Abstract

This project tackles issues of scalability and reality transfer in evolutionary robotics through the *hard* task of scoring goals in *robot football*.

Evolutionary Robotics raison d'etre is to allow behavioural complexity beyond that imposed by the limitations of design, and as such scalability of techniques to more complex tasks is a central issue. The use of vision addresses sensory complexity and a novel technique of object level representation, using *virtual sensors* to provide such a description of the world, is implemented and seen to work. The use of *task decomposition* addresses task complexity, allowing division of a complete task into simpler sub-behaviours, making a potentially intractable problem tractable to solution by the evolutionary algorithm. Results show that designer misconceptions can reduce the efficacy of the latter approach.

Good controllers were consistently evolved in simulation using genetic programming of logic-level controllers, and for the most part successfully transferred to reality. A controller that could score goals in a real-world environment using visual location of ball and goal was evolved. The complexity of this task, within evolutionary robotics work, goes some way to validating the scalability of the approaches.

Acknowledgements

Thanks to my supervisor, John Hallam, for helpful advice and discussion, and always being willing to proof read, write a Perl script, etc. Thanks to Rens Kortman for introducing the Khepera and Andrew Eremin for discussions on evolutionary robotics. Thanks to all those who made the terminal room a better a place to be.

Finally, I thank EPSRC who provided financial support during my year of study, without which this project would not have been possible, via studentship ref: 9740124X.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There is a growing realisation within the field of intelligent robotics that behavioural complexity is approaching a limit imposed by difficulty of design. Evolutionary robotics, the application of evolutionary algorithms to robot synthesis, has emerged as an attempt to breach this impasse. Time constraints typically require evolution in simulation. Such an approach introduces the problem of successfully transferring the evolved controller from the virtual to the real world. Such reality transfer is a central issue in evolutionary robotics, without attempts at which, work remains unvalidated.

The aims of this project were to work at the current limit of behavioural complexity in evolutionary robotics, developing techniques that would allow progression beyond this. This was done through application to a complex task — robot football. Crossing the reality gap under pinned all aspects of design, and a data-intensive look-up table approach was taken: response of modelled aspects in the real world was sampled and recalled in simulation to recreate a virtual reality.

The work is characterised by its use of Genetic Programming, task decomposition, and object-level *virtual sensor* approach to vision. The common theme of these techniques is their scalability in principle. Scalability is identified as a key issue in evolutionary robotics, being essential if it is to realise its potential as a means of extending the complexity of robot behaviour past the 'design limit': At present the, still nascent, field is struggling to progress from the domain of 'toy problems'. The intended scalability of task decomposition and *virtual sensors* comes from the designer's insertion of domain knowledge to guide evolution and prune the search space — they are engineering/evolutionary hybrid approaches.

Genetic Programming and the style of controllers it encourages have been largely overlooked, and in some cases dismissed, by the evolutionary robotics community. Results here, and in [Lee *et al.* 97] show that logic level controllers are highly tractable to Genetic Programming and merit further consideration.

Evolutionary algorithms are not all powerful[1]! Through task decomposition complex behaviours can be recursively broken into simpler sub-behaviours, reducing an intractable problem to many tractable, independently evolvable ones. It is hoped that these can then be effectively recombined to give the complete behaviour. Such success will depend upon the design of the decomposition and results presented show how bad design[2] can impose constraints that hinder evolution and performance: A monolithic controller is seen to perform better than a hierarchical one. In some cases decomposition may be necessary and some guidelines about how to go about this are presented based on lessons learnt.

Tackling the problem of vision is also important if evolutionary robotics is to progress to real world problems. Vision introduces problems to simulation, and complexity in evolution. Both of these are circumvented by abstracting

[1]EA generated solutions are often presented with no impression of the work involved in getting the EA to solve the solution, perhaps portraying them as more automatic/powerful than they are. Throughout this thesis details of the steps involved in tuning the fitness functions and EA parameters have been endeavoured to be documented to avoid this potential illusion.

[2]not intended as such

vision to the object level: '*virtual sensors*' are evolved to give such object-level descriptions using supervised learning on a training set of images. This is a novel approach, and is open to criticism, particularly that of reverting the problem of vision to the domain of static observer and classical machine vison rather than exploiting the interactive nature of robot perception. Indeed it proved a struggle to get good sensors using this technique, but given such sensors this level of abstraction is attractive: Controller evolution using vision was as tractable as for simple low bandwidth sensors, even with highly unreliable *virtual sensors*. Simulation was simple and fast.

The project is based on work by [Lee 97] with extensions of vision and complexity of task. Implementation of the above techniques, particularly that of *virtual sensors* were time consuming and as such the work is largely a development of a technology. The techniques were applied to the task of visually locating a ball and pushing it to a visual goal — inspired by robot football. This is a hard task in the sense of other achievements in evolving robot controllers and is at the current state of the art in the field. A controller that could score goals in the real-world, starting facing up to 90° from the goal, and dribbling the ball from the half way line, was evolved. The general success stands as an existence proof, showing these techniques to work, and calls for further development and comparative work to assess their true worth. The complexity of the task goes some way to validating the scalability of the approaches and it is hoped that the principles of scalability upon which *virtual sensors* and task decomposition were based will allow application to more complex problems.

1.1 Thesis Structure

Chapter 2 introduces the field of evolutionary robotics, introducing relevant concepts and providing some context for the work presented.

Chapter 3 introduces the controller architecture and genetic programming as the means of evolving such controllers.

Chapter 4 introduces the robot used — a LAMI Khepera with K213 vision turret, the environment — a flat surface bordered by uniform walls with a vertically striped goal, and the task — pushing a tennis ball to the goal.

Chapter 5 details the simulation of the Khepera and the environment. The approach of look-up tables is used to create a 'virtual reality' of IR sensor response and Khepera movement. Problems with the intended methods of vision modelling, leading to the idea of *virtual sensors*, are explained.

Chapter 6 explains the method and principles behind the *virtual sensor* technique, along with a conceptual argument for and against such an approach.

Chapter 7 chronicles the *virtual sensor* development and describes the characterisation of the evolved sensors for modelling in simulation.

Chapter 8 describes the evolution of the controllers. The decomposition and fitness functions are described. Controller modules are evolved and the subsequent reality transfer is presented. Some analysis of controllers at the behavioural and logic level is made. An experiment on controller decomposition is included.

Chapter 9 summarises the work and suggests future directions.

Chapter 2

Evolutionary Robotics

Evolutionary robotics is the application of evolutionary algorithms to the synthesis of robot controllers, and in some cases robot morphology [Hautop-Lund *et al.* 97]. It has emerged as a response to the growing realisation that designing increasingly complex robots using the behaviour-based paradigm, e.g. [Brooks 91a], is disproportionately difficult and that "interesting robots may be too difficult to design" [Harvey *et al.* 93]. It is hoped that by using the semi-automatic approach of evolutionary algorithms robotic behaviours can breach this impasse.

2.1 A Difficult Design Problem

A number of observations have been made about why behaviour-based robotics controllers are difficult to design. Behaviour-based controllers are characterised by their behavioural, rather than functional, decomposition. In the archetype architecture, Brooks's subsumption architecture, behaviour modules are wired together in task producing layers linking perception and action in a highly reactive manner [Brooks 85, Brooks 90]. More complex task functionality is added in the form of new sense$\rightarrow$action layers that interact with existing layers solely by means of suppression, inhibition, or message passing to modules therein. Brooks alludes to natural evolution as the inspiration for this incrementalness [Brooks 90, Brooks 91b]. This is in contrast to the classical treatment, prevalent until the late 80's, of sense, planning and action as being largely independent of each other.

A behaviour based controller is thus a highly distributed system with the associated complex dynamics, and [Resnick 97] notes the problems designers can have with distributed systems in general. Whilst there have been many successes with simple, insect-like behaviours, there is a problem in design scalability because the complexity can scale with the interactions between modules: exponential with respect to modules in the worst case. [Gomi & Griffith 96] note that examples of early behaviour-based controllers were no larger than several dozen competence modules; [I. Harvey 97] put the figure at 10 layers.

Further problems arise from the robot being an embodied agent, having interactions with an external environment. The result of actions is often hard to foresee and uncertain. The sensors are only measuring devices, often noisy and non-linear, they do *not* give descriptions of the world in terms of objects [Brooks 92]. Good characterisation is needed if the designer is to see the world from a robot-eye view and utilise this to give effective perceptual constructs. Even with a good understanding of the response of a sensor it is hard for the designer to consider all secondary sources the sensor will respond to. In addition, [Nolfi *et al.* 94] notes that action selection design is made difficult by the strong sense-action coupling in behaviour based systems: the robot's behaviour depends on current stimuli, which depends on previous actions, ..., and a loop is formed. A choice of motor action is thus linked to previous actions and we now have temporal agent-environment interaction complexities.

2.2 Evolution as the Solution?

Given the above burdens on the designer and the resulting limitations design is placing on the behaviours achievable, using the semi-automatic method of genetic algorithms is attractive. These population based, empirical credit assignment [Angeline 94] search methods have been applied successfully to many AI fields including the evolution of robot controllers. Their efficacy aside they are also superficially attractive for their biological metaphor and its relevance to evolving agents.

Using evolutionary algorithms allows the roboticist to declare the robot controller at a task or behaviour level, ideally without having to worry about any lower level mechanisms. This is done through the user defined fitness function, which also interacts with the environment, to define successful behaviour.

A minimum evolutionary run might involve a population of 30 robots, running for 30 generations. It can be seen that a fitness evaluation time of the order of minutes will give a run time of the order of days. Such assessment times may be practical for trivial behaviours such as obstacle avoidance but as complexity increases more time is needed to encounter and check for the full repertoire of behaviour. Multiple trials comprising a single assessment have also shown to be beneficial [Reynolds 94b, Jakobi 94a], further increasing run times. Operating over such long periods of time the robot physiology is susceptible to change — batteries lose capacity and drive-trains wear, leading to altered dynamics — something the evolutionary algorithm will have to adapt to. Examples of real-world assessment are [Floreano 97] where collision avoidance takes 2-3 days to evolve and [Cliff *et al.* 92] where location and movement to a visual target takes approximately 1 day to evolve.

It can be seen that the principal problem for evolutionary robotics is one of time duration. Solutions include assessment in simulation during evolution then down load of the fittest controller to the real robot [Nolfi *et al.* 94, Jakobi 94a, Cliff *et al.* 93], and running parallel assessments as in nature (suggested by [Floreano & Mondada 96]). The common choice is simulation and indeed most evolutionary robotics is done in simulation. Such an approach can accelerate assessment by a factor of 10^3 [section 5.1, below] and reduces the resources required, robotic and environmental, but brings with it a new set of challenges; Section 2.3 below details these.

As well as being a necessary solution to human designer limits there is a stronger viewpoint of evolutionary robotics as being emphatically preferable to hand-coded design: By choosing a low-level representation for the genetic algorithm the controller can be more plastic than a prototype used by a designer, allowing the controller structure to emerge from the evolutionary process. In this way, with no dogmatic adherence to, e.g., behavioural or functional decomposition, the best controller type [or in practice degree of each] for the job can be selected [Harvey *et al.* 93]. Such an argument is more fundamentally about introducing assumptions/knowledge into the controller, and there are proponents for and against this. The task decomposition used in this project is an example of introducing domain knowledge and section 3.3 looks at this issue in more detail.

2.3 Simulation as the Solution?

Accepting that simulation is neccesary if robot controller evolution is to occur in feasible time introduces the central issue of simulation to reality transfer of the evolved robot. In an early evolutionary robotics paper [Brooks 92] highlights some of the expected pitfalls: That effort will go into solving problems that do not come up with the real robot and world (regular validation on real robots is recommended). That there is a near certainty that programs evolved in simulation will fail on transfer to the real world because of the differences in sensing and actuation arising from the difficulty to simulate the actual dynamics of the real world. These are valid cautionary points and were perhaps a reaction to artificial life style 'grid worlds' being used at the time, e.g. [Pollack & Ringuette 90]. Since then there has been much work, and success in "crossing the reality gap" e.g.[Lee *et al.* 97, Jakobi *et al.* 95, Nolfi ss].

A number of approaches have been used and shown to work, for simple problems at least:

Mathematical modelling has been used by e.g. [Jakobi *et al.* 95]. A two-dimensional, spatially continuous world with discrete 100ms time steps was modelled, with the kinematics, sensor response and propagation of radiation based on control theory and physics equations respectively with empirically defined constants. Noise at the real-world level is added. The level of detail and decidedly salient features to model were reached through intuition and experimentation, and a need to keep computational costs for a single assessment low. Obstacle avoidance and light-seeking behaviours were evolved and transferred to reality with no qualitative changes in behaviour.

The look-up table approach [Nolfi *et al.* 94] is a data intensive method that involves sub-sampling of sensor response over all possible scenarios that can be encountered in simulation. For example an IR sensor response to a wall is measured over 20 distances over 180 orientations to the wall. Given a particular situation in simulation the actual responses of the sensors can then be recalled — recreating a faithful representation of the robot's world. Such an approach takes into account the idiosyncracies of each sensor and [Nolfi *et al.* 94] found that each sensor responded in a significantly different way from other 'identical' sensors when exposed to the same situation. This approach can also be applied to the dynamics of the robot, measuring movement for combinations of motor commands.

In contrast to mathematical modelling, the look-up table approach offers greater faithfulness to the real world and lower computational overheads, but has diminished generality. [Nolfi *et al.* 94] recognise that the look-up table approach becomes more costly as environmental complexity increases: Without any generic assumptions, samples of every obstacle must be taken, perhaps resulting in a too highly memory intensive simulation. Combining sensor measurements from individual objects is a potential problem — combinatorics prevent sampling of all relational positions. Non-symmetrical objects require sampling from many view points; in [Nolfi *et al.* 94, Lee *et al.* 97] circular obstacles are used. For these reasons the scalability of such an approach is questionable.

The issue of scalability is of vital importance if evolutionary robotics is to transcend its relatively simple controller behaviour achievements, to fulfill its raison d'etre as a means for taking robotics past the limitations of human design.

The Mathematical model approach has been validated in terms of coping with sensor model complexity up to the level of IR and ambient light sensors in fabricated 'two dimensional' environments containing a few geometrically regular objects. The level of dynamical complexity modelled by either approach is so far very basic, typically involving no interaction with the environment except between the wheels and floor. The current state of the art using either approach is no interaction with walls and perhaps pushing of a cylindrical box [Lee *et al.* 97]. Using the *radical envelope of noise approach* and *minimal simulation*, see section 2.3.1, [Smith 97] has modelled interaction with walls and ball-pushing. It should be noted that in nearly all cases mentioned a LAMI Khepera research robot is used. The properties of light weight, small size, cylindrical shape and precise P.I.D controlled stepper motors make it far easier to build an accurate simulation of a robot of this sort than it would be for many others [Harvey *et al.* 93].

