Swarm-Bot: a New Distributed Robotic Concept

FRANCESCO MONDADA, GIOVANNI C. PETTINARO, ANDRE GUIGNARD, IVO W. KWEE, DARIO FLOREANO, JEAN-LOUIS DENEUBOURG, STEFANO NOLFI, LUCA MARIA GAMBARDELLA AND MARCO DORIGO

Autonomous Systems Lab (LSA), EPFL-STI-I2S, Lausanne, Switzerland
francesco.mondada@epfl.ch
andre.guignard@epfl.ch
dario.floreano@epfl.ch
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), USI/SUPSI, Manno (Lugano), Switzerland
giovanni@idsia.ch
ivo@idsia.ch
luca@idsia.ch
Center for Nonlinear Phenomena and Complex Systems (CENOLI) - Université Libre de Bruxelles, Belgium
jldeneub@ulb.ac.be
Institute of Cognitive Sciences and Technologies - CNR, Roma, Italy
nolfi@www.ip.rm.cnr.it
Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle (IRIDIA) - Université Libre de Bruxelles, Belgium
mdorigo@ulb.ac.be

Abstract. The swarm intelligence paradigm has proven to have very interesting properties such as robustness, flexibility and ability to solve complex problems exploiting parallelism and self-organization. Several robotics implementations of this paradigm confirm that these properties can be exploited for the control of a population of physically independent mobile robots.

The work presented here introduces a new robotic concept called *swarm-bot* in which the collective interaction exploited by the swarm intelligence mechanism goes beyond the control layer and is extended to the physical level. This implies the addition of new mechanical functionalities on the single robot, together with new electronics and software to manage it. These new functionalities, even if not directly related to mobility and navigation, allow to address complex mobile robotics problems, such as extreme all-terrain exploration.

The work shows also how this new concept is investigated using a simulation tool (swarmbot3d) specifically developed for quickly designing and evaluating new control algorithms. Experimental work shows how the simulated detailed representation of one *s-bot* has been calibrated to match the behaviour of the real robot.

Keywords: Swarm Intelligence, Swarm Robotics, Distributed Robotics, Reconfigurable Robotics, Collective Robotics, Physics-Based Simulation.

1. Introduction

Applications like semi-automatic space exploration (Visentin et al., 2001), rescue (Casper et al., 2000), or underwater exploration (Ayers et al., 1998) need robust and flexible robotic systems. Most of these applications require systems combining the following three basic characteristics:

Robustness. Unstable, very complex or extreme environments require robustness to severe hardware failures.

Versatility. The complexity of the task needs versatility in hardware shape and functionality. The robot has to perform well in very different terrains and in very different tasks such as displacement, exploration or object transportation.

All Terrain Navigation. Complex unstructured environments such as distant planets or catastrophic environments need a very flexible and efficient all-terrain navigation.

The SWARM-BOTS project[1] aims at combining swarm intelligence (Bonabeau et al., 1999) and physical self-assembling features to provide the above mentioned characteristics to a group of 35 robots.

The swarm-bot robot concept, as well as the hardware implementation, have been developed in parallel to a simulator. This last is intended to provide the following supporting functionalities:

- the accurate prediction of both kinematics and dynamics of a *swarm-bot* in 3D;
- the evaluation of hardware design options for different components;
- the design of *swarm-bot* experiments in 3D worlds; and
- the efficient investigation of different distributed control algorithms.

The work presented here reports the hardware and software development carried out from October 2001. It is expected that a group of 35 real robots are going to be available by the Fall of 2004. Currently, two single robot prototypes are available. Preliminary control tests on a group of more than two robots (see the companion paper in this issue (Dorigo et al., 2004)) are for the moment possible only within the simulation environment (swarmbot3d).

The next section presents the *swarm-bot* concept in more details and places it into its research context (Section 2). The physical hardware implementation of a *swarm-bot* component (*s-bot*) is illustrated in Section 3. The swarmbot3d simulation environment is discussed in Section 4. Section 5 is dedicated to the experimental comparison between simulated and real *s-bots* while final conclusions are drawn in Section 6.

2. Concept and Related Work

The objective of the SWARM-BOTS project is to study a novel approach to the design, hardware implementation, test, and use of a self-assembling, self-organizing, metamorphic robotic system called *swarm-bot*. This approach finds its theoretical roots in recent studies in the field of swarm intelligence, that is, in studies exploiting the self-organizing and self-assembling capabilities shown by social insects and by some other animal societies (Bonabeau et al., 1999).

An important part of the project consists in the physical construction of at least one *swarm-bot*, that is, a self-assembling and self-organizing robot colony composed of a number (30-35) of smaller devices, called s-bots (Figure 1). Each *s-bot* is a fully autonomous mobile robot capable of performing basic tasks such as autonomous navigation, perception of the environment and

[1] *Swarm-Bots* is a European IST-FET (Future and Emerging Technologies) project, grant IST-2000-31010, for more details see `http://www.swarm-bots.org`.

Figure 1. A graphic visualization of the *s-bot* concept. The main body (turret) is equipped with passive and active gripping facilities, sensors and electronics. The lower body (traction system) is equipped with tracks and it hosts the batteries. The diameter of the main body is 116 mm.

grasping of objects. In addition to these features, one *s-bot* is able to communicate with other *s-bots* and physically connect to them in flexible ways, thus forming a so-called *swarm-bot*. Such a robotic entity is able to perform tasks in which a single *s-bot* has major problems, such as exploration, navigation, and transportation of heavy objects on very rough terrain (see Figure 2 for an example). This hardware structure is combined with a distributed adaptive control architecture loosely inspired upon ant colony behaviors (Dorigo et al., 2004).

The final goal of the SWARM-BOTS project is illustrated by a scenario describing the type of operation that this novel robotic concept aims at achieving. This scenario consists in transporting a very heavy object from its initial location to a target defined by a light. The light cannot be seen from the area where the object is initially placed and there are several possible paths for the transport. These paths have different lengths and include large obstacles and holes. The scenario itself is split in four stages. During the first stage a group of *s-bots* searches for a heavy object,

Figure 2. Graphic visualization of how the rigid gripper can be used to connect in a secure way *s-bots* among themselves so to form chains for overcoming large obstacles or holes.

grasps it and starts moving it in a collective way. In parallel, another group of *s-bots* disperses in the environment to search for the goal. In the second phase the *s-bots* create a path linking initial and goal positions for the object. In the third and fourth phases, the *s-bots* transporting the object have to move over a hole and through a narrow passageway by reconfiguring their position around the object.

This scenario emphasizes several key features of the *swarm-bot* concept, and particularly the three aspects mentioned in the introduction (Section 1), that is, robustness, versatility, and rough terrain navigation. This concept would therefore be well adapted for exploration in extreme rough terrain. The collective robustness and the self-assembling versatility can be used to climb obstacles and transport objects also in situations where a single robot acting alone could not succeed. This gives a *swarm-bot* a clear advantage over existing collective robotic systems in rough terrain conditions. Additionally, distributed hardware and control provide strong robustness to failures, which is an advantage over both classic rovers and self-reconfigurable robots, that often have a centralized control (Kamimura et al., 2001; Yim et al., 2002). Even if a *swarm-bot* concept cannot be used to form complex 3D structures, it fits well rough terrain situations and can perform well in search tasks where dynamic assembly and disbanding are required.

In the next sections the details of the *swarm-bot* concept are presented within the research context of exploration in rough terrain conditions. The description is structured following the three main features needed by this type of robots: robustness (Section 2.1), versatility (Section 2.2), and rough terrain navigation (Section 2.3). In each section, a comparison is drawn between the *swarm-bot* concept and the related state-of-the-art robots.

2.1. Robustness to Hardware Failures

The problem of robustness to physical damages plays a crucial role in unstructured and unstable environments, such as those found in post-catastrophic situations or space exploration. Large obstacles, holes in the ground, unstable hindrance, fire, explosions, water, chemicals or other dangerous agents can cause damages to a robotic system. In order to ensure the most efficient task execution, a system has to be fault tolerant and ensure operation even if a large part of it, such as half of the hardware, is lost.

2.1.1. State of the Art. A widely used technique to overcome hardware failures is redundancy. Most of the literature on fault tolerant systems deals with minor failures that can be corrected with a robust control or with systems which have intrinsic redundancy, like distributed communication networks. A typical example exploiting intrinsic redundancy is the failure of a node in a communication network. In this case the system, if well controlled, can continue to operate using the remaining working parts. To face this type of partial failure, which is the most common in engineering systems, the main design effort has to be placed in the control part of the system (Stengel, 1991). An efficient fault tolerant control is based on failure detection and correction. Both of them need a major design effort and an accurate model of the system. To correct major failures, additional and specific hardware redundancy becomes necessary.

In case of exploration in extreme environments, hardware failures can be frequent and major. Here robust control is not anymore sufficient and redundancy has to be introduced also at the hardware level, building in this way multi-robot systems.

Most of the research done in this direction is known under the name of *collective robotics* and represents a very active field. An overview can be found for instance in a survey by Parker et al. (Parker et al., 2000). A major focus of this research community is distributed control, and there is little research on exploiting the collective self-organization at the hardware level. The main motivation of collective robotics research is the coordination of several systems (Gerkey and Matarić, 2002; Agassounon et al., 2001; Melhuish, 1999; Flocchini et al., 2000) and the robustness that can be achieved by the redundancy of the whole system (Parker, 1998; Goldberg and Matarić, 2002; Fukuda et al., 1999). An increasing number of applications, such as space robotic missions (Chien et al., 2000; Earon et al., 2001) where there is a strong advantage in obtaining a more robust system, plan to exploit this type of information processing.

Hardware modularity and redundancy can also be found in the field of *self-reconfigurable robots*. There the research activity is complementary to that of collective robotics and is mainly focussed on hardware modularity with relatively little research on autonomous perception and action in the environment.

Pioneering examples of self-reconfigurable robots are MTRAN (Kamimura et al., 2001) and PolyBot (Duff et al., 2001). An overview of existing systems and characteristics can be found in the work of Kamimura et al. (Kamimura et al., 2001), or in the work of Yim et al. (Yim et al., 2002)). MTRAN and PolyBot use both a large number of simple modules, they both have been physically implemented, and they both can self-reconfigure. Despite their very good hardware flexibility, both MTRAN and PolyBot have been designed with a centralized control perspective, which, in comparison with the decentralized ones, shows reduced robustness to failures. The latest articles on these two research works show that MTRAN is keeping the centralized control approach (Kurokawa et al., 2003) while the PolyBot team is working on new decentralized approaches under the name of Phase Automata (Zhang et al., 2003).

The first 3D self-reconfigurable robot with decentralized control has been the CONRO hardware (Castano et al., 2000) which runs the decentralized control developed by Støy et al. (Støy et al., 2002) or the one developed by Salemi et al. (Salemi et al., 2001). These controllers allow the robot's hardware modules to change their relative position while the system is running. During this dynamic change, each involved module re-adapts autonomously its behavioral role in the system. Although this demonstrates software robustness towards structure modifications and failures, automatic hardware failure correction is not yet implemented and would require a major redesign effort. Hardware failure detection and correction are in fact known to be hard to implement in a reliable way (Blanke et al., 1997).

2.1.2. *Swarm-bot* Robustness. Robustness is ensured within the SWARM-BOTS project by distributed hardware and control. Each *s-bot* is a simple but fully autonomous unit capable of displacement, sensing and acting based on local information and decisions. This is a clear distinction from self-reconfigurable robots, where each unit has no mobility, very limited sensing capabilities and acts often under the control of a central unit. The self-assembling ability of the *swarm-bot* is added on top of the *s-bots*, enabling a swarm behavior at the level of the physically connected *swarm-bot* system. The global task execution is obtained by the exploitation of robot-robot and robot-environment properties, without centralized planning and control. Both the *swarm-bot* control strategy and the distributed hardware ensure good robustness to hardware failures.

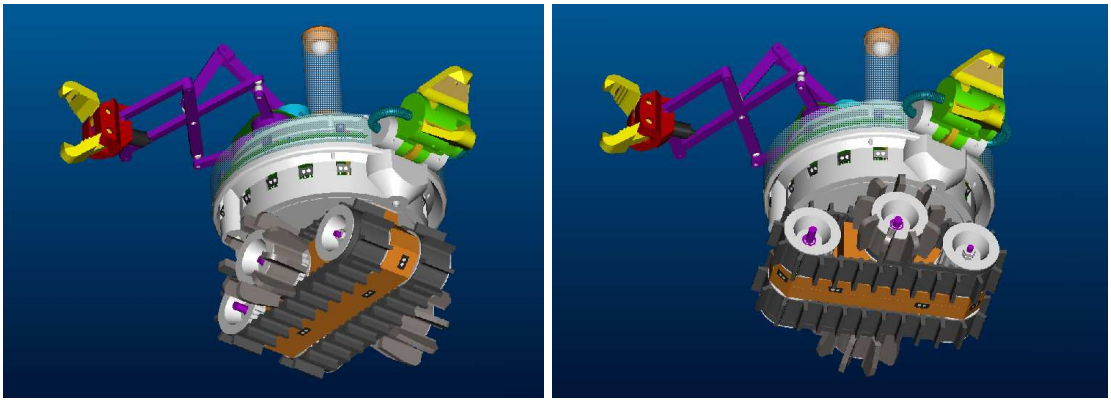


Figure 3. Bottom view of the *s-bot* robot. A differential treels© drive ensure the displacement of the *s-bot*. The motor base with the treels© can be oriented independently of the main body.

As mentioned above, the robustness of the concept is based on the distribution of the task over a group of *s-bots*, each of them able of autonomous displacement, sensing, acting and control.

The mobility of the system is ensured by a combination of two tracks and two wheels, as illustrated in Figure 3. Each track is connected to the wheel of the same side and it is controlled by an independent motor. Wheel and track on the same side are driven by the same motor, building a differential drive system controlled by two motors. This combination of tracks and wheels was labelled *Differential Treels© Drive*[2]. Such a combination has two advantages. First, it allows a more efficient rotation on the spot due to the larger diameter and position of the wheels. Second, it gives to the traction system a shape close to the cylindrical one of the main body (turret), avoiding in this way the typical rectangular shape of simple tracks and thus making navigation simpler.

The differential treels drive allows each *s-bot* to move in moderately rough terrains[3], while more complex situations are handled by *swarm-bot* configurations.

