Evolving Formation Movement for a Homogeneous Multi-Robot System: Teamwork and Role-Allocation with Real Robots.

Matt Quinn, Lincoln Smith, Giles Mayley and Phil Husbands

C.S.R.P. 515

May 2002

ISSN 1350–3162

Cognitive Science
Research Papers

Evolving Formation Movement for a Homogeneous Multi-Robot System: Teamwork and Role-Allocation with Real Robots.

Matt Quinn, Lincoln Smith, Giles Mayley and Phil Husbands

May 2002

Abstract

In recent years a number of researchers have successfully applied artificial evolution approaches to the design of controllers for autonomous robots. To date, however, Evolutionary Robotics research has focussed almost exclusively on the design of single-robot systems. We are interested in the evolution of controllers for multi-robot systems that are capable of exhibiting cooperative and coordinated behaviour. We report on recent work in which we employed artificial evolution to design neural network controllers for small, homogeneous teams of mobile autonomous robots. The robots are evolved to perform a formation movement task from random starting positions, equipped only with infrared sensors. The dual constraints of homogeneity and minimal sensors make this a non-trivial task. We describe the behaviour of a successful evolved system, in which robots work as a team, adopting and maintaining distinct but interdependent roles in order to achieve the task. We believe this to be the first successful use of evolutionary robotics methodology to develop cooperative, coordinated behaviour for a real multi-robot system.

All correspondence to Matt Quinn, e-mail: matthewq@cogs.susx.ac.uk.
Giles Mayley is currently based at Mathematique Appliqué, Sussex Innovation Centre, University of Sussex, Brighton, BN1 9QG, U.K. The other authors are members of the School of Cognitive and Computing Sciences (COGS) and the Centre for Computational Neuroscience and Robotics (CCNR) at the University of Sussex.

1 Introduction

In this paper we report on our recent work evolving controllers for robots which are required to work as a team. The word 'team' has been used in a variety of senses in both the multi-robot and the ethology literature, so it is appropriate to start the paper with a definition. We will adopt the definition given by Anderson and Franks in their recent review of team behaviour in animal societies. They identify three of defining features of team behaviour. Firstly, individuals make different contributions to task success, i.e. they must perform different subtasks or roles (this does not preclude more than one individual adopting the same role). Secondly, individual roles or sub-tasks are interdependent (or "interlocking") requiring structured cooperation; individuals operate concurrently, coordinating their different contributions in order to complete the task. Finally, a team's organisational structure persists over time, although its individuals may be substituted, or swap roles (Anderson and Franks, 2001).

The designer of a multi-robot team faces a number of challenges. One of which arises because a team is a structured system. Robots must be designed to behave in such a way that the team will both become and remain appropriately organised. This requires ensuring that all the individual roles or sub-tasks are appropriately allocated. One way to address this problem is to design a team in which each individual's role is predetermined (see e.g. Balch and Arkin, 1998). This has the additional advantage of specialisation, in that each robot's behavioural and morphological design can be tailored to its particular task. In natural systems, this type to team organisation is often found amongst eusocial insects, where roles may be caste-specific (see e.g Detrain and Pasteels, 1992). Despite the organisational advantages of system heterogeneity and the efficiency benefits of specialisation, we are interested in the design of homogeneous systems. In a homogeneous multi-robot system, each robot is built to the same design, and has an identical controller. Our interest in homogeneous robot teams stems from their potential for system-level robustness and graceful degradation due to the interchangeability of team members (although this is not an issue that we will be addressing in this paper). Since each robot is capable of performing any role or sub-task, homogeneous systems are potentially better than heterogeneous teams at coping with the loss of an individual member. Lack of role specialisation also has potential benefits for organisational flexibility (Stone and Veloso, 1999). However, from the perspective of team organisation, the constraint of homogeneity makes the design task more difficult. In a homogeneous team there are no differences between robots' control systems or morphologies which can be exploited for the purposes of team organisation. Other mechanisms must be employed to facilitate the dynamic allocation and coordination of roles.

Dynamic role allocation and closely coordinated co-operation are two areas which have been addressed by a number of researchers in the field of multi-robot systems, resulting in successful implementations of such tasks such as cooperative transport (Chaimowicz et al., 2001), robot football (Stone and Veloso, 1999), and coordinated group movement (Matarić, 1995). Solutions have relied

heavily on the use of essentially global information, shared by radio communication. For example, in Matarić's implementation of coordinated movement with homogeneous robots, robots made use of a common coordinate system (through radio beacon triangulation) and exchanged positional information via radio communication in order to remain coordinated (Matarić, 1995). Mechanisms for dynamic task or role allocation rely on communication protocols by which robots globally advertise or negotiate their current (or intended) roles (e.g. Chaimowicz et al., 2001; Gerkey and Matarić, 2001; Matarić and Sukhatme, 2001; Stone and Veloso, 1999).

Our work differs from that of these researchers. We wish to design teams in which system-level organisation arises, and is maintained, solely through local interactions between individuals which are constrained to utilise minimal and ambiguous local information. Systems capable of functioning under such constraints have some interesting potential engineering applications (see, for example, Hobbs et al.'s (1996) discussion of the need for minimal systems in the space industry). However, they are also interesting from an adaptive behaviour perspective, providing an example of a phenomenon often referred to as 'self-organising' or 'emergent' behaviour (Camazine et al., 2001).

Imposing the joint constraints of homogeneity and minimal sensors leaves us with a complex design task. One which cannot easily decomposed and addressed by conventional 'divide and conquer' design methodologies. Instead, it is a problem exhibiting significant interdependence of its constituent parts. For this reason, we have adopted an evolutionary robotics approach, employing artificial evolution to automate the design process, since such an approach is not constrained by the need for decomposition or modular design. It is worth noting that—insofar as we are aware—the work reported in this paper represents the first successful use of evolutionary robotics methodology to develop cooperative, coordinated behaviour for a real multi-robot system. (See Meyer et al. (1998) and Nolfi and Floreano (2000) for good surveys of evolutionary robotics research).

The work which we will describe in this paper represents our first experiments in the evolutionary design of homogeneous multi-robot teams. We used three robots, each minimally equipped with four active infra-red sensors, and two motor-driven wheels. There is no radio communication between robots. Controllers were evolved to perform a formation movement task, in a featureless, obstacle-free environment, starting from random initial positions. The robots and their task are introduced in more detail in section 2. In order to build a fast-running simulation from which we could successfully transfer evolved controllers from simulation to the real world we adopted Jakobi's minimal simulation methodology (Jakobi, 1998b). Section 3 details how the minimal simulation was constructed. Robots were controlled by neural networks, loosely based on spiking neuron models; the topology of these networks, along with weights, thresholds and decay constants, was placed under evolutionary control, with minimal constraints being placed on the size and connectivity. The networks, their encoding, and the associated evolutionary machinery are described in section 4. Finally, section 5 presents description and analysis of the

successful behaviour of one of the evolved teams in some detail, showing that task success is dependent on the robots adopting and maintaining separate roles performing as team, in accordance with definition given at the beginning of this paper.

2 The Robots and their Task

2.1 The Robots

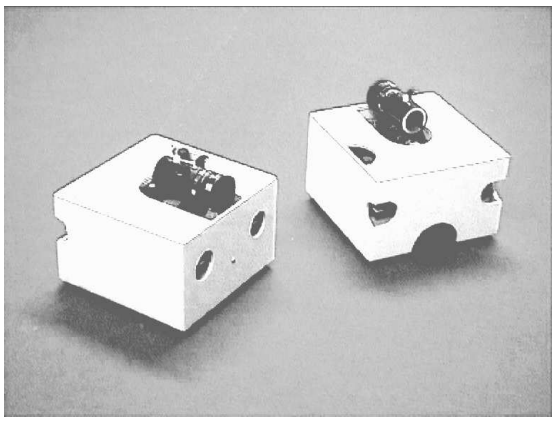


Figure 1: Two of the robots used in the experiment. The cameras shown are not used for the experiments described in this paper.

Two of the robots used in these experiments are shown in figure 1. Each robot's body is 16.75 cm wide by 16.75 cm long by 11 cm high (this excludes the additional height of the camera). Two motor-driven wheels, made of foam-rubber, are arranged one on either side of the robot and provide locomotion through differential drive; the robots have a top speed of just over 6cm/s. An unpowered castor wheel, placed rear-centre, ensures stability. In the experiments described in this paper, a robot's *only* source of sensory input comes from its four active infrared sensors, each comprising a paired infrared emitter and receiver. Each robot has two infrared sensors at the front and two at rear, as illustrated in figure 2. Although the robots are also equipped with a 64 pixel linear CCD array camera (shown in the photograph), a 360 degree electronic compass, and wheel rotation sensors (i.e. shaft encoders), the controllers we evolved were prevented from making use of these devices.

The robots are controlled by a host computer, with each robot sending it's sensor readings to, and receiving it's effector activations from this machine via a radio communication system. Note that radio communication is only between a controller and its physical robot, not between different robots. Each robot uses a 80C537-based micro-controller board for low-level control. The host computer is responsible for running the controller for each robot, updating each controller's inputs with the sensor readings from the appropriate robot, and transmitting the controller's output to the robot. It should be noted that although the physical instantiation of the robots has been implemented as a

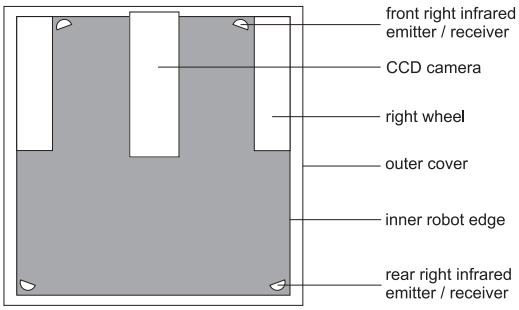


Figure 2: Plan view of a robot, drawn to scale, showing location of the IR sensors and wheels.

host/slave system, conceptually the robots are to be considered as independent, autonomous agents by virtue of the logical division of control into distinct and self-contained controllers on the host machine.

2.2 Infrared Sensors

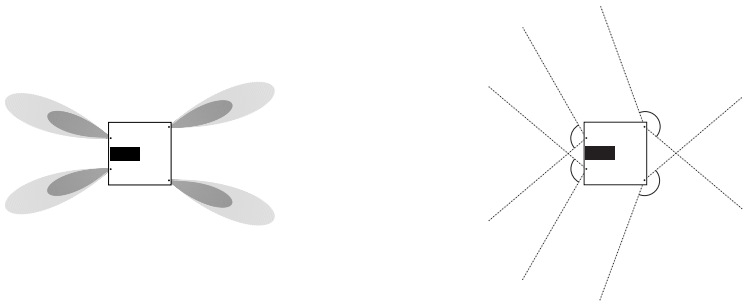


Figure 3: **Left:** An approximate illustration of the extent to which reflected IR can be sensed (dark grey area), and the extent to which IR beam is perceptible to other robots (light grey area). **Right:** The angles from which a robot can perceive the IR emissions of others.