Although in principle entirely extensible, in practice limitations of the computational requirements of the model and the skill/resources of the modeller have prevented this. As mathematical models strive for increased realism at some point simulation-time or design-time will make the approach slower than real-world evolution. The success of such an approach therefore relies on the designer making the right approximations such that reality transfer is still possible.

2.3.1 Noise

A number of experiments into the effect of noise in simulation have shown that it is important to faithfully represent the real environment's noise levels. In the extreme case of zero noise, evolution will exploit the fact that the robot will behave identically in similar situations [Harvey *et al.* 93], but such reliability cannot be counted on in the real world. In the case of too much noise one can get freak peak or trough levels that no longer trigger behaviours in the real world. [Nolfi *et al.* 94] discovered that adding a translational positional noise they termed 'conservative noise' to the robot in simulation eliminated the drop in fitness upon reality transfer observed with conventional or no-noise in simulation.

In [Jakobi *et al.* 95], it is noted that noise added above the level of that found in the real environment "may help to cope with the inevitable deficiencies of the simulation by blurring them". This has since developed into the radical envelope of noise hypothesis [Jakobi 97a] where a distinction is made between features which may have some bearing on the robot behaviour (*base set aspects*) and those that should not affect behaviour (*implementation aspects*). Non-reliance on *implementation aspects* is ensured through making them unreliable through variation over trials; in such a situation fit individuals must ignore these aspects of the simulation. This approach is powerful in terms of allowing generalisation — e.g. the colour of a ball could be made an *implementation aspect*, ensuring that the controller would not be sensitive to the actual colour of the ball in the real world — and evolving robots that operate in more complex environments, by selectively removing the need to model certain aspects by making them *implementation aspects*[1]. The emphasis of simulator design moves away from trying to minimise all differences between simulation and reality to acknowledging these and preventing the controller relying on them. This technique has been successfully applied to evolution in simulation, and subsequent transfer to reality, of corridor following, visual target seeking,[Jakobi 97b] and robot football [Smith 97].

2.4 Vision

This project uses camera-like (one-dimensional) vision. It is intended that this be used for navigation purposes, facilitating identification of objects at a distance. Given that the complexity of sensor required is related to the range of behaviours and environment in which the tasks are to be performed, it is essential evolutionary robotics tackle vision if it is to progress past 'toy problems'.

Examples of simulating vision in evolution are [Floreano & Nolfi 97] and [Smith 98] using thresholded and grey level Khepera vision module images respectively. [Jakobi 98] and [Jakobi ng] use 32 pixel and video resolution images, modelled using the *radical envelope of noise* approach.

2.5 Comments

This chapter has focused on the issues of evolutionary robotics relevant to this project. Other issues/challenges, not necessarily focussed on in this project, outlined by [Matarić & Cliff 95] include:

- Fitness Function Design: It is non-trivial expressing behaviours in the

[1]although the base set must of course be adequate for the desired behaviour to be a priori possible

form of a fitness function. The exploitative nature of the GA will often 'cheat', finding a way to satisfy the fitness function that does not produce the behaviour intended. The design process is often one of incremental trial and error, which unfortunately is often masked by failure to report on this process.

High level fitness functions, e.g. goals scored, are often inadequate as they do not provide enough guidance — the step from not scoring to scoring is a large one and the fitness landscape has no intermediate points, up which the evolution can gently climb. Given this, componential fitness functions are required, for example the standard collision avoidance one consists of move-fast, move-straight and avoid-objects components. For more complex problems the interactions between these sub-goals and their relative weightings further exacerbate design. Work towards high level fitness functions working in conjunction with ecological constraints by [Floreano & Mondada 97] is a potential solution. The decomposition used in this project, see chapter 3, is another.

- Co-evolution: As a powerful method for searching fitness landscapes, in particular for evolving group or adversarial behaviour. See e.g. [Miller & Cliff 94, Floreano & Nolfi 97].
- Genetic Encodings: The controller must be expressible as an encoding, phenotypic or genotypic, suitable for manipulation by a genetic algorithm. See chapter 3.
- Evolving Morphology: Brooks advocates this in [Brooks 92] as a method for reducing the size of the search space. Robot morphology ostensibly has a large effect on behaviour, and co-evolution of body and controller can optimise both and the interaction between the two. Taking this approach is consistent with the emerging view of mind-body as a strongly coupled whole [Clark 97]. [Hautop-Lund *et al.* 97] evolve

wheel-base and radius, body size, and sensor positions along with the controller, in simulation only.

The motivations behind evolutionary robotics have been outlined. It promises a great deal although in its present nascent stage is failing to work at even a labour-saving level, with evolved controllers taking more or the same amount of effort as would have been required had they been hand crafted [Matarić & Cliff 95]. The author considers the central problem to be one of the [inevitable] simulator design and subsequent need to cross the reality gap, with the scalability of technique used being paramount. There are advances to be made in terms of complexity of dynamics and sensory environments. Additional approaches to crossing the reality gap include evolution in simulation followed by further evolution in the real environment [Nolfi *et al.* 94], and choosing controller representations that are robust to real world transfer [see section 3.2, below].

Chapter 3

Genetic Programming the Controller

The Genetic algorithm of choice for this project is Genetic Programming, GP, an evolutionary search algorithm with a tree data-structure encoding. Individual trees are (typically) functional, LISP-like programs. The possible functions, located at the branching nodes of the tree, form the *function set* and the possible atoms, located at the leaf nodes, form the *terminal set*. A population of trees is manipulated by an evolutionary algorithm, briefly described in section 3.4. [Koza 92b] provides a comprehensive introduction to GP and it will only be discussed further in the context of the variant used in the project. Evaluating the effects of variations of GP and GP parameters on the evolution is not an aim of the project. The controller representation used was developed by [Lee *et al.* 97]. The 'sgpc' program [Tackett and Carmi] is used as the GP engine.

3.1 GP in Evolutionary Robotics

The first example of GP use in evolutionary robotics is [Koza 92b], with wall following and box pushing behaviours evolved in simulation. The function set was {IFLTE (if arg1 less-than-or-equal-to arg2, do arg3 else do arg4)[1], PROG2 (a Lisp construct for sequencing evaluation of its two arguments)}; the terminal set consisted of sonar sensors (returning distances to the nearest wall at which they were pointing), actions (moving backwards/forwards, turning by a constant amount), and carefully chosen constants such as the minimum safe distance to avoid hitting a wall. As a piece of evolutionary robotic research[2] this work remains unvalidated since no attempt was made to test the controller in reality, and so it should be considered as Artificial Life. It has been criticised as: Relying on the simplicity of the simulated environment [Brooks 92], with no noise modelling; Choosing the terminal/function set on the basis of an existing control program [Matarić 90] that could perform similar tasks, thus providing no evidence for GP as a general method; Using artificial sensors, returning an absolute description rather than a response, and artificial actions, being absolute distances of motion not motor commands.

Another example is the sequence of work [Reynolds 94c, Reynolds 94a, Reynolds 94b] evolving corridor following and collision avoidance. The function set was {+,-,*,% 'protected divide' (preventing overflow if division by zero is attempted),ABS,IFLTE,TURN,LOOK_FOR_OBSTACLE (returning the distance to the nearest obstacle in the direction of its argument)}, the terminal set was real constants. This work is more realistic than [Koza 92b] with noise and robustness of controllers with respect to e.g. different starting positions being considered. However, the sensors used still return a distance rather than a real sensor response and the evolved controllers have not been validated in reality. The tree representation used was real-valued with the overall value of the tree being transformed to a steering direction, in which

[1]A conditional statement returning a real number — consistent with the rest of the tree nodes

[2]which the author may not have intended it be

the robot moves for a fixed period of time.

The work of [Lee *et al.* 97] is the only example, known to the author, of GP evolved controllers being validated on real robots, and hence could be said to be the sole example of GP evolutionary robotics, as opposed to GP artificial life. Box pushing towards a light was evolved and this is at the current state of the art in terms of behavioural complexity in evolutionary robotics. Lee's controller architecture is used in this project and is detailed in the next section.

3.1.1 Controller Architecture

It is an entirely reactive, combinatorial logic system in which the output is determined solely by the sensor state at evaluation [Lee *et al.* 97]. A combinatorial logic system can be mapped to a boolean network which can in turn be mapped to a boolean tree [Bryant 92], giving a architecture suitable for use with a standard GP representation. The function set of the boolean valued tree is {AND, OR, NOT, XOR}[3]. The leaves of this boolean tree are structured sensor conditionals, consisting of a comparison function (>= is used) taking terminals as arguments. The terminal set consists of real constants and sensor values. Thus sensor values are compared with other sensor values or 'thresholded' against real constants (taking values in the range of the sensor values) to give a boolean value compatible with the higher level branches of the tree. Such a tree (see figure 3.1), with two data types, functions designed for specific data types, and syntactic constraints on tree structure is termed a *strongly-typed* GP [Montana 93]. It would be possible to do without syntactic constraints but would result in a higher proportion

[3]although some of these are superfluous in terms of being able to define any boolean network this is not necessarily a bad thing. Although enlarging the search space, strictly unnecessary functions may make a solution easier to find. Selection of function set remains something of an art form.

of ineffective subtrees.

As described so far, a tree will transform sensor state to a boolean value. A number of boolean values are required if the output is to be used to specify the states of two motors (e.g. for a Khepera) so the values returned by N such trees are used, these trees are joined by a dummy root node. With N=6, as used in this project, the output of such a tree can be interpreted as 8×8 possible motor commands, and this is deemed continuous enough for the purposes of the tasks attempted.

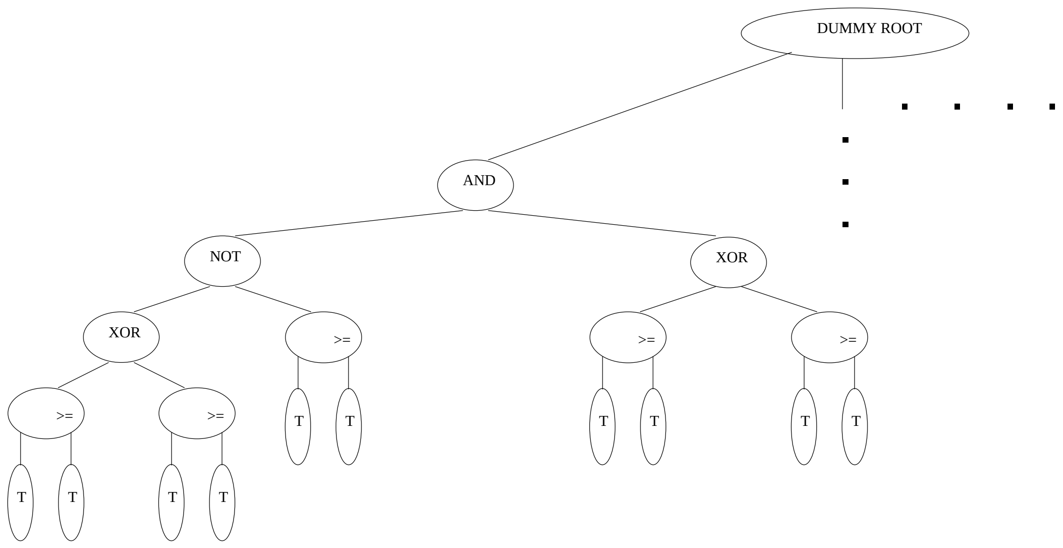


Figure 3.1: A partially drawn controller-module. T denotes member of terminal set {SENSOR, REAL-CONSTANT}

3.2 Should GP be used?

The review in section 3.1 shows the use of GP in evolutionary robotics to be fairly limited. Neural networks (NNs) evolved using a genetic algorithm, are the prevalent choice and a number of reasons have been cited for their use:

- That small variations in the structure or synaptic values of a NN result in small variations of the behaviour of the controlled robot, and that the resulting smooth fitness landscape of the search space aids the genetic algorithm [Floreano & Mondada 96].

- Their resistance to noise and breakdown, and their generalisation abilities [Floreano & Mondada 96].

- That they have the ability to adapt through learning after evolution [Nolfi *et al.* 94]. [Brooks 92] cites this as an essential feature of any controller evolved in simulation.

- That the primitives manipulated in the evolutionary process should be at the lowest level possible since higher level semantics restrict the possible set of controllers available to the evolutionary search [Harvey *et al.* 93].

None of these properties of NNs, which GP programs may or may not have to a lesser extent, is grounds for dismissing GP. [Floreano 97], who uses NNs, concedes that there is not enough evidence for the superiority of one type with regard to generalisation and complexity of behaviours of the evolved robot. Indeed the lack of work with GP, and the success of [Lee *et al.* 97] means it is a potentially fruitful area of research. The last objection is largely a matter of whether one sees designer input as misguided prejudices hindering the evolutionary process or expert domain knowledge cutting down the search space and guiding the evolutionary process. It is more relevant to the higher level *behaviour language* genetic programming representation proposed by [Brooks 92] than the low-level logic networks used in this project.

Genetic programming is distinguished from the standard genetic algorithms typically used to evolve neural net robot controllers by its variable length tree representation. Such extensibility is attractive if it is to be applied to evolution of arbitrarily complex robot controllers. Work at Sussex [Harvey 92] uses a variable length genotype genetic algorithm for similar means. In principle incremental evolution, starting with shorter genotypes with presumably simpler behaviours and moving towards longer genotypes until sufficient behavioural complexity is encoded, is possible. Given this the, roboticist need place fewer prior restrictions on the size of controller; underestimation may prune the set of all good controllers from the search space, overestimation may unnecessarily slow the evolution.

3.3 Shaping Through Decomposition

It is proposed that, at some level of behavioural complexity, evolution of complete homogeneous controllers will become intractable [Perkins & Hayes 96]. An incremental method/architecture is thus eminently more scalable and the architecture of [Lee *et al.* 97] has this property via task decomposition and subsequent recombination through arbitration. Task decomposition defines an incremental path through lower level behaviours being combined to form more complex behaviours.

