extrinsic activities continuing searching for further energy harvesting or to fulfill the actual goal. In this stage of the survival cycle, robots decide either to stay in swarm mode or, when the power sources are unreachable or the number is limited, to assemble into a robot organism.

Aggregation Phase
When the shape of desired robot organism is determined by self-assembling algorithms, robots start the corresponding self-assembly process aggregating into 3D configuration.

Locomotion Phase
In the Locomotion Phase, robots are planning and generating locomotion gaits or joint trajectories in order to navigate and move towards the desired direction. Robots are able to perform either 2D, 2,5D or 3D locomotion dependent on robot types assembled in the robot organism. Organisms with docked Active Wheels are able to perform energy efficient 2D locomotion even on rough terrains, while robot organisms assembled of Backbones and Scouts are capable to move in 3D by using legged robot structures.

Sustainability Phase
In this phase, robots discover survival strategies for a certain period of time. Dependent on the achieved results, robots decide either to stay in the organism mode or, when they fail to reach the power sources, to re-assemble into another configuration.

Disaggregation Phase
After achieving the goal, robots disaggregate and transit back to the swarm mode giving again the possibility to explore the environment or go into another transient state of the SOS cycle.

Action Selection
The whole framework is controlled by an action selection mechanism, which gives a global control about the scenario. It guides all individual robots as well as controls the data sharing and synchronization activities.

III. Implementation of SOS Cycle

The cognitive framework explained above brings together many different robotics research fields. Each phase of the framework contains diverse strategies and technologies, which together enable autonomous and adaptive robot behaviour. In this section, we describe the technologies used to tackle the complexity of such tasks that can be handled by the robots. The SOS cycle can be split into two main parts: the environment-driven tasks, which handle the subtasks for interaction with the environment, and the goal-driven tasks, which relate to fulfill the targetted goal.

A. Environment-Driven Tasks

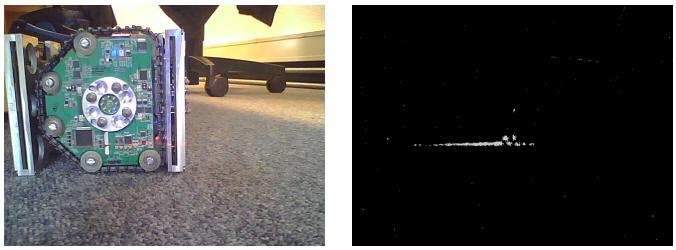
At the beginning of every mission, the robots start as a swarm collecting first of all the information about the environment (Figure 4).

Figure 4: Swarm Phase: Robot performing collision avoidance and random walk.

The capability of recognizing objects enables to determine the current situation and helps to achieve the current goal (Exploration Phase). In this stage, all possible sensors are used in order to get the most reasonable information. Modular robots, which have been developed in the Replicator project are equipped with diverse optical sensors such as IR distance range sensors, cameras and laser. For this reason, different object recognition methods explained below have been developed during the runtime of the project.

Laser-based Exploration

The line-laser on the robots can be mounted horizontally or vertically. Horizontally, its reflections form a feature vector that subsequently can be used in a supervised fashion to classify a range of objects (Figure 5). For example, robots can be distinguished from flat objects such as walls by a large difference in the overall variance of the feature vector. If



(a) (b)

Figure 5: (a) Laser reflecting on another Replicator robot, (b) Difference image of the red channel.

mounted vertically, the laser can be used to detect the height and distance of objects (Figure 6). The horizontal distance between the laser (mounted in the center of the front of the robot) and the camera (mounted at the top-left) makes it possible to detect the distance to an object. The further the distance to the object, the larger the shift on the image sensor

through the pinhole camera. This shift in image coordinates is translated into a distance in world coordinates by a truncated polynomial.

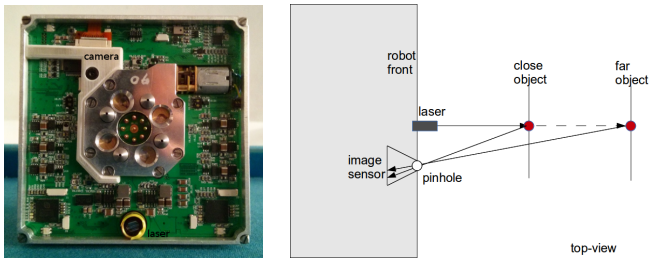


Figure 6: The horizontal distance between laser and camera makes depth perception possible.