The reader may not be familiar with the limitations of active infrared sensors, especially those peculiar to a multi-robot scenario, so we will address them in some detail. An active infrared sensor comprises a paired infrared emitter and receiver. Its normal function is to emit an infrared beam and then measure the amount of infrared which reflects back from nearby objects. In this way our robots can use their sensors to detect other robots up to a maximum of about 18cm (i.e. just over one body length away). The dark grey beams in the left-hand panel of figure 3 approximately indicate the limited areas in which a robot can detect other robots in this manner. IR sensors are sometimes referred to

as proximity sensors, however this is somewhat misleading. Whilst the sensor reading due to reflected IR is a non-linear, inverse function of the distance to the object detected (see figure 4), it is also a function of the angle at which the emitted beam strikes the surface of the object, and of the proportion of the beam which strikes that object. It is because an IR sensor reading combines these three factors into a single value that, even in normal function, sensor readings are ambiguous.

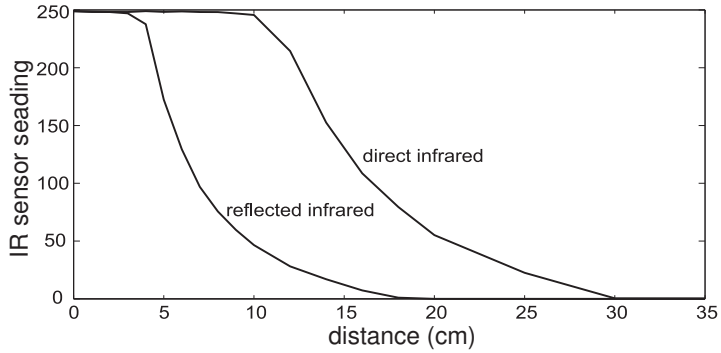


Figure 4: How the maximum infrared sensor reading varies with distance, for reflected IR (normal sensor function) and for directly received IR (interference from other robots' sensor emissions). Note that direct IR can be detected from almost twice the range at which a robot can detect an object by reflected IR.

The ambiguity of IR sensors is significantly increased in a multi-robot scenario, because the robots' sensors interfere with one another. Since each robot is constantly emitting IR, a robot's IR emissions can also be directly sensed by other robots. The light grey beams in the left-hand panel of figure 3 indicates the approximate area in which a robot's infrared emissions may be directly detected by other robots. The maximum range at which emissions can be detected is approximately 30cm—almost twice the range at which a robot can detect an object by reflected IR (see figure 4. The right-hand panel of the same diagram illustrates the range of angles at which a robot can receive the IR emissions of other robots. The sensor value due to receiving another's IR emissions is also the combined function of a number of factors: It will depend on the distance between the robots, but also the angle at which the emitted beam strikes the other robot's receiver, and which portion of the beam strikes the receiver (IR is significantly more intense at the centre of the beam than at the edges). Readings due to direct IR are thus ambiguous for the same reasons as reflected IR. However, ambiguity is compounded by the fact that, to the robot, readings due to reflected IR are indistinguishable from those due to the reception of IR emissions of other robots. Moreover, a sensor reading may be the result of a combination of both reflected and direct IR and it may be due to one or both

of the other robots.

2.3 The Task

The task with which we present the robots is an extension of that used in previous work which involved two simulated Khepera robots (Quinn, 2001a,b). Adapted for three robots, the task is as follows: Initially, the three robots are placed in an obstacle-free environment in some random configuration, such that each robot is within sensor range of the others. Thereafter, the robots are required to move, as a group, a certain distance away from their initial position. The robots are not required to adopt any particular formation, only to remaining within sensor range of one another, and to avoid collisions. During evolution robots are evaluated on their ability to move the group centroid one metre within the space of three simulated minutes. However, our expectation was that a team capable of this would be able to sustain formation movement of much longer periods. The robots are not required to adopt any particular formation, only to remaining within sensor range of one another, and to avoid collisions. Since the robots start from initial random configurations, we anticipate that successful completion of the task will entail two phases. The first entailing the team organising itself into a formation, and the second entailing the team moving whilst maintaining that formation.

From the characterisation of the robots' sensors in the previous section, it should be clear that these impose significant constraints. They provide very little direct information about a robot's surroundings. Any given set of sensor input can be the result of any one of large number of significantly different circumstances. Furthermore, outside the limited range of their IR sensors, robots have no indication of each other's position. Any robot straying more than two body-lengths from its teammates will cease to have any indication of their location (which can also occur at much closer distances, as can be seen from figure 3). Of course, a robot controller may employ strategies to overcome some of the limitations of it sensors. For example, additional information can be gained by strategies which combine sensing and moving, and the integration of sensor input over time. However, it should be clear that the team's situation contrasts strongly with previous work in which robots utilised shared coordinate systems and global communication. It is worth noting that biological models of 'self-organising' coordinated movement typically assume that agents are presented with significantly more information about their local environment than these robots have. For example, in models of flocking and shoaling, agents are typically assumed to have ideal sensors and are provided with the location, velocity and orientation of their nearest neighbours (see Camazine et al. (2001) for an extensive review of such biological models, see also Ward et al. (2001) for a recent evolutionary model).

A further constraint is placed on the team by their homogeneity. Consider that the team will move from their initial random configuration into the formation in which they will maintain whilst moving. In so doing it is inevitable that different robots will come to adopt functionally distinct roles (for example, a

leader and two followers). For this reason, the team will encounter the problem of role allocation. For a heterogeneous team, as we noted earlier, this problem need not arise, since each individual can be assigned a fixed, predesignated role (see Balch and Arkin (1998) for an example of how team heterogeneity can be exploited in this way to facilitate formation movement). For a homogeneous team, of course, such a strategy is not an option, since each robot has an identical controller. The lack of any intrinsic differences between the controllers makes the task considerably harder. Roles will have to be dynamically allocated; the robots will have to find some way to 'decide' which robot will take which role within the formation. In the absence of intrinsic differences, the only exploitable differences that exist between the robots are the differences in their relative positions, and the resulting differences in sensory input. The controllers will need to find some way to exploit these differences, through interaction, in order to successfully complete their task. The problem of role allocation is, of course, made all the more challenging by the poverty of the robots' sensory input.

3 Building the Simulation

Controllers were initially evolved in simulation, before being transferred to the real robots. A big problem with evolving in simulation is that robots may become adapted to inaccurate features of the simulation, not present in the real world (Brooks, 1992; Husbands and Harvey, 1992). However, building completely accurate simulation model of the robots and their interactions would be an onerous, and potentially impossible task, moreover, it would be unlikely that such a simulation would have significant speed advantages over evolving in the real world (Husbands and Harvey, 1992; Matarić and Cliff, 1996). To avoid such problems we employed Jakobi's minimal simulation methodology (Jakobi, 1997, 1998b). This enabled us to build a relatively crude, fast-running simulation model of the robots and their interactions, based on a relatively small set of measurements. The parameters of this model were systematically varied, within certain ranges, between each evaluation of a team. Parameters included, for example, the orientation of each robots' sensors, the manner in which a robot's position was affected by motor output, and the effects of IR interference. Whilst it was generally either difficult or time-consuming to measure parameters needed for the simulation with great accuracy on the robots, it was relatively easy to specify a range within which each of the parameters lay, even if that range was wide. Varying parameters within these ranges meant that a robot of capable of adapting to the simulation would be adapted to a wide range of possible robot-environment dynamics, including those of the real world. In addition to compensating for inaccuracies in our measurements, variation was used in the same way to compensate for inaccuracies in our modelling, since we were able to estimate the error due to these inaccuracies and adjust parameter ranges to compensate. More importantly, this approach allowed us to sacrifice accuracy for speed and employ cheap, inaccurate modelling where more accurate modelling would have incurred significant computational costs.

That fact that we are attempting design of controllers for a homogeneous system had particular consequences for the design of our simulation. As stated at the beginning of this paper, a homogeneous system is one in which all robots are built to the same design, each having essentially the same physical attributes (morphology, sensor and motor capabilities etc.). Of course, no real system will contain *identical* robots. Robots will inevitably differ to some extent, due to component variations, calibration differences and general accumulated wear and tear. Our robots are no exception; indeed, the fact that the robots were hand-built has tended to exacerbate inter-robot differences. There are, for example, minor differences in sensor placement and motor speeds between robots. Physical differences can result in behavioural differences, and the effects of even small behavioural differences can be progressively amplified over the course of the robots' interactions with each other and with their environment. For this reason, physical variation between robots is an issues, not just for the designers of heterogeneous systems, but also for the designers of homogeneous systems (a point recently stressed by Balch and Parker, 2000).

In the context of our simulation, we identified two interrelated issues arising from the existence of inter-robot variation. First, since we are aiming to evolve identical controllers to control each robot, we need controllers to be capable of functioning effectively with each individual robot, despite the existence of differences between them. A controller which evolves to become closely adapted to the specific characteristics of one particular robot may well prove less effective at controlling a different robot[1]. Thus, the existence of variation between robots needs to be reflected in the simulation, to allow evolving controllers to become adapted to its existence. This brings us to the second issue. We need to avoid the possibility that robots might evolve to exploit a particular set of differences that exist on the real robots; this would ultimately be a heterogeneous solution to the task. Ideally, we would like solutions which work on any three robots built to this specification, solutions that are successful in spite of inter-robot variation, rather than because of it. We addressed both of these issues in the context of the minimal simulation methodology, incorporating our measurements and estimates of the differences between the robots into the ranges of variation associated with the relevant simulation parameters. Evolving controllers were thus exposed to differences within the ranges we observed on the four robots that we own. However, the particular set of differences between robots was also varied between trials, so that successful controllers would be capable of functioning given any degree of variation within these ranges. In this way, successful solutions will have be robust to the existence of inter-robot variation, but unable to rely any on particular set of differences between robots. Note that this is a property that we can test, at least minimally, by examining the performance of the evolved controllers on different combinations of three out of the four existing robots.

The remainder of this section gives a detailed description of the implemen-

[1] See Thompson's work with evolvable hardware for a particularly pronounced example of this type of phenomenon (Thompson, 1996; Thompson and Layzell, 1999).

tation of our minimal simulation. The simulation was updated at a rate of 5Hz, roughly the same update rate as the real robots. IR sensors were updated using look-up tables. The first two sections below describe how look-up tables were generated and employed for updating sensors with respect to reflected IR (section 3.1) and to directly received IR (section 3.2). Next, we describe how direct and reflected IR readings are amalgamated to produce a single sensor reading (section 3.3). Finally, we explain the way in which robot movement is modelled (section 3.4) and how collisions are handled (section 3.5).