Decomposition is done using the behaviour based archetype, where behaviour producing primitives are coordinated through arbitration. The primitives have the form of the controllers already described and the arbitrators are similarly reactive networks, their boolean output switching activation between behaviour primitives rather than being interpreted as motor output. Using such an approach, hierarchies of arbitrators and behaviour primitives can be evolved so as to select the 'correct' behaviour according to sensor state. Such a plasticity affords evolutionary adaptation at the sensorimotor level *and* behaviour selection level. There is now a requirement for a designer to select the decomposition of the task into sub-behaviours, and it is hoped that the modularisation chosen will be effective and that it will make tractable, or speed up, the evolutionary process. This is likely since human designers are good at decomposing complex tasks at a coarse, or behaviour scale, or are at least better at this than at a lower level sensorimotor scale. Such a belief in the designer is the foundation of *robot shaping* [Perkins & Hayes 97], where a similar hybrid engineering-evolution approach is proposed.

Using such a decomposition, suitable sub-sets of sensors can be selected for each primitive/arbitrator. Such selection is again imparting domain knowledge, and prevents the evolutionary algorithm from having to ascertain e.g. that an IR sensor is useless when it comes to finding the goal-mouth in a robot football game. Decomposition also makes design of the fitness function easier by reducing the number of components per function (see section 2.5)

An example of a hierarchical architecture as used in this project is shown in figure 8.1: using an example of a robot footballer, the root node might be an arbitrator between 'find-ball' and 'push-ball-to-goal' behaviour-producing subtrees.

3.4 The Evolutionary Algorithm

The genetic programming works on a population of controller trees. The initial population is randomly generated in this work although it can be seeded with hand-coded controllers in an attempt to guide evolution. The *generational* variant is used initially, where the fitness of the entire population is evaluated before fitness based selection [4] of individuals to pass their genetic material to the next generation. Selected individuals are either copied into the next generation, mixed through crossover to produce offspring, hopefully combining good traits of both parents to produce even better individuals, or mutated. Mutation involves pruning and randomly re-growing sub-trees,

[4]tournament or fitness-proportional methods were used, the choice was found to be insignificant in terms of algorithm performance

crossover involves swapping sub-trees between individuals, the details can be found in [Koza 92b]. The syntactically constrained nature of the controller tree requires syntax preserving crossover, which is simply a matter of selecting crossover points in the parents to be at the same node class (logic-function, $>=$ function, and terminal). The crossover, mutation and copying probabilities are fixed and user-defined. The success of the algorithm was found to be dependent on, although not overly sensitive to, these parameters. As a rough heuristic crossover values below 0.4, and a split of the remaining 0.6 between mutation and copying, worked equally well.

The evaluation involves running the robot in simulation for a given number of time steps, possibly over multiple trials, and calculating its fitness over these based on the *fitness function*. The controller is evaluated every 200ms, roughly approximating a continuous reactive controller, on the current sensor state, the motor speeds being set according to the controller output and remaining unchanged until the next update. The total fitness is typically the accumulation of the fitness at each time step.

Selectional pressure in the evolutionary process causes convergence. Premature convergence, where the population is suboptimal and homogeneous, should be avoided. Diversity can be maintained through lower selectional pressure (e.g. via smaller tournament size in selection) and through an island or cellular model GP, both of which were available in the GP system used. In the island model smaller populations are evolved in parallel with occasional migration of the best individuals between populations. The migration parameters must provide a balance between global mixing and maintaining local diversity. In the cellular GP individuals inhabit cells in a grid, mating is more likely with nearby individuals and so diversity can be maintained through geographical isolation.

Multiple trials are important to [statistically] counter the effects of non-determinism [Matarić & Cliff 95] and to control for the effects of *implement-*

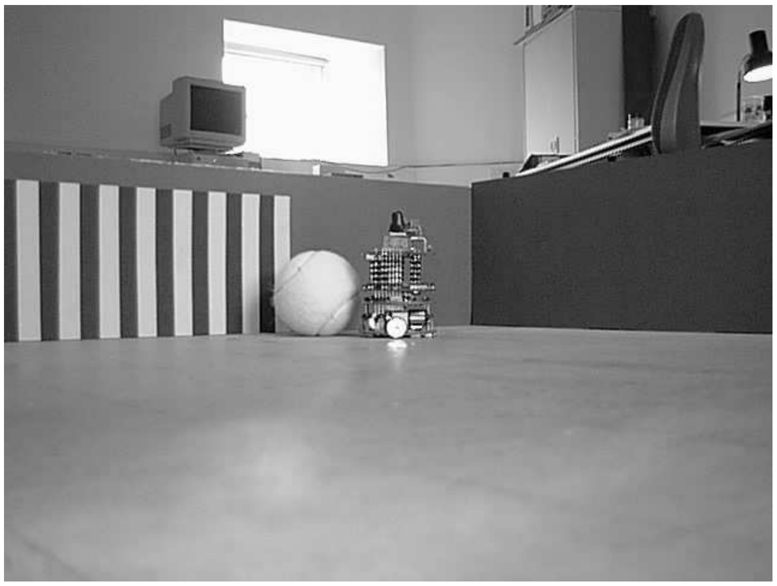
ation aspects — for example the starting position of the robot. As in the Jakobian definition, section 2.3.1 above, 'implementation aspects' are those that the evolved robot should not rely on. Variation of these prevents 'brittle' controllers from over-fitting, where exploitation of a regularity prevents generalisation to other conditions. Experimentation has been done on the best method of combining scores from multiple evalutations to give the fitness: [Jakobi 94b] notes how taking the mean gives a fitness closer to the true value[5] but does not promote robustness; a controller that performs very well on most of the training examples but fails on the others could be considered fitter than one which performs moderately on all of them. In contrast, taking the worst score ensures robustness although information is discarded; performing badly on all runs gives equivalent fitness to scoring badly on just one run. [Smith 97] prefers median-score as the fitness measure. [Lee 97] investigates random selection of a sub-set, from a super-set, of starting positions at each generation, finding that it reduces the computational cost of the evolution[6]. The method of choice is still an open issue, being largely problem specific.

[5]The fitness averaged over N trials, as N tends to infinity

[6][Gathercole 98] gives a thorough treatment of *Dynamic Subset Selection* in evolving GP classifiers

Chapter 4

The Robot, the Task and the Environment



The task was inspired by robot football. The use of ‘football’ to describe the behaviour is perhaps hyperbolic; more prosaically it is pushing an object to a target. In terms of behavioural complexity in evolutionary robotics this task is state of the art. The task and environment were shaped by the properties/limitations of the Khepera robot and the K213 vision turret.

4.1 The Khepera

The LAMI Khepera research robot with K213 vision turret is shown in figure 4.1. It is small in size ($57mm$ diameter, $60mm$ high), light weight with a smooth cylindrical plan profile, has an on-board 32 bit Motorola 68331 processor, is equipped with eight short range ($\approx 4cm$) infra-red proximity, and ambient light sensors (not used here). It has two powered wheels, each with a feedback controller (P.I.D) with highly accurate positional encoding ($\sim 10^{-1}mm$).

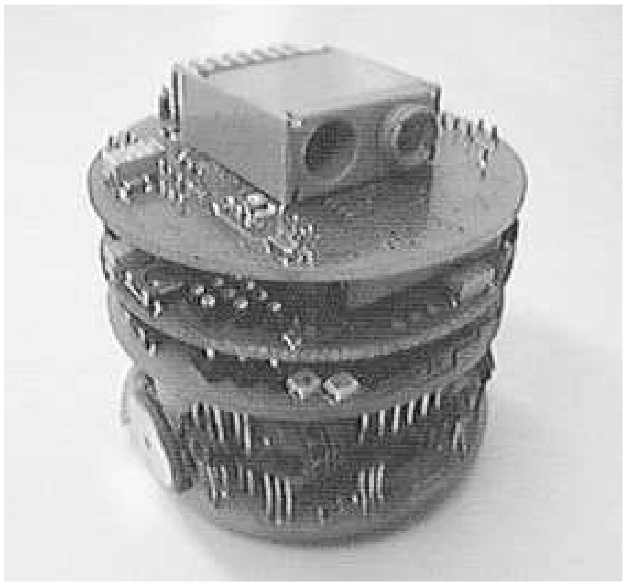


Figure 4.1: Khepera with K213 vision turret

The Khepera can run autonomously (for up to 30 minutes), using an on-board battery, or tethered with power being supplied by an overhead umbilical cord. Programs can be written in ANSI C using Khepera BIOS functions, compiled, then down loaded to the Khepera via this cord from a workstation serial line. The line also allows two-way communication with, and control of the Khepera using a communications package such as *Kermit*.

Multi-tasking is possible, allowing the behaviour primitives to work in parallel with the output being selected by arbitration. In practice a serial implementation is used for reasons of efficiency: Arbitration gives the behaviour primitive that has control, this is then evaluated to give the output. For this project a control program involves a 'sense $\rightarrow$ evaluate controller $\rightarrow$ send motor commands $\rightarrow$ wait' loop with each loop-cycle duration being $200ms$.

4.1.1 The Vision Module

The K213 vision module, figure 4.2, plugs into the top of the Khepera. It contains a one-dimensional 64 pixel line camera[1] and an ambient light intensity sensor. Output is a grey scale (256 levels with 0 being black) linear image: figure 5.3 shows images of background and ball against background. The view-angle is approximately 36 degrees and the focal range is 5 to $50cm$. The scanning period can be set as fast $50ms$; adequate for the $200ms$ time slicing used in the project.

The turret will return the brightest and darkest pixel indices if requested. Capture of sub-sampled images is also possible to give 16 or 32 pixel images.

The light intensity sensor is used to adjust the integration time of the camera according to the total ambient light level. This simulates an iris, improving the contrast of images and enabling operation over a wider range of lighting levels. An additional effect is to remove any cues of absolute light levels from the image; a white background can appear the same as a black background due to iris compensation. This was to have repercussions in the modelling of the vision. The iris functionality is hardwired and cannot be switched off, though the light intensity sensor can be read.

[1]Texas Instruments TSL213

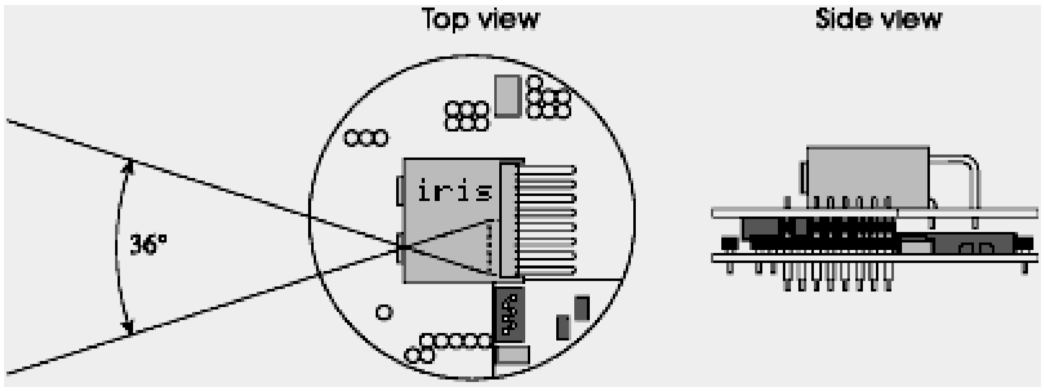


Figure 4.2: Schematic of the K213 vision turret

The vision and IR sensors are noisy with respect to time, given a constant input. Response characteristics vary over IR sensors and pixels in the camera.

4.2 Robot Football

Robot football is a burgeoning field within AI/robotics. The task of football has been proposed as a benchmark problem as it ultimately requires technologies of: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning and sensor-fusion [Kitano *et al.* 97]. There is a robot football world cup, the 'RoboCup', receiving 100 entrants this year. The style of football herein is closer to that in the Autonomous Robot Football Tournament with its single player sides and rejection of over-field cameras (permitted in RoboCup) allowing global position knowledge. In the ARFT single robots (invariably Khepera's with K213 vision modules because of size regulations) must push a tennis ball to a goal whilst an opponent is doing the same. Goals are coloured uniform black, the ball is white, the walls uniform grey and the Khepera's are dressed in black and white vertical stripes. The winner of this year's tournament scored 6 goals in 20 minutes.

For the purposes of this project the opponent is removed; resources did not allow the whole issue of coevolution that this would necessitate. Given this simplified interpretation of robot football there are still significant challenges a successful controller must deal with. It must locate and move towards a ball using a limited (low resolution and one-dimensional) vision sensor, then push the ball towards the goal which must also be visually located. Whilst pushing, the ball will tend to slide away laterally from the Khepera because of the point contact between two circular objects. The roll of the ball also introduces a natural time scale to the dynamics that the Khepera must keep up with.

Modifications to the Khepera

A fundamental problem is that the line camera is blinded by the ball when pushing it; the ball fills the field of view and the goal cannot be seen. Pushing a ball to goal with a reactive controller given such a situation seems infeasible. This is overcome by lifting the vision turret; the ball can be looked over when it is close but comes into view as separation increases because of lens dispersion in the vertical plane.

4.3 The Pitch (no grass)

This is a $800 \times 700mm$ smooth surface. Walls are used to act as neutral backgrounds, making the task of identifying ball and goal easier. For this purpose they are dark green. They are visible up to a range of $40mm$ by the IR sensors, making collision avoidance possible. A tennis ball is used, this is a highly visible fluorescent yellow, it is visible up to a range of approximately $30mm$ to the IR sensors and the length of the pitch to the vision. The goal is a black (darker than the walls) and blue (lighter than the walls) vertically striped region. In an image containing ball, goal and background there is good contrast between the three. The goal stripes provide a potential mechanism for discrimination between ball and goal: It should be remembered that because of the iris, objects can not be reliably distinguished on intensity alone. The stripes are wide enough ($15mm$) so that the striped region can be resolved at distances of up to the maximum focal range using 32 or 64 pixel images.

The pitch is in a room that is consistently well lit with diffuse overhead fluorescent lighting. Steps were taken to remove any dependence on this, see section 6.4, so that e.g. moving the pitch to another room will not result in failure.

4.4 Comments

The Khepera is a very convenient desk top research robot, but this is also a matter for caution: Its small size and desk top nature almost necessitate construction of special environments which can tend to be unnatural and simplified. The Pitch used here is a case in point. Success of a technique in any such environment is diminished by issues of scalability to the 'real world'. It is however useful as an initial testing ground for the methods and the author is sceptical of the capacity of the vision sensor to cope with more complex environments. If the work is successful, time allowing, realism could be increased by removing the visual screens (i.e. walls) for instance.

Chapter 5

Modelling the Khepera

The infra-red sensor response and dynamics are modelled using the look-up table approach introduced in section 2.3, above. Such a method can be used because of the simplicity of each of these aspects for the environment-task-robot combination used and confers increased realism and less computational intensity than the mathematical modelling approach. Modelling the vision proved more of a challenge, so a new approach of a higher-level 'virtual' or 'software sensor' was used.