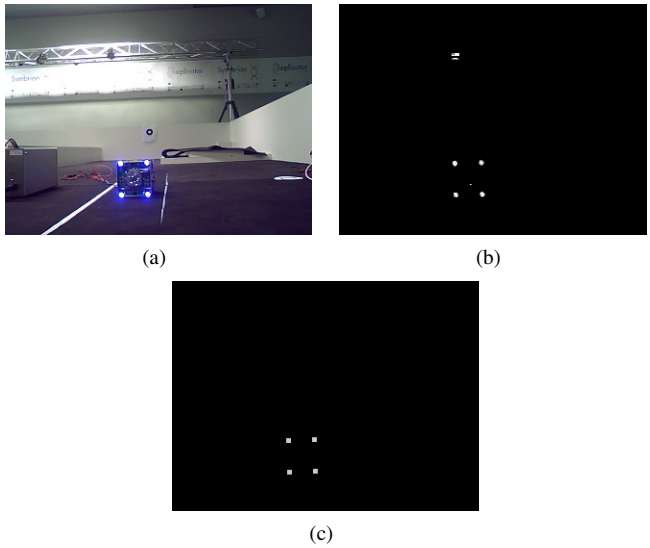
The height of the vertical line can be used to detect the height of the object encountered to distinguish walls (hard to climb) from steps (easy to climb as an organism). A Hough Transform [12] turns out to be too computationally demanding. An efficient filter is used which removes the side margins and implements a low-pass filter to remove noise. A smoothing average across several image samples provides a reliable height estimation of objects.

During laser-based exploration, the robots update the map by sending the detected object description (such as a wall, or a step) over ZigBee to the other robots. A robot knows its own position through the UbiSense indoor positioning system and corrects these absolute coordinates with its own distance and orientation estimate to the object.

Camera Based Object Recognition

All robot platforms are equipped with one or two on-board mobile phone style colour cameras that are connected directly to the main on-board processor. The cameras can acquire images with a resolution of 640x480 pixels (VGA) and a frame rate of approximately 10 fps. For this task, we are using feature extraction algorithms (running on the on-board processor) to detect the presence of other objects or robots in the scene, as well as power sockets, steps in the arena and other features of interest. To detect other robots in the scene at a sufficiently high frame rate, we are exploiting the colour features generated by four LED's that are located on each side of the robots (Figure 7). By grabbing the images in YUV format, we are able to easily separate luminance and chrominance information, therefore identifying the position of the four (switched on) LED's in the image. As we only need to identify the position of three LED's to detect position and orientation of the robot, this solution provides some degree of robustness when one of the LED's is not being recognised or is not lit.

Based on the known geometrical information relative to the position of the four LED's on the robot board, we are able to filter out clutter and other noise in the image (Figure 7) and use the remaining four (or three) blobs to guide the robot in the direction of the target robot up to a distance that is dependent on the optical characteristics of the camera module (i.e. minimum working distance). Below this distance, images go out of focus, therefore LED localisation in the image becomes unreliable.



(a) (b) (c)

Figure 7: Detecting robot's LEDs from on-board camera: (a) Original image; (b) Image after segmentation ; (c) Image after blobs removal

In a similar way, for power socket recognition, we attach coloured markers next to the power socket and run a colour blob detection algorithm to localise the position and orientation of the power socket with respect to the observing robot.

Docking Approach

The ability to identify and reach a docking mechanism of other robot or a charging station is crucial for both the aggregation and the sustainability phases.

To achieve autonomous docking during the aggregation phase, IR-based sensing - including proximity detection, docking alignment detection and local communications circuits - has been developed for all REPLICATOR robots, see [13] for detailed implementation. Figure 8(a) illustrates the average density of an IR beacon measured using a Backbone robot. The area surrounded by the white dashed line represents the region that another robot can detect the signals and behave accordingly, given the triggering threshold value 8. This provides a 90 cm long narrow region, with each side 15 cm from the centre.