3.1 Reflected IR

The robots' IR sensors are designed to emit IR and measure the IR that is reflected back from any objects within the range of the sensor. In the experiment described in this paper, the only objects which will come within range of a robots sensors are other robots. To this end, we needed to characterise the response of infrared sensors to the proximity and relative orientation of other robots, and incorporate this characterisation into the simulation. As described in section 3.1.1, reflected IR was characterised by first measuring the IR received by a sensor pointing at a white wall from a variety of angles and distances and then extrapolating from these measurements by means of a crude ray-tracing procedure and some assumptions about (unmeasured) properties of the IR emitter. Sensor values generated by this method were incorporated into a lookup table; section 3.1.2 describes how the lookup table was used update the robots' sensors in the simulation, and explains our minimalist solution to the problem of sensor occlusion.

3.1.1 Measurement and Extrapolation

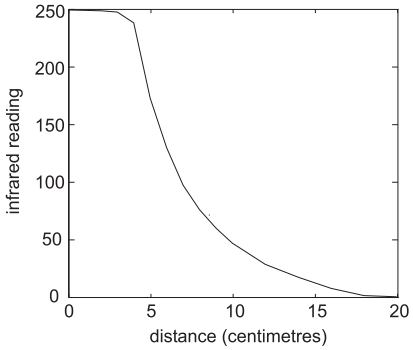


Figure 5: Reflected IR: The maximum sensor reading recorded at each distance

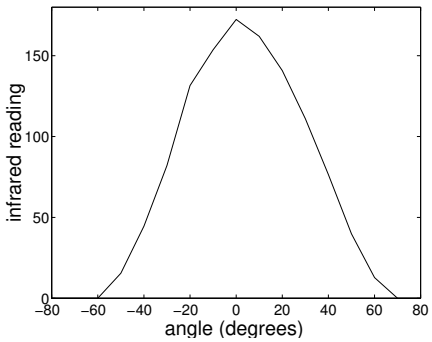


Figure 6: Reflected IR: An example of how sensor readings varies angle θ (distance is 5cm)

The first step in characterising reflected IR consisted in taking measurements of the IR received by a sensor pointing at a white wall. The wall was wide enough

that all of the beam emitted from the sensor would strike the wall. IR reflected from a wall can be calculated as a function of the distance between sensor and wall, d, and angle of the sensor relative to a line orthogonal to the wall, θ, as illustrated in figure 7 . We took measurements for $d = 0, 2, \ldots 20$cm and $\theta =$ -90, -70,... 90 degrees (i.e., the full range for which the wall could be sensed). The variation of IR reading with distance for reflected IR is illustrated in figure 5, and an example of how readings vary with the angle θ is shown in figure 6. Only one set of measurements was taken, that is, we measured one sensor on one robot only.

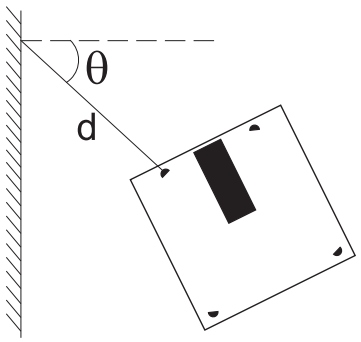


Figure 7: Reflected IR was treated as a function of the distance, d, and angle of the sensor relative to a line orthogonal to the wall, θ

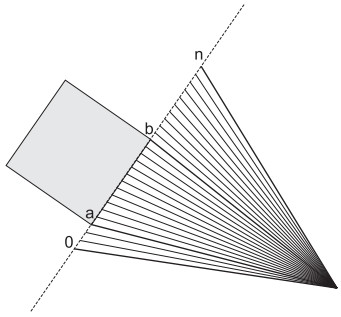


Figure 8: Illustrating the 2-d ray-tracing procedure. Of all the rays, **0** through **n**, only a subset, **a** through **b**, hit both the robot and the wall

A function, $V_{wall}(d, \theta)$, was devised which generated IR values with a fairly reasonable fit to the data. This characterisation of reflected IR from walls was used to estimate the IR reflected from the side panel of a robot. To this end, we used a simple, two-dimensional, ray-based approach based on that employed by Jakobi (1994). We treat the side-panel of a robot as a sub-section of a wall, as illustrated in figure 8. Given the expected reflected IR reading for an entire wall given d and θ, and it is clear that the sensor reading for the subsection of the wall comprising the robot's side-panel is some proportion (≤ 1) of $V_{wall}(d, \theta)$. In order to estimate this proportion we draw $n = 500$ rays which are evenly spread across the emitter beam. Each ray is then given a contribution value such that, if a ray travels d_r before it strikes a surface at an angle θ_r , then its contribution to the final IR value is assumed to be proportional to $V_{wall}(d_r, \theta_r)$ multiplied by an angular intensity factor. (This scaling factor, defined below, is necessary because the intensity of light emission from the diode varies in intensity with the angle of ray emission.). We use the ratio between the sum of the contributions of all rays hitting the side-panel and the sum of the contribution of all rays hitting the entire wall to estimate what proportion of $V_{wall}(d, \theta)$ comprises the sensor reading for the IR reflected from the side panel. More formally, the sensor

reading from a single panel is taken to be:

$$V_{wall}(d_w, \theta_w) \frac{\sum_{i=a}^{b} I(\alpha_i) \,.\, V_{wall}(d_i, \theta_i)}{\sum_{i=0}^{n} I(\alpha_i) \,.\, V_{wall}(d_i, \theta_i)}$$

Here a and b are, respectively, the indices of the first and last rays hitting the robots side-panel, as illustrated in figure 8. α_i is the angle at which the i^{th} ray is emitted from the sensor, and I is the intensity factor function, (range [0:1]), which was introduced previously. We had little information on exactly how the ray intensity varied with the angle of emittance, except that maximum intensity occurred in the direction which the emitter is facing and that intensity was zero beyond the maximum angle of emittance (i.e. $\pm\frac{\pi}{8}$). We assumed that, were our model fully 3-dimensional, the intensity of a ray would vary with the cosine of its angle of emission, if that angle was first mapped onto the range $\pm\frac{\pi}{2}$. However, since we are working with a 2-dimensional approximation, our 'rays' are, in effect, formed by integrating over horizontal slices through the conical beam of IR emitted by the sensor. Treating our rays as slices means that the intensity factor, I, as function of the angle of ray emission, α, becomes:

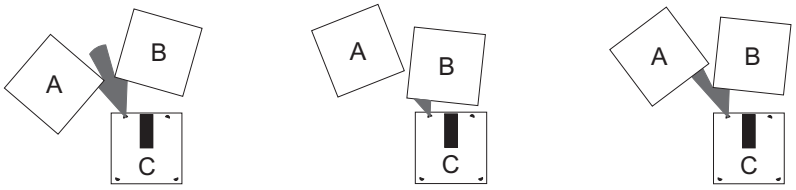
$$I(\alpha) \,=\, sin\left[\, \frac{\pi}{2}\, cos\big(\, 2\alpha \,\big) \,\right]$$

The procedure described above allowed us to estimate the IR sensor reading generated by IR reflected from the side-panel of robot. From many positions, two side-panels (of the same robot) will fall within the IR beam. In such cases, the value for each panel is calculated separately, in the manner described above, and then these values are summed.

3.1.2 Reflected IR in the Simulation

Ray-tracing is a relatively computationally expensive technique to employ in an evolutionary simulation. So instead we used a look-up table to assign sensor values during simulation. The ray-tracing procedure just described was used to generate the lookup table. The lookup table stored the sets of IR sensor values that one robot would receive if another robot were within sensor range, indexed by the distance between the robots (45 indices at 0.5cm intervals), and the orientation of each robots, relative to the other's position (100 indices each, at $2\pi/100$ intervals in both cases). At the beginning of each trail, an angular offset, in the range ± 5 degrees, and distance offset $\pm$1cm, was generated at uniform random and associated with each IR sensor of each robot. When looking up a sensor's value in the look-up table, the distance between the robots and the orientation of the robot whose sensor was being updated were first adjusted by the appropriate offset associated with that sensor. In effect then, these offsets reposition and reorient each sensor of each robot at the beginning of every trial. This strategy was intended to smooth over the inevitable gaps between our estimated readings and the those which real IR sensors would receive. In addition, it reflected the fact that there was some variation (between robots) in the exact positioning and orientation of their IR sensors.

The IR lookup table stores only the sensor values which are generated by the presence of one other robot. Nevertheless, the simulation must be able to deal with the many occasions when two robots are simultaneously within range of a third robot's sensor(s). In such cases, two sets of readings can be taken from the look-up table, one for each of the two robots that are within range of the third robot's sensors. However, the problem arises of how these two sets of values are to be combined to yield a single set of sensory readings. This is a problem because one robot may occlude the other, and the look-up table provides no information as to whether occlusion is taking place. Detecting occlusions and calculating their effect would be a significant additional computational expense in the simulation—it would necessitate ray-tracing across the path of the IR beam. To avoid this, we opted for a minimal and computationally inexpensive solution to the problem. Namely, occlusion is accommodated but not simulated.



i. No occlusion. ii. Full occlusion. iii. Partial occlusion.

Figure 9: When two robots, A and B, are in range of a sensor belonging to a third robot, C, the look-up table gives two independent values for C's sensor; the sensor value due to A's proximity, S_A, and the value due to B's proximity, S_B. In an accurate model, occlusion would be taken into account when combining S_A and S_B. Thus, in panel i, where there is no occlusion, C's sensor value should be $S_A + S_B$. In panel ii, where A is fully occluded by B, C's sensor value should be S_B. In panel iii, where A is partially occluded by B, C's sensor value should be S_B, plus some proportion p, $(0 < p < 1)$, of S_A.