5.1 The Simulator

A simulator developed by Wei Po Lee [Lee 97] and integrated with *sgpc* was adapted and used for the project. The simulator was ideal for adaptation and use in the project; being designed for Khepera and circular-box-pushing-to-a-target. The simulator does not model interaction with walls; collision with a wall results in termination of the run. This is fine for the football task as the walls were only introduced to provide a [IR] visible boundary to the pitch and to act as sight screens. The Khepera should not exploit physical interaction with the walls when playing.

The IR sensor, vision and dynamics models detailed in the coming sections were implemented. A run-time MATLAB graphical display was added; figure 8.6 shows a screen shot. This proved invaluable for debugging and can be run at speeds that allow the evolutionary process to be observed in reasonable time. Watching the evolution can help identify how individuals are exploiting the fitness function to behave in unintended ways. It also became apparent that such a facility would be very useful for run-time environmental shaping, with the author frequently wanting to modify parameters to drive the evolution in the direction that was really wanted. For example, as the robot comes to master the problem from all the starting positions specified it sometimes becomes apparent that its control strategy may fail in certain scenarios. Being able to introduce such scenarios would be extremely useful. Run time interaction with the evolutionary algorithm would also open up other shaping possibilities, incrementalness in terms of task/environmental complexity for instance. The author is not aware of any such research in evolutionary robotics.

The box kinematics model in the existing simulator treated the box as moving freely when being pushed by the Khepera: moving along the normal to the point of contact, and being stationary otherwise. This was extended to ball-like kinematics by letting the ball roll after it has been pushed; at the velocity and direction it is pushed in, with deceleration at each time step. Noise is added to the deceleration, the direction of velocity, the direction of movement during pushing and the separation of the ball from the Khepera after each pushing time-step. These aspects were designed to capture the irregularities of the tennis ball's movement and shape, with the last aspect approximating the tendency of the ball to rebound or roll away from the Khepera during pushing. In practice the ball-like aspects of the model were 'switched off' by choosing box-like values for the coefficients because accurately determining these was not possible given the lack of suitable experimental equipment. Section 8.4.1 details an (unsuccessful) attempt at introducing ball dynamics through choice of coefficients based on rough observation.

The simulator, with graphics disabled, gives speed up of order 10^3 over real world evolution, allowing most runs to be completed in the order of minutes.

5.2 The Infrared Sensors

Infrared snapshots of the wall and ball are taken starting from the point of saturation (2mm) and moving backwards in 2mm increments until the object in question becomes invisible to the sensors (28mm and 48mm for the ball and wall respectively). At each separation a 360 degree turn in steps of 2 degrees is made, with 30 readings being made for each sensor, with the average and standard deviation being recorded. The positional sampling frequency is adequate, with the variations due to changes in distance/angle being below the level of noise. Analysis of the standard deviations shows the noise characteristics to be independent of the IR sensor and object being viewed, but a strong function of the IR sensor reading, see figure 5.1. The noise is modelled by calculating the mean standard deviation [1] over IR sensor value 'buckets' to produce a noise look-up table, indexed by IR sensor reading.

The IR data is used by the simulator to produce a two-dimensional look-up table, for wall and ball, that can be indexed by distance and angle to object. The symmetry of the ball and walls requires only sampling over distance-from-object and angle-to-object. The indexed element is an array of 8 IR sensor readings to which the empirically derived noise is added. For distances-to-object outside the range of the look-up table, the values at the

[1]it can be seen that for medium band IR readings the distribution of standard deviations is skewed. This was ignored and it was assumed that using the mean standard deviation would provide an accurate enough model of the noise.

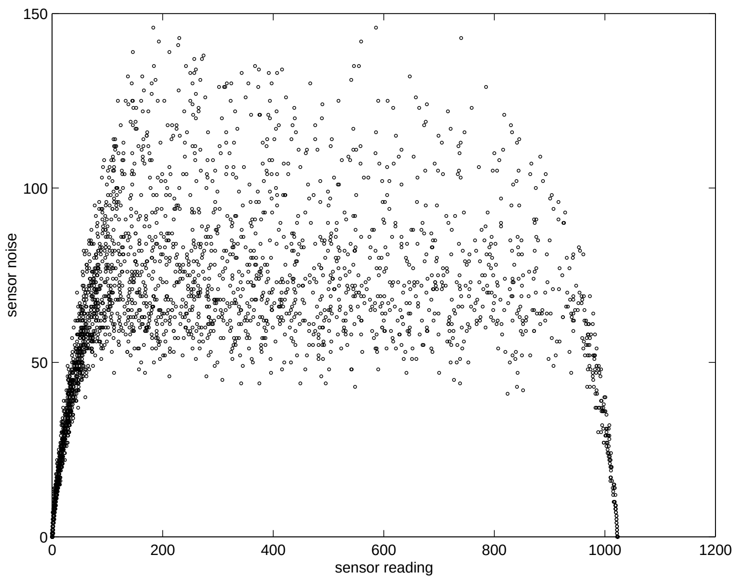


Figure 5.1: Scatter plot of IR sensor noise as a function of sensor value for data collected

appropriate limit are used.

5.3 The Dynamics

The movement of the Khepera, given a left and right motor command for the time slice duration, is described by a change in heading and a translation of the Khepera's centre.

Experiments showed that the on-board odometry corresponded well to the absolute movement of the Khepera. This allows high resolution [units of 0.08mm] on board measurement of the short distances moved by each wheel for each pair of motor commands. By measuring movement for sequences of motor commands it was found that it was reasonable to consider each pair of motor commands in isolation, with the movement being produced being largely independent of the prior and following commands. This approximation starts to break down with oscillations between full forward and full backward speed [$\pm 10\ 0.08mm/10ms$] where the positional deviation is still less less than 5%. It is attractive to keep this approximation since the size of the motor look-up table will cube if the immediate context of the motor state is considered.

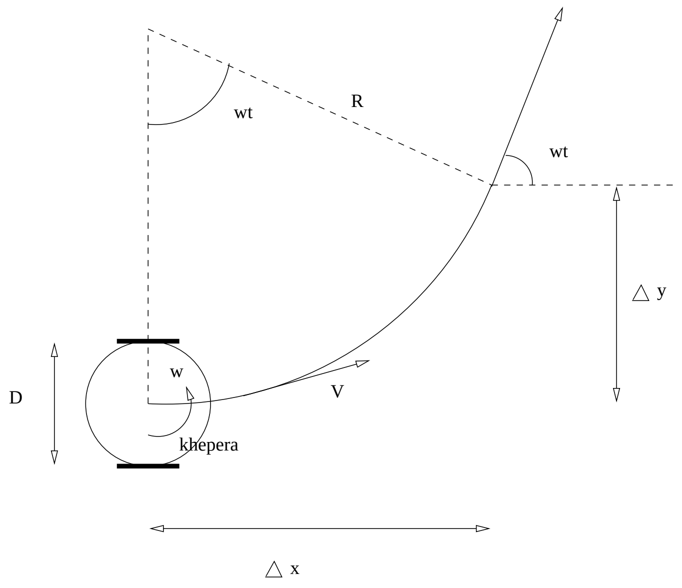


Figure 5.2: Movement of Khepera

All of the 8×8 combinations of motor commands were executed 10 times, the average and standard deviation movement of each wheel were recorded. These wheel movement measurements are transformed into translation and change-in-heading look-up tables using a model of the khepera following a circular arc, with a constant angular velocity [2]. See figure 5.2 and table 5.1.

Error for each of the quantities is also calculated and stored in look-up tables, combining the error in the measured quantities using the maximum

[2]this was introduced to the author by John Hallam and Graham Horn

$\triangle x$ and $\triangle y$ the x and y translation respectively of the Khepera centre and the change in heading $\triangle h$, in radians, are given by:

$\triangle x = Rsin(\omega t)$	
$\triangle y = R(1 - cos(\omega t))$	from trigonometry, where R is the radius of the arc followed, ω the angular velocity, t the time slice.
$\triangle h = \omega t$	
where ωt is given by:	
$\omega t = (\nu_r t - \nu_l t)/D$	
$= (d_r - d_l)/D$	where D is the wheel base of the Khepera, and $\nu_{l,r}$ are the over-ground velocities of the left and right wheels and are related to the over-ground movements of the left and right wheels, $d_{l,r}$, as is usual.
and where R is given by:	
$R = \frac{D}{2}\frac{d_l+d_r}{d_r-d_l}$	
since:	
$R = \frac{\nu}{\omega}$	from standard equations of motion, where ν is the tangential velocity given by $\nu = (\nu_l + \nu_r)/2$, where $\nu_{l,r}$ are related to $d_{l,r}$ in the usual way

Table 5.1:

possible error approach. Translational noise is of order 10^{-1}mm, heading noise is of order 10^{0} degrees, and in both cases is a function of the relevant quantity. These tables give movements in the Khepera's frame of reference. In simulation the values are indexed according to the motor commands given by the controller, noise added, and then transformed to global coordinates to give the absolute movement of the Khepera.

Interaction with walls does not have to be modelled due to the termination of the run upon contact with them. It was provisionally assumed that the Khepera's movement would not be altered when pushing the ball, with a view to modifying this if this assumption caused problems upon reality transfer.

The accurate P.I.D. controller and positional encoding of the Khepera greatly simplify modelling the dynamics.

5.4 The Vision Sensor

The initial intention was to use an extended look-up table approach for the vision modelling — 'Extended' because images of lone objects would have to be fused to give images of scenes containing multiple objects as a result of the impracticality of capturing a database of images of all scenarios. With a single object it is possible to do this but with multiple objects the exponential combinatorics make it totally unscalable. The idea is similar to cinema 'blue screen', where independent foreground and background images are combined to give a seamless composite image. In the case of this project look-up tables of images of the wall, goal and ball were to be made, at each time step the relevant individual images could be extracted and overlaid [with the wall at the bottom and the ball at the top] with some smoothing at the seams to avoid false edge artifacts. The pixel values of this composite image would then be used in the GP as terminals. Such an approach is potentially scalable, with calculations of occlusion, and false edge artifacts, and the implicit assumption that the appearance of the object is independent of its position/environment (e.g. what about shadows, changes in lighting) becoming an issue in the more complex case.

Characterisation of the K213 vision turret revealed some idiosyncracies that eventually contributed to the rejection of this technique. The scaling effect of the iris means that it is not a matter of simply superimposing ball on background. Figure 5.3 shows how the image of a ball against a background is dependent on the brightness of the background, information that is not contained in an image of the background-without-ball due the iris's intensity normalisation.

A potential solution would be to scale all images according to the iris value to give absolute intensity images. But, the iris does not normalise the image in the sense of giving a constant average intensity [or minimum or maximum intensity], nor is the average intensity a reliable function of the iris reading: A simple constant scaling or scaling based on an observable quantity would not give an absolute value. In the end the iris was 'paralysed' by plugging it with an LED, which went part way to solving the problem, although the contrasts of ball/goal/wall were greatly reduced.

There were also problems in determining where exactly the ball extremities were in the ball images[3] so that the correct section of image could be projected onto the background images. It was also expected that the GP representation would not be powerful enough to deal with pixel level input of the images, perhaps requiring an automatically defined function [Koza 92a] extension to the GP engine used[4]. These problems are probably surmountable but an alternative approach of 'virtual sensors' seemed fruitful...

[3]positional measurements of the ball and Khepera were not accurate enough to use geometry reliably

[4]a considerable undertaking given the time available

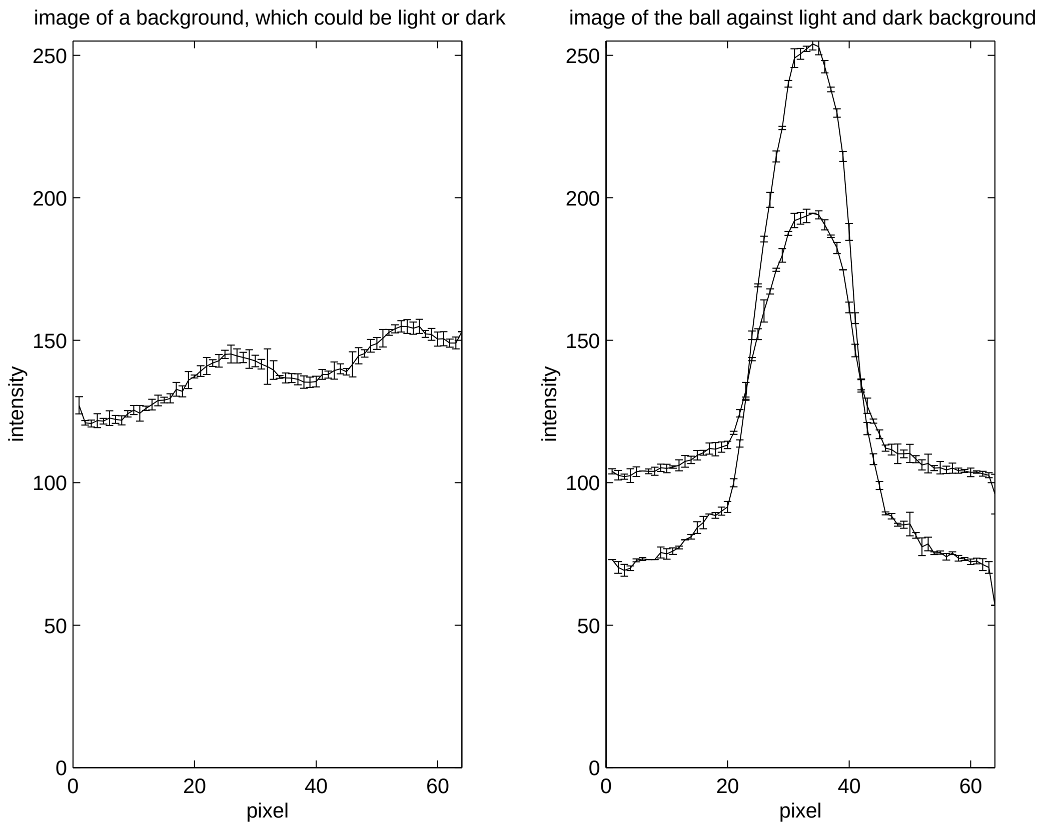


Figure 5.3: Image 1, a background, contains no clues as to how an image of a ball against it would appear (image 2) since the background could be light or dark

Chapter 6

Virtual Sensors

This chapter details the vision modelling technique used. The shortcomings of existing methods have already been discussed in section 5.4.