In general, the IR docking approach between two robots can be divided into three stages:

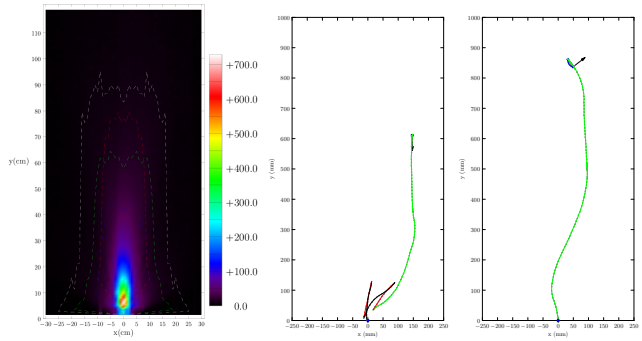
1 Facing – when a robot exploring in the environment detects beacon signals, i.e. at least one of its docking alignment sensor readings reaches certain threshold. It will first execute a manoeuvre to adjust its heading until two docking sensors located on the Front side can detect the signals.

2 Aligning – using two symmetrically placed Front docking sensors, the robots try to move towards the beacon along the central line of the IR beam. i.e., to minimise the difference of readings between two sensors.

3 Tuning – When robots are getting closer, the docking

robot uses extra sensors, e.g. proximity sensor to perform a fine adjustment on its pose. Once two docking units are in good positions, robots initialise the physical locking routine to finalise docking.

However, these three stages can not guarantee the success of the docking due to the noise of sensor inputs or mechanical interference. For instance, the locking bolts prevent two robots moving closer if they are not perfectly aligned. An extra behaviour is therefore added for robot to give up. Normally, this is achieved by moving backward by some distance and then transferring into stage 2 to resume the docking process. The following two conditions decide when a robot needs to give up: 1) the duration that robot enters into stage 3 exceeds certain level and 2) the difference between two sensors value exceeds some threshold. Figure 8(b) – 8(c) show some selected trajectories of docking procedure for a Scout robot.



(a) (b) (c)

Figure 8: (a) Coverage of IR docking beacon. Beacon is located at (0,0). White dashed line indicating the edge of detectable region. (b),(c) Trajectory of IR Docking for Scout robot.

Unlike in the aggregation phase, where the recruited robot actively indicates its position by turning on its IR beacon, the charging dock is entirely passive. Therefore, the aforementioned IR-based approach cannot be used and we have decided to base the docking to a charging station on a computer vision technique. Each charging station is tagged with a black and white elliptical pattern, which allows to determine its position and orientation by means of a fast algorithm suitable for a low-power processor [14]. The algorithm can process the image with speeds exceeding the camera frame rate and calculate the motor inputs solely from the information provided by the robot's on-board camera.

Once a robot decides that the charging station is reachable, the relative position of the robot to the charging station is determined and a sequence of controllers is executed. These controllers first guide the robot close to the station, position it in front of the docking mechanism and align the robot heading with the dock orientation. After that, the actual docking is performed by means of intuitive visual servoing technique (Figure 9). Detailed information on the docking controller can be found in [15].

Visual Based Navigation and Mapping

During the exploration phase, the robots can use the visual dock-detection algorithm not only to estimate positions of