The motor base with the treels© can rotate with respect to the main body by means of a motorized axis, as illustrated in Figure 3. This ensures an independent movement of the upper part where the sensors and the physical connections to other robots are located.

Each *s-bot* is equipped with sensors necessary for navigation, such as infrared proximity sensors, light and humidity sensors, accelerometers and incremental encoders on each degree of freedom. In addition, each robot is equipped with sensors and communication devices to detect and communicate with other *s-bots*, such as an omni-directional camera, color LEDs all around the robot, local color detectors and sound emitters and receivers. In addition to a large number of sensors for detection of the environment, several sensors provide each *s-bot* with information about physical contacts, efforts, and reactions at the interconnection joints with other *s-bots*. These include torque sensors on most joints as well as traction sensors on the connection belt.

Research on social insects (Camazine et al., 2001) suggests that collective robotics could benefit from multi-range and multi-modal sensing in order to perceive and exchange signals at multiple levels and in several circumstances. Because of this, as well as for more practical reasons of

[2] Treels is a contraction of TRacks and whEELS

[3] Obstacles no more than 3–4 cm high, 30% slope, and gaps not larger than 3–4 cm.

Figure 4. The environment in rescue operation is composed of obstacles of very different size and shape, including wires, walls, tubes and gaps.

interference, infrared proximity (active) sensors mainly have a very short range. Sound has instead a much longer range span. The camera, which is a passive sensor, is thought to be used both for long and short range sensing, depending on the features extracted from the image.

The control architecture of a *swarm-bot* consists of distributed algorithms based on local information and simple self-organization rules inspired upon ant colony behaviors (Şahin et al., 2002; Dorigo et al., 2004). Although this type of control algorithm does not need much computational power, the large number of sensors and degrees of freedom requires fast pre-processing and efficient control. Therefore *s-bots* are equipped with a network of several processors, each of them responsible for a particular sub-task in the system. The main processor is in charge of the management of the entire system and of the communication with a base station for monitoring purposes. This processor runs a standard Linux OS allowing in this way the use of standard development tools, such as compilers and debuggers, as well as an easy porting of custom made robotic development tools, such as specific control libraries or monitoring tools. *S-bots* are also equipped with a radio link to a base station just for monitoring purposes (not for control).

2.2. Versatility

The environment where a robot has to move about in applications of extreme exploration includes a large number of obstacles anywhere and of any kind: from fissures to deep vertical holes, from small pebbles to large rocks, from wires to walls, from long tubes to compact blocks, etc. (*e.g.*, Figure 4). It may happen, for instance, that robots need to be introduced into small holes, and once inside they need to overcome large gaps, to descend a vertical duct ending in a large void, and finally to pass in other narrow passageways. Robots designed to cope with only one or two of these features are surely challenged by the other ones. It may also happen that a mission starts with a goal and ends up with another one. For instance, a mission can start with an exploratory phase and finish with a transportation task. To be successful, a robot has therefore to be very versatile, that is, capable of dynamically changing shape and control functionality depending on the situation it faces.

2.2.1. State of the Art. Modularity is a widely used technique to ensure versatility. At the control level, modularity is often implemented by distributed approaches in the structure of the control system (Callen, 1998; Zhang et al., 2001) or in the control process itself, as in collective robotics. At the hardware level, modularity and versatility are clearly represented by the field of self-reconfigurable robots.

Modularity provides versatility at several levels in collective robotics. A physically distributed system allows distributed sensing, acting and processing. Simultaneous distributed sensing delivers high flexibility in placing the sensors according to the configuration of the search space, thus improving search efficiency. A good example of this type of situation is given by Hayes et al. (Hayes et al., 2001) where a very difficult search task, plume tracing, is performed using a swarm of robots equipped with odor sensors.

Distributed acting allows versatility in transport tasks (Kube and Bonabeau, 2000; Groß and Dorigo, 2004), exploiting the possibility to change the number of agents involved depending on the effort needed. Sorting is another example where multiple agents can improve versatility of the system (Martinoli et al., 1999; Wilson et al., 2004). Transport (Detrain and Deneubourg, 1997), sorting (Deneubourg et al., 1991), or structure building (Camazine et al., 2001) are typical tasks where collective robotics can take inspiration from the behavior of social insects (Bonabeau et al., 1999) providing efficient and versatile solutions.

At the hardware level, advanced modularity and versatility are shown by self-reconfigurable robots. These systems are built with a large number of physical modules acting together within a unique body. Each module provides few degrees of freedom. These modules, when assembled together, give the body an extraordinary physical versatility. An additional feature is given by the possibility of the system to connect or disconnect modules autonomously, enabling self-reconfiguration. Based on such a characteristic, a robot can change shape depending on the environment, as shown by PolyBot (Yim et al., 2000a) and by other robots such as MTRAN (Kamimura et al., 2001). The structure of the modules and of the possible configurations change very much across existing systems. The most advanced ones show 3D configurations like snakes, tracks, spiders, and quadruped legged systems. Both PolyBot and MTRAN have displayed transition between shapes in hardware.

Experiments of mobile robots equipped with connection capabilities showed that it is possible to create larger structures. This is the case of the Millibot robot units, which have been modified to enable the creation of a Millibot train (Khosla et al., 2002). Such a structure is equipped with one degree of freedom between each robot, enabling a rotation around a horizontal axis. This allows each robot placed inside the structure to lift vertically the next robot connected to it, bending the train vertically. Although the whole structure seems to bring some additional mobility when facing large obstacles, it offers a very limited flexibility and lateral mobility. Moreover, the version of Millibot able to create trains has very limited sensor capabilities, due to mechanical constraints and small size.

2.2.2. *Swarm-bot* Versatility. Versatility is given in a *swarm-bot* by the presence of many entities that can self-assemble in a unique body and disband when the union is no longer necessary. This feature combines the properties of control versatility found in collective robotics with some hardware versatility found in self-reconfiguring robots. Since each *s-bot* is a fully autonomous mobile robot, a *swarm-bot* can not only self-reconfigure, but it can also self-assemble and disassemble efficiently. *S-bots* can leave a *swarm-bot* configuration, move around it and

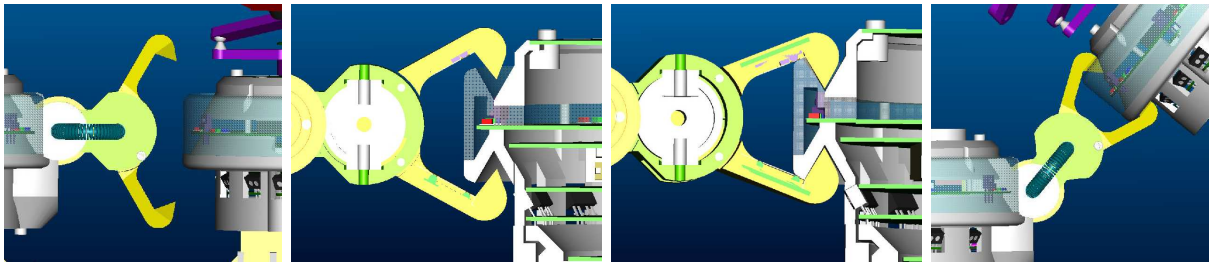


Figure 5. The *s-bot* rigid gripper rotates around a horizontal axis. It can connect either ensuring a rigid grip or leaving some freedom of movement.

join it again when necessary. This is a major additional feature with respect to existing self-reconfigurable robots, which form a unique and monolithic structure. Compared with a Millibot train, a *swarm-bot* can form more complex and flexible configurations, due to better mechanical and electronics capabilities. Each *s-bot* has, in fact, about 50 sensors and 9 actuators, as opposed to a Millibot unit which has just about 10 sensors and 3 actuators. The large number of sensors and actuators allows *s-bots* to ensure more efficient connections and operations.

S-bots have two types of possible physical interconnections for self-assembling into a *swarm-bot* configuration: rigid and semi-flexible.

Rigid connections between two *s-bots* are established by a rigid gripper mounted on a horizontal active axis (Figure 5). Such a gripper has a very large acceptance area allowing it to realize a secure grasp at any angle and, if necessary, allowing it to lift another *s-bot*. Similar connections are made by ants when they build bridges or other rigid structures (Lioni et al., 2001). The large acceptance area is a very significant aspect for connections taking place among independent autonomous robots on rough terrains. Building a self-assembling *swarm-bot* by means of interconnecting robots is a very different task than interconnecting modules in a self-reconfigurable robot. This last can in fact compute the exact position of each module in order to ensure precise positioning during interconnection (Agrawal et al., 2001). This is not the case in a *swarm-bot* where there is freedom of connecting at several angles and with less accuracy. This is a very crucial difference with respect to a Millibot train, whose units must align very accurately in order to interconnect.

The *s-bot* rigid gripper can grasp other *s-bots* on a T-shaped ring around the main *s-bot* body (turret). If it is not completely closed, such a grasp lets the two joined robots free to move with respect to each other while navigating on a rough terrain. If the grasp is firm, the gripper ensures a very rigid connection which can even sustain the lifting up of another *s-bot*. However, lifting with the rigid gripper more than one *s-bot* is not possible. This is a major difference between a *swarm-bot* and other self-reconfigurable robots, which can instead form quite complex 3D shapes while moving and overcoming obstacles. Nevertheless, a *swarm-bot* does not require complex 3D shapes, since its mobility is guaranteed by the combined effort of each *s-bot*.

Semi-flexible connections (Figure 6) are implemented by a gripper positioned at the end of a flexible arm actuated by three servo-motors positioned at the point of attachment on the main body. The three degrees of freedom allow to extend and move laterally and vertically the arm (Figure 7 and 8, respectively). This structure is a modified version of the DELTA robot (Clavel, 1988). The gripper at the end of the arm (called in the following "flexible gripper") is

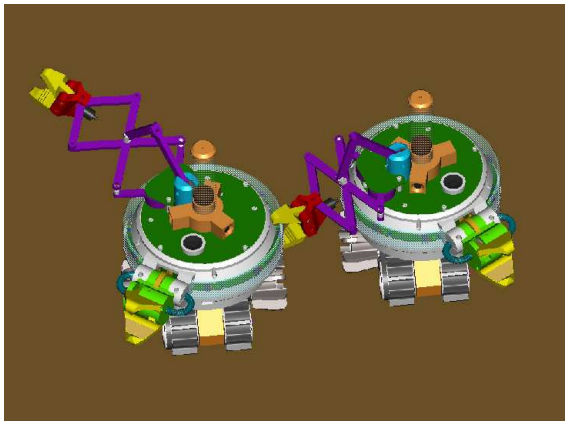


Figure 6. Two *s-bots* connected using semi-flexible connections.

similar to the rigid gripper mentioned above. The orientation of this gripper is kept in a default position by the cables flexibility, but can rotate around a vertical axis. Rigid and semi-flexible connections have complementary roles in a *swarm-bot*. A rigid connection is mainly used to form solid chains for passing large gaps or obstacles (Figure 2). A semi-flexible connection is instead used for configurations where *s-bots* need to stay close to each other but at the same time they still retain relative freedom of movement with respect to each other (Figure 9). A *swarm-bot* can also have mixed configurations, which include both rigid and semi-flexible connections. A third type of connection among *s-bots* can take place through an external object in case of a transporting task (Figure 10).

Rigid and semi-flexible connections are not designed to create complex 3D structures. Most of the configurations envisioned are close to the examples shown in Figure 2 and 9. In any case, a rigid connection allows the creation of simple 3D structures, for instance where peripheral *s-bots*

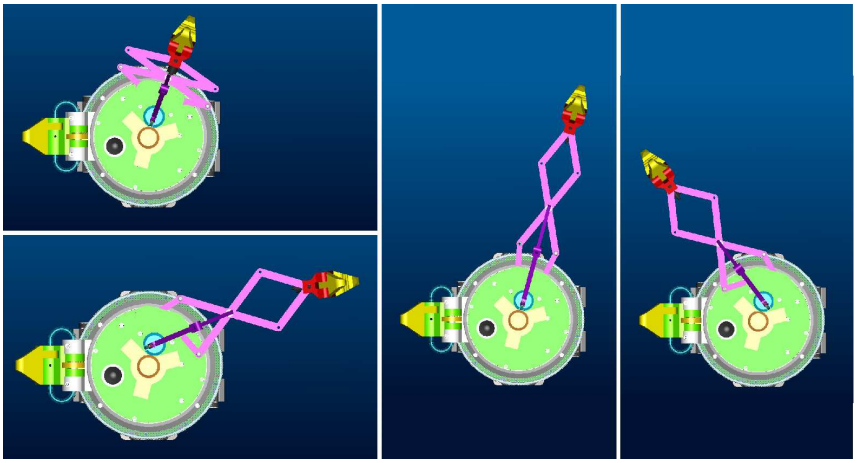


Figure 7. The semi-flexible connection can be extended, retracted, and moved laterally.

Figure 8. The semi-flexible connection can be moved also vertically.

Figure 9. Graphic visualization of how lateral semi-flexible connections are going to be used to keep relative mobility between *s-bots* while they are in a *swarm-bot* configuration. This flexible structure can help for instance to pass local small obstacles.

are placed vertically to help a *swarm-bot* to overcome obstacles. This type of 3D flexibility is exploited mainly for climbing obstacles too steep for the tracks of a single *s-bot*.

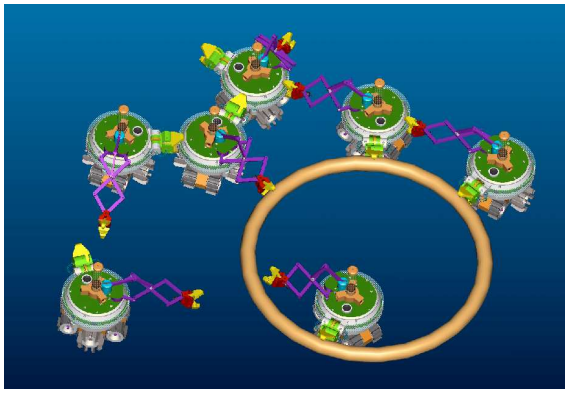


Figure 10. Most *swarm-bot* configurations will include both rigid and semi-flexible connections.