6.1 A What?

A 'virtual sensor' here is a software sensor or detector that returns an object level description of an image. For example a ball detector that returns TRUE when a ball is visible, FALSE otherwise, or a goal position sensor that returns a numerical value correlated to the position of the goal in the image.

These sensors could conceivably be hand crafted if one possessed enough expertise, however the problem can be posed as one of supervised learning and hence its solution (hopefully) automated. To synthesise a ball detector for instance, involves training a detector to output TRUE when presented with an image containing a ball, FALSE otherwise.

Given such high level sensors, vision at the pixel level need not be simulated, greatly simplifying the modelling of vision. Instead one need only know the output of the sensor for the particular positional scenario. Accurately predicting the output of the sensor requires identification of all the factors affecting its performance, which might be the angular width of the ball in the image or whether or not the ball is overlapping the goal for instance. With careful characterisation it should be possible to accurately predict the sensors' output for purposes of simulation. All parameters that could conceivably be factors were recorded during the collection of the images for the supervised learning. These were angular width of ball/goal, distance to ball/goal, angle to ball/goal and class of image with respect to occlusion, of which there were 30, shown in figure 6.1. The last factor classifies the ball as 'partially visible off the left of the scene', 'fully visible and occluding the right edge of the goal', 'no ball', etc. The classification was geometrically derived, being calculated from the relative position of the Khepera, ball and goal. A minor complication was the visible width of the ball changing with distance from the Khepera as a result of the vision turret being set up to peer over the ball when close: the coefficients of a parametric model of visual radius as a function of separation were derived empirically. Inspection of images showed that the geometrical classifier, the labeller of the images for the supervised learning, was reasonably reliable.

6.2 Genetic Programming

Genetic Programming was chosen, somewhat arbitrarily, as the supervised learning method. Alternatives were decision trees [1] and, the more established, neural nets.

Most of the examples of Genetic Programming image classification use feature rather than pixel level input. An exception is PADO(parallel Algorithm Discovery and Orchestration) [Teller & Veloso 95] which classifies 256*256 256-level grey scale images. The function set includes arithmetic

[1]in preliminary tests classifying images containing balls from those containing just backgrounds genetic programming outperformed the c4.5 decision tree algorithm by 97% to 90% images classified correctly respectively

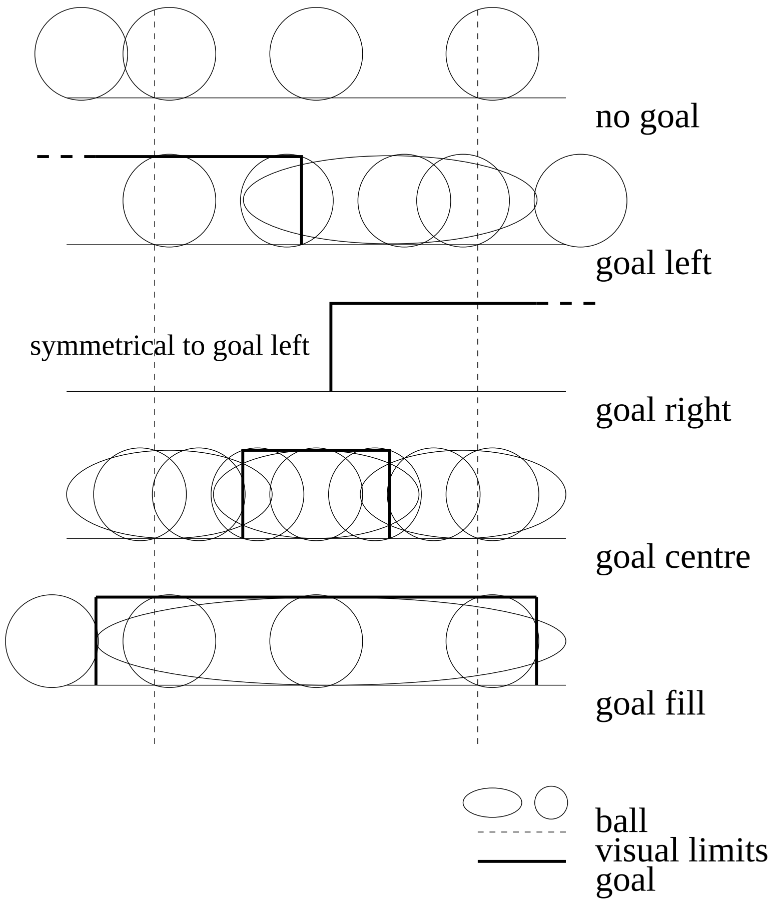


Figure 6.1: Classes of image with respect to overlapping edges of ball and goal and visual limits

operations, conditional constructs, pixel access, regional properties (such as maximum and average), indexed memory and evolveable library, functions. This is sophisticated (including parallel orchestration of a set of classification algorithms and "not simply a genetic algorithm" [Teller & Veloso 95]) and computationally intensive well beyond what is possible here. [Johnson *et al.* 94] evolve programs to give the position of a person's hand on a full body *thresholded* image, reporting that evolved routines perform better than those they were able to write by hand. [Tackett 93] evolves classifiers to distinguish images containing visual targets from those that do not, using a statistical feature vector of the image as input. The function set is {+,-,*,%,IFLTE}. GP is compared to multi-layer perceptrons and binary tree classifiers and found to outperform them.

6.3 The Implementation

Consideration of the Khepera CPU processing power, see appendix **??**, shaped the choice of function set. Arithmetic operators {+,–,*,%} and IFLTE comprise the function set, these can be combined to give any vision processing function one might reasonably require. More powerful operators such as the regional property functions in PADO are not used. Pixel intensities are used as input and two methods were implemented: One where each pixel is a distinct terminal, its value being the pixel's intensity; One where a PIXEL function accesses the pixel at an index given by its numerical argument and returns that pixel's intensity. In both cases the terminal set includes real constants (randomly generated in the initial population) in the range of pixel intensities or indices, respectively. The second method is more powerful, being able to direct attention to pixels depending on the image — the tree beneath a PIXEL function can contain other PIXEL functions. It is also more scalable with respect to larger images; a terminal set of size 64, as is required using the pixel intensities method for the highest resolution K213 images, is getting to the point where certain pixels may not be well represented in the initial population. The first method was implemented as a safeguard; [Reynolds 94a] associated failure of a corridor following robot with the use of directional attention [2]. The terminal set is completed with the maximum and minimum pixel intensities or indices, depending on the implementation, to give some global information, and since these are available 'free' from the

[2]Referred to as 'roving eyes' in [*ibid*]

Khepera's on-board vision processing.

The fitness of a detector depends on its performance on the training set: a database of labelled images. For presence detectors a tree returning a positive value for an image is interpreted as a positive identification. The fitness is the proportion of correct outputs over the whole training set, with 1.0 corresponding to a perfect sensor. For the position sensors, a rank based correlation method was used to avoid constraining the output to being anything more specific than a monotonically increasing/decreasing function of position. The images in the training set are ranked according to ball position and output of detector elicited. The correlation co-efficients of the two ranked sets are then calculated using the standard r-formula [3]. Again, a fitness of 1.0 corresponds to a perfect sensor.

Evolution is allowed to run for a fixed number of generations, or until a good enough sensor is produced. The true performance of the sensors is judged by evaluating fitness on an independent *validation set*. A sensor that can generalise[4] past the training set to the distribution from which those samples were drawn will perform equally well on the validation set. If overfitting to the training set is found to be a problem regular testing on the validation set can be used to stop the evolutionary process when there is a down turn in performance on the validation set.

6.4 Training Data

The success of any supervised learning is dependent on its training data. A good data set is one that is representative in some sense. In this case it should give reasonably dense sampling over the images the robot will encounter with

[3]the Spearman rank based correlation equation could not be used because it does not take account of equal rankings

[4]"The ability to deal with sparse and nonuniform statistical covering of sample space", [Tackett 93]

the distribution of sampling matching the probability of the image being encountered. This matching of distributions is important if the learning is to focus on the important cases, i.e. those that are encountered most often. Given that we are evolving a detector for a robot whose behaviour can not be known, such matching is unattainable and an educated guess about the common/uncommon types of images is the best that can be done.

A program was written to semi-automate the data collection process. Given relative positional information about the ball and goal the Khepera can use geometrical calculations to rotate almost past an object — until it is just visible at the far edge of the field of view, then rotate in 2 degree increments in the opposite direction until no object is visible. It repeats this, reversing each time. It was straight forward to sample over all the possible ball and goal images, the former just requiring reversing away from the ball, the latter requiring additional repetition over offsets from the centre of the goal due to the non-symmetry of the goal. Combinatorics of the images containing ball and goal required more configurations for this class of image; the procedure for taking pictures of just the goal was repeated, at a coarser scale, with the ball being moved away from and offset from the Khepera for each configuration. Images of just the background were also taken. Whilst the images were being taken changes in lighting were simulated by intermittently shading the Khepera/ball/goal — the evolved detectors should now be robust to such changes in lighting. From this database of images a training and validation set had to be selected to best meet the criteria outlined above. They were chosen to consist of 30% ball, 30% goal, 20% ball-and-goal, and 20% just background images making 50% of images contain ball and 50% contain goal. Within the set of ball-and-goal images, classes were represented in proportion to that in the collected data set. This was deemed to reflect the likelihood of such images given random movement, and no better approach was available. The selection of data within these constraints was done randomly, with data going into the training or validation set at random also.

Collecting large amounts of data is straightforward and so training and test sets can be made by simply randomly splitting the data and without having to invoke methods, such as *cross-validation*, advisable for small sets of data. The training and validation sets each contained approximately 1000 images giving a dense sampling of the image space whilst maintaining reasonable computational requirements for the evolutionary run.

Chapter 7 details the process involved in evolving the detectors.

6.5 A Good Idea?

The advantage in terms of ease of modelling for simulation of the virtual sensor approach has been noted. This section discusses other issues on the efficacy of such an approach.

The problem of evolving sensors becomes a *time independent* supervised learning problem. This has advantages in terms of evolving sensors to work in more general environments: a ball detector could be made to respond to golf balls as well as tennis balls by inserting suitably labelled images of golf balls into the training set. General environments can thus be expressed as a superset of specific environments and each of these specific environments need only be sub-sampled. Evolving a similar sensor with simulated vision would require complete capture of each environment in order to build a comprehensive model, then evaluation in each environment, in turn. By removing the time dimension it is true that dynamic sensors [5], combining movement and sensor readings are not possible. The evolved sensor could,

[5]There is no ostensible requirement for such a sensor in this project. However, an example is a distance-to-goal sensor that could conceivably scan the goal by rotating, measuring the intensity oscillation frequency arising from the stripes moving across the image. Being a function of distance from goal, this frequency is a depth cue.

however, be used by the evolved robot in a dynamic manner.

The main issue is one of *active perception* versus designer foresight or know-how when it comes to what should be sensed. Active perception is used here, following [Floreano & Mondada 94] as meaning:

> An autonomous choice of the sensory information extracted from that available and of the type of pre-processing performed on it.[*ibid]*

This extraction and pre-processing, given that primitive sensor information is available, will emerge during the evolutionary process as agents interact with their environment, with those individuals that attend to salient[6] information — using it in an effective manner, perhaps combining it with other sensory information, and filtering out irrelevant information — perpetuating their genetic material [*ibid*]. Use of the virtual sensor approach prevents such autonomous extraction by imposing a representation, in this case at the level of object (ball or goal), and discarding all lower-level raw-measurement information. Using the virtual sensor approach one might expect there to be less harmony between sense and action than when sense and action-selection can co-evolve; adapting to, exploiting and shaping each others functionality. An example of this was encountered during the collection of the training set data: With little knowledge about which images would be important to, and frequently encountered by, the as yet unevolved controller, uniform sampling and penalty in the fitness function was the best one could do. With coevolution the perception[7] need only perform well in frequently encountered situations. In both cases the controller can still adapt to the sensors, compensating for deficiencies in certain situations by learning to avoid these.

One can counter that for this task, and perhaps most others, it is fairly

[6]With respect to performing the declared behaviours

[7]it is perhaps a false dichotomy to be talking of perception and controller modules when this is the case, but it is done for comparison with the virtual sensor case

obvious what the robot needs to sense: in this case it needs to find and push a ball to a goal so a ball and goal detector will be useful. If the designer makes a good choice of representation it could be expected that a 'solution' controller would be more tractable than the case where one uses low level sensory information. This potential for increased tractability, the reason behind most engineering-evolution hybrid approaches, is promising in terms of scalability: the virtual sensor effectively hides the complexity of the vision from the controller by putting it in a 'black box'.

The complex problem of vision is not removed, it is merely being juggled with. There is still the problem of building the virtual sensor, which lies in the field of classical machine vision or pattern recognition. And that within these fields object recognition in noisy dynamic environments is recognised as a challenging problem [Boyle & Thomas 88]. It may be that the problem is being phrased in a more difficult way and there is always the possibility (or maybe certainty) that, by hand selecting what is sensed, one may have missed out on exploration of simple and more efficient visual mechanism that will solve the problem.

The experimental results of this project will contribute towards answering these issues of shaping versus unconstrained evolution: Whether inserting domain knowledge makes evolution easier by guiding the genetic algorithm in the right direction, reducing the search space, or whether the designer's [8] preconceptions (perhaps misconceptions) are misguided, pruning simpler, more efficient solutions from the search space.

[8] or rather my own

Chapter 7

Evolving the Virtual Sensors is Hard

This chapter describes the (lengthy) process involved in evolving the virtual sensors and the subsequent performance characterisation in preparation for evolution, in simulation, of the controller.

7.1 Synthesis

Initial results with the GP supervised learning setup detailed in the previous chapter were not 'satisfactory'. After a number of strategies were progressively employed, most notably an edge detector addition to the function set and large population size, a ball detector that correctly classified approximately 90% of the validation set and a goal detector that correctly classified approximately 80% of the validation set were synthesised. This poor upper limit of goal detector performance was to have repercussions in controller evolution. Preliminary attempts were made at evolving a ball-position sensor, though this was not used in the controller.

It is difficult to know what level of performance will be sufficient for effective control; manual characterisation of the evolved sensors is time consuming and hinders the use of feedback of evolved controller performance to determine whether the virtual sensors being evolved are good enough. The performance striven for was largely a matter of intuition and compromise. As good as possble virtual sensor performance was deemed important since the robot must take the sensor values as read; for each sensed property there is only one sensor, so there is no context on which the robot can base sensor reliability. Also, being solely reactive, the controller cannot integrate sensor readings over time to glean a more reliable indication of world-state.

7.1.1 The Ball Detector

Setting Parameters

A search over various parameters was undertaken to ensure the GP was being used to full effect.