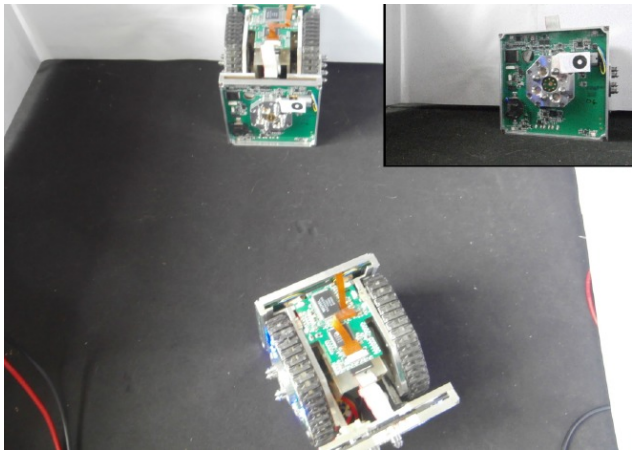


Figure 9: Replicator robot during visual docking

the charging stations, but also for their own self-localization. This ability allows each robot to run an EKF-based SLAM method and build a local map of the surrounding charging stations [16]. This map can be later reused in case the robot needs to reach a particular place in its operational environment. Problems arising from poor EKF-SLAM scalability and low computational power of the robots can be avoided by employing a computationally efficient bearing-only navigation method proposed in [17].

B. Goal-Driven Tasks

The second part of the SOS cycle contains all tasks that are related to the achievement of the goals of the presented scenario. These tasks handle the challenges for navigation, control and survival strategy for robot organisms.

3D Locomotion

Modular robot organisms benefit from a high number of DOFs including at least one actuator in every module. This feature makes the robot assembly flexible and adaptable to uncertain terrains, however requires sophisticated and smart control systems. There exist diverse methodologies of control for hyper redundant systems following bio-inspired or classical paradigms see e.g. [18], [19] or [20]. In both cases, the complexity grows rapidly with the increased number of DOFs. In this paper, we introduce a model based approach for generation of motion equations followed and also model based control strategies. The concept is illustrated in Figure 10 and has two main branches: the gait generation and the model generation.

The adaptive gait generator is responsible for the generation of trajectories of each individual module. Two different methods are used for serial and for branched multi-legged robot organisms. Serial robot configurations are better controlled by periodical gaits, whereas other configurations require more sophisticated approaches. One possibility for complex robot structures is to use self-organized pattern generators. Different periodical and self-organized approaches for gait generation are explained in detail in [22] and in [23].

The gait for locomotion is often not sufficient to move a complex robot assembly in unpredictable and uncertain environments because of high dynamic influences. To handle

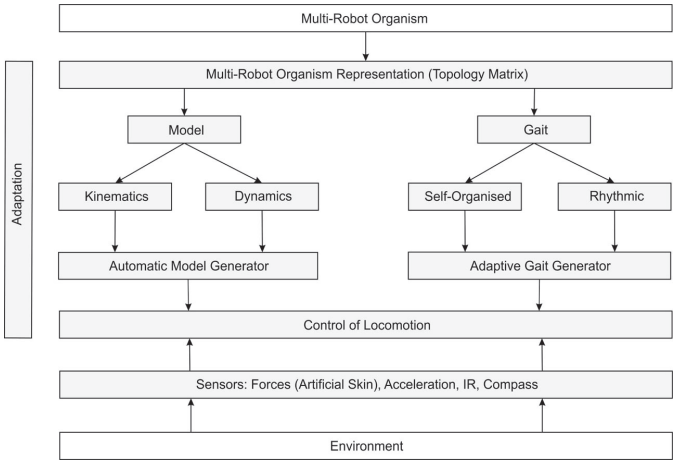


Figure 10: Framework for 3D locomotion [21].

the dynamics, a geometrical modelling approach [24] is used ending up with a set of motion equations of the form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + N(q) = \tau \quad (1)$$

where $M(q)$ is the mass matrix; $C(q,\dot{q})$ describes the Coriolis and centrifugal accelerations and $N(q)$ represents the gravitational and external forces.

Once the motion equations are generated, diverse control strategies known from the field of control theory can be applied. In our approach, we use an exact linearization and

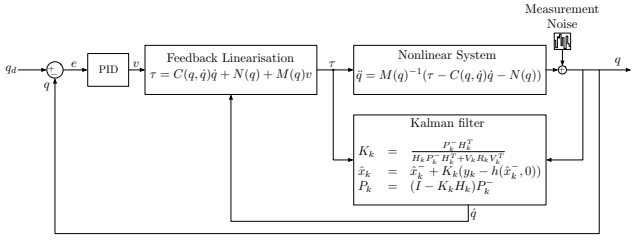


Figure 11: Schematic of system structure with computed torque, EKF and PID control [24].

an Extended Kalman Filter (EKF) [25] as not all the states of the robot can be measured but have to be estimated, instead. Figure 11 shows the complete control strategy.