Consider that two robots, A and B, are both within range of a sensor belonging a third robot, C. Using the look-up table, we can establish C's expected sensor reading due to the presence of A, call this S_A, and its reading due to B, call this S_B. We need to establish the the combined, final value, S_{final}. We proceed by finding the possible range of values within S_{final} can lie. The two extremes, non-occlusion and full occlusion, correspond to the extremes of the range of possible values for S_{final}. If we know that neither robot occluded the other, as in the situation illustrated in figure 9(i), then we could simply to sum the sensor value due to each, i.e. $S_{final} = S_A + S_B$. This is the maximum possible value of S_{final}. Alternatively, if one robot fully occludes the other, as for example in figure 9(iii) B fully occludes A, then there will be no sensor contribution from the occluded robot and the final value will simply be the reading due to occluding robot; thus, in the scenario depicted in figure 9(iii), $S_{final} = S_A$. Hence, the minimum possible value for S_{final} is the value due to the closest

robot. In the case of partial occlusion, illustrated in figure 9(ii), S_{final} will be the value due to the occluding robot plus some proportion (< 1) of the value due to the partly occluded robot. To calculate the exact value due to the partly occluded robot would require ray-tracing across the part of the beam hitting that robot, or some other form of numerical integration. Nonetheless, it is clear that the cases of partial occlusion yield values of S_{final} greater than that of full occlusion, but less than in cases where there is no occlusion. Therefore, given values for S_A and S_B we know range within which S_{final} lies. In the simulation, we set the combined, final value, S_{final} as:

$$S_{final} \;=\; \max(S_A, S_B) \;+\; c_R\big[\; S_A + S_B - \max(S_A, S_B) \;\big]$$

where c_R is a robot-specific scalar set randomly in the range [0:1] for each robot at the beginning of each trial. Note that to avoid calculating which robot is closest to the sensor we make the approximation that the larger value of S_A and S_B will be the value due to the closest robot[2]. Note that this method will hardly ever produce the correct value for S_{final} and controllers will have to adapt so as to be capable of dealing with any value of S_{final} within range specified above. However, any controller capable of this will be capable of dealing with correct values of S_{final}, since these will always lie within that range.

3.2 Direct IR

3.2.1 Measurement and Extrapolation

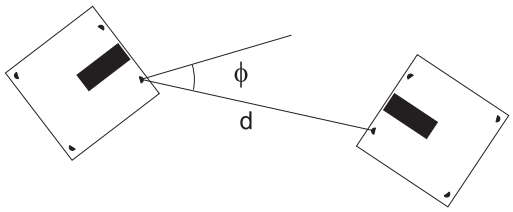


Figure 10: Direct IR was measured as a function of the distance, d, and the angle, ϕ.

Measuring direct IR was more difficult than measuring reflected IR, since it required lining up two robots' sensors, and eliminating any reflected IR. Readings were taken with the emitting robot's emitter facing directly at the receiving robot's receiver, and we varied the angle of reception ϕ and the distance between the sensors d, as illustrated in figure 10. We took measurements at d=0,2, ... 20,

[2]The larger value will typically the produced by the closest robot. Exceptions can occur, but errors due to this approximation will be small, and the correct value will anyway typically lie within the range of variation that we have specified here, or be covered by the other variation added to the generation of sensor values.

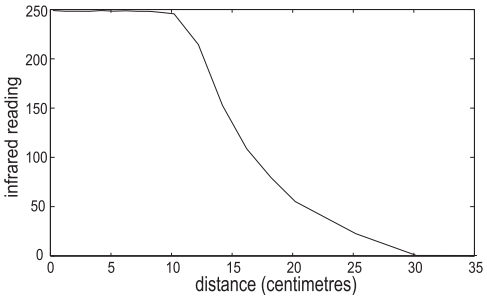


Figure 11: Direct IR: The maximum sensor reading recorded at each distance.

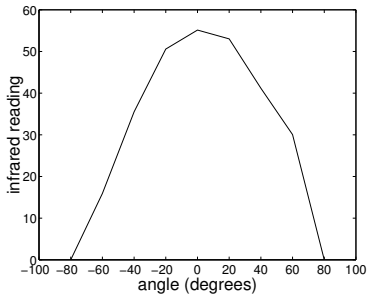


Figure 12: Direct IR: An example of how sensor readings varies with angle ϕ (distance is 18cm)

25, 30, 35cm and at angles ϕ = -80, -60, ... 80 degrees, the general shape of these readings can be seen in figures 11 and 12. We generated a function, $V_{direct}(d, \phi)$, to fit these data. Since these values only cover cases where an emitter is pointed directly at a receiver we need to cover those cases where the beam of IR is not directly centred on a receiver. Hence we factored in the angle β, between the direction of the emitter and a line drawn between the emitter and the receiver, and employed the angle of emission intensity factor, I, defined above, giving a final IR reading, R_{direct} as a function of d, ϕ and β, namely:

$$R_{direct}(d, \phi, \beta) \;=\; V_{direct}(d, \phi).I(\beta)$$

3.2.2 Direct IR in the Simulation

As with reflected IR, direct IR was incorporated into the simulation by means of a look-up table which allowed one robot's sensors to be updated as a function of its position relative to a second robot. As before, this look-up table was indexed by the distance between the robots (70 indices at 0.5cm intervals) and the orientation of each robot, relative to the other's position (100 indices each, at $2\pi/100$ intervals in both cases). Sensor-specific offsets were used when looking up values for each sensor, in exactly the same way as with the reflected IR, and for the same motivations. At the beginning of each trail, an angular offset, in the range ± 5 degrees, and distance offset $\pm$1cm, was generated at uniform random and associated with each IR sensor of each robot. Note, however that the offsets used for the direct IR are distinct from those used for reflected IR. Each sensor of each robot thus had two pairs of offsets associated with it, one pair was used for updating direct IR, the other for reflected. This resulted in a degree of variation into the exact relationship between received and direct IR within the simulation.

Occlusion is also a problem when simulating direct IR, but the solution adopted for reflected IR is not applicable here. With reflected IR, occlusion placed a sensor value in a relatively constrained range between that IR reflected

from the closed robot and the sum of the IR reflected from both. However, in the case of direct IR, occlusion can reduce an IR value from the maximum possible value down to zero. Hence, we contrived to model the occlusion of direct IR, albeit as minimally as possible. The procedure we adopted required making some simplifying assumptions, which reduce the accuracy of our model, but made the occlusion-handling process much faster. Our first simplifying assumption is based on the following observation: When one robot receives IR emitted by a second robot, this usually only involves one emitter and one receiver. This is simply due to the positions of the robots' sensors and the properties of the IR beam. In certain positions the beam emitted from one receiver may strike two receivers, or two beams (emitted by the same robot) may strike two separate receivers. However, in such cases, the amount of IR received by one, or both, of the two receivers will be low. For example, for one beam to strike both receivers, the two robots must either be far apart, or the beam must strike at an acute angle, or only the edges of the beam will strike one or both receivers. This observation enables us to take a simplifying step. For any given position in which one robot receives IR emitted by a second robot, we will only check for occlusion of the line between the most significant receiver and emitter, that is, receiver which receives the most IR, and the emitter whose beam contributes most to that receiver's sensor value. Note that this is not information that can be gathered from the look-up table. However, it can be established during the construction of the lookup table. Thus, associated with each entry in the lookup table was the index of the most significant emitter and receiver for that set of sensor values.

During simulation, when robot A, is updated with respect to robot B's IR emissions, and there is some possibility that a third robot C could cause an occlusion, the procedure adopted was as follows. The IR beam is treated as a line segment, travelling from the centre of B's emitter to the centre of A's receiver, where the emitter and receiver were those specified by the lookup table. The third robot, C, is deemed to cause an occlusion if the distance between the centre of C and the line segment is less than some distance, r (specified below), as illustrated in figure 13. Occlusion is an all-or-nothing event. If the beam is occluded then robot A's sensors receives no IR from robot B's emitter. Alternatively, if there is no occlusion, A's sensors will receive the sensor values specified in the lookup table. The value r specifies the minimum distance between the centre of robot C and the beam segment, necessary to cause an occlusion. Modelled accurately, the value of r is some function of C's orientation relative to the direction of beam segment. Its maximum value is the distance between the centre and a corner (11.84cm), its minimum value is half the length the robot's side (8.375cm). At the beginning of each trial the value of r was set at uniform random within that range. This was done for two reasons. First, it avoided the expense of calculating the value every time that a potential occlusion was checked. Second, it made the exact details of the occlusion model less easy for controllers to rely upon.

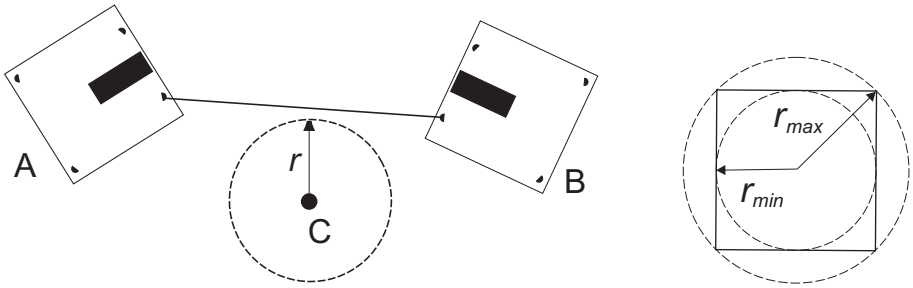


Figure 13: Checking occlusion by robot C of the beam from A's sensor to B's sensor: The beam is modelled as segment between the two sensors, occlusion occurs if the distance between the centre of C and the beam is smaller than r. In the case illustrated, there no occlusion is assumed. r is set randomly each trail in the range from r_{min}, the shortest distance between the centre of the robot and its sides, to r_{max}, the distance between the robot centre and a corner.

3.3 Combined IR: Final Sensor Values

At each update of the simulation, each robots' sensors' direct and reflected IR values are established in the manner set out in the sections above. A sensor's direct and reflected values are combined simply by summing them, and capping the result if it exceeded the maximum possible value the could be returned by an IR sensor. The last step in generating a sensor value is the addition of noise. Very little sensor noise is observed on the real robots; sensors return values in an integer range [0:249] and the majority of variation due to noise lies in the range ±2. We could (and possibly should) have characterised the range and distribution of the sensor noise more accurately. However, we simply modelled noise as a pseudo-gaussian offset for each sensor at each update. The noise offset, $f_{noise}(N)$, is an integer offset in the range $\pm N$ generated thus:

$$f_{noise}(N) = random(N) - random(N)$$

where $random(N)$ is function returning an integer drawn at uniform random from the range [0:N]. The value N is set anew, at the beginning of each trial, from a uniform random distribution over the range [0:8].