2.3. Rough Terrain Navigation

A robot capable of navigating in unstructured environments should be able to get across rough terrains as well as through cavities and narrow passages. A *swarm-bot* offers within this context an innovative solution in improving mobility by exploiting physical collaboration of a collective system.

2.3.1. State of the Art. Navigation in rough terrain conditions is mainly addressed by articulated rovers and reconfigurable robots. Examples of rovers include the *shrimp* robot (Estier et al., 2000), the family of space exploration robots by ESA[4] (Visentin et al., 2001), the *pathfinder* rover used on Mars (Stone, 1996), as well as other specific rovers for missions like volcano explorations (Bares and Wettergreen, 1999). This type of research is mainly focussed on mechanical structures of articulated wheels and tracks and their ability to pass obstacles. Although most of these rovers are remotely controlled, research aims also at developing sensors for autonomous operation (Vandapel et al., 1999) or to help the remote operator (Matthies et al., 2002).

Some researchers consider multiple rovers for all-terrain exploration (Chien et al., 2000; Earon et al., 2001) exploiting distributed hardware and, in some cases, distributed control to obtain a more robust system and better exploration performances[5]. To the best of the authors' knowledge, nobody has yet tried to take advantage of the collective aspect for obstacle climbing, except for some preliminary experiments using the modified version of Millibot mentioned earlier.

Research in self-reconfigurable robots addresses the same problem in a totally different way, building modular systems that are flexible and can walk, creep, and roll in rough environment conditions. Simulations of PolyBot have been based on an all-terrain scenario (Yim et al., 2000b) and the typical goal of the CONRO system is earthquake search-and-rescue and battlefield surveillance and scouting (Castano et al., 2000). Despite these goals, the sensors included in these developments are mainly used for perception of the internal state of the system and there is practically no perception of the environment. This is motivated in some cases by pure tele-operation. Pure tele-operation, however, may not be sufficient for efficient operation. The remote perception of the environment is in fact limited by time delays, communication bandwidth, and representation of the environment to the remote operator (Murphy et al., 2000; Matthies et al., 2002). Semi-autonomous tele-operation, using local information and performing local control, can improve operability and allow to achieve the task in an efficient way. There is therefore a need for including sensors on this type of robots, as shown by recent work on the CONRO system (Støy et al., 2002).

Another problem in rough terrain operation of self-reconfigurable robots is the contact of the robot with the ground. Although snake-like structures can be quite efficient, performance is less convincing in legged configurations. The contact with the ground is guaranteed in this latter case by the module at the end of the chain. Such a module is in this way forced to have its inter-modules connector, that is its most sensitive part, in contact with the ground. A future reconfiguration of the system might therefore fail due to possible severe damage of the modules previously used as feet. Solutions to this problem are yet to be found.

[4] European Space Agency
[5] http://www.sandia.gov/isrc/Swarm.html

The approach taken by a Millibot train is much closer to that of the *swarm-bot* concept. However, despite its ability to self-assemble and form chains that can climb large obstacles, the very limited capacities of each module and the limited lateral mobility of the one-dimensional trains show strong limitations of the entire concept.

2.3.2. Swarm-bot Ability to Deal with Rough Terrain. The overall mobility of a *swarm-bot* is guaranteed by the mobility of each single *s-bot* composing it. *S-bots* are not designed to be used as modules of a leg, as it happens for instance in the case of self-reconfigurable robots. The gripper used for the interconnection between two robots does not have sufficient torque to support this type of structure. The configurations displayed by a *swarm-bot* are mainly bi-dimensional with the possibility of lifting up lateral *s-bots* in order to overcome large obstacles. The tracks of one *s-bot* are therefore always the point of contact to the ground for a *swarm-bot*. The possibility of rotating the tracks with respect to the turret (Figure 3) ensures suitable mobility of the entire structure even when *s-bots* are rigidly connected.

The control of a *swarm-bot* structure in rough terrain conditions is strongly inspired on insect behavior (Lioni et al., 2001; Şahin et al., 2002). Most structures are built of chains of *s-bots* combined with lateral connections for overall stability. The process of passing an obstacle is based on local push-pull operations. The self-assembling feature is strongly exploited: a *swarm-bot* structure is assembled, if necessary, and disbanded as soon as possible, using in this way the robots as much as possible as independent units.

As a final remark, the limited size of one *s-bot* fits very well the constraints of a catastrophic search operation which requires introduction of a robotic unit into very narrow entry points. This characteristic gives to the *swarm-bot* entity the possibility of accessing internal voids. Once inside, a *swarm-bot* navigates adapting to the environment conditions which may demand self-assembling of the swarm and disaggregation of the same when the union is no longer needed.

3. Hardware Implementation

This section illustrates the feasibility of the *swarm-bot* concept, showing how it has been implemented and briefly summarizing some preliminary results. The discussion presents an overview of the mechanical (Section 3.1), electronic (Section 3.2), and software (Section 3.3) implementations of the first prototype.

3.1. Mechanics

The design described in Figure 1 was done so as to include all necessary details to build a real robot. All parts were designed to be feasible and most mechanisms tested during the design. Figure 11 shows all major parts included in this design. Each of them was translated into a technical manufacturing drawing (blueprint) and then produced. Figure 12 shows the corresponding real parts.

The production methods employed were very different depending on the type of part. Standard machining was used for very simple components, such as some bars of the flexible arm, or parts

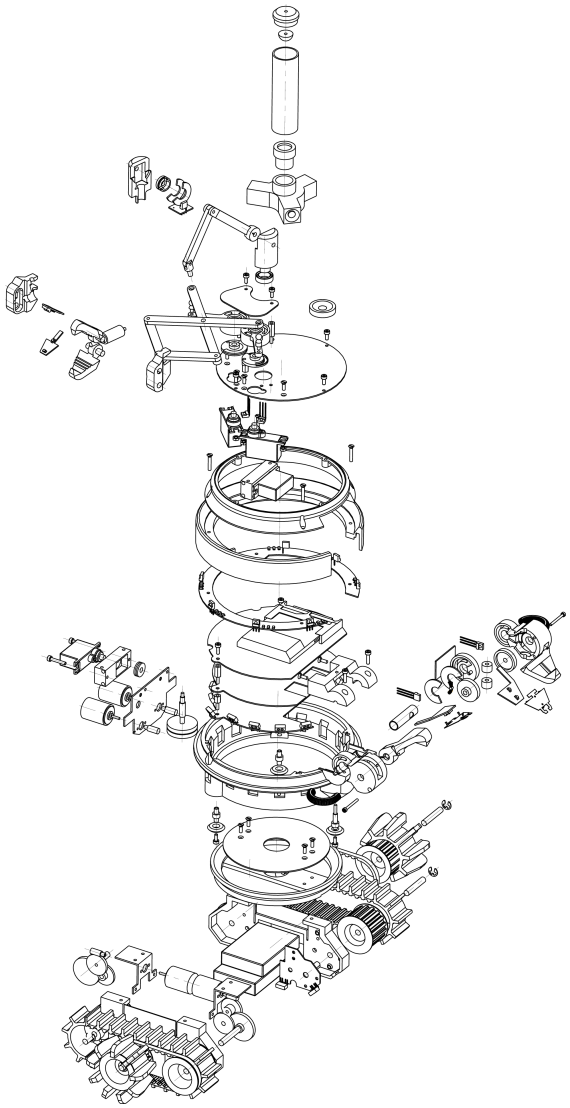


Figure 11. Exploded view of most of the components.

that needed to be metallic, such as some gears or axes. Chemical machining was instead employed for flat parts like the electric contacts molded inside the gripper (Figure 17). Most of the parts were molded, which implied first to manufacture a mold and then to create the part itself. This manufacturing approach asks for a bigger initial effort but it allows to reproduce parts very easily. Due to the number of *s-bots* that we plan to produce (thirty-five), this method allows a cheap and fast production.

At the time of writing, two *s-bots* have been fully assembled and tested (Figure 13). By March 2004, the plan is to have a *swarm-bot* of 10 fully operational *s-bots*.

The treels© mechanism has shown very good performance during tests in rough terrain conditions (Figure 14). The association between tracks and wheels performs very well both in

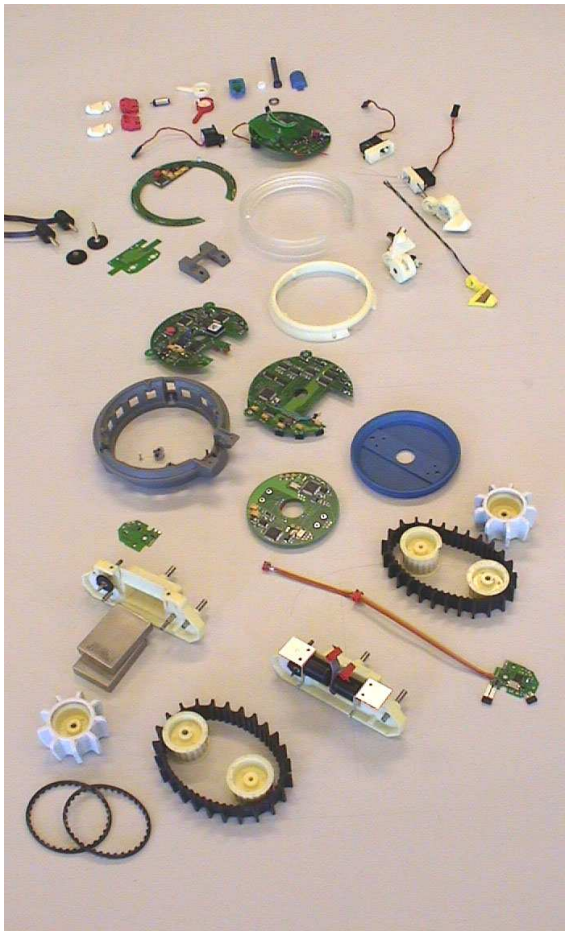


Figure 12. Exploded view of most of the real parts.

straight motion, where tracks ensure a powerful displacement, and in sharp turns, where the wheels, which are bigger than the tracks and placed on a bigger radius, play a key role and ensure a very good rotation.

The rigid gripper (Figure 15) is another important part of an *s-bot*, and it has been crafted so that it can lift another *s-bot*. This feature requires a good torque and a good rigid connection between the gripper and the *s-bot* to be lifted. The gripper is able to grasp rigidly another *s-bot* using a lockable gripper, which can be locked mechanically to keep its position.

Comparison results between simulated and real robots both for a single *s-bot* and for a *swarm-bot* configuration are presented in Section 5.

3.2. Electronics

An overview of the electronic structure controlling the robot is given in Figure 16. The CPU is an Intel XScale processor running Linux OS and controlling directly the sound and camera interfaces. The camera is a standard color web cam with a resolution of 640x480 pixels connected to the main

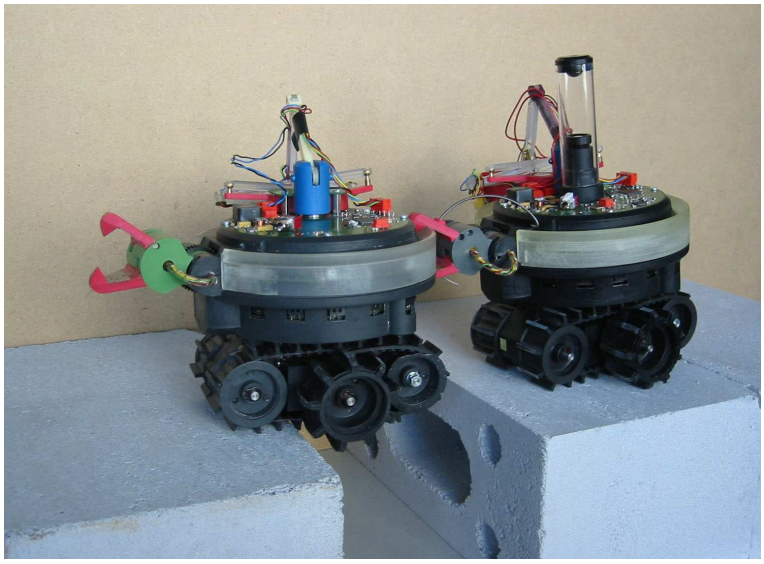


Figure 13. Swarm-bot prototype using the rigid connection to pass a gap.

processor using a USB bus. A spherical mirror facing the vertical optics allows to have a 360° panoramic view. All other devices on the robot are controlled by local PIC^{TM} micro-controllers[6] communicating with the main processor using an I^{2}C bus.

The electronics is mainly included in the central *s-bot* body (turret), but several printed circuits are located in places where they support sensors or local control electronics. In some cases the printed circuit is molded inside the mechanical parts, as seen for instance in Figure 17.

The most important integration effort in size, power consumption, and computational power has been made at the level of the main XScale Linux board. Developed to fit in a very small size (just about a credit card), this board has been integrated successfully inside the *s-bot* after a long

[6] PIC^{TM} micro-controllers are products of Microchip Corp. See `http://www.microchip.com` for more details.

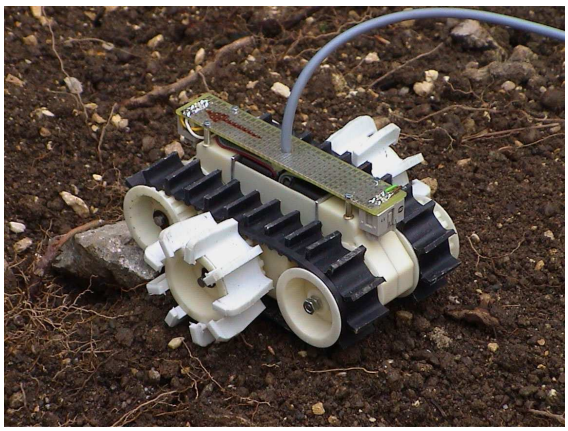


Figure 14. Treels© mechanism during tests.