The proposed methods have been first developed and evaluated in MATLAB and finally have been ported to a C++ based framework running on the modular robots [26]. Figure 12 shows an example robot configuration, which was used in the final demonstration of the Grand Challenge.

High-level motion planning

To achieve a desired goal, e.g. to approach a power socket, pure locomotion generation cannot ensure reaching the goal, as it does not consider the overall situation in the environment. In such a case, motion planning of the robot is required. Motion planning of such a system with many degrees of freedom can be solved by sampling-based methods, that create a plan by

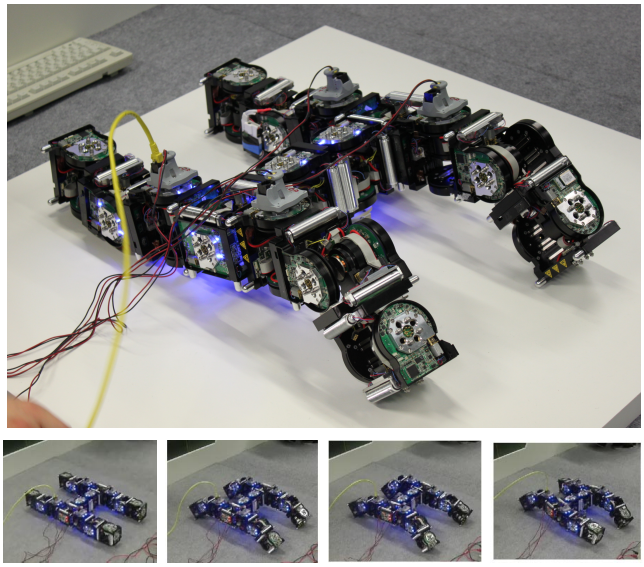


Figure 12: Implementation on modular robots using Backbones.

randomly sampling of robot's configuration space. As the complexity of the sampling-based planners can grow with the number of actuators in the organism, the naive implementations of sampling-based planners cannot be utilized [27]. To decrease the complexity of the planning task, we propose the concept of motion planning with motion primitives [28]. In this approach, a robot organism is equipped with several motion primitives, such as 'move-forward' or 'step-up'. These primitives can be generated by many approaches like Central Pattern Generators (CPGs) or the 3D locomotion generator described above. The primitives are considered as atomic actions in a high-level motion planner, which tries to combine the motion primitives to achieve the goal. Therefore, the motion planner does not need to handle all the actuators, which significantly reduces the search space and, consequently, speeds up the planning process. An example of a high-level motion plan is depicted on Fig 13.

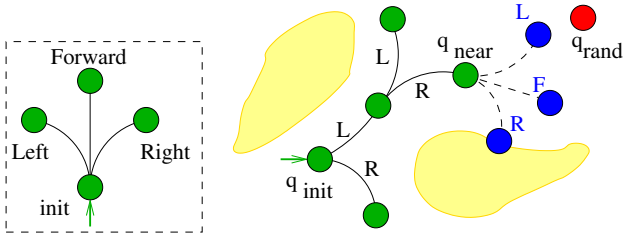


Figure 13: Example of a motion plan built using RRT-MP (Rapidly Exploring Random Tree with Motion Primitives) [28] motion planner for a robot equipped with three motion primitives ('go-left', 'go-right' and 'go-forward'). The green nodes represent feasible configurations, the blue nodes are possible actions to approach the red goal.

The sampling-based motion planning relies on a motion model, that computes motion of a robot after a control signal is applied. In the early version of our work, a physical

simulator [29] was used. The simulation can provide a detailed motion model (it simulates motion of all modules), but it is computationally demanding and it cannot be run on board. Therefore, a Simplified Motion Model (SMM) can be used [30] instead of the simulation. The idea of the SMM is to describe only changes of the robot's state after a motion primitive is applied without computing intermediate states. This can be described as a combination of two rotations and a translation. Such a model has only three parameters, that can be easily estimated from the simulations or even on-line based on the robot's real performance. The SMM can be easily programmed and its evaluation is fast, which allows to run the whole motion planning on board without the necessity to employ an external computation facility. The motion planning in an arena of size 5x5m takes approx. 5 seconds running on board. The high-level