During the data collection process 16, 32 and 64 pixel images were taken. Inspection shows that 32 pixels give high enough resolution to capture the goal stripes and ball. With 16 pixels some stripes are missed. The GP was run on all three image resolutions: In line with the above observation the performance of the evolved goal detectors on validation sets shows that 32 pixel images are adequate, giving significantly better performance than with 16 pixel images[1] and no difference to when 64 pixel images are used. 32 pixel images were chosen, being more amenable to any on-board-Khepera processing that might be required and perhaps improving GP tractability by removing redundant information.

Using PIXEL function accessing (see section 6.3) gives significantly better performance than pixel accessing via terminals, and is the chosen method.

[1]Where comparisons between variants are made, mean performance of the best individual in the last population on the *validation set* is taken over a number of runs (typically 5 because of computation-time constraints). 'Significance' in performance difference is used in the statistical sense and is determined using a t-test with ρ of 0.05.

Training set performance is the same for both approaches; the more powerful function accessing method is better able to generalise.

A coarse grained search of GP parameters was made, varying each independently for the most part:

- *Steady-State* versus *Generational*: The GP algorithm described earlier was generational, having distinct generations where the whole population is evaluated and then individuals are selected to form a new population to replace the previous generation. In the *steady-state* GP a single pair of mates is selected from the population one at a time, their offspring replacing weaker members of the population [through non-deterministic selection]. There are no discrete generations but it is convenient to consider a population-size number of such events as a generation. The *steady-state* GP gave significantly better performance and is the chosen variant. The *sgpc* program requires tournament selection[2] to be used with this variant.

- Tree Depth: Trees deeper than 8 and 14 levels give no significant increase in performance for ball and goal detectors respectively. Using larger trees increases the GP computational requirements and often increased the evaluations (the product of population size and generations) required before validation fitness stopped decreasing. Depths of 8 and 14, respectively, were selected for use.

- Genetic Operator Probabilities: The probability that a selected individual will undergo crossover at any node, crossover at a function node, copying and mutation were varied over all combinations of {0.0, 0.2, 0.4},{0.0, 0.2, 0.4} and {0.1, 0.3, 0.5} respectively, with the mutation rate set so as to make the sum of probabilities 1.0. The performance

[2]the fittest of N randomly picked individuals is selected

was largely insensitive to these and values of 0.2, 0.15, 0.15, 0.5 were selected.

- Cellular GP: Cellular occupancy of {10,2,50} by {1,5,10} grids with localised selection gave no significant increase in performance.

After search and selection of these parameters the upper limit of performance was approximately 80% correct. An experiment was run to indicate whether this was a true upper limit: Parameters were configured to give low selectional pressure and encourage diversity (tournament-size-two selection, cellular occupancy, Large populations and higher mutation probability) and no limit was placed on the number of generations before stopping. Large cpu-time allocation was required because of the the non-aggressive nature of the search (population size 1500 was used). Configured in this way the GP is resistant to premature convergence and is closer to truly open-ended evolution, i.e it is more likely to find the 'solution' given enough processing time. Figure 7.1 shows how the performance continued to increase on the training set but the validation fitness reaches a limit. The continued improvement on the training set indicates that evolution is continuing, and in conjunction with the over-fitting to training data suggests that 80% is an a priori limit on the performance of a ball detector using the tree representation and training method detailed so far.

Problem Images

It was possible that the bad performance was in part due to 'anomalous' images in the training set. Such images, e.g. a background image incorrectly geometrically classified as a ball, may be few but could have disproportionately adverse effects on the training. To check for this the final population is run on the training set and the incorrect classifications for each image are accumulated: anomalous images should be revealed by being incorrectly

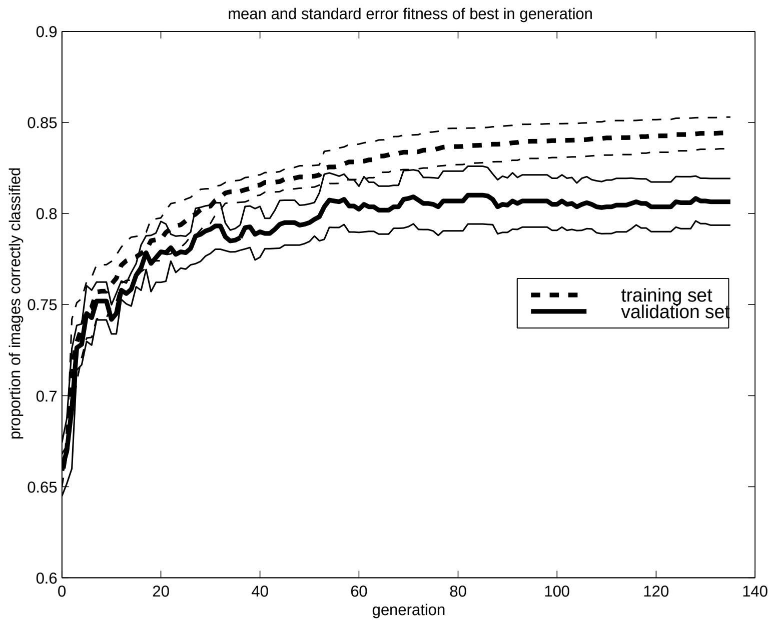


Figure 7.1: Fitness as a function of generation: Showing over-fitting to the training data and the upper limit of classification performance on 'unseen' data

classified by a high proportion of the population.

Inspection of such images shows the geometrical classification to be reliable. Most of the problem images correspond to: Situations where the view-angle of the ball is comparable to the view-angle of the goal's stripes; making identification of ball against goal very difficult (see figure 7.2a). Or situations where the goal is going off the edge of the image with only one stripe being visible; being essentially equivalent to a partially visible ball image. It is doubtful whether such images are classifiable, the author has trouble doing this visually. There are a subset of background images that are truly anomalous, see figure 7.2b. The strong peaks could be caused by the contrast enhancement effects of the iris. It was decided to remove these from the training set so not as to 'confuse' the GP with images that were similar to ball images but were not. This is bona fide since performance of the evolved sensors is measured against the validation set, which still contains such images.

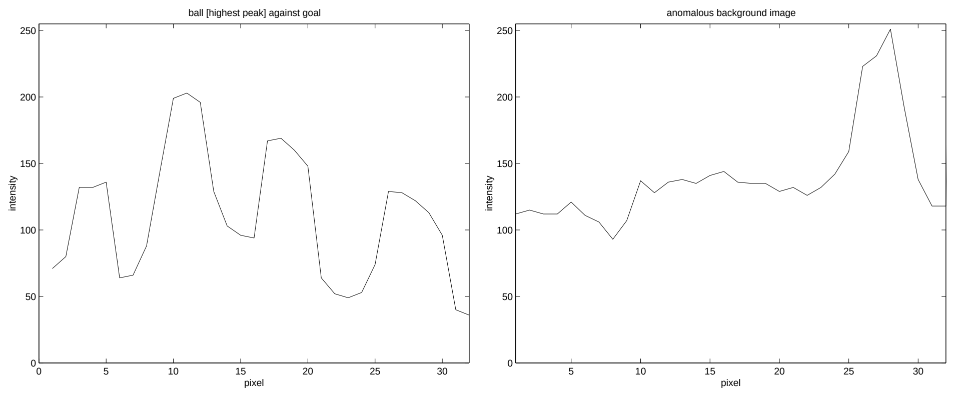


Figure 7.2: Problem Images: a) 'spot the ball'. b)A uniform-background image!

With the cleaned up training set the performance increased significantly, but only by a small amount: $\approx 2\%$ more correctly classified.

Edge Detection

The next essay at getting decent performance was to increase the power of the GP function set by adding an edge detector. The number and separation of edges in an image are strong indicators of the image class and give the GP a higher level representation to work with. A weighted second order derivative and first order derivative function were tried. See equations 7.1[3] and 7.2 respectively, where $I(x)$ is the intensity of pixel x. A ball-and-goal image processed using these functions is shown in 7.3.

[3]Used in [Smith 97] for Khepera vision processing, this responds to curves.

$$edge(x) = I(x) - (0.25I(x-1) + 0.5I(x) + 0.25I(x+1)) \tag{7.1}$$
$$edge(x) = I(x+1) - I(x) \tag{7.2}$$

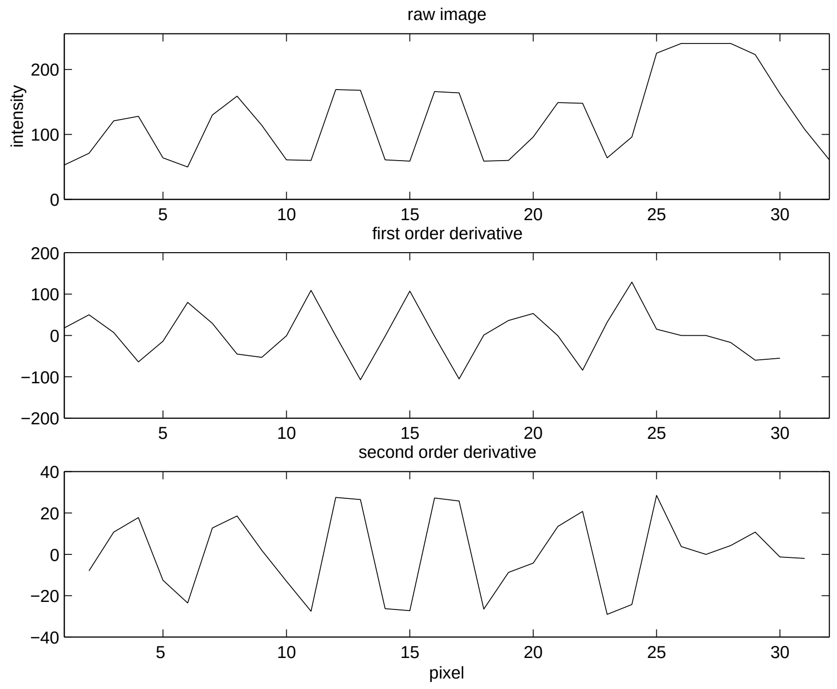


Figure 7.3: Edge Detection on an image of the goal and ball (far right)

Adding the first order edge detector to the function set gave significant improvement in performance; to 87% correct classification at convergence with population size 1000. Adding the second order edge detector to the function set gave no significant improvement in performance, perhaps because of the extra sensitivity to noise inherent in a second order derivative calculation. Performance with the function set containing the first order derivative edge detector was deemed good enough to work with; a ball detector evolved at this stage was characterised and used in the controller evolution.

7.1.2 Ball-position Sensor

Preliminary experiments evolving a ball-position sensor with a data set containing only just-ball images showed that the GP could essentially make a ternary classification of position, see figure 7.4. Comparable performance could probably be achieved by looking at the index of the MAX pixel. The detector was not developed further; although conducive to a higher performance controller it was deemed unessential.

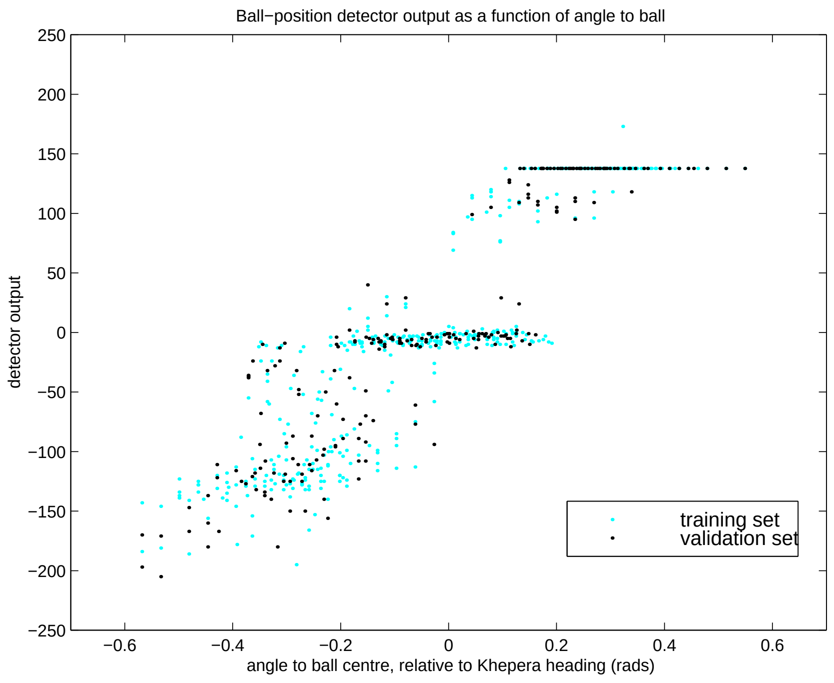


Figure 7.4: Ball-position Sensor: showing essentially 3-way classification

7.1.3 Goal Detector

Using the same approach as with the final ball detector a goal detector with 79% correct classification was evolved. The efficacy of the detector was worse than this percentage suggests since it tended to give false negatives, resulting in missing the goal 25% of the time. As a final attempt to get decent performance a more specialised goal detector was evolved:

It is noted above that images containing ball and goal are potentially harder to classify [4]. By removing these from the training and validation set the learning process can concentrate on the simpler, and most important task of distinguishing goal from ball and background. This will be detrimental to the performance on ball-and-goal images but this may not be so bad in terms of controller performance; a goal made 'invisible' by a ball in the field of view will appear as the Khepera's viewpoint changes. A training and test set were compiled that contained 50% just-goal, 30% just-ball, and 20% background images.

Using this approach performance increases to 82% correct classification, with 78% performance on just-goal images, at convergence with population size 1000. This is not ideal but it was decided to work with this in the controller evolution and see how the controllers could cope with such unreliability.

7.1.4 Why does GP Struggle?

Beyond the fact that it is a vision classification problem with the associated difficulties of high dimensional complex and noisy input, and that there are ambiguous images, can reasons for the relatively poor performance of the GP system be given?

One is the difficulty of acquiring global information given the GP tree representation: for example a tree version equivalent of the MAX terminal would require a tree of depth order log_2 number of pixels, with a highly regular structure; each node being an IFLTE function taking PIXEL($index_1$),

[4]30% of ball-and-goal images against 20% of just-goal images were misclassified by the aforementioned goal detector, although it should be noted that the training set contained more of the latter so better performance on these could be expected

performance with MAX and MIN	without MAX and MIN
0.969 ± 0.004	0.85 ± 0.01
0.92 ± 0.01	0.83 ± 0.04
0.96 ± 0.02	0.81 ± 0.01

Table 7.1: Mean $\pm$ standard error performance. Rows are over different GP parameter settings

$index_1$, PIXEL($index_2$), $index_2$ as arguments. The importance of such global properties to a successful ball detector, and the difficulty GP has in evolving sub-trees that approximate these was demonstrated by running the GP with and without the {MIN ,MAX} terminals. Table 7.1.4 shows the results for data sets containing just-ball and background images (simplified from the general case being evolved for, above) over various other parameters.