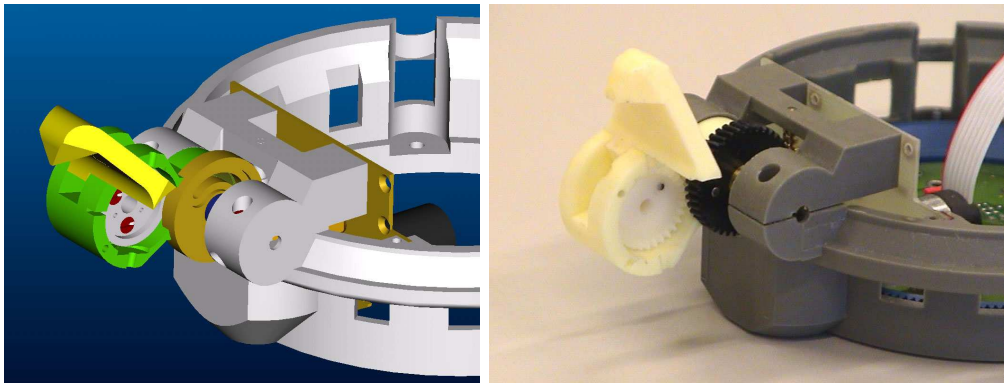


Figure 15. CAD and real view of the gripper mechanism partially assembled to show the internal mechanics. On the real view the black gear ensures the elevation of the gripper. Inside the gripper support a white gear ensures the symmetrical configurations of the jaws.

process of prototyping and software development. The main characteristics of this board are: 64MB RAM memory, 32MB Flash memory, two slots for compact-flash cards (able to support radio-ethernet or bluetooth), USB master and slave interfaces, I^2C bus and serial port. The XScale processor runs at 400 MHz. Tests of Linux running on the board have shown a power consumption of 750 mW. Computational tests have shown that this type of processor can process simple algorithms on full color images (640x480) in 100–200 ms.

Each *s-bot* is equipped with two Lithium-ION accumulators placed between the tracks. The capacity of these accumulators is 10 Wh. Preliminary measurements show a power consumption of one *s-bot* between 3 and 5 W, which ensure continuous operation for at least two hours.

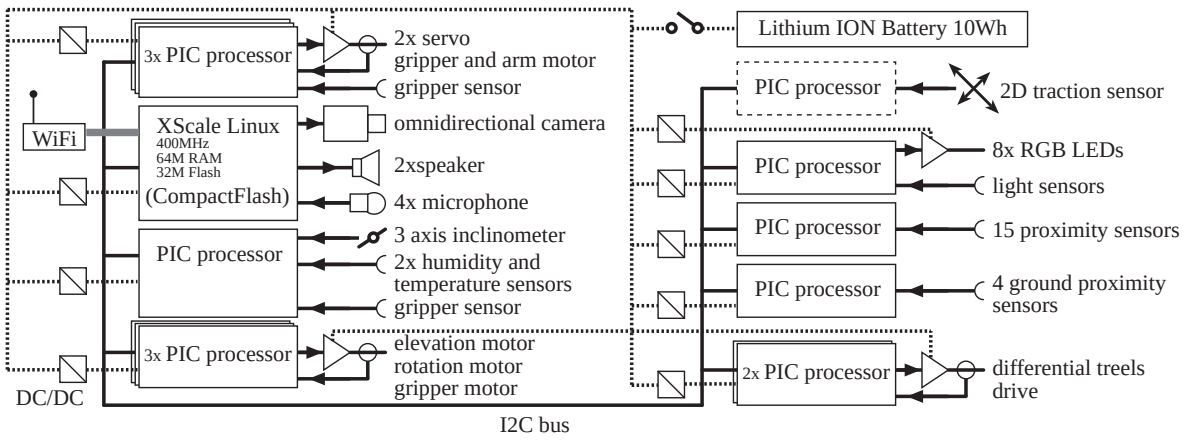


Figure 16. Overview of the electronics controlling the *s-bot*.

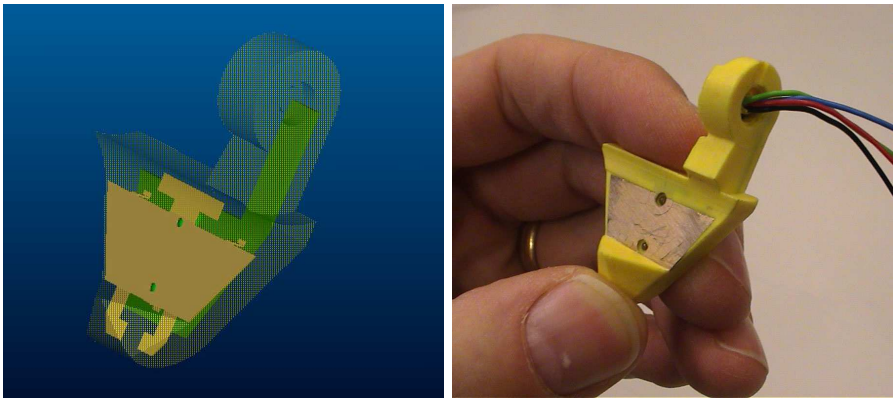


Figure 17. Gripper teeth: 3D model showing the internal printed circuit and electrical contact (left) and real part (right).

3.3. Software

The low level software is distributed among the 14 processors controlling all the functionalities. One of them is the ARM main processor running Linux and described above. The ten other processors are PIC^{TM} micro-controllers, each programmed for a very specific task. Five of them perform motor control, while the other five perform sensor processing. The five PIC^{TM} motor controls have been programmed in assembler, and the remaining other eight in C. All of these processors have a part of the code managing the communication through the I^{2}C bus, whereas the rest manages the custom functionalities they are responsible for. This last part of code can include some preprocessing or some local control loop which can be supervised by the main CPU with commands sent over the I^{2}C bus. As an example, it can be mentioned the control loop using the torque sensors on the motors or the IR sensors data preprocessing.

4. The Simulation Tool

This section presents the simulation environment (swarmbot3d) complementing the hardware part of the *swarm-bot* concept described earlier.

The simulator was planned in order to cover the current lack of commercial products or research prototypes allowing to tackle all the aforementioned aspects of the SWARM-BOTS project at the same time. Most of the tools available on the market concentrate, in fact, on specific aspects of the distributed intelligence paradigm and they generally deal with 2D worlds only.

Swarmbot3d was developed to work as an aiding tool for accurately predicting 3D kinematics and dynamics of a single *s-bot* in a *swarm-bot*, for evaluating possible new options for hardware parts, for designing new experimental set-ups in 3D, and for quickly evaluating new distributed control ideas before porting them to the real hardware (Pettinaro et al., 2002).

The main characteristics of this simulation environment can be summarized as follows.

3D dynamics. It is a 3D dynamics simulator of a multi-agent system (*swarm-bot*) of cooperating robots (*s-bots*).

Hardware *s-bot* compatibility. It provides *s-bot* models with the functionalities available on the real *s-bots*. It can simulate different sensor devices such as IR proximity sensors, an omni-directional camera, an inclinometer, sound, and light sensors.

Software *s-bot* compatibility. Controllers that are developed using swarmbot3d can be ported directly to the hardware *s-bot* due to a common application programming interface.

Interactive control. It provides online interactive control during simulation, useful for rapid prototyping of new control algorithms. Users can try, debug and inspect simulation objects while the simulation is running.

Multi-level models. It provides most robot simulation modules at different levels of detail. It also provides a hierarchy of four *s-bot* reference models with increasing level of detail. *Dynamic model switching* is an included feature which allows to change the robot representation model in real-time. This allows a user to switch between a coarse and a detailed level of simulation model to improve simulation performance at any time.

Swarm handling. It allows to handle a group of robots either as independent units or in a *swarm-bot* configuration, which can be thought of as an entity made of *s-bots* connected to each other. The connections are created dynamically at simulation time and can be eliminated when the components disband. Connections may be of a rigid nature giving to the resulting structure the solidity of a whole entity. This feature is unique with respect to other existing robot simulators.

This section is dedicated to present several aspects of the swarmbot3d simulation tool: from its internal structure to how robots have been modeled.

4.1. State of the Art

Simulation of multiple robot systems has been addressed mainly in the fields of multi-agent systems, artificial life, distributed AI and autonomous mobile robotics. The simulation tools developed for these areas, depending on the abstraction level, have ranged from simple cellular automata to highly distributed realistic environments such as, for instance, MissionLab (MacKenzie et al., 1997), which supports execution of multiple robots both in 2D-simulation and on actual robotics platforms.

Another simulation package for large multi-agent systems is Swarm (Minar et al., 1996) developed at the Santa-Fe Institute. The modeling formalism adopted by Swarm is a collection of independent agents interacting via discrete events. Each entity can generate events that affect the entity itself and other agents. A simulation consists of scheduling the interactions among agents. Although Swarm simulates multiple agents, this discrete-event simulator is not appropriate for simulation of mobile robots.

There are three further multiple robot simulators worth being mentioned: Player/Stage, TeamBots, and MuRoS. They are all designed to deal with 2D worlds, and, because of this, they do not comply with the basic requirement of simulating *s-bots* in 3D. Player/Stage (Gerkey et al., 2001) is a public domain simulator developed at the Robotics Lab of the University of South California (USC). Its characteristic is that of being a scalable multiple mobile robot simulator with each robot moving about and sensing a two-dimensional bit-mapped environment. TeamBots (Balch,

1998) is a Java-based 2D simulator for multi-agent mobile robotics research. Its distribution is written entirely in Java and its release is open source. Last, MuRoS (Chaimowicz et al., 2001) is a simulator developed at the University of Pennsylvania. Such a simulator allows 2D simulation of several multi-robot applications such as cooperative manipulation, formation control, foraging, etc. It is interesting to point out that in MuRoS tasks both loosely and tightly coupled can be simulated. With the exception of the restriction to 2D environments, MuRoS seems to address the same goals of the SWARM-BOTS project, including the formation of flexible and rigid cooperating structures.

It should be mentioned for the sake of completeness that there exists a variety of robotic soccer simulators such as the official RoboCup simulator (Noda, 1995), JavaSoccer, SoccerBots (part of the TeamBots package). However, all of them are used to simulate only disconnected robots and they do not consider the possibility of dynamically creating inter-robot connections.

Finally, we mention Webots, a simulator which was originally developed for the Khepera^{TM} robot but which is now able to support any type of autonomous vehicle, including wheeled, legged and flying robots (Michel, 1998). Earlier versions of this simulator were purely kinematics based. However, its latest release has been extended to handle dynamics using a physics engine based on the Open Dynamics Engine[7] libraries. The software includes a complete library of actuators and sensors for building customized robots. Webots would have been an interesting candidate for simulating a *swarm-bot*; unfortunately, its version using dynamics became available too late to be considered in the project.

4.2. Structure

Swarmbot3d is a 3D dynamics simulator. This means that it is able to take into account physical laws related to properties such as mass, friction, or acceleration in the usual Euclidean space. The simulator is built on top of Vortex[8], a commercial physics engine used in many applications, including the pioneering work of Karl Sims on evolved creatures (Sims, 1994).

Simulation models of environments and robots, as well as world properties such as friction, gravity, and so on, are all defined in an external text file written in XML format. Robot control programs are expressed in terms of the same application programming interface (API) available in the real robot hardware. This guarantees full portability of any control developed using swarmbot3d.

The simulator provides a simple graphic user interface (GUI) developed in Python, a high level interpreter language, for controlling the simulator's parameters (such as gravity, simulator speed, time step, and so on) and for controlling directly each *s-bot*. This Python-based GUI provides also a command line interface for directly interacting with the simulated robots. This feature has shown to be very useful for online debugging, scripting, and rapid prototyping of control strategies.

[7] Open Dynamics Engine (ODE) is an open source project.
[8] `http://www.cm-labs.com`.

Table 1. Modularity of the simulated *s-bot* in subsystem components with different levels of abstraction.

Subsystem	**Abstraction**
treels©	⇒ 2 spherical wheels ⇒ 6 spherical wheels ⇒ 6 detailed teethed wheels
turret	⇒ cylinder ⇒ detailed description
rigid gripper	⇒ box with on-off connection capability ⇒ detailed toothed jaws
flexible gripper	⇒ detailed scissor-like arm

4.3. S-bot *Modeling*

Swarmbot3d has been designed with three main features in mind: (i) modularity, (ii) multi-level modeling, and (iii) dynamic model switching. Such a multi-featured design philosophy allows to build a flexible and efficient simulator.

4.3.1. Modularity. This characteristic allows users to have a large freedom in customizing their own swarm according to their specific research goals. This revealed to be very useful during the early prototype stages of the robot hardware when its specifications often changed. Since the simulation models were developed in parallel with the hardware prototype, the use of modularity allowed not only to re-model a specific *s-bot* geometry, but also to extend swarmbot3d with new mechanical parts or new sensor devices which became available throughout the hardware development. To implement the modular design philosophy, one *s-bot* model was divided in 4 subsystems: the treels©, the turret, the rigid gripper, and the flexible gripper. Some of these subsystems have been implemented at different abstraction levels, as explained in the next subsection (see also Table *1*).

4.3.2. Multi-Level Modeling. This characteristic provides different models for the same part, so that an end-user is given the possibility to load the most efficient and functionally equivalent abstraction model among those available to represent the real *s-bot*.

For example, one *s-bot* may be loaded as a detailed model when an accurate simulation is needed; or it may be loaded as a crude abstraction for the evaluation of a big swarm, in which case the accuracy of a single robot might not be a crucial aspect. People working with learning or swarm intelligence techniques might in fact be more interested in simple coarse *s-bot* models. Conversely, those experimenting with the interaction of relatively small groups of *s-bots* (between 5 and 10) might prefer to use a more refined *s-bot* model.

Viewed in terms of subsystems, defining an abstraction for one *s-bot* consists in defining an opportune combination of some of the subsystems at the desired level of detail (Table *1*). Since the use of a particular model refinement influences considerably the time required to simulate

it on a given computing hardware, the level of abstraction has to be chosen as an opportune trade-off between simulation efficiency (speed) and accurate reproduction of reality.

As an aid to the end-user, 4 reference *s-bot* descriptions differing in their level of abstraction have been defined and prepackaged: *detailed*, *medium*, *simple*, and *fast*. Users can anyhow still select among the *s-bot* subsystems the combination of approximations which best suits the specific research goals they intend to pursue. A brief description of each reference model is outlined in the following.

Detailed s-bot This robot model is a quite faithful replica of the real *s-bot*: all its mechanical parts are reproduced with all the degrees of freedom required (Figure 18 on the left). This model replicates in details the geometry of the real hardware (Figure 19) as well as the masses, centre of masses, torques, accelerations, and speeds.

The *detailed* model comprises four mechanical modules: treels, turret, rigid gripper, and flexible gripper. Its main characteristics are reported below.