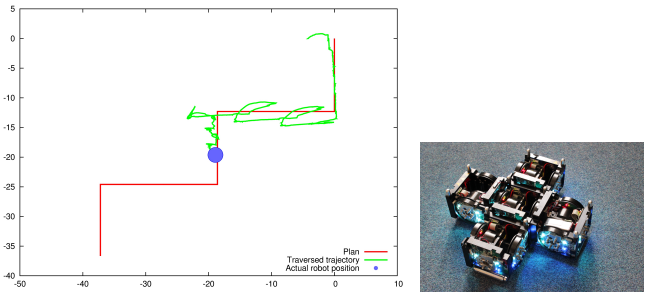


Figure 14: Navigation of a small Cross organism along a planned path.

planner provides a plan as a sequence of motion primitives together with their duration. Such a plan can be easily executed by switching the locomotion generators used to model the individual primitives. During the motion, the robot's position is tracked by the global localization. When a robot deviates from the planned path or when the environment changes, a new plan can be immediately generated. An example of a navigation along a planned path is depicted on Figure 14.

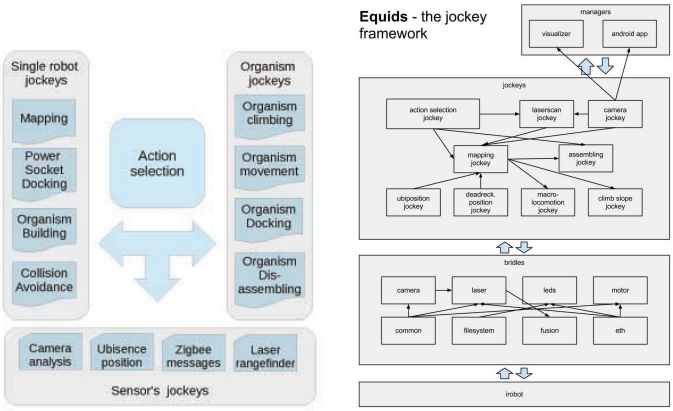
Jockey Framework

The overall software framework has two main capabilities. First, knowledge sharing between the robots in swarm, organism mode, or any transition in between. Second, smooth selection from one controller to the next depending on locally or globally defined states.

One of the challenges in modular robotics is acquisition, maintenance and usage of the knowledge of the operational environment. The main issue is that the knowledge is distributed across the robotic swarm and it is unfeasible to assume that the individual robots can possess the complete information of the environment. Moreover, the sensor range of the individual robots is limited and the environment might lack features which allow reliable robot self-localization and precise mapping. Therefore, representing the knowledge of the environment in a form of a consistent metric map is unfeasible.

To deal with the above issues, we have proposed to represent the environment in the form of a directed multigraph. While the vertices of the multigraph represent certain salient places (e.g. a charging station or a step), the edges represent navigation algorithms that allow to move between these places. The fact that switching an algorithm n_0 at a place v_0 brings the robot to place v_1 with probability p is represented by an edge associated with the probability p and necessary knowledge for the algorithm n_0. The proposed framework does not impose any restriction of this knowledge representation - it can be a landmark map for a traditional navigation method, configuration of a neural network which guides the robot or a sensorymotor pattern indicating the correct direction of movement. To emphasize that the controllers represented by the edges actually drive the robot around, we have decided to call them 'jockeys'. Creation and maintenance of the above environment representation is based on the notion of subjective logic that allows for efficient reasoning about graphs representing uncertain spatial knowledge [31]. The aforementioned reasoning allows to merge, distribute and compress the spatial knowledge of any group of robots that can communicate with each other. Therefore, if a group of robots aggregates to form an organism, they not only increase their physical ability, but also share the spatial knowledge they acquired so far.

The exact layout of all the different jockeys and their interactions would take us too far. The most important facets are visualized in Figure 15.



(a) (b)

Figure 15: (a) List of implemented jockeys, (b) Example of Jockey dependencies of targeted Grand Challenge scenario.