3.4 Movement

Measuring the robot's speed, we found between robot variation in the range 6cm/s to 6.6cm/s. There were also a small, but noticeable differences between the speeds of individual wheels on each robot, so that robots do not move exactly in a straight line. The deformable nature of their foam-rubber wheels introduces further variation when robots are turning, accelerating or decelerating. In addition, the trailing (unpowered) castor wheel is slow to become realigned after a turn; indeed the castor wheel typically does not become fully realigned after a

turn, meaning that a robot attempting to move in a straight line will tend to veer slightly in the direction of its most recent turn (this is in addition to the robots tendency not to go straight even when the castor wheel is aligned forwards). To avoid trying to accurately simulate the robot's movement we introduced three kinds of between-trial variation. At the beginning of each trial, the mean speed of each robot was set randomly in the range 5.985-6.615cm/s (i.e., a variation of ± 5% around the mean robot speed of 6.3cm/s). In addition, for each robot, some random difference between the maximum speed of the robot's two motor-driven wheels was introduced, resulting in some consistent directional deviation. The between-wheel speed difference lay in the range 0-0.63cm/s (i.e., up to 10% of mean robot speed), and was applied in such a way that the robot's mean wheel speed remained unchanged. These two forms of variation should ensure that evolved controllers are robust to variations, overall speed, between-wheel speed and directional biases due to the trailing castor wheel. The remaining aspects of movement variation on the real robots, due to the deformation of the foam rubber wheels, and interactions between the castor wheel and the motor-driven wheels, were addressed by simply adding noise to the simulated robots' motor output, with the magnitude of this noise being varied on a per-trial basis, as detailed in the following paragraph.

The movement of a robot with respect to its controller's motor output was modelled extremely simplistically: A robot's neural network controller produces left and right motor outputs from the set $\{-1, 0, 1\}$. Each motor output is scaled to provide motor velocity by multiplying by the maximum speed of its associated wheel. Noise was then applied independently to each motor velocity, by multiplying that velocity with a value chosen at uniform random from the range $[1 - m : 1 + m]$. For 25% of trials, $m = 0$ for all robots, otherwise, m was set a uniform random in the [0:0.1] independently for each robot[3]. After the application of noise, the resultant left and right motor velocities, v_l and v_r, were used to update the robot's orientation, θ, and the position of the centre of the robot's wheel axis, the polar vector $\vec{P}$, as follows:

$$\theta^{t+1} = \theta^t + \frac{v_r^t - v_l^t}{wu} \qquad \vec{P}^{t+1} = \vec{P}^t + \left(\frac{v_l^t + v_r^t}{2u}, \theta^{t+1} \right)$$

Here u and w are constants, respectively, the number of simulation updates per second (5) and the distance separating a robot's two wheels (13.2cm).

[3]We were concerned that the controllers might come to rely or exploit the existence or structure of the motor noise. We have employed it to avoid making any attempt at modelling various interactions between the robot's motor-driven wheels, the castor wheel and the floor, but the effects of such interactions are likely to bear no resemblance to uniform noise. By imposing a significant number of trials with no motor noise, we ensure that high-scoring controllers are cannot rely on the existence of motor noise, and hence cannot rely on the structure of motor noise in the simulation. They should therefore be robust to the existence of a variety of deviations from our simple model of robot movement, but unable to rely on how these deviations are manifested on the real robots.

3.5 Collision Handling

Accurate modelling of collisions between robots would be both complex and computationally expensive; fortunately however, this is unnecessary. Collisions are penalised during the evolutionary process, so robots are expected to evolve to avoid collisions, and successful strategies are not expected to attempt to exploit the effects of collision. For this reason collisions were only very minimally modelled: If, at any given time-step, the displacement component of a robot's movement was such that the robot would come into contact with another, then neither robots' position and orientation were updated for that time-step.

4 The Evolution of Controllers

4.1 Neural Network

The robots were controlled by artificial neural networks. Since it was unclear how the task would be solved, we could estimate little about the type of network architecture that would be needed to support the required behaviour. Thus we attempted to place as much of the design of the networks as possible under evolutionary control, namely the thresholds, weights and decay parameters, and the size and connectivity of the networks. Each neural network comprised 4 sensor input nodes, 4 motor output nodes, and some number of artificial neurons. These were connected together by some number of directional, excitatory and inhibitory weighted links. The network has no explicit layers, any neuron may connect to any others, including itself; and may also connect to any of the sensory or motor nodes. No upper limit is placed on the number of neurons in the network, although the number of connections is limited by the number of neurons. Each neuron can connect to all, some, or none of the network's neurons and nodes. However, there can be duplication of connections; no two connections may have the same origin and destination. Thus, each neuron can have no more than one recurrent connection; there can be no more than two connections between any two neurons (one in each direction), no more than one connection to any neuron from the same sensor node (although a neuron may take input from more than one sensor), and no more than one connection from any neuron to the same motor node (although a neuron may output to more than one motor)[4].

The neurons we use are loosely based on model spiking neurons (see Gerstner and Kistler, 2002, for a comprehensive review of such models). At any time-step, the output, O_t, a neuron is given by:

$$O_t = \begin{cases} 1 & \text{if } m_t \geq T \\ 0 & \text{if } m_t < T \end{cases}$$

[4]For a network comprising N nodes, there can be up to $N^2 - N$ connections between neurons, N recurrent connections, $4N$ sensor to neuron connections and $4N$ neuron to motor connections, giving an upper limit of $N^2 + 8N$ connections.

Here T is the neuron's threshold. m_t is a function of a neuron's weighted, summed input (s), and the value of m_{t-1} scaled by a temporal decay constant, such that:

$$m_t = \begin{cases} (\gamma_A)m_{t-1} + \sum_{n=0}^{N} w_n i_n & \text{if} \quad O_{t-1} = 0 \\ (\gamma_B)m_{t-1} + \sum_{n=0}^{N} w_n i_n & \text{if} \quad O_{t-1} = 1 \end{cases}$$

where γ_A and γ_B are decay constants, and w_n designates the weight of the connection from the n^{th} input (i_n) that scales that input. γ_A and γ_B are constrained to the range [0:1], the values of weights and thresholds are unconstrained. For certain parameter settings this neuron will behave like a simple spiking neuron, accumulating membrane potential, firing and then discharging (i.e., with $\gamma_A > \gamma_B$ and $T > 0$). However, the neurons also exhibit a range of other interesting temporal dynamics for different parameter ranges.

Each sensor node outputs a real value, in the range [0.0:1.0], which is simple linear scaling of the reading taken from its associated sensor. Motor outputs consist of a 'forward' and a 'reverse' node for each motor. The output, M_{out}, of each motor nodes is a simple threshold function of its summed weighted inputs:

$$M_{out} = \begin{cases} 1 & \text{if} \sum_{n=0}^{N} w_n i_n > 0 \\ 0 & \text{if} \sum_{n=0}^{N} w_n i_n \leq 0 \end{cases}$$

The final output of the each of the two motors is attained by subtracting its reverse node output from its forward node output. This gives three possible values for each motor output: {-1,0,1}.

4.2 Encoding Scheme

The network is encoded by a topological encoding scheme, designed to enable the size and connectivity of the network to be placed under evolutionary control. A main concern in the design of this encoding scheme was that topology was not determined by the position at which individual neurons and connections were encoded on the genotype, as it is in standard fixed-sized encoding schemes. Such position-dependent encoding schemes are particularly vulnerable to disruption by the insertion and deletion of genetic material, for the obvious reason that insertions and deletions will alter the positions of the encoded variables (Jakobi and Quinn, 1998). The encoding scheme used in this paper employs a position-independent encoding, whereby each encoded neuron is associated with an identity tags, and these tags are used to specify connections between neurons, in a manner detailed in the following paragraph.

The procedure for encoding and decoding networks is as follows: Each neural network controller is encoded in a 'genotype', this consists of a list of 'genes'. Each gene encodes the parameters for an individual neuron and its associated connections. The form of each gene was { X, T, γ_A, γ_B, L_{in}, L_{out} }, where T, γ_A and γ_B are the (real-valued) threshold and decay parameters, and L_{in} and L_{out} are lists of input and of output connections respectively. X is an integer which serves as an 'identity tag'. This is assigned to a gene upon its creation

(genes are created either in the first GA generation when a population is created, or when a new gene is added to a genotype as a result of a macro-mutation). The identity tag (i.d.) assigned to a new gene is initially unique – no other gene will have the same i.d. at the time it is first assigned. The i.d. is not subject to mutation, but it *is* inherited. Hence if two genes (on two different genotypes) have the same i.d., it will be because both genes have a common ancestor. Sensor and motor nodes also have unique identity tags. The identity tags of the neurons, and the sensor and motor nodes, are used to determine network connectivity; the encoded connections specify a source or destination by reference to these tags. Each element in the connection lists, L_{in} and L_{out}, takes the form, $\{Z, w\}$, where w is the connection weight, and Z specifies an identity tag of the node (sensor, motor, or neuron) to which it should connect. In decoding the network, the source (or destination) of each connection in the list is determined by matching the tag it specifies with the i.d. of a neuron in the network, or that of a sensor or motor node. A connection may specify the tag of the neuron from which it originates, resulting in a recurrent connection being made. Note that Z is inherited and is subject to mutation (see below).

4.2.1 Initialisation

At the beginning of an evolutionary run, genotypes are randomly initialised. This proceeds as follows: Each genotype was assigned 3 genes and each gene was assigned a unique identity tag. Next, each gene was given between 0 and 7 input connections, and between 1 and 7 output connections. The number of connections was chosen at uniform random for each of the connection lists belonging to each gene. (Output lists were non-empty to ensure that each neuron was at least potentially functional.) Each connection element in each gene was then randomly assigned a tag, such that it was equally likely to connect to any neuron specified by the genotype or to any sensor node, or any motor node. Finally the real-valued parameters, i.e., all thresholds, decay parameters and connection weights were initialised to random values: weights and thresholds in the range [-5:5] and decay parameters in the range [0:1].

4.3 Evaluation

A single genotype was used to generate a team by 'cloning', that is, decoding the genotype and then making one copy of the resulting controller for each robot. Each clonally generated team was evaluated over a number of trials from different starting positions. Each trial was scored according to the task evaluation function, set out below, and the team's fitness was then its the average score over this set of trials. Insofar as was possible, we tried to ensure that different controllers were evaluated under the same set of environmental conditions, so that differences between teams' scores maximally reflected differences in their ability. Given that different starting positions will present different challenges, it was important each team was evaluated over the same set of initial positions. To this end, at each generation of the evolutionary algorithm, a set of starting

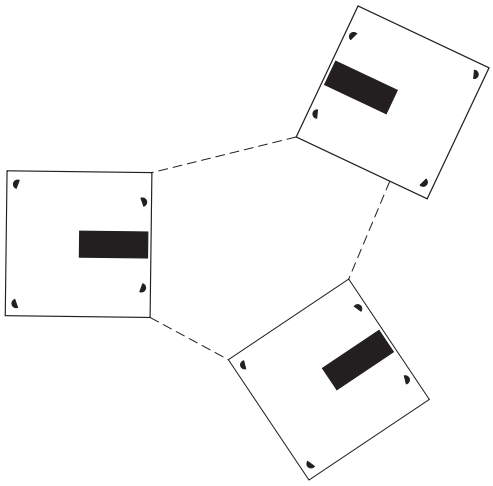


Figure 14: **Generating a random starting position:** Each robot's orientation is set randomly in the range [0:2π], and the minimum distance between the edges of each robot and its nearest neighbour is set randomly in the range [10cm:22cm].

positions was randomly generated, as detailed figure 4.3, to be used for the evaluation all the teams. Each member of the population was evaluated for one trial from each the different starting positions. An additional source of potential variation between teams was due to the variation in simulation parameters that we had introduced as part of the minimal simulation methodology (e.g. wheel speeds, sensor positions etc.). To counteract this, we associated a full set of simulation parameters, randomly generated within the appropriate ranges, with each of the different starting positions. As with the starting positions, these simulation parameter sets were also generated anew at the beginning of each generation of the evolutionary algorithm. At the beginning of an evolutionary run we used a set of 60 starting positions; there were thus 60 evaluations per team. However, once the population had begun find reasonable strategies (when scores began to exceed 70% of the maximum possible), we increased the number of starting positions, and hence the number of evaluations, to 100.