- A detailed chassis description comprehensive of 4 ground IR sensors (2 at the bottom, 1 in front and 1 on the back).
- Six teethed wheels, three on each side, with the two middle ones slightly larger and located outward as in the hardware *s-bot*.
- A detailed turret representation.
- A rigid gripper hinged on the front of the turret and endowed with two teethed jaws.
- A flexible gripper attached through three hinges to the *s-bot*'s body and endowed with two teethed jaws.

Although the *detailed* model closely matches the *s-bot* hardware, it lacks the caterpillar-like rubber teethed band joining the inner wheels of each track. The simulation of this track was computationally very expensive and provided only a negligible gain in the realism of simulating the real hardware.

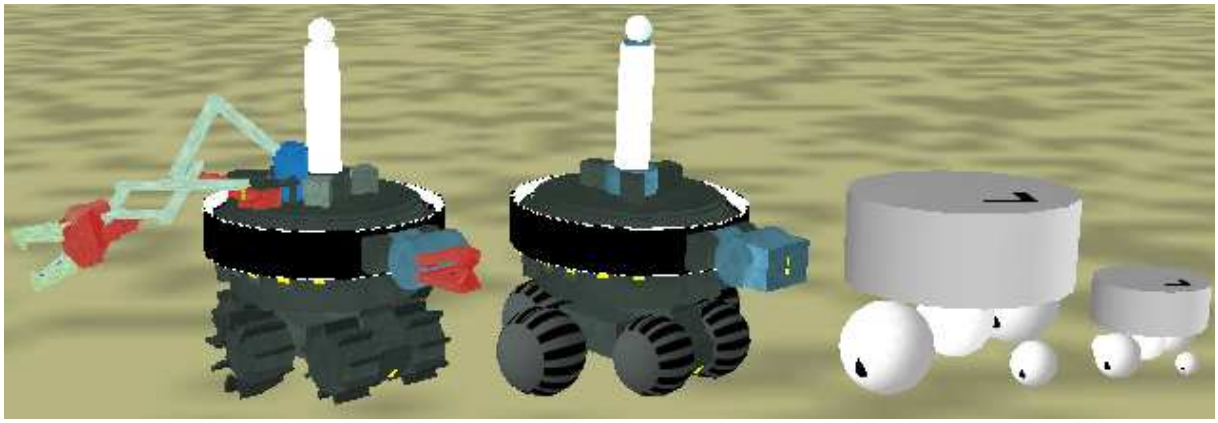


Figure 18. Detailed (first from left), *medium* (second from left), *simple* (third from left), and *fast* (fourth from left) *s-bot* models. *Fast* and *simple* models have both 2 wheels and a very simplified rotational turret. The *medium* model differs from the *detailed* model only by having 6 simplified spherical wheels, a simplified rigid gripper and lacking the flexible side arm.

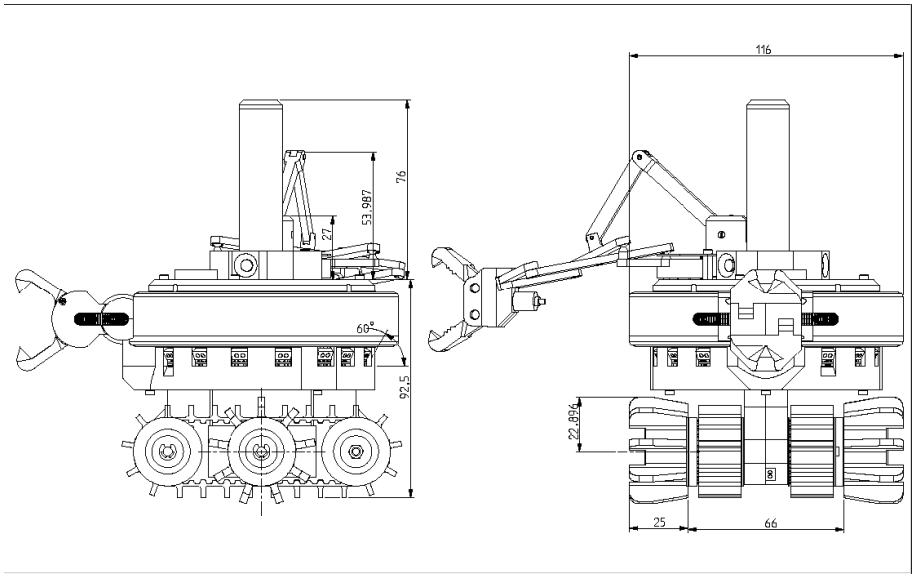


Figure 19. Mechanical diagram of a real *s-bot*.

Medium s-bot This *s-bot* model is shaped with a detailed description of the treels system, although the 6 wheels are in this case defined as simple spheres (Figure 18, second from the left). The turret is defined in full detail but without the presence of the flexible gripper. A coarse representation of the rigid gripper (a hinged box) gives this model the possibility of realizing limited on/off connections to other *s-bots*. The turret model implements all main hardware sensors such as infrared, sound, camera, and light sensors.

This model is still reasonably efficient and therefore it allows to develop distributed controllers over quite large groups of units on smooth planes as well as on uneven terrains. The simplified rigid gripper and the absence of the flexible gripper, however, limit its use just to single *s-bot* missions or to collective tasks in groups of *s-bots* with simple rigid connections.

Simple s-bot This model is a minimalistic abstraction of the most salient characteristics identifying one *s-bot* (Figure 18, third from the left). It has a traction system made of one sphere with two spherically shaped wheels hinged to its sides, and a turret made of a bare cylinder. Caster wheels are added to give mechanical stability to the model. Mass and size are roughly the same as those of the real *s-bot*.

This is a model designed to test and validate control algorithms, developed using the *fast* model, in environment with real gravity pull.

The *simple* model, benefitting from its minimal structure and therefore from its high simulation speed, is very useful for investigating computation intensive distributed control policies spread over a large number of units (*e.g.* genetic algorithms) in environments using the real gravity pull. However, because this model implies real masses and forces, it is not possible to run simulations using large time steps without incurring in problems of instability. Furthermore, its applicability is strongly limited to environments with very limited roughness: when this constraint ceases to hold, a more refined level of detail is required.

Table 2. Comparison of features among *detailed*, *medium*, *simple*, and *fast s-bot* simulation models. Yes and no respectively indicate the presence or absence of a particular feature in the model.

Model	**Detailed**	**Medium**	**Simple**	**Fast**[1]
driving wheels #	6	6	2	2
IR proximity sensors	yes	yes	yes[2]	yes[2]
ground sensors	yes	yes	no	no
camera	yes	yes	no	no
sound module	yes	yes	yes[2]	yes[2]
rigid gripper	yes	yes[4]	no[3]	no[3]
flexible gripper	yes	no	no	no
dynamical bodies #	23	8	6	6
typical RTM	3	31.8	94.7	960[1]
time step	10	10	10	100[1]

1) Measurement taken with 1/50th of gravity and 1/20th of mass.
2) Using sample-based look-up table.
3) No physical gripper model: connections are possible using virtual links.
4) Coarse version, i.e. sticking box.

Fast s-bot This model is a scaled down version of the *simple* model with the same modelling structure (Figure 18, first from the right). However, its linear dimensions are halved and its mass is 1/20th with respect to the *simple* model.

It is meant for simple tests in environments with 1/50th of the real gravity pull. The reduced masses and unrealistic gravity enables to use a large time step without getting unstable simulations, at least in flat environments. This possibility allows to increase the simulation speed up to 10 times, although the use of this abstraction on terrains modeled as mesh surfaces may lead to unstable simulations if the time step is too big ($> 10ms$).

To establish how computationally heavy each of the 4 models described above is, a performance evaluation experiment was set. This consisted simply in loading and using one *s-bot* at each different abstraction level on a horizontal plane. The performance result for each type of model is presented here in terms of its *Real Time Multiplier* (RTM) value which refers to how fast real time can be simulated. Table *2* summarizes each model characteristics and gives a typical RTM value for each of them. RTM values were obtained running the tests on a dual 3.06GHz Xeon PC with one nVidia QuadroFx 1000 graphics card.

4.3.3. Dynamic Model Switching. Using the hierarchical abstraction levels introduced above, it has been implemented in the simulator a way to change the used *s-bot* representation during simulation (Pettinaro et al., 2003). The availability of such a feature allows a user, for example, to start a simulation with the simplest abstraction level for one *s-bot* when the terrain onto which it moves is flat and to switch to a more refined model representation when the environment or the interaction among *s-bots* require a more detailed treatment. Dynamic model changing allows swarmbot3d to increase simulation speed by introducing complexity only when it is needed.

Table 3. Sensors types and their implementations in the simulation environment. Each sensor may have multiple implementations which can be chosen by the user.

***S-Bot* Sensors**	**Implementations**
proximity sensor	Sample table based
	Ray tracing
ground sensor	Ray tracing
sound	Instant on-off with spatial decaying intensity
	Sound wave propagation with spatial decaying intensity
speed sensor	Standard library function
torque sensor	Standard library function
inclinometer	Comparison with the absolute XZ plane
light sensor	Ray tracing based with shadowing
camera	Abstract camera using high level objects
	Low-level fish-eye view

This feature, however, assumes that all representation models are *compatible* with each other, that is, all models show the same behaviour when a particular robot command is issued. It is therefore important to adjust each model so that speed, mass, and geometry are calibrated to ensure compatibility, even if they differ in representation detail. For this reason, it was developed an extension of the *simple* model named *basic* model possessing a simple on/off connecting device, which was previously available just for the *medium* model, and a ray traced model of the IR sensors (see Section 4.4).

Thanks to this model changing mechanism, users can use the *basic* model when the terrain is flat, change to the *medium* model when the terrain gets rough or change to the *detailed* model when the flexible arm is needed. Currently, such a model switching has to be carried out manually, however work is in progress for investigating ways for automating this feature, so to leave to the simulator the decision on when changing the abstraction level.

4.4. Sensor Modeling

Real *s-bots* are equipped with several types of sensors which are read by the control program running on each *s-bot*. Swarmbot3d implements 8 types of sensors matching those available on the real units. Some sensors (such as the speed and torque sensors) have been implemented using standard library functions of the underlying physics engine, while others needed implementation and calibration with the real sensors. Table *3* summarizes the virtual sensors and their implementation currently available in swarmbot3d.

The infrared sensors used by the proximity sensor and the ground sensors are simulated using ray-tracing by probing the sensor vicinity with 5 rays (1 central and 4 peripheral) within the sensing cone of each virtual sensor. The 5 rays detect any intersection of nearby objects and compute an average distance value which is subsequently converted to an integer sensor response. The mapping function has been obtained by linear regression of the experimental values.

Table 4. Simulator performance for different abstraction models. The numbers represent *real time multipliers*, *i.e.*, the ratio between simulated real time and simulation time (the higher, the better).

Model	**Disconnected *S-Bots*** (smooth plane)						**Conn. *S-Bots***	**Disconn. *S-Bots***
	1	2	5	10	20	40	5 on smooth plane	5 on rough terrain
Fast[1]	960	490	200	96	43.3	17.6	88.1	147
Simple	94.7	49	20	9.6	4.3	1.8	8.5	14.9
Medium	31.8	15	4.7	2	0.7	0.1	2.2	2.4
Detailed	3	1.4	0.5	0.2	0.09	0.04	0.2	0.4

1) Measurements taken with a time step of 100 ms and 1/50th of the real gravity pull.

The light sensors are modelled in the simulator by summing up contributions of all known light sources weighted by their inverse squared distance and some scaling factor. At present, light shadowing is not implemented. Sound sensors are implemented in a similar way, disregarding also in this case shadowing.

4.5. Performance Evaluations of the Different Abstraction Models.

A crucial point concerning swarmbot3d is how it performs with respect to an increasing number of units populating its world and for different abstraction models.

To evaluate the computational load, the 4 model abstractions introduced in Section 4.3.2. were examined. The experimental test was carried out by loading into the simulator an increasing number of disconnected *s-bots* of the same kind. The simulator performance was quantified by checking, as done with the single *s-bot*, the *real time multiplier* (RTM) value. The hardware employed in this performance evaluation was a dual 3.06GHz Xeon PC with one nVidia QuadroFx 1000 graphics card.

The readings obtained for a smooth plane terrain are reported in Table *4*, where, for comparison purposes, data is shown also for 5 connected *s-bots* on a plane and for 5 disconnected *s-bots* on a rough terrain. All measurements were taken using a time-step of 10 ms, except for the *fast* model, that, because of the large step used with it, showed instability when used on rough terrain.

5. Comparisons between Simulated and Real *S-bot*

This section presents a number of experiments conducted to compare the mechanical behaviour of the various simulated *s-bot* models with respect to the real *s-bot*. A good correspondence was found between the *detailed* model and the real *s-bot* in all cases. The *medium* model was able to approximate acceptably well the *detailed* one in many situations, whereas the *fast* and *simple* models were sufficient only in certain simple environments.

5.1. Motion Comparison

This experiment compares how different models differ during forward linear motion on terrains with different levels of roughness. To do so, each simulation model was placed randomly on a

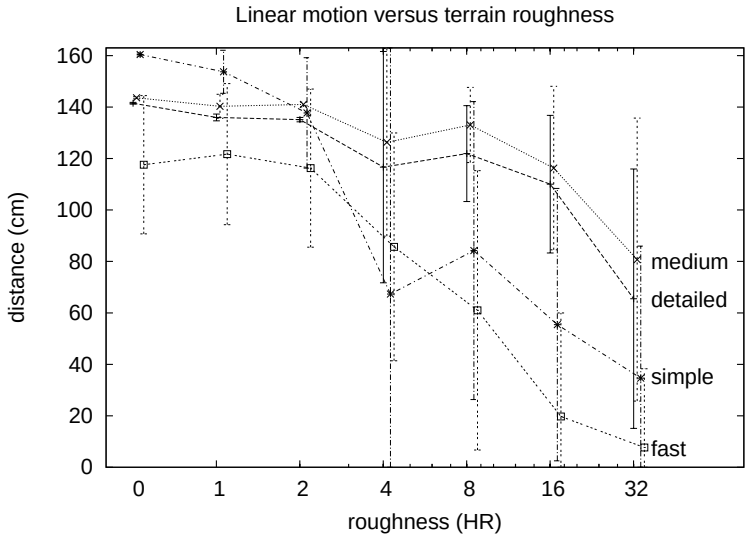


Figure 20. Comparison of linear motion errors of the three *s-bot* simulation models with respect to different terrain roughness. The HR values correspond to descriptive values: 0 for flat, 1 for almost flat, 2 for minimally rough, 4 for little rough, 8 for mildly rough, 16 for rough, and 32 for very rough.

terrain and then assigned a random heading. The terrain roughness was controlled within the simulator by specifying a height range (HR) parameter. Given the reduced size of the *fast* model, the terrain used for that *s-bot* abstraction was also scaled down by half its original size.