The jockey framework is, on a technical level, quite advanced. Different from alternatives [32], [33] it is meant to run embedded on the robot itself (and is hence very lightweight) and is tailored to a decentralised setting: peer-to-peer communication is no problem. The jockeys communicate with each other using multiple communication channels, Ethernet between connected robots, wireless Ethernet and ZigBee between physically unconnected robots. Each robot runs a controller that defines its local state and allows it to switch from one type of behaviour to the next. When certain local conditions are met, a robot can also switch to a behaviour that characterizes another mode and, for example, starts recruiting other robots for an assembly task. There is no global controller that defines when the robot swarm as a whole switches its behaviour. This is done on a local level. The jockey framework is one of the first frameworks that allows for such a decentralised approach.

IV. Conclusion

The scientific impact of the set of one hundred robots is not only their ability to react to changing environmental conditions by a catalog of self-organization but also to be completely aware for the reasons to change the global behavior of the swarm, or to modify the organism shape or to repair defect part by themselves. This ability is not achieved by ingenious engineering of different mainly physical/chemical operating components (like molecules) but by a deliberate use of capabilities and restrictions. The technical impact of our approach is the design of reliable and stable robots that can operate over a long period without human support. In the far future this might also revolutionize the design methods of Mechanical Engineering.

Acknowledgment

The "REPLICATOR" project is funded by the European Commission within the work programme "Cognitive Systems, Interaction, Robotics" under the grant agreement no. 216240. The authors would like to acknowledge the contribution of all REPLICATOR partners. In particular, we are indebted to Sergej Popesku, Florian Schlachter, Stefano Marrazza, Jens Liedke and Timo Koch for the hardware and software design of modular robots.

References

[1] "REPLICATOR: Robotic Evolutionary Self-Programming and Self-Assembling Organisms, 7th Framework Programme Project No FP7-ICT-2007.2.1," 2008-2013.

[2] P. Levi and S. Kernbach, Eds., *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer-Verlag, 2010.

[3] R. Pfeifer and C. Scheier, *Understanding Intelligence*, ser. Bradford Books. MIT Press, 2001. [Online]. Available: http://books.google.de/books?id=iIv6-rxTCkoC

[4] T. Fukuda, H. Hosokai, Y. Kawauchi, and M. Buss, "Dynamically Reconfigurable Robotic System (DRRS) (System Configuration and Implementation as CEBOT)," in *Proceedings of 5th Int'l Symp. of Robotics Research*, 1989.

[5] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. Chirikjian, "Modular Self-Reconfigurable Robot Systems," *IEEE Robotics & Automation Magazine, Magazine*, pp. 43–52, 2007.

[6] R. Groß and M. Dorigo, "Self-Assembly at the Macroscopic Scale," pp. 1490–52, 2008.

[7] J. Dorrier, "MITs M-Blocks: A New Class Of Robot Cubes That Self Assemble," *IEEE Spectrum Magazine*, Vol. October, pp. 43–53, 2013.

[8] S. Kernbach, O. Scholz, K. Harada, S. Popesku, J. Liedke, R. Humza, W. Liu, F. Caparrelli, J. Jemai, J. Havlik, E. Meister, and P. Levi, "Multi-Robot Organisms: State of the Art," *CoRR*, Vol. abs/1108.5543, 2011.

[9] J. Liedke and R. Matthias and L. Winkler and H. Wörn, "The Collective Self-Reconfigurable Modular Organism (CoSMO)," in *Proceedings of IEEE/ASME Inter- national Conference on Advanced Intelligent Mechatronics (AIM2013)*, 2013.

[10] S. Russo, K. Harada, T. Ranzani, L. Manfredi, C. Stefanini, A. Menciassi, and P. Dario, "Design of a Robotic Module for Autonomous Exploration and Multimode Locomotion," *Mechatronics, IEEE/ASME Transactions on*, Vol. PP, No. 99, pp. 1–10, 2012.

[11] S. Popesku, E. Meister, F. Schlachter, and P. Levi, "Active Wheel - An Autonomous Modular Robot." in *Proceedings of International Conference on Robotics, Automation and Mechatronics (RAM) (accepted)*, Manila, Philippines, 2013.