4.3.1 Task Evaluation Function

Reflecting the task description, the evaluation function seeks to assess the ability of the team to increase its distance from its starting position, whilst avoiding collisions and staying within sensor range. It therefore consists of three main components. First, at each time-step of the trial, the group is rewarded for any gains in distance. Second, this reward is multiplied by a dispersal scalar, reducing the fitness increment when one or more robots are outside of IR sensor range. Third, at the end of a trial, the group's accumulated score is reduced in proportional to the number of collisions which have occurred during that trial. (The maximum number of allowed collisions per trial was 20, if this number was

reached, the trial was ended). More specifically, a team's trial score, is:

$$P.\Big(\sum_{t=1}^{T}\Big[f\big(d_t,\, D_{t-1}\big).\big(1\,+\,\tanh\left(s_t/20.0\right)\big)\Big]\Big)$$

Here P is a collision-penalty scalar in the range $[0.5 : 1]$, such that, if c is the number of collisions between robots, and c_{max} is the maximum number of collisions allowed, then $P = 1 - c/2c_{max}$. The distance gain component is given by the function f. This measures any gain that the team have made on their previous best distance from their initial location. Here a team's location is taken to be the centre-point (or centroid) of the group. If d_t is the Euclidean distance between the group's location at time t and its location at time $t = 0$, D_{t-1} is the largest value that d_t has attained prior to time t, and D_{max} is the required distance (i.e. 100 c.m.), then the function f is defined as:

$$f\left(d_t, D_{t-1}\right) = \begin{cases} d_t - D_{t-1} & \text{if } D_{t-1} < d_t < D_{max} \\ 0 & \text{otherwise} \end{cases}$$

The final component of a team's trial score, is the scalar s_t, is a measure of the team's dispersal beyond sensor range at time t. If each robot is within sensor range of at least one other, then $s_t = 0$. Otherwise, the two shortest lines that can connect all three robots are found, and s_t is the distance by which the longest of these exceeds sensor range. In other words, the team is penalised for its most wayward member. Note that s_t used in combination with a **tanh** function, this ensures that as the robots begin to disperse, the team's score increment falls away sharply, the gradient of the tanh curve falls off as the distance between the robots increases, ensuring that increases in distance will still receive some minimal reward, even when the robots are far apart.

4.4 The Evolutionary Machinery

A simple, generational, evolutionary algorithm (EA) was employed for these experiments. An evolutionary population contained 50 genotypes, initially created at random (see section 4.2.1). At the end of each generation (i.e. after all individuals had been evaluated), genotypes were ranked by score, the 10 lowest scoring individuals were discarded and the remainder used to generate a new population. The two highest scoring individuals ('the elite') were copied unchanged in the new population, thereafter, genotypes were selected randomly with a probability inversely proportional to their rank. 60% of new genotypes were produced by recombination, and mutation operators were applied to all genotypes except the elite.

4.4.1 Mutation Operators

Genotypes were subject to both macro-mutation (i.e. structural changes) and micro-mutation (i.e. perturbation of real-valued parameters). Micro-mutation

entailed that a random Gaussian offset was applied to each real-valued parameter encoded in the genotype with a small probability, such that the expected number of micro-mutations per genotype was 2.0. The mean of the Gaussian was 0, and its s.d is 0.33 of that parameter's initialisation range. The threshold and weight parameters were unbounded, but the decay constants were restricted to their initialisation range of [0:1]. In cases where the mutated value of a bounded parameter fell outside a bound, its value was set at uniform random to a value between the bound and its pre-mutated value.

Three types of macro-mutation were employed. The first two involve the addition and deletion of genetic material. Ideally we would like to balance the rate at which new genetic material is added to and removed from the population, facilitating steady growth as the added material becomes a functional part of the genotype. We have found that, on average, addition is less disruptive that deletion, so in an attempt to maintain a balance, we have set addition rates lower than deletion rates; individuals subject to addition mutations are more likely to remain in the population than those subject to deletion mutations. The first type of macro-mutation involved the addition or deletion of genes. An addition mutation occurred with a probability of 0.004 per genotype, with the new gene being created and added to the genotype by the same procedure described in section 4.2.1 above, except that the maximum number of connections per connection list was limited to two (in order to minimise disruption). Deletion occurred with probability 0.01, and was applied in one of two ways. With a probability of 0.5, one gene was selected at uniform random and removed from the genotype, otherwise a gene was chosen for removal at biased random, using roulette-wheel selection, with a probability inversely proportional to its age (i.e. the number of generations that it had been in the population)[5]. The second type of macro-mutation involves the addition and deletion of connections. With probability 0.02, a new connection was created (following the procedure described in section 4.2.1); the connection had an equal probability of being an input or an output, and was added to a randomly chosen gene. With a probability of 0.04, a gene was selected at uniform random, a connection list chosen (with equal probability) and, unless that list was empty, a randomly chosen connection was deleted. The final type of macro-mutation was reconnection. With a probability of 0.04, the genotype was subject to a reconnection mutation: one gene was selected at uniform random, one of its connection list was then randomly selected and a connection element randomly chosen. This connection was then reconnected, by randomly assigning it a tag such that it was equally likely to connect to any neuron or sensor node (in the case of input connections) or to any neuron or motor node (in the case of output connections).

[5]Biasing the deletion operator towards the removal of newer genes over older ones will tend to bias it toward the deletion of non-functional genes over functional ones, hence making it less disruptive. As discussed elsewhere, the older a gene is, the more likely that it is being maintained by selection, and hence functional rather than neutral (Jakobi and Quinn, 1998).

4.4.2 Recombination

The recombination (or 'cross-over') operator creates two new 'offspring' individuals by combining encoded variables from two 'parent' genotypes. Since this is variable-size encoding, the addition and deletion of genetic material means that two genes at the same position on their respective genotypes may not be correlated at the phenotypic level. If crossover were simply based on genotype position, as it is with standard fixed-length encoding schemes, it would often be highly disruptive, with unrelated variables being crossed-over (Jakobi and Quinn, 1998). To avoid this problem, the recombination operator utilises gene's i.d. tags, rather than their position on the genotype, in order to maintain the structural integrity of the resulting genotypes. The i.d. tag is used by the recombination operator to pair genes with a common ancestor, and thereby helps to ensure that genes with similar phenotypic functionality crossed. Crossover takes two parent genotypes, P_1 and P_2, and generates two offspring, O_1 and O_2 in the following manner: Each of offspring initialised as a copy of a different parents (e.g., $O_1 = P_1$ and $O_2 = P_2$). The offspring genes are then paired by identity number; any gene that O1 has in common with O2 is crossed (i.e. swapped) with a probability of 0.5. Any remaining un-paired genes are not crossed.

5 Evolved Behaviour

To date, we have undertaken a total of ten evolutionary runs. Four of these were terminated at early stage because they seemed unpromising. The remaining six runs produced teams capable of a consistently high standard of success after being left to evolve for between two and five thousand generations. There were significant behavioural differences between the successful teams, and we have chosen to focus on a single team rather than attempt to summarise them all. In describing the behaviour of the team, we wish primarily to achieve two objectives. The first is to demonstrate that the robots' behaviour is indeed that of a team, in the sense in which the term was introduced at the beginning of this paper. The second is to illustrate the process by which these roles become allocated in the absence of any intrinsic differences between the robots.

It is perhaps a testament to Jakobi's minimal simulation methodology that the controllers we evolved in simulation transferred to the real robots at the first attempt. However, it is also likely that there was an element of luck involved—we had expected to have to fine-tune the simulation. Paper really is too static a format to demonstrate how well the team transferred from simulation to reality, a problem which is lamented in more detail elsewhere (Jakobi, 1998b). We can only report that the behaviour observed in simulation was qualitatively reproduced in reality. In simulation, averaged over 5000 trials, this team achieve a mean score 99.7 (out of a possible 100). We have not completed nearly so many trials with real robots. To date, we have conducted a total of 100 trials, from random starting positions, with the real robots. This included testing all

combinations of three robots from the four we own. In each case, the robots successfully completed the task, first organising themselves into formation, and then maintaining that formation as they travelled away from their initial positions.

5.1 Formation Movement

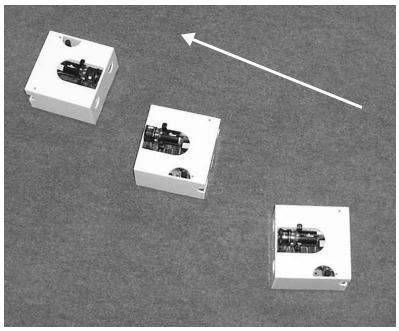


Figure 15: Video still of the robots moving in formation. The arrow indicates the direction of overall movement. The front-most robot travels in reverse, the two remaining robots move forwards.

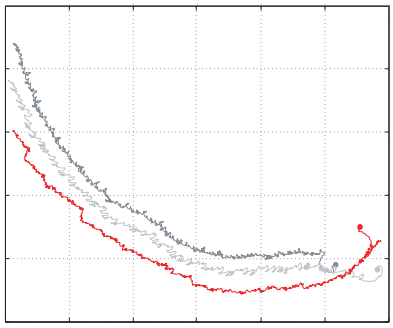


Figure 16: An example trajectory, tracing the position of each robot over a 5 minute period. Grid divisions are at 50cm intervals. Robots start bottom right (indicated by dots). Grid divisions are $50cm^2$. Data generated in simulation.