Figure 20 shows the motion errors which are obtained by letting each *s-bot* model run at medium speed (15.24 cm/s) for 10 seconds. The vertical axis plots the projection of the travelled distance onto the randomly selected initial direction. Depending on the terrain roughness this distance decreases because the *s-bot* is not able to retain a straight course. The small differences in distance on flat terrain are caused by calibration errors of the velocity due to differences in the wheel diameters among the various *s-bot*.

Figure 20 shows that the rough terrain motion of the *medium* model closely follows the behaviour of the *detailed* one: the constant offset is due to differences in wheel size (see above). Both *simple* and *fast* models quickly fail to retain linear motion even on minimally rough terrains. Since the *detailed* model replicates quite closely the behaviour observed on the real robot, this suggests that also the *medium* one can reasonably approximate it, at least as far as pure locomotion on rough terrain is concerned. The *fast* and *simple* models therefore are not suitable for experiments involving very rough terrain.

5.2. Passing a Gap

In rough terrain situations, it may be the case that one *s-bot* or a group of *s-bots* have to pass gaps or holes. We ran therefore an experiment to study how *s-bots* behave in these situations. To quantify the behaviour of the different simulation models with respect to the real *s-bot*, two planes were placed close to each other with a variable gap (Figure 21). We observed how each model reacted to changes in the size of the gap and compared the results with what observed using one real *s-bot*. This experiment was carried out for one *s-bot* and then repeated for connected *s-bots*.

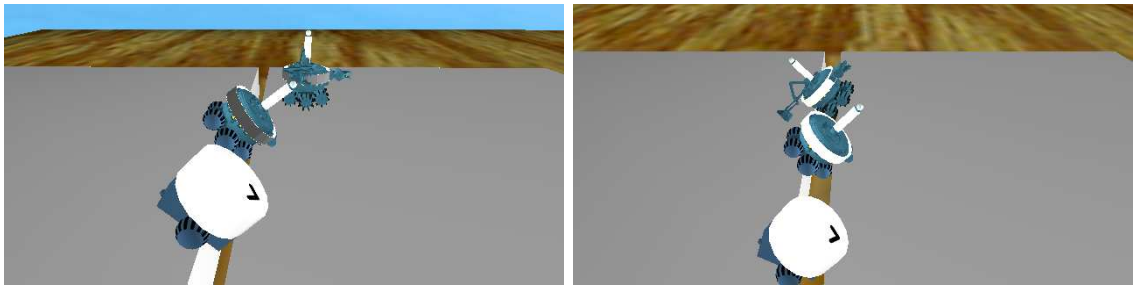


Figure 21. From left to right: The 3 *s-bot* models while traversing a gap of 40 mm and 60 mm, respectively.

5.2.1. Single *S-bot*. The results of the experiment described above are reported in Table *5*. Each table entry corresponds to the modal value of 12 observations. All tests were carried out by using a low speed of 2.8 cm/s.

By observing the table, we see that the *simple* model can cope with gaps up to 40 mm, starts having trouble with gaps of 45 mm, and gets stuck with gaps of 50 mm or wider. The *medium* and *detailed* models, instead, do not have problems with gaps up to 40 mm. Beyond this size, the presence of the teeth on the wheels of the *detailed s-bot* makes a difference. This feature, in fact, by acting as a surrogate of caterpillar tracks, mimics remarkably well the behaviour observed on the real *s-bot* which gets also stuck with gaps of 55–57 mm.

The *fast* model was tested in a separate environment with low gravity (1/50th of the normal one) and with a high time step value (100 ms). This model overcomes gaps of 60 mm and behaves therefore similarly to the *detailed* model, although it moves in an unreal environment. Thus, it can be used as a rough approximation of the real *s-bot* functional behaviour.

5.2.2. Connected *S-bots*. While for a single *s-bot* the maximum traversable gap width is around 60 mm, a connected structure is expected to pass wider gaps, even larger than the

Table 5. Gap experiment for a single *s-bot*. The *s-bot* had to maneuver across a gap of different width. Each row corresponds to a certain gap width and reports the results.

Width (mm)	**Fast**[1,2]	**Simple**	**Medium**	**Detailed**	**Real**
40	overcome	overcome	overcome	overcome	overcome
45	overcome	overcome (70%) stuck(40%)	overcome (90%) stuck(10%)	overcome	overcome
50	overcome	stuck	stuck	overcome	overcome
57	overcome	stuck	stuck	overcome (50%) stuck(50%)	stuck
60	overcome	stuck	stuck	stuck	stuck
65	stuck	stuck	stuck	stuck	stuck

1) Observations taken separately with 1/50th of gravity and 1/20th of mass.
2) Since the *fast* model is half the size of the *simple* model, the gaps' widths should be divided by 2 for having a correct comparison.

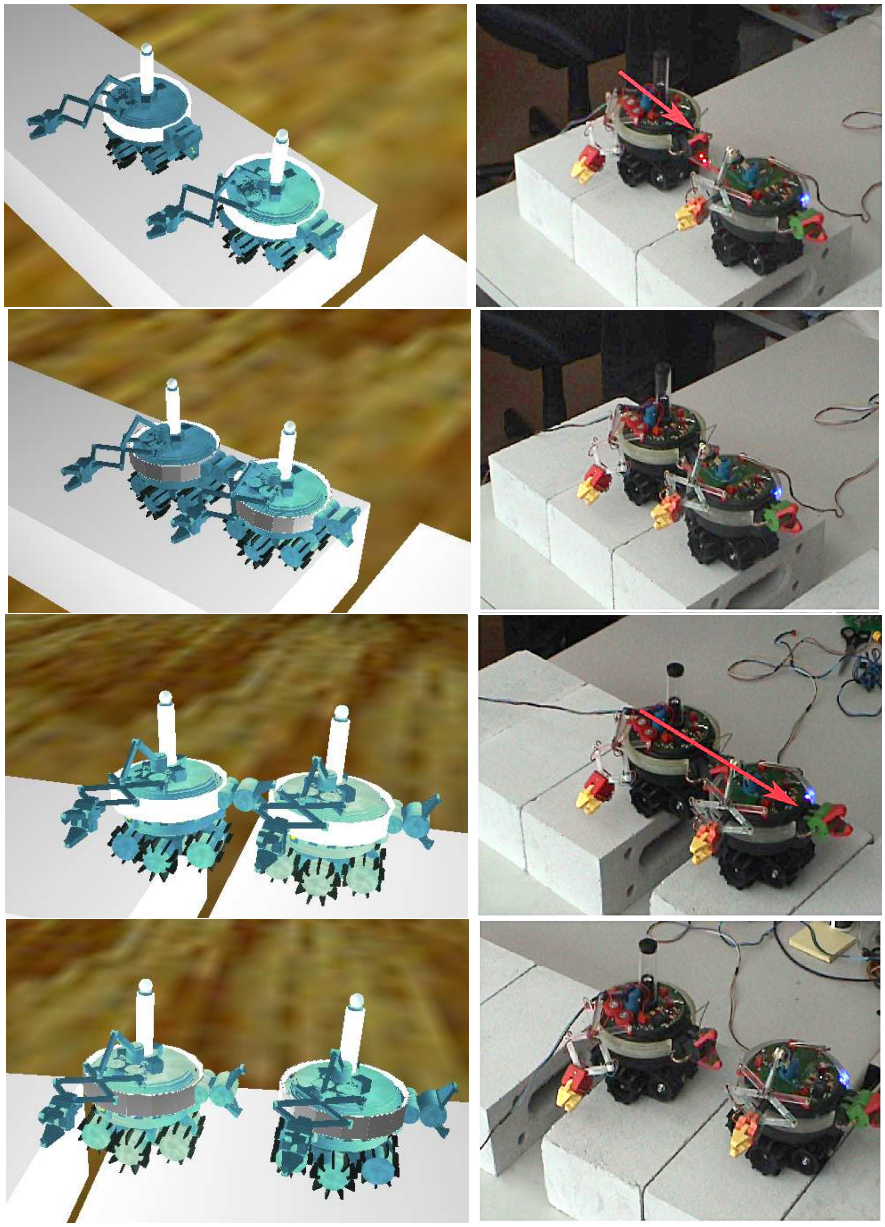


Figure 22. Gap traversal sequence using two *s-bots*. The same sequence is reported in the left column for the simulated *s-bots* and in the right column for the real *s-bots*. From top to bottom: (1) the first *s-bot* in front of the gap calls for help, (2) the second *s-bot* connects with the rigid gripper, (3) both *s-bots* pass the 70 mm gap, (4) the second *s-bot* releases its gripper. A movie of the experiment is available at `www.swarm-bots.org`.

diameter of a single *s-bot* (about 116 mm). We ran an experiment using two connected *s-bots* both in simulation and in the real world (Figure 22). The gap passing experiment was repeated, only in simulation, using chains of three and four robots. Table *6* summarizes the observed maximum gap widths for successful traversals. Each table entry for the simulated *s-bot* corresponds to the modal value of 12 observations.

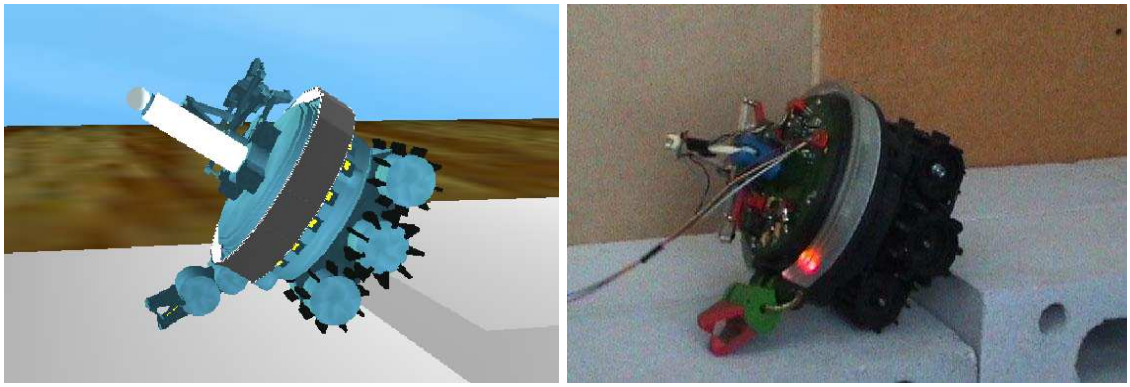


Figure 23. Equivalent behaviour of simulated and real *s-bot* with respect to a step size of 32 mm.

5.3. Climbing a Step

When navigating on rough terrain one *s-bot* has to confront itself with all sorts of hindrances. Some may be overcome by getting around them, some others can instead be overcome by climbing over. Climbing a step is one of these latter and it is important to evaluate how *s-bots* behave when they have to face such an obstacle. In order to do so, two experiments were carried out: one using a single *s-bot* and one using two *s-bots* in a connected configuration. The capability of overcoming a step was in both cases quantified by progressively varying its height with respect to the ground.

5.3.1. Single *S-bot*. In this experiment one *s-bot*, *detailed* model, placed on the ground started to move towards a step. The experiment was performed with the *s-bot* moving both forward and backward. The height of the step was varied in intervals of 1 mm.

It was observed that the maximum step height which a single *detailed* *s-bot* was able to cope with was 15 mm when moving backward and 23 mm when moving forward (Table 7). Each table entry for the simulated *s-bot* is the outcome of 12 observations. The difference between the

Table 6. Gap traversal experiment using swarm-bot3d simulation. The values represent observed maximum gap width for successful traversal for different numbers of connected *s-bots*.

***S-bots* #**	**Gap Width** (mm)	
	Detailed	**Real**
1	57	57
2	100	100
3	190	not available
4	240	not available

forward and backward behaviours both in the simulated *s-bot* and in the real one is due to the center of mass of the turret which is 40 mm off centered towards the gripper.

Figure 23 shows that with a step of 32 mm, which is the height at which two connected robot can pass (see next section), one single *s-bot* going backward topples over both in the simulated world and in reality.

5.3.2. Connected *S-bots*. This experiment was set by letting a pair of simulated *s-bots*, *detailed* model, first connect and then navigate backward towards a step. By varying the height of the step, it was observed that the two robots were able to pass steps up to 32 mm, in accordance with what experienced with the real *s-bot*.

Figure 24 shows four stages in passing the limit step of 32 mm. First, the two *s-bots* approach the step in backward formation. Second, as soon as the first robot senses the step with its rear ground sensor, it starts lifting itself using its connected rigid gripper. During traversal, the robot bends its rigid gripper in the opposite direction (downward) pushing itself up. Finally, the first robot continues its backward motion and pulls in this way the second one over the step to complete the procedure.

6. Conclusions

The work reported in this paper presented a new robot concept, called swarm-bot. Such a concept shows to possess the three major characteristics needed for rough terrain exploration: robustness, versatility, and all terrain navigation. These characteristics were used to discuss how the *swarm-bot* concept compares with similar existing systems.

A *swarm-bot*, with its self-assembling capability added on top of fully autonomous robots, opens up a new research field situated between self-reconfigurable and collective robotics. The concept combines hardware versatility found in self-reconfigurable robots with control versatility found in distributed control for collective robotics. This fundamental property of a *swarm-bot* plays a key role in robotic operations to be performed on rough terrains and it allows to carry out different tasks while facing complex and harsh environments usually found in exploration missions.

A second and fundamental property of a *swarm-bot* is robustness, provided by distributed hardware and control. This feature is also essential for exploration operation where the unknown and unstable environment can cause loss of robotic units.

Table 7. Maximum step climbing height for one *s-bot* moving backward (bw) and forward (fw).