Another potential problem is the lack of re-usability of sub-trees. Image processing algorithms often involve repeated application of a procedure to groups of pixels in an image. There is no mechanism for this in the monolithic GP tree representation used here. Automatically defined functions [Koza 94] are such a method for co-evolving sub-routines within GP, their implementation was not possible given time resources.

7.2 Characterisation

The performance of the virtual sensors must be faithfully reproduced in simulation if the evolved controllers are to transfer to reality. Behaviours reliant on non-existent properties of the sensors are likely to fail completely when using the real sensors.

The method of characterisation is data intensive and in the ethos of the look-up table approach rather than mathematical modelling: The chosen detectors are run on those images collected but not used in the training or validation set — this will be referred to as the *test set*, it contains approximately 12000 images — with each classification being recorded as right or wrong. The data is then divided into 'buckets' with respect to the parameters recorded during data collection, such as angle-to-ball and distance-to-goal, and the performance of the detector over the data in each bucket calculated. Analysis of variance is then used to look for buckets over which the performance differs[5]. The performance over homogeneous buckets can be represented simply by the mean performance over such buckets. This gives the empirically derived probability that an image of the scenario described by the said range of parameters will be correctly classified. Parametric models were not fitted.

One-way factorial without repeated measures analysis of variance was performed on each of the parameters to look for factors that had significant (at $\rho = 0.01$) effects on virtual sensor performance. Using just one-way analysis when there is more than one effective parameter it is possible to miss significant parameters because of variance from other factors contributing to within-bucket variance: such variance can swamp between-bucket variance of the parameter under consideration, hiding the dependency. To try and counter this, one-way analysis over all parameters is done for each class separately; since image class was shown to have the largest effect on performance. Visualisation techniques were used in conjunction with the analysis of variance to identify such cases. At the end of the classification each class has a look-up table, ranging from zero to two dimensions, to be indexed by the value of the relevant parameters (those that were determined to have an effect on performance) during simulation.

Figure 7.2 shows the performance (proportion *mis*classified) of the ball

[5]Analysis of variance is a statistical technique used to test for equality of population means

detector as a function of image class. Figure 7.2 shows performance of the goal detector on a single image type as a function of two paramaters: the matrix plotted is the look-up table used in simulation.

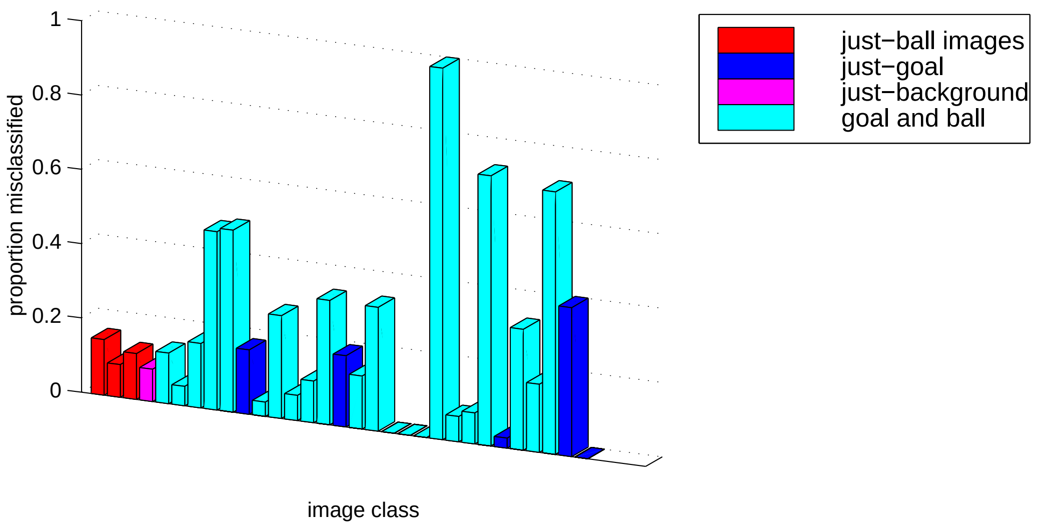


Figure 7.5: Misclassification rate as a function of image type for the ball detector used. The image classes consistently misclassified typically contain few (< 10) images.

7.3 Comments

Whatever the advantages of the virtual sensor approach, the work chronicled in this chapter supports the reservations in section 6.5 about shifting the problem of perception into the non-temporal object recognition domain: evolving sensors was a challenge and in some cases the final product was less than desirable. It remains to be seen whether or not controllers can evolve to cope with this unreliability...

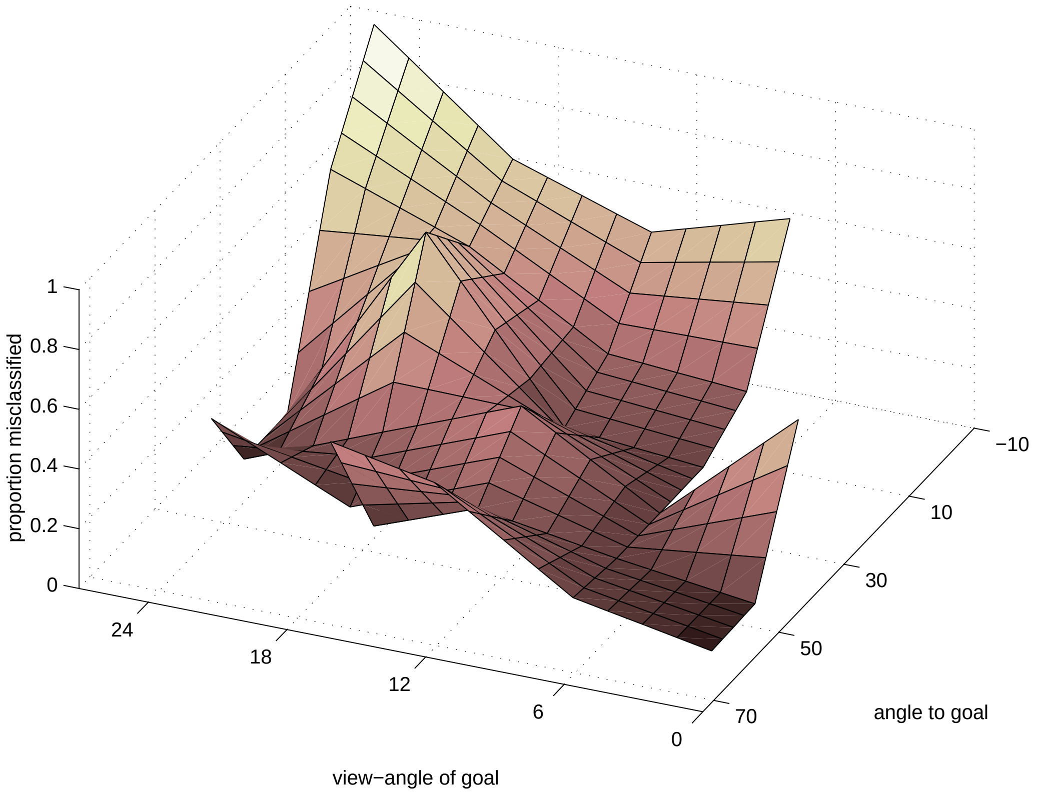


Figure 7.6: Misclassification rate for images of goal going off-image to the left as a function of angular width (view-angle) of goal and angle to goal. (Note that the graph is interpolated; one patch does not correspond to one 'bucket' or entry in the look-up table.)

Chapter 8

Controller Evolution

Virtual sensors have been synthesised and the sensory and dynamical aspects of the Khepera and environment characterised and simulated. Now the controller is to be evolved and successfully transferred to reality (or not) on the strength of these.

8.1 Evolution Preliminaries

Fitness functions are all scaled to be in the range $\{0..T\}$, where T is the maximum number of time-steps for a single trial, following the convention of increasing fitness with numerical value. Sensor values are normalised and consequently the constants [1] are in the range $\{0..1\}$. The best of run individual is defined as the fittest controller in the last generation.

The number of runs for each starting position is chosen as a function of the noise of the sensors in the terminal set — assuming this to be the principal source of statistical variation in fitness. Choice and number of starting positions is guided by the same issues of *representative-ness* as in chapter 6, but with the constraint of much smaller training sets; necessary

[1]Randomly generated in the initial population

with the increased computation of evaluating performance on a single training instance, or in this case a run. Subset selection was found to inject a lot of noise into the evolutionary process — effectively producing varying fitness functions, according to the particular sub-set selected, across generations. The degree to which this is detrimental to the selectional process is not known, however applications of dynamic subset selection in [Gathercole 98] use sub-sets of order 10^2 training instances, where statistical fluctuations will be at a minimum. In any case, the speed of the simulator meant that sub-set selection was obsolete as a means of reducing run-time. Preliminary runs, taking the mean fitness over multiple trials produced controllers that worked well over all the starting positions, and this is method of fitness combination used.

8.2 The Decomposition

The hierarchical structure of the controller, whose modules are to be evolved, is shown in figure 8.1. It is similar to that used in [Lee *et al.* 97]. The complete task of scoring goals is decomposed into homing in on the ball and pushing the ball towards the goal. The root node will switch between these two behaviours, requiring the ball detector and IR sensors to determine when the ball is contiguous and so start pushing-to-goal behaviour. Homing behaviour comprises moving within close vicinity of the ball as quickly as possible and the sensors to be used for this are the ball detector and the IR s (for any collision avoidance that might be required). Pushing-to-goal behaviour is decomposed in to ball-pushing and ball-spiraling. The intention being that the arbitrator will switch between pushing the ball when it is aligned with the goal along the Khpera's line of sight, and search for such an alignment otherwise. Hence it requires the goal detector. In this context ball-pushing comprises fast pushing of the ball in a straight line; IR s are

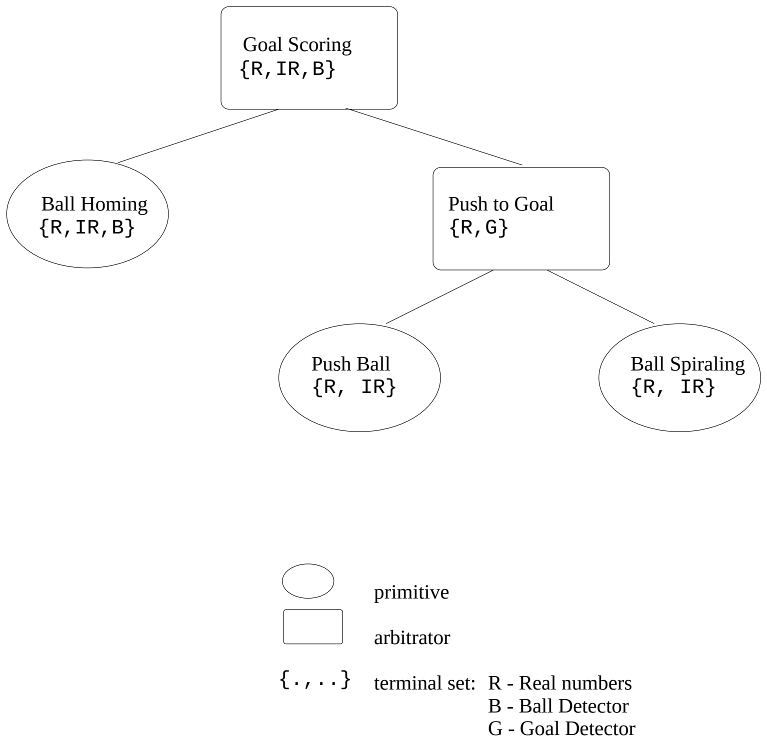


Figure 8.1: The hierarchical decomposition of the task showing the arbitrators, primitives and relevant terminal sets.

used for the proximal sensing this demands. Ball-spiraling is a behaviour that will bring the Khepera, ball, and goal into alignment quickly, allowing for detection of this by the arbitrator. IR sensors are used, for the same demands.

The choice of decomposition arises from expectations of tractability, perception of 'natural' division of behaviours, and ease of fitness function design. It is a highly unscientific design process!

8.3 Ball Homing

The fitness function expressing the requirement to get within close vicinity of the ball as quickly as possible is shown in equation 8.1.

```
if      ( centre to centre Khepera-ball separation $< \tau$ )
then    fitness $=$ time steps remaining;
        terminate trial;                                        (8.1)
else    fitness $= 0$
```

Having a single term it has no parameters to tweak. Unlike the fitness functions in sections 8.4 and 8.5, it contains no sensor readings — it is said to be an *external fitness function* — because the inherent noisiness of the ball detector was deemed a bad parameter to base the fitness on. A τ of 66mm is used to get the Khepera in the vicinity of being able to see over the ball to spot the goal, whilst still being able to sense the ball with the IRs.

Six Khepera starting states were chosen for the evolution, tending to be distanced from the ball rather than random. Three runs of each starting state were made because of the noisy nature of the ball detector being used. Each run lasted a maximum of 200 time steps. GP parameters are shown in table 8.1 and were chosen to be similar to those in [Lee *et al.* 97]. They did not have to be tweaked.

Eight evolutionary runs were made. Figure 8.2 shows the progression of fitness during these, the asymptotic best of run fitness corresponds to reaching the ball in approximately five seconds.