[12] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern recognition*, Vol. 13, No. 2, pp. 111–122, 1981.

[13] W. Liu and A. Winfield, "Implementation of an IR approach for autonomous docking in a self-configurable robotics system," in *Proceedings of Towards Autonomous Robotic Systems.*, 2009, pp. 251–258.

[14] T. Krajník, M. Nitsche, *et al.*, "External Localization System for Mobile Robotics," in *Proceedings of 2013 IEEE International Conference on Advanced Robotic*. Montevideo: IEEE, 2013, To appear.

[15] V. Šalanský and T. Krajník, "Docking procedure for the Replicator project," Master's thesis, Czech Technical University in Prague, 2013.

[16] R. Pěnička and T. Krajník, "Acquisition and representation of spatial knowledge in the REPLICATOR project," Master's thesis, Czech Technical University in Prague, 2013.

[17] T. Krajník *et al.*, "Simple yet stable bearing-only navigation," *Journal of Field Robotics*, Vol. 27, No. 5, pp. 511–533, 2010.

[18] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, Vol. 315, No. 5817, pp. 1416–1420, 2007.

[19] C. Liu, Q. Chen, and D. Wang, "CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots." *IEEE Transaction on Systems, Man, and Cybernetics.*, Vol. 41, No. 3, pp. 867–80, 2011.

[20] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices." *Trans. ASME J. Applied Mechanics*, Vol. 22, pp. 215–221, 1995.

[21] E. Meister, "Adaptive Locomotion of Modular Recofigurable Robotic Systems," Ph.D. dissertation, University of Stuttgart, Germany, 2013.

[22] E. Meister, S. Stepanenko, and S. Kernbach, "Adaptive Locomotion of Multibody Snake-like Robot," in *Proceedings of Multibody Dynamics 2011, ECCOMAS Thematic Conference*, P. F. J.C. Samin, Ed., Brussels, Belgium, 2011.

[23] S. Kernbach, M. E., F. Schlachter, and O. Kernbach, "Adaptation and Self-adaptation of Developmental Multi-Robot Systems," *International Journal On Advances in Intelligent Systems*, Vol. 3, pp. 121–140, 2010.

[24] E. Meister and A. Gutenkunst, "Dynamics and Control of Modular Self-Reconfigurable Robotic Systems," *International Journal On Advances in Intelligent Systems*, Vol. 6, No. 1 & 2, pp. 121–140, June 2013.

[25] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," 1995.

[26] S. Kernbach, F. Schlachter, R. Humza, J. Liedke, S. Popesku, S. Russo, R. Matthias, C. Schwarzer, B. Girault, and . P. Alschbach, "Heterogeneity for Increasing Performance and Reliability of Self-Reconfigurable Multi-Robot Organisms." in *In Proceedings IROS-11*, 2011.

[27] V. Vonásek, K. Košnar, and L. Přeučil, "Motion planning of self-reconfigurable modular robots using Rapidly Exploring Random Trees," in *TAROS*, 2012.

[28] V. Vonásek, M. Saska, K. Košnar, and L. Přeučil, "Global motion planning for modular robots with local motion primitives," in *ICRA*, 2013.

[29] L. Winkler, V. Vonásek, H. Worn, and L. Přeučil, "Robot3D - A Simulator for Mobile Modular Self-Reconfigurable Robots," in *IEEE International Conference on Multisensor Fusion and Information Integration*, 2012.

[30] V. Vonásek, L. Winkler, J. Liedke, M. Saska, K. Košnar, and L. Přeučil, "Fast on-board motion planning for modular robots," in *ICRA*, 2014, Accepted.

[31] K. Košnar, T. Krajník, V. Vonásek, and L. Přeučil, "LaMa - Large Maps Framework," in *Proceedings of Workshop on Field Robotics, Civilian-European Robot Trial 2009*. Oulu: University of Oulu, 2009, pp. 9–16.

[32] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: yet another robot platform," *International Journal on Advanced Robotics Systems*, Vol. 3, No. 1, pp. 43–48, 2006.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, Vol. 3, No. 3.2, 2009.