Step (mm)	**Detailed *S-bot***		**Real *S-bot***	
	bw	fw	bw	fw
≤ 15	pass	pass	pass	pass
16	fail	pass	fail	pass
23	fail	pass	fail	pass
24	fail	fail	fail	fail

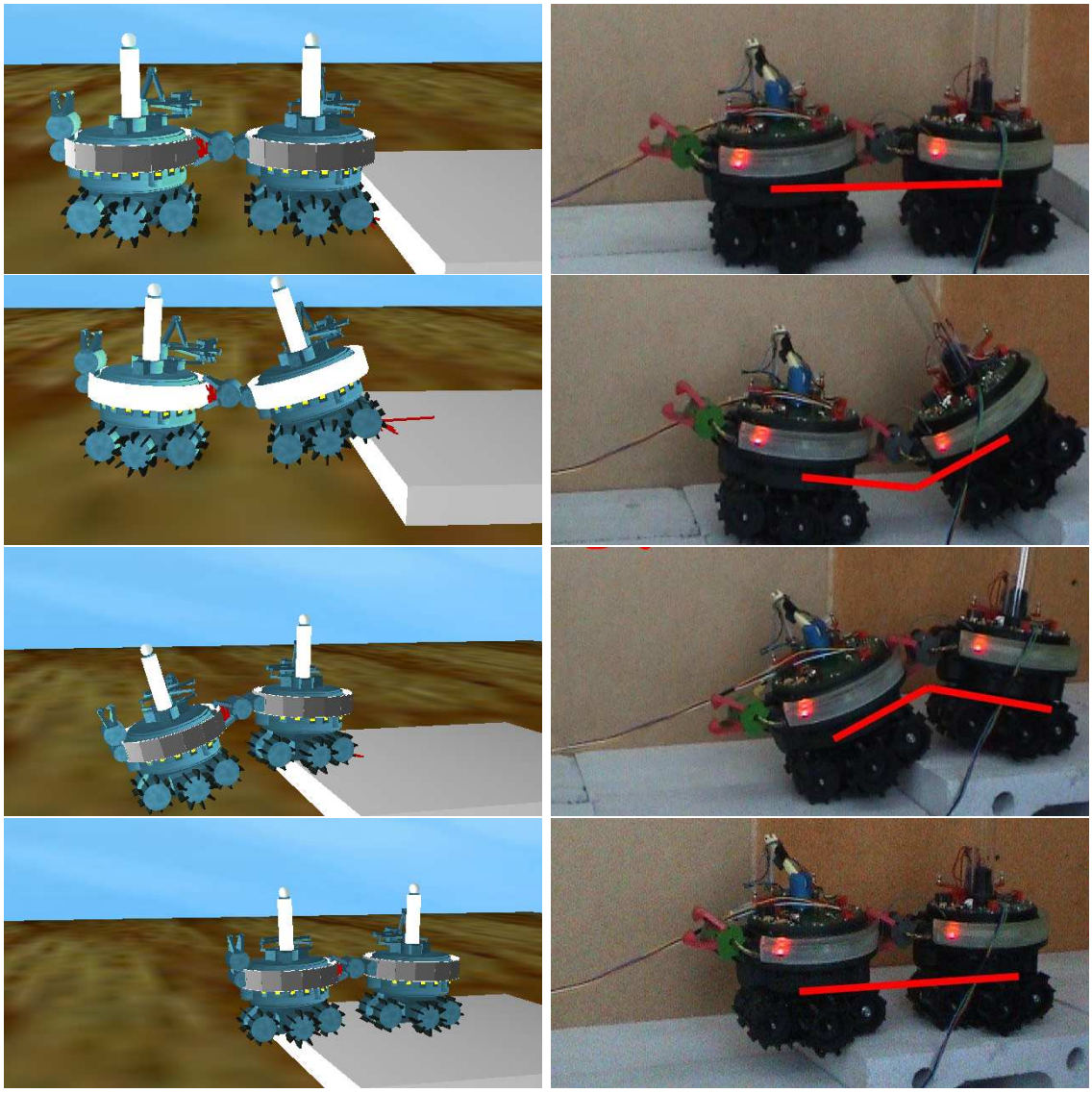


Figure 24. From left to right: the different phases of the step passing for the limit step of 32 mm. The first four pictures refer to the simulated *s-bot* and the last four refer to the actual *s-bots*. A movie of the experiment is available at `www.swarm-bots.org`.

The feasibility of the concept has been shown by presenting the construction of the first physical prototype as well as of a 3D dynamics simulation package (swarmbot3d) complementing it. The usefulness of this software package has been shown for accurately simulating both the kinematics and the dynamics of a single *s-bot* as well as of an entire *swarm-bot*, for evaluating hardware design options for different robot components, for designing *swarm-bot* control experiments in 3D worlds, and for investigating distributed control algorithms.

The simulation environment features modularity, multi-level modeling, and dynamic model switching. *S-bots* are defined in terms of modules with each modules expressed at different

levels of detail. These characteristics make simulated *s-bots* fully customizable. Dynamic model switching is unique in its kind. Such a feature gives to the simulator the power of reducing the computational cost while keeping the accuracy of predicting a *swarm-bot* behaviour within acceptable limits.

Acknowledgements

Many thanks to Michael Bonani, Daniel Baer, Pierre Bureau, Michel Lauria, and Vito Trianni for their help, comments, and ideas.

This work was supported by the SWARM-BOTS project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-2000-31010. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. The Swiss participants to the project are supported under grant 01.0012 by the Swiss Government. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate, through the grant "Virtual Swarm-bots", contract no. 9.4515.03, and from the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

Agassounon, W., Martinoli, A., and Goodman, R. 2001. A Scalable, Distributed Algorithm for allocating Workers in Embedded Systems. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 5. pp. 3367–3373. IEEE Press, Piscataway, New Jersey, USA.

Agrawal, S. K., Kissner, L., and Yim, M. 2001. Joint Solutions of Many Degrees-of-Freedom Systems Using Dextrous Workspaces. In: W. H. K. et al. (ed.): *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*. pp. 2480–2485. IEEE Press, Piscataway, New Jersey, USA.

Ayers, J., Zavracky, P., McGruer, N., Massa, D. P., Vorus, W. S., Mukherjee, R., and Currie, S. N. 1998. A Modular Behavioral-Based Architecture for Biomimetic Autonomous Underwater Robots. In: *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium*. Naval Postgraduate School, Monterey, California, USA.

Balch, T. 1998. Behavioral Diversity in Learning Robot Teams. Ph.D. thesis. Georgia Institute of Technology. Atlanta, Georgia, USA.

Bares, J. and Wettergreen, D. 1999. Dante II: Technical Description, Results and Lessons Learned. *International Journal of Robotics Research* **18**(7), 621–649.

Blanke, M., Izadi-Zamanabadi, R., Bøgh, S. A., and Lunau, C. 1997. Fault Tolerant Control Systems - A Holistic View. *Control Engineering Practice* **5**(5), 693–702.

Bonabeau, E., Dorigo, M., and Theraulaz, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, NY.

Callen, J. N. 1998. Distributed Control enables Flexible Intelligent System Development. *IEEE Intelligent Systems* **13**(4), 14–17.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. 2001. *Self-Organisation in Biological Systems*. Princeton, New Jersey, USA: Princeton University Press.

Casper, J., Murphy, R. R., and Micire, M. 2000. Issues in Intelligent Robots for Search and Rescue. In: R. W. G. Grant R. Gerhart and C. M. Shoemaker (eds.): *Proceedings of the SPIE Unmanned Ground Vehicle Technology II*. Vol. 4024. Orlando, Florida, USA. pp. 292–302. SPIE, Bellingham WA, USA.

Castano, A., Shen, W.-M., and Will, P. 2000. CONRO: Towards Deployable Robots with Inter-Robot Metamorphic Capabilities. *Autonomous Robots* **8**, 309–324.

Chaimowicz, L., Campos, M., and Kumar, V. 2001. Simulating Loosely and Tightly Coupled Multi-Robot Cooperation. In: *V Brazilian Symposium on Intelligent Automation (SBAI)*. Canela, RS, Brazil.

Chien, S., Barrett, A., Estlin, T., and Rabideau, G. 2000. Three Coordinated Planning Methods for Cooperating Rovers. In: *Intelligent Automation and Control, Proceedings of the International Symposium on Intelligent Automation and Control, World Automation Congress (ISIAC-WAC, Maui, Hawaii, USA)*. Vol. 9. TSI Press, Albuquerque, NM, USA.

Clavel, R. 1988. DELTA, a Fast Robot with Parallel Geometry. In: C. Burckhardt (ed.): *Proceedings of the 18th International Symposium on Industrial Robots (ISIR '88)*. pp. 91–100. Springer Verlag, Berlin, Germany.

Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrin, C., and Chatier, L. 1991. The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot. In: J. A. Mayer and S. W. Wilson (eds.): *Simulation of Adaptive Behavior: From Animals to Animats*. pp. 356–365. MIT Press, Boston, Massachusetts, USA.

Detrain, C. and Deneubourg, J.-L. 1997. Scaveging by *Pheidole Pallidula*: a Key for understanding Decision-Making Systems in Ants. *Animal Behaviour* **53**, 537–547.

Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., and Gambardella, L. M. 2004. Evolving Self-Organizing Behaviors for a Swarm-bot. *Autonomous Robots* **17**(2–3).

Duff, D., Yim, M., and Roufas, K. 2001. Evolution of PolyBot: a Modular Reconfigurable Robot. In: *Proceedings of COE/Super-Mechano-Systems Workshop*. Tokyo, Japan.

Earon, E. J. P., Barfoot, T. D., and D'Eleuterio, G. M. T. 2001. Development of a Multiagent Robotic System with Application to Space Exploration. In: *Proceedings of 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. Vol. 2. pp. 1267–1272. IEEE Press, Piscataway, New Jersey, USA.

Estier, T., Crausaz, Y., Merminod, B., Lauria, M., Piguet, R., and Siegwart, R. 2000. An innovative Space Rover with Extended Climbing Abilities. In: W. C. Stone (ed.): *Robotics 2000. Proceedings of the ASCE Conference on Robotics for Challenging Environments*. pp. 333–339. ASCE, American Society of Civil Engineers, Reston, VA.

Flocchini, P., Prencipe, G., Santoro, N., and Widmayer, P. 2000. Distributed Coordination of a Set of Autonomous Mobile Robots. In: *Proceedings of the 4th IEEE Intelligent Vehicles Symposium, (IVS 2000)*. pp. 480–485. IEEE Press, Piscataway, New Jersey, USA.

Fukuda, T., Mizoguchi, H., Sekiyama, K., and Arai, F. 1999. Group Behavior Control for MARS (Micro Autonomous Robotic System). In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*. pp. 1550–1555. IEEE Press, Piscataway, New Jersey, USA.

Gerkey, B. P. and Matarić, M. J. 2002. Pusher-watcher: an Approach to fault-tolerant tightly-coupled Robot Coordination. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002)*. IEEE Press, Piscataway, New Jersey, USA.

Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhtame, G. S., and Matarić, M. J. 2001. Most Valuable Player: A Robot Device Server for Distributed Control. In: T. J. T. et al. (ed.): *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*. Vol. 2. pp. 1226–1231. IEEE Press, Piscataway, New Jersey, USA.

Goldberg, D. and Matarić, M. J. 2002. Design and Evaluation of Robust Behavior-Based Controllers. In: T. Balch and L. E. Parker (eds.): *Robot Teams: From Diversity to Polymorphism*. A. K. Peters, Natick, MA, USA. pp. 315–344.

Groß, R. and Dorigo, M. 2004.. Evolving a Cooperative Transport Behavior for Two Simple Robots. In: P. Liardet, P. Collet, C. Fonlupt, E. Lutton, and M. Schoenauer (eds.): *Artificial Evolution – 6th International Conference, Evolution Artificielle, EA 2003*. Vol. 2936 of *Lecture Notes in Computer Science*. pp. 305–317. Springer Verlag, Berlin, Germany.

Hayes, A. T., Martinoli, A., and Goodman, R. M. 2001. Swarm Robotic Odor Localization. In: T. J. T. et al. (ed.): *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*. Vol. 2. pp. 1073–1078. IEEE Press, Piscataway, New Jersey.

Kamimura, A., Murata, S., Yoshida, E., Kurokawa, H., Tomita, K., and Kokaji, S. 2001. Self-Reconfigurable Modular Robot–Experiments on Reconfiguration and Locomotion. In: T. J. T. et al. (ed.): *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS2001*. Vol. 1. pp. 606–612. IEEE Press, Piscataway, New Jersey, USA.

Khosla, P., Brown, B., Paredis, C., Grabowski, B., Navarro, L., Bererton, C., and Vandeweghe, M. 2002. Millibot Report. Report on millibot project, darpa contract DABT63-97-1-0003. Carnegie Mellon University. Pittsburgh, Pennsylvania, USA.

Kube, C. R. and Bonabeau, E. 2000. Cooperative Transport by Ants and Robots. *Robotics and Autonomous Systems* **30**(1–2), 85–101.

Kurokawa, H., Kamimura, A., Yoshida, E., Tomita, K., Kokaji, S., and Murata, S. 2003. M-TRAN II: Metamorphosis from a Four-Legged Walker to a Caterpillar. In: C. S. G. Lee and J. Yuh (eds.): *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS 2003)*. pp. 2454–2459. IEEE Press, Piscataway, New Jersey, USA.

Lioni, A., Sauwens, C., Theraulaz, G., and Deneubourg, J.-L. 2001. Chain Formation in Oecophylla Longinoda. *Journal of Insect Behaviour* **15**, 679–696.

MacKenzie, D. C., Arkin, R. C., and Cameron, J. M. 1997. Multiagent Mission Specification and Execution. *Autonomous Robots* **4**(1), 29–52.

Martinoli, A., Ijspeert, A. J., and Mondada, F. 1999. Understanding Collective Aggregation Mechanisms: from Probabilistic Modelling to Experiments with Real Robots. *Robotics and Autonomous Systems* **29**, 51–63.

Matthies, L., Xiong, Y., Hogg, R., Zhu, D., Rankin, A., Kennedy, B., Hebert, M., Maclachlan, R., Won, C., Frost, T., Sukhatme, G. S., McHenry, M., and Goldberg, S. 2002. A Portable, Autonomous, Urban Reconnaissance Robot. *Robotics and Autonomous Systems* **40**, 163–172.