The team travel in a line formation, as can be seen from the video still in figure 15. The lead robot travels in reverse, whilst the middle and rear robot travel forwards. When travelling in formation, the team move at just over 1 cm/s, a relatively slow speed compared the 6 cm/s maximum speed that an individual robot is capable. The video still of the robots in formation fails to catch the dynamics of the team's movement; this entails each robot swinging clockwise and counterclockwise whilst maintaining its position—watching the video footage sped up, team locomotion appears almost snakelike. The sequence of diagrams in figure 17 is an attempt to illustrate this aspect of the team's locomotion. Note from these diagrams, that the robots rely almost entirely on the direct perception of each other's IR beams (i.e. sensory interference) in order to coordinate their movement. One illuminating way of illustrating relational movement patterns is by plotting changes in an individual's orientation relative to the position of the individual with which it is interacting (see e.g. Moran et al., 1981). Relative orientation is an egocentric measure; the orientation of A relative to the position B is the angle between A's orientation and the line AB. Figure 18 shows the orientation of each robot relative to its neighbours during a period of formation movement. It illustrates the high degree of coordination between the front and middle robot, each responding closely to the other's

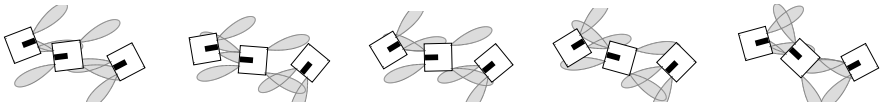


Figure 17: Time sequence illustrating relative positions during formation movement over a short (4 second) period. Robots maintain contact through direct sensing of each other's IR beams.

movements. It also illustrates the much lower degree of coordination between the middle and rear robots, and the difference, with respect to the frequency of angular oscillation, between the movement of the rear robot and the leading pair. Despite the oscillating angular displacement of the robots, their formation is extremely robust. The formation is maintained indefinitely[6], despite robots only having been evolved for their ability to move the group centroid one metre.

5.2 Roles

It should be clear from the above that robots perform the task we have set them. But are the robots actually operating as a team? In what follows we briefly show that each robot makes some necessary contribution to overall success and that these contributions are different and persist over time. To this end, we are interested in what each individual contributes to the maintenance of the formation and its continued movement. Perhaps the simplest way to assess individual contributions is simply by considering the effects of the removal of individual robot from the formation. To this end, we consider the effects of the removal of either the front or the rear robot (removal of middle robot is unilluminating, merely leaving the remaining two robots out of sensor range).

If the rear robot is removed from the formation, the locomotion of the remaining pair ceases, there is no further significant displacement of their position. However, this is the only significant effect. The pair maintain the same configuration as when in full formation. Their cycle of angular oscillation relative to one another remains in anti-phase, although the pattern becomes more regular, as illustrated in figure 19. This is a dynamically stable configuration, tightly constrained by sensory feedback, which will persist indefinitely. If the rear robot is replaced, the group will move away once more. Now we consider the front robot. If this is removed from the full formation, the middle robot swings round toward the rear robot, and—after some interaction—the two robots form an opposed pair which maintain the same dynamically stable configuration as was just described.

[6]Strictly speaking, we should state that the robots maintain formation indefinitely in simulation. This also appears to be the case with the real robots, although we have yet to test this in a sufficiently large space. In our current testing area, robots move in formation until they encounter a wall, however we have observed them moving together over distances of up to 10 metres, and have yet to see the formation break down in the absence of an obstacle.

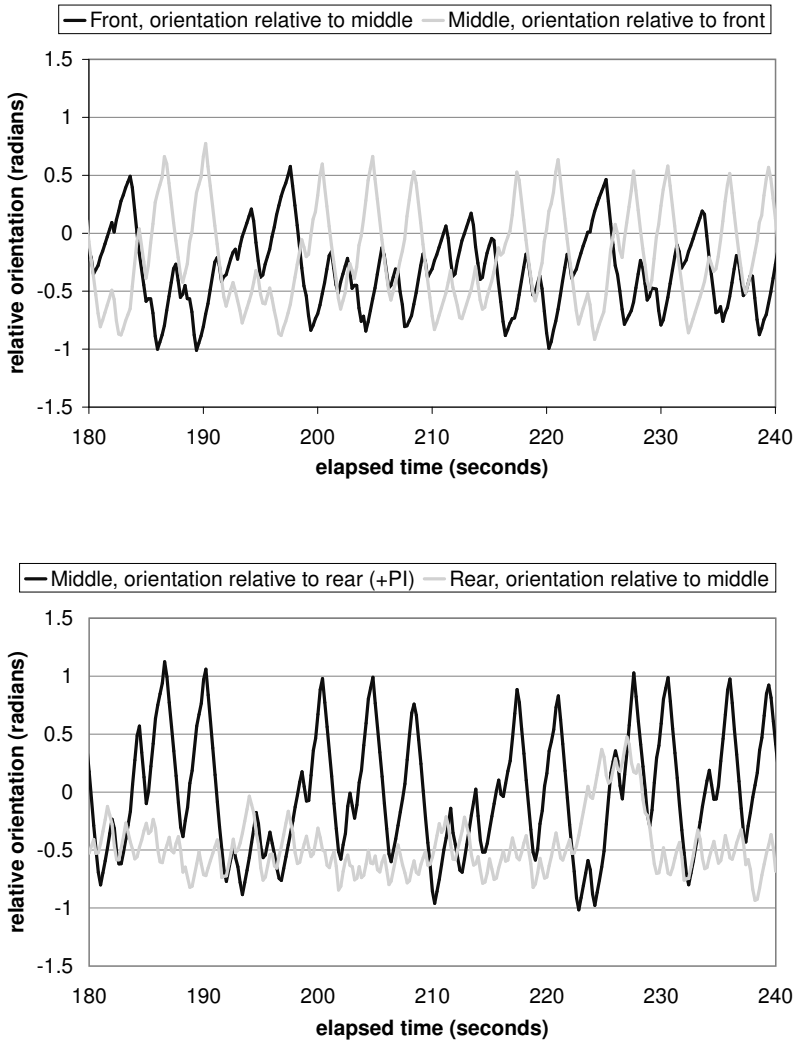


Figure 18: Relative orientations of robots in formation over a 60 second period. (Data taken from simulation.) *Top:* The angular movements of the front and middle robot are closely coordinated, with relative orientations predominantly in anti-phase. *Bottom:* The coordination of the middle and rear robot is much looser (Note: To facilitate comparision, the relative orientation of the middle robot has been offset by π in the bottom graph.)

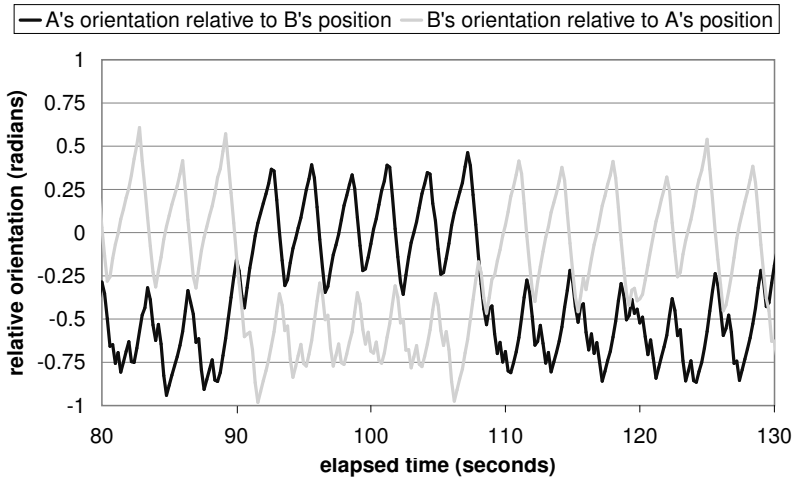


Figure 19: Relative orientations of two robots, A and B, operating in the absence of a third robot: Similarly to the front pair in a full formation, orientations are in anti-phase, although here the pattern is more regular. The configuration (and the pattern) is asymmetric, and the pattern is maintained although robots periodically swap roles within the configuration (seen at 90 and 110 seconds in the figure). Note that there is no significant displacement of the pair's position.

From the above, we can say the following: Firstly, the rear robot has no significant effect on the other two robots' ability to maintain formation, but it is crucial to sustaining locomotion. Secondly, it is clear that the middle robot responds to the presence of the rear robot by moving forwards, since in the absence of the rear robot, the remaining pair cease to travel. For locomotion to continue, the configuration of the rear and middle robot must persist. That is, the middle robot must continue to sense the rear robot with its back sensors. Finally, in the absence of the front robot, the configuration adopted by the middle and rear robot in the formation is unstable.

This analysis is sufficient to show that these robots are working as a team, concurrently performing separate but complementary roles which, in combination, result in coordinated formation movement. A more precise characterisation of each robot's contribution is difficult without presenting detailed analysis of the close sensorimotor coupling between the opposed front pair, and how this coupling is perturbed, but not completely disrupted, by the presence of the rear robot. Nevertheless, it is possible to say something further about the team's organisation through investigating the effects of reorganising its formation. Firstly, when the middle robot is quickly picked up and rotated by 180 degrees, the formation is maintained and the team start to move in the opposite direction, with the robots which were previously front and rear adopting the roles appropriate to their new positions in the formation. Secondly, if the rear robot is removed from the formation and appropriately placed behind the

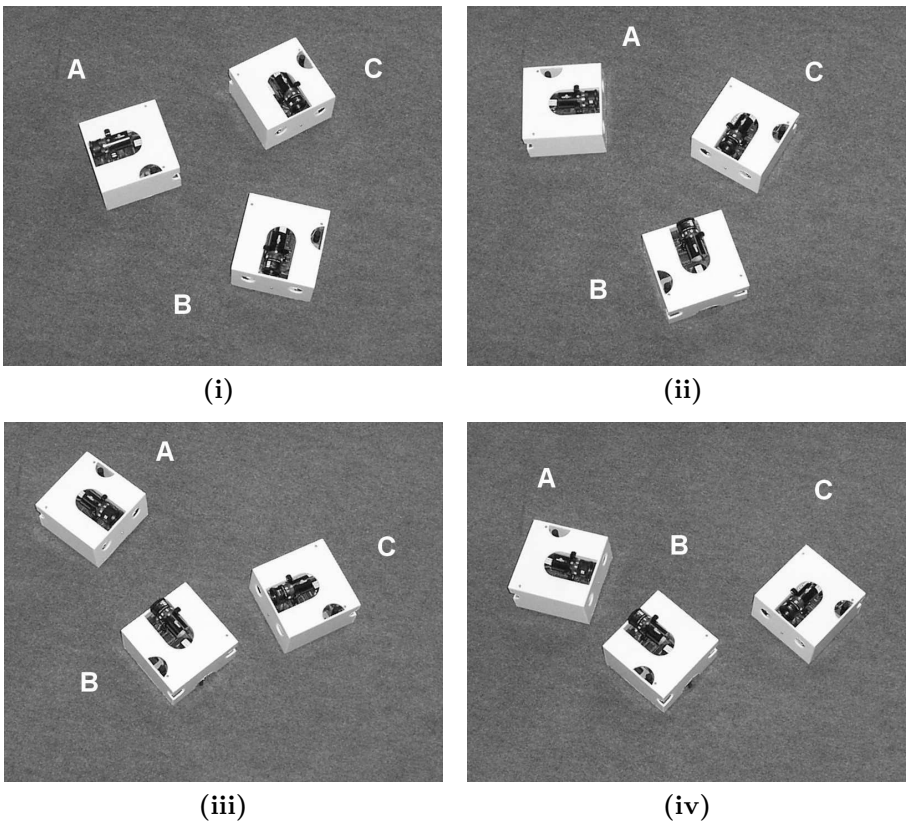


Figure 20: **An example of the team moving into the formation positions.** (i) The robot's initial positions. Initially, C is attracted B's rear sensors, causing B to turn tightly, A circles away, clockwise (ii) B and C begin to form a pair as A circles round towards them (iii) A disrupts the pair formation of B and C, subsequently pairing with B. (iv) C becomes attracted to B's rear sensors and begins to move into position. Shortly after this, the team achieve their final formation.

front robot, formation again move off in the opposite direction, with each robot performing the role appropriate to its position. Thus, the fact that each robot remains in the same role within the formation is solely by virtue of the spatial organisation of the formation, rather than any long-term differences in internal state[7].