Melhuish, C. 1999. Exploiting Domain Physics: Using Stigmergy to Control Cluster Building with Real Robots. In: D. Floreano, J.-D. Nicoud, and F. Mondada (eds.): *Proceedings of the 5th European Conference on Artificial Life (ECAL '99)*. pp. 585–595. Springer Verlag, Berlin, Germany.

Michel, O. 1998. Webots: Symbiosis between Virtual and Real Mobile Robots. In: *Proceedings of the First International Conference on Virtual Worlds (VW '98)*. Paris, France. pp. 254–263. Springer Verlag, Berlin, Germany.

Minar, N., Burkhart, R., Langton, C., and Askenazi, M. 1996. The Swarm Simulation System, a Toolkit for Building Multi-Agent Simulations. Working Paper 96-06-042. Santa Fe Institute,New Mexico, USA.

Murphy, R., Casper, J., Hyams, J., Micire, M., and Minten, B. 2000. Mobility and Sensing Demands in USAR. In: *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instrumentation (IECON 2000)*. Vol. 1. pp. 138–142. IEEE Press, Piscataway, New Jersey, USA.

Noda, I. 1995. Soccer server : a Simulator of Robocup. In: *Proceedings of the AI symposium '95*. pp. 29–34. Japanese Society for Artificial Intelligence, Tokyo, Japan.

Parker, L. E. 1998. ALLIANCE: an Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation* **14**(2), 220–240.

Parker, L. E., Bekey, G., and Barhen, J. (eds.) 2000. *Distributed Autonomous Robotic Systems 4*. Springer Verlag, Berlin, Germany. Collection of accepted papers to DARS 2000.

Pettinaro, G. C., Kwee, I. W., and Gambardella, L. M. 2003. Acceleration of 3D Dynamics Simulation of S-Bot Mobile Robots using Multi-Level Model Switching. Technical Report IDSIA-20-03. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA).

Pettinaro, G. C., Kwee, I. W., Gambardella, L. M., Mondada, F., Floreano, D., Nolfi, S., Deneubourg, J.-L., and Dorigo, M. 2002. Swarm Robotics: A Different Approach to Service Robotics. In: *Proceedings of the 33rd International Symposium on Robotics (ISR 2002)*. pp. 71–76. International Federation of Robotics, Paris, France.

Şahin, E., Labella, T. H., Trianni, V., Deneubourg, J.-L., Rasse, P., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., and Dorigo, M. 2002. SWARM-BOT: Pattern Formation in a Swarm of Self-Assembling Mobile Robots. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '02)*. Vol. 4. IEEE Press, Piscataway, New Jersey, USA.

Salemi, B., Shen, W.-M., and Will, P. 2001. Hormone Controlled Metamorphic Robots. In: W. H. K. et al. (ed.): *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*. pp. 4194–4199. IEEE Press, Piscataway, New Jersey, USA.

Sims, K. 1994. Evolving Virtual Creatures. In: *Proceedings of the 1994 ACM Conference on Computer Graphics (SIGGRAPH '94)*. pp. 15–22. ACM, New York, USA.

Stengel, R. F. 1991. Intelligent Failure-Tolerant Control. *IEEE Control Systems Magazine* **11**(4), 14–23.

Stone, H. W. 1996. Mars Pathfinder Microrover: a Low-Cost, Low-Power Spacecraft. In: *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics*. Madison, Wisconsin, USA.

Støy, K., Shen, W.-M., and Will, P. 2002. Global Locomotion from Local Interaction in Self-Reconfigurable Robots. In: C. T. M. Gini, W.-M. Shen and H. Yuasa (eds.): *Proceedings of the 7th International Conference on Intelligent Autonomous Systems (IAS-7)*. pp. 309–316. IOS Press, Amsterdam, The Netherlands.

Vandapel, N., Moorehead, S., Whittaker, W. R., Chatila, R., and Murrieta-Cid, R. 1999. Preliminary Results on the Use of Stereo, Color Cameras and Laser Sensors in Antarctica. In: P. I. Corke and J. Trevelyan (eds.):

Experimental Robotics VI, Proceedings of 6th International Symposium on Experimental Robotics (ISER 1999). Vol. 250 of *Lecture Notes in Control and Information Sciences*. Springer Verlag, Berlin, Germany.

Visentin, G., Winnendael, M. V., and Putz, P. 2001. Advanced Mechatronics in ESA Space Robotics Developments. In: T. J. T. et al. (ed.): *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*. Vol. 2. pp. 1261–1266. IEEE Press, Piscataway, New Jersey, USA.

Wilson, M., Melhuish, C., Sendova-Franks, A. B., and Scholes, S. 2004. Algorithms for Building Annular Structures with Minimalist Robots Inspired by Brood Sorting in Ant Colonies. *Autonomous Robots* **17**(2–3).

Yim, M., Duff, D. G., and Roufas, K. D. 2000a. PolyBot: a Modular Reconfigurable Robot. In: *Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*. Vol. 1. pp. 514–520. IEEE Press, Piscataway, New Jersey, USA.

Yim, M., Duff, D. G., and Roufas, K. D. 2000b. PolyBot: a Modular Reconfigurable Robot. In: *Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000)*. Vol. 1. pp. 514–520. IEEE Press, Piscataway, New Jersey, USA.

Yim, M., Zhang, Y., and Duff, D. 2002. Modular Robots. *IEEE Spectrum* **39**(2), 30–34.

Zhang, Y., Roufas, K. D., and Yim, M. 2001. Software Architecture for Modular Self-Reconfigurable Robots. In: T. J. T. et al. (ed.): *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS2001*. Vol. 4. pp. 2355–2360. IEEE Press, Piscataway, New Jersey, USA.

Zhang, Y., Yim, M., Eldershaw, C., Duff, D., and Roufas, K. 2003. Phase Automata: A Programming Model of Locomotion Gaits for Scalable Chain-type Modular Robots. In: C. S. G. Lee and J. Yuh (eds.): *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS 2003)*. pp. 2442–2447. IEEE Press, Piscataway, New Jersey, USA.

Francesco Mondada has an MSc in Microengineering and a PhD from the Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland). He is a member of the group who developed the Khepera mobile robot. He is co-founder of two companies: K-Team (robotics) and Calerga (scientific software). He has been president and director of K-Team for 5 years. He is currently a senior researcher at the Autonomous System Laboratory of the Swiss Federal Institute of Technology (EPFL), Lausanne. His interests include miniature robotic design, mechatronics, bio-inspired robotic research, the development of tools to perform this research and the transfer of robotics technology to industry.

Giovanni C. Pettinaro graduated in Computer Science at the Università di Milano, Italy, in 1989. He received the MSc in Information Technologies and the PhD in Intelligent Robotics from the University of Edinburgh in 1992 and 1996, respectively. He joined in 1997 ABB Corporation in Västerås, Sweden, where he worked as a consulting researcher in mechanical

design for ABB Robotics AB, Sweden. From 1998 till 2001, he worked as a researcher in mobile robotics and artificial intelligence within the group of Mechatronics and Software Systems at the Department of Industrial IT of ABB Corporate Research. In 2001, he joined the Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), a multi-disciplinary research institute within the Università della Svizzera Italiana (USI) and the Scuola Universitaria Professionale della Svizzera Italiana (SUPSI). Since then he has worked as a researcher in robotics control and 3D simulation within the SWARM-BOTS project. He is a member of IEEE, ACM, and AAAS. His research interests are in mobile robotics, distributed robot control, sensor fusion, robot architectures, robot learning, and swarm robotics.

Andrè Guignard got a Swiss federal certificate of proficiency as watchmaker and afterwards an engineering degree in electronics in Lausanne, Switzerland. He is a member of the group who developed the Khepera mobile robot. He is co-founder of the two companies K-Team (robotics) and CFG (electronics). He is currently senior engineer at the Autonomous System Laboratory of the Swiss Federal Institute of Technology (EPFL), Lausanne. His interests include miniature mechatronics applied to autonomous robotics, manufacturing techniques for miniature mechanics, and microsystems.

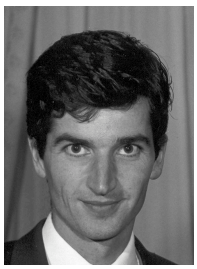
Ivo W. Kwee was born in Surabaya, Indonesia, in 1969. He received his Engineering degree in Technical Physics from the Delft University of Technology, The Netherlands, his MSc in Applied Physics from the Hokkaido University, Japan, and his PhD in Biomedical Physics from the University College London, UK, in 1991, 1994 and 2000 respectively. Currently he is researcher

at IDSIA, Switzerland, working on the SWARM-BOTS project, in particular on developing the 3D simulator and on developing learning algorithms for swarm-bot control.

Dario Floreano (MSc, 1992; PhD, 1995) is a professor of the Swiss National Science Foundation at the Swiss Federal Institute of Technology in Lausanne (EPFL) where he is the director of the Institute of Systems Engineering. His research activities include artificial neural networks, evolutionary robotics, swarm intelligence, bio-mimetic electronics and robotics, and artificial life. He held senior research positions at the National Research Council (CNR) in Rome, at the University of Stirling, at the Swiss Federal Institute of Technology in Lausanne, and at Sony Computer Science Laboratory in Tokyo. He published more than 100 peer-reviewed papers, authored 2 books, and edited 3 other books. His book with Stefano Nolfi, Evolutionary Robotics, has been reprinted by MIT Press three times since 2000. He co-organized three international conferences and joined the program committee of more than 60 other conferences. He delivered more than 100 invited talks all over the world to academic, industrial, and public audiences. He is on the editorial board of the journals Neural Networks, Genetic Programming and Evolvable Machines, Adaptive Behavior, Artificial Life, Connection Science, and IEEE Transactions on Evolutionary Computation. He is also co-founder and co-director of the International Society for Artificial Life (Inc., USA), member of the Board of Governors of the International Society of Artificial Neural Networks (Inc., USA), and member of various international societies. He frequently serves as advisor to the Research Division of the European Commission, to the U.S. National Science Foundation, and to other governmental and private institutions. He aims at building autonomous machines that have life-like properties, that is, machines which are able to reproduce, adapt, and evolve without human intervention.

Jean-Louis Deneubourg received his doctoral degree in Sciences (Chemistry) from the Université Libre de Bruxelles in 1979. From 1980 to 1989 he was a research fellow at the Service de Chimie Physique and at CENOLI, Université Libre de Bruxelles. From 1989, he has been a researcher of the FNRS, at CENOLI. From 2003, he is, with C. Detrain, co-director of the Department of Social Ecology. He is the author or co-author of around 180 papers, the coeditor of two books (From individual to collective behavior in social insects with J.M. Pasteels Birkhäuser;

Information Processing in Social Insects with C. Detrain, & J.M. Pasteels, Birkhäuser) and the co-author of one book (Self-Organization in biological systems with S. Camazine, N. Franks, J. Sneyd, E. Bonabeau & G. Theraulaz, Princeton, in press). He is (was) member of the editorial board of numerous international journals and was involved in the organization of many international conferences. His research concerns the collective intelligence in animal societies and their application to artificial and human systems. He has developed integrated experimental and theoretical tools for the study of complexity and self-organisation in biological systems. Current research projects deal with decision-making, information flow, building behavior and pattern formation in insect societies and in group-living organisms. Offshoots in applied research include collective robotics and transportation systems. He was awarded two prizes of the Belgian Academy and, in 2004, the french Prize Goeffroy Saint-Hilaire for his work on collective intelligence.



Stefano Nolfi is a senior researcher at the Institute of Cognitive Sciences and Technologies of the National Research Council (CNR) in Rome, where he leads the Laboratory of Artificial Life and Robotics. He is also associate professor at Lumsa University in Rome. His research interests are in the field of neuro-ethological studies of adaptive behavior in natural and artificial agents and include: evolutionary robotics, artificial life, complex systems, neural networks, and genetic algorithms. The main tenets underlying his work are that behavioural strategies and neural mechanisms are understood better when an organism (living or artificial) is caught in the act, that is, when one considers situated and embodied agents in their interaction with the environment; and that to understand how natural agents behave and to build useful artificial agents one should study how living organisms change, phylogenetically and ontogenetically, as they adapt to their environment. He has published more than 70 peer-reviewed articles and a book on Evolutionary Robotics.

Luca Maria Gambardella is Research Director at IDSIA. His major research interests are in the area of optimization, simulation, robotics learning and adaptation, applied to both academic and real-world problems. In particular he has studied and developed several ant colony optimisation algorithms to solve scheduling and routing problems. In these domains, the best-

known solutions for many benchmark instances have been computed. He is responsible for IDSIA robotics projects. He has led several research and industrial projects both at national (Swiss) and European level.

Marco Dorigo received the Laurea (Master of Technology) degree in industrial technologies engineering in 1986 and the doctoral degree in information and systems electronic engineering in 1992 from Politecnico di Milano, Milan, Italy, and the title of Agrégé de l'Enseignement Supérieur, from the Université Libre de Bruxelles, Belgium, in 1995. From 1992 to 1993 he was a research fellow at the International Computer Science Institute of Berkeley, CA. In 1993 he was a NATO-CNR fellow, and from 1994 to 1996 a Marie Curie fellow. Since 1996 he has been a tenured researcher of the FNRS, the Belgian National Fund for Scientific Research, and a research director of IRIDIA, the artificial intelligence laboratory of the Université Libre de Bruxelles. He is the inventor of the ant colony optimization metaheuristic and one of the founders of the swarm intelligence research field. Its current research interests include metaheuristics for discrete optimization, swarm intelligence and swarm robotics. Dr. Dorigo is an Associate Editor for the journals: Cognitive Systems Research, IEEE Transactions on Evolutionary Computation, IEEE Transactions on Systems, Man, and Cybernetics, and Journal of Heuristics. He is a member of the Editorial Board of numerous international journals, including: Adaptive Behavior, AI Communications, Artificial Life, Evolutionary Computation, Information Sciences, and Journal of Genetic Programming and Evolvable Machines. He is the author of three books: *Robot Shaping*, MIT Press, 1998; *Swarm Intelligence*, Oxford University Press, 1999; and *Ant Colony Optimization*, MIT Press, 2004. In 1996 he was awarded the Italian Prize for Artificial Intelligence and in 2003 the Marie Curie Excellence Award for his work on ant colony optimization and ant algorithms.