5.3 Role Allocation

How are the roles initially allocated within the team? This is essentially to ask how the robots achieve their formation positions from random initial positions, since as has already been noted, that the maintenance of individual roles is a function of the spatial organisation of the team formation. Any discussion of the initial interactions of the robots will be difficult without at least some information about how the robots responds to sensory input, so we will start by giving a very simplified explanation. In the absence of any sensory input, the robots move in a small clockwise forwards circle (the motor output is a cyclic pattern of left motor forward for 3 time-steps, followed by one times-step of right motor forward). A robot is generally 'attracted' to any source of front sensor input. It will rotate anticlockwise in response to any front left input and clockwise in response to front right input. Activation of either (or both) of the rear sensors in the absence of significant front sensor input causes the robot to turn more tightly in a clockwise direction (i.e. the fourth step of the basic motor pattern is removed). This is an incomplete description, but should be sufficient for the purposes of our explanation.

From its initial position, a robot will begin to circle clockwise until its senses another robot. Recall that a robot can sense both IR reflected off the body of another robot and the IR beam of another robot, the latter being perceptible from twice the distance than the former. For this reason a robot will typically first encounter either the front or rear IR beams of another robot (direct IR), or one of its side panels (reflected IR). A robot 'attracted' to the side of another robot will simply be ignored as it cannot be sensed. A robot attracted to the rear IR beams of another robot will in turn activate that robot's rear sensors, causing it to turn sideways on. If however a robot becomes attracted to the front IR beams of another, it will in turn activate the front sensors of that robot as it approaches, both robots will turn to face each other—mutually attracted. The remaining robot will subsequently become attracted to rear sensors of one of the pair, bringing the formation into completion. Prior to the arrival of the third robot, the facing pair maintain the dynamic, stable configuration which was described in the previous section (illustrated in figure 19). In the present context, this serves as 'holding pattern', in which the pair await arrival of the remaining team member before they begin to move off.

The process of achieving formation is not always quite as simple as the above description might imply. The pairing process may have to be resolved between three robots (as for example, in panels ii and iii of figure 20) where one robot may disrupts the pair-forming of the other two. However, the explanation given above should be sufficient to inform the reader of the basic dynamics of the process of team formation. This is a process which can be seen as a one of progressive differentiation. The robots are initially undifferentiated with respect to their potential roles. The opposed pairing of two robots partially

[7]This is not to say that the robots' behaviour is reactive. We know from analysis (not presented here) that the evolved networks rely heavily on temporal dynamics, such as short-term transient states. However they do not rely on internal state to maintain their roles.

differentiates the team. The excluded robot's role is now determined—it will become the rear robot in the formation. Further differentiation occurs when the unpaired robot approaches the back sensors of one of the waiting pair, thereby determining the final two roles.

6 Conclusion

The structured cooperation required for the performance of a team task presents interesting problems for a distributed control system. This is particularly true when individuals are homogeneous, and constrained to only make use of limited local information. We have suggested that artificial evolution is a useful tool for automating the design of such systems, and presented an example of an evolved homogeneous multi-robot team. We have shown that the evolved system is capable of organising itself into formation, adopting functionally distinct roles, and maintaining this organisation over time.

It is worth noting the novelty of this work within the field of evolutionary robotics. To date, this research field has focussed almost exclusively on single robot systems (a notable exception, using real robots, being Nolfi and Floreano's (1998) work with co-evolved predator-prey pairs). Insofar as we are aware, the work reported in this paper represents the first published example of cooperative and coordinated behaviour for a real multi-robot system designed by artificial evolution. By virtue of involving multiple robots, it is also one of the few examples of evolutionary robots research in which controllers engage with a non-static environment (see Nolfi and Floreano, 1998; Jakobi, 1998a; Smith, 1998, for exceptions).

Finally, we suggest that such a system would be extremely difficult to design by hand, given the sensory constraints and the close coupling of the individual robots. Of course, this not an easy claim to prove. However, we will conclude with a quote from someone with a great deal of experience in hand-designing multi-robot systems. Discussing the need for more complex sensors in the design of a following, behaviour Matarić comments: "If using only IRs, the agents cannot distinguish between other agents heading toward and away from them, and thus are unable to select whom to follow" (Matarić, 1995).

Acknowledgements

We would like to thank Adrian Thompson for investing time and effort in robot modifications. Thanks also to Nick Jakobi and Kyran Dale for useful discussion. This work was funded by the B.N.S.C. Space Foresight project IMAR.

References

Anderson, C. and Franks, N. (2001). Teams in animal societies. *Behavioural Ecology*, 12(5):534–540.

Balch, T. and Arkin, R. (1998). Behavior-based formation control for multiagent robot teams. *IEEE Transactions on Robotics and Automation*, 14(6).

Balch, T. and Parker, L. (2000). Guest editorial. *Autonomous Robotics*, 8(3):207–208. Special Issue on Heterogeneous Multi-Robot Systems.

Brooks, R. (1992). Artificial life and real robots. In *Proceedings of the First Conference on Artificial Life*. MIT Press/Bradford Books.

Camazine, S., Denouberg, J.-l., Franks, N., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2001). *Self-Organization in Biological Systems*. Princeton University Press.

Chaimowicz, L., Sugar, T., Kumar, V., and Campos, M. (2001). An architecture for tightly coupled multi-robot cooperation. In *Proc. IEEE Intl. Conf. Robotics and Automation*.

Detrain, C. and Pasteels, J. (1992). Caste polyethism and collective defense in the ant *pheidole pallidula*. *Behavioural Ecology and Sociobiology*, 29:405–412.

Gerkey, B. and Matarić, M. (2001). Principled communication for dynamic multi-robot task allocation. In Rus, D. and Singh, S., editors, *Experimental Robotics VII, LNCS 271*, pages 253–362. Springer-Verlag.

Gerstner, W. and Kistler, W. (2002). *Spiking Neuron Models*. Cambridge University Press.

Hobbs, J., Husbands, P., and Harvey, I. (1996). Achieving improved mission robustness. In *Proc. 4^{th} E.S.A. Workshop on Advanced Space Technologies for Robot Applicatations*.

Husbands, P. and Harvey, I. (1992). Evolution versus design: Controlling autonomous robots. In *Proc. 3^{rd} Intl. Conf. on Artificial Intelligence, Simulation and Planning*. MIT Press, London.

Husbands, P. and Meyer, J.-A., editors (1998). *Evolutionary Robotics: Proceedings of the First European Workshop, EvoRobot98*. Springer.

Jakobi, N. (1994). Evolving sensorimotor control architectures in simulation for a real robot. Master's thesis, School of Cognitive and Computing Sciences, University of Sussex.

Jakobi, N. (1997). Half-baked, ad-hoc and noisy: Minimal simulation in evolutionary robotics. In Husbands, P. and Harvey, I., editors, *Fourth European Conference on Artificial Life*. MIT Press/Bradford Books.

Jakobi, N. (1998a). Evolving motion-tracking behaviour for a panning camera head. In *Proc. 5^{th} Intl Conf. Simulation of Adaptive Behaviour*. MIT Press.

Jakobi, N. (1998b). *Minimal Simulations for Evolutionary Robotics*. PhD thesis, University of Sussex.

Jakobi, N. and Quinn, M. (1998). Some problems (and a few solutions) for open-ended evolutionary robots. In Husbands and Meyer (1998).

Matarić, M. (1995). Designing and understanding adaptive group behaviour. *Adaptive Behaviour*, 4(1).

Matarić, M. and Cliff, D. (1996). Challenges in evolving controllers for physical robots. *Robots and Autonomous Systems*, 19(1):67–83.

Matarić, M. and Sukhatme, S. (2001). Task-allocation and coordination of muliple robots for planetary exploration. In *Proc. 10^{th} Intl Conf. Advanced Robotics*.

Meyer, J.-A., Husbands, P., and Harvey, I. (1998). Evolutionary robotics: A survey of applications and problems. In Husbands and Meyer (1998).

Moran, G., Fentress, J., and Golani, I. (1981). A description of relational patterns of movement during 'ritualized fighting' in wolves. *Animal Behaviour*, 29:1146–1165.

Nolfi, S. and Floreano, D. (1998). Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? *Artificial Life*, 4(4):311–335.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence and Technology of Self-Organizing Machines*. MIT Press.

Quinn, M. (2001a). A comparison of approaches to the evolution of homogeneous multi-robot teams. In *Proc. 2001 Congr. on Evolutionary Computation*.

Quinn, M. (2001b). Evolving communication without dedicated communication channels. In *Proc. 6^{th} European Conf. Artificial Life*, Prague, Czech Republic.

Smith, T. (1998). Blurred vision: Simulation-reality transfer of a visually guided robot. In Husbands and Meyer (1998).

Stone, P. and Veloso, M. (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence Journal*, 100(2).

Thompson, A. (1996). An evolved circuit, intrinsic in silicon, entwined with physics. In Higuchi, T., editor, *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96)*. Springer-Verlag.

Thompson, A. and Layzell, P. (1999). Analysis of unconvential evolved electronics. *Communications of the Association of Computing Machinery*, 42(4).

Ward, C., Gobot, F., and Kendal, G. (2001). Evolving collective behaviour in an artificial world. *Artificial Life*, 7(2):191–210.