All of the best of run individuals could successfully home in on the ball in simulation and displayed qualitatively identical behaviour: Moving towards the ball at full speed when it is seen and circling-search for the ball when it is not, see figure 8.3a. Occasionally the Khepera is deceived by the goal, moving

generations	50
population size	30
Steady State	ON
max depth for new trees	5
max depth after crossover	8
max mutation depth	3
selection	Tournament, size 2
crossover probability copied probability	0.4 0.15
mutation probability	0.55
parsimony factor	0.00000

Table 8.1: GP parameters for Ball-homing evolution

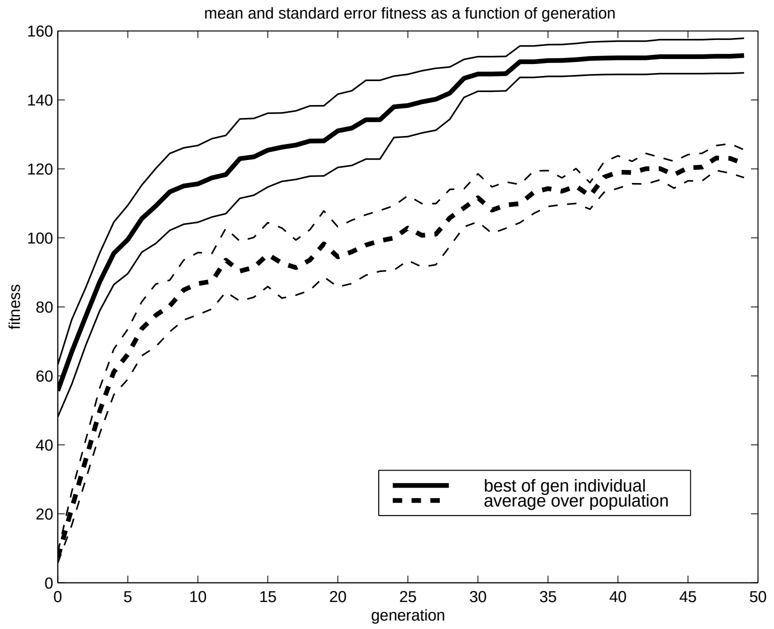


Figure 8.2: Fitness as a function of generation for ball homing.

a substantial distance towards it, or loses the ball whilst heading towards it — in which case it circles until it picks it up again: see figure 8.3b. Both of these are a result of the unreliability of the ball detector.

It was noted that the circling-search radius corresponded to just less than the closest Khepera-wall separation in the training set. Such a strategy had avoided the need for collision avoidance but is an example of over-fitting — when the Khepera was placed nearer to a wall it circled into the wall and exhibited dysfunctional behaviour (going off on a wall following tour!), having not evolved behaviour for such a situation. The original behaviour was perfectly valid (perhaps optimal) given the training set but illustrates how the fitness function *and* the training set interact to define the evolved behaviour. The run was repeated with two extra starting positions $5cm$ from a wall with the Khepera facing a wall and corner respectively. Figures 8.4a,b show that the subsequently evolved controllers included collision avoidance components.

It is arguable that ball homing behaviour is trivial given the high level sensor used — just moving forward when the ball is visible, turning to search otherwise, would work. The behaviour evolved even appears sub-optimal at first glance — wouldn't spinning on the spot to search for the ball be faster? In fact, the controllers are found to be behaviourally more advanced than at first glance and the strategy evolved performs better than the obvious approach: Inspection of the sensor and motor values shows that when the ball is seen the controller switches between moving forward at full speed and intermittently slowing one of the motors so as to turn in the opposite direction to the circling-search. This behaviour compensates for the turning behaviour, required when the ball is not in the field of view, activating when the ball momentarily disappears, and produces approximately straight line movement to a ball once it has been 'locked on' to even though it is not always detected. Evolution has found a way to cope with the unreliability of the

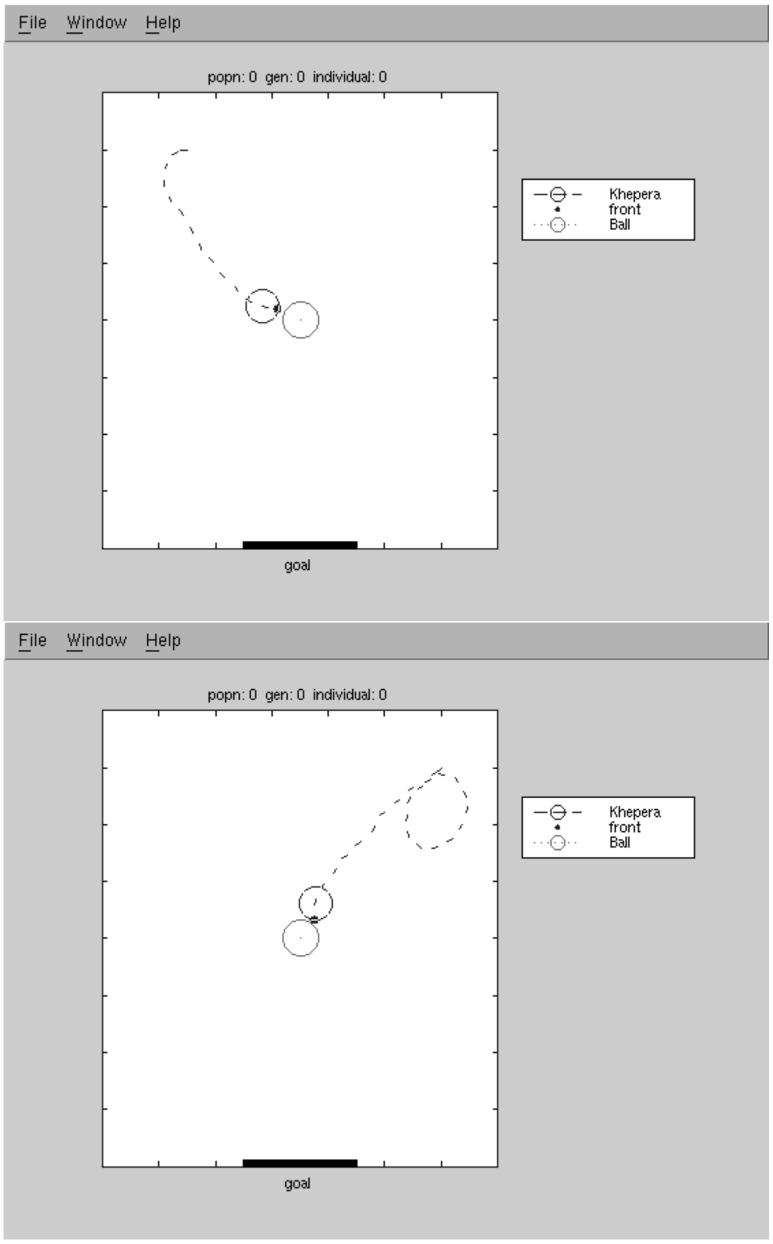


Figure 8.3: Aspects of a single Ball-homing controller's behaviour: a) circle $\rightarrow$ locate $\rightarrow$ move to ball. b) ball 'disappears' $\rightarrow$ circling to relocate.

virtual sensor. If one considers the obvious approach again then it becomes apparent that whilst moving towards the ball the Khepera will continually

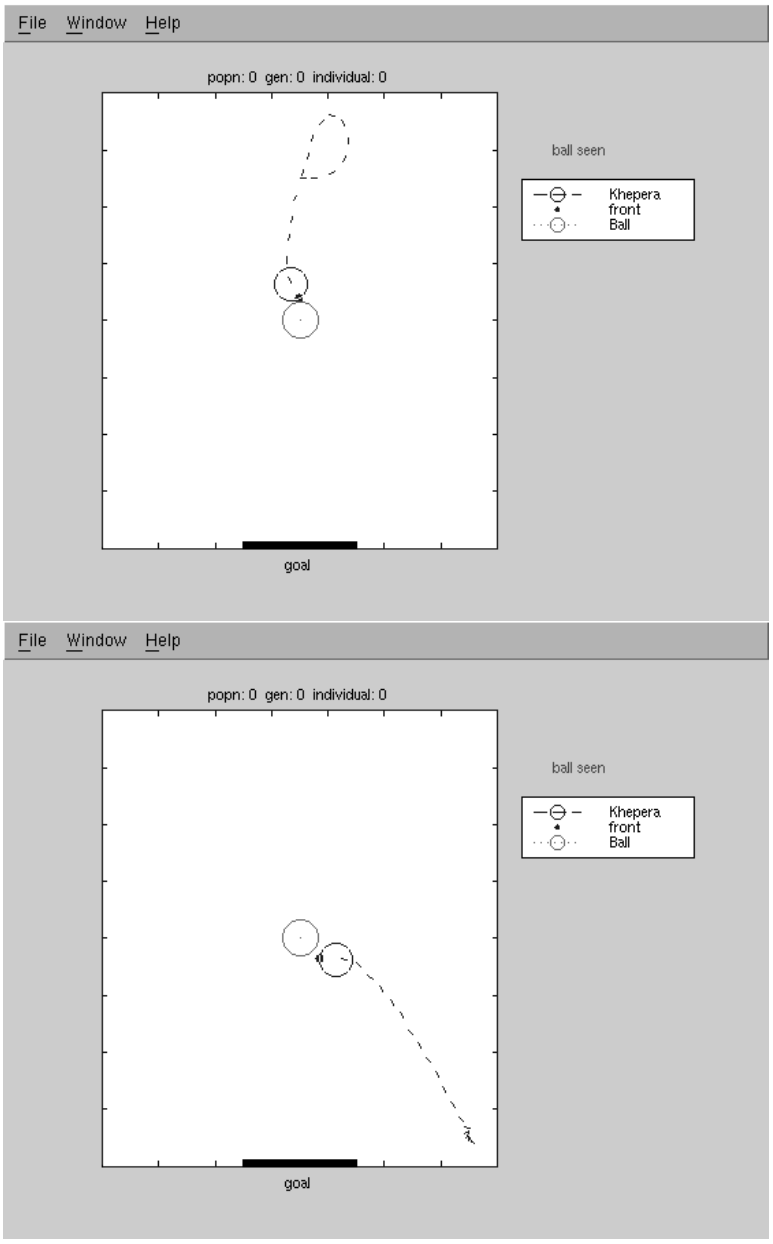


Figure 8.4: A Ball Homer with collision avoidance skills: a) during circling search. b) escaping from a corner.

be stopping as the ball intermittently 'disappears'.

The way this behaviour is produced, by exploiting the noise of the IR sensors, is interesting too: Analysis of one of the controller trees shows the switching, in this particular case, to be dependent on whether IR detector six is reading larger or equal to IR detector one. Whilst the Khepera is moving towards a ball it is generally in the open and its IRs will be returning noise about the zero level. The IR6>=IR1 is thus equivalent to a uniform $\{1, 0\}$ random number generator, and the controller is using this to overcome the deterministic nature of the tree evaluation and provide stochastic switching between behaviours. A reactive controller that produces different behaviours given the same perceptual cues has been evolved!

8.3.1 Reality Transfer

A single controller was transferred to reality and tested over the eight starting positions used in training, three times each. The behaviour was qualitatively the same (the noise compensation behaviour was observed). The behaviour was found to be quantitatively the same too: The fitness of each run was recorded and the mean over all trials did not differ [2] from identical trials in simulation:

fitness in reality	in simulation
156±19	166±7

The success of the reality transfer validates the modelling of the ball detector.

8.4 Ball Pushing

The fitness function expresses ball pushing as moving forward fast, in a straight line with the ball to the front of the Khepera; having a component for each [3]. See equation 8.2

[2]a t test gave $\rho_{means\ different} < 0.7$ (d.o.f. = 12)

[3]as used by [Lee *et al.* 97]

$$
\begin{aligned}
fitness &= \sum_{t=1}^{T} \alpha\, a(t) + \beta\, b(t) + \gamma\, (1 - c(t)) \qquad (8.2)\\
& a(t) \text{ is the average of the front two normalised IR readings}\\
& b(t) \text{ is the normalised average wheel speed: } \frac{v_l + v_r}{2\, v_{max}}\\
& c(t) \text{ is the normalised difference in wheel speeds: } \frac{v_l - v_r}{2\, v_{max}}
\end{aligned}
$$

The weightings used were $\alpha = 0.6$, $\beta = 0.2$, $\gamma = 0.2$, as in *[ibid]*. Initial runs indicated the need for an extension to the function: terminating the run upon collision with a wall was preventing further fitness accumulation, effectively penalising straight line pushing and hence was in conflict with the latter two components in the fitness function. Evolved controllers would push the ball in a circle of diameter approximately equal to arena size. The simple solution was to add a perfect fitness score for each time step left if the ball was pushed into contact with the wall. In fact it was found that the rest of the fitness function is now unnecessary[4] — such a monolithic fitness function does not have the disadvantage of the need for parameter tuning.

Six Khepera starting states were chosen for the evolution, each within τ[5] of the ball centre. Only one run of each starting point was made because of the relatively low noise in the IR sensors. A trial lasted for up to 200 time steps. The GP parameters were as in section 8.3. A ball-as-box model was used, given the lack of ball dynamics characterisation, with a view to augmenting this if it was detrimental to reality transfer. Seven runs were made, the graph of the evolutionary process is shown in figure 8.5.

Six of the seven best of run controllers were high performance ball pushers [6]; turning to meet the ball then pushing at maximum forward speed for

[4]Although the results are for the original fitness function

[5]the distance used in section 8.3, for compatibility

[6]the other pushing backwards, more slowly

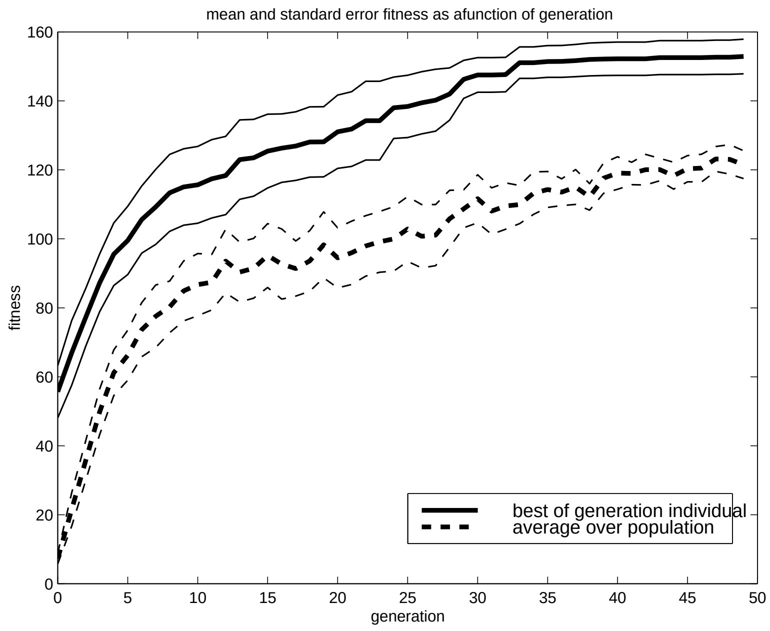


Figure 8.5: Fitness as a function of generation for ball homing.

the most part with occasional switching of one motor to the next fastest speed to keep the ball centred. Figure 8.6 shows a typical push.

8.4.1 Reality Transfer

The ball pushing transferred qualitatively to reality, but with a performance decrease; perhaps inevitable given the approximations of the ball model. As in simulation the Khepera makes a tight turn to meet the ball then commences pushing. The path taken by the ball veers more than in simulation, and the Khepera spends more time making heading corrections.

An attempt was made at improving the reality transfer by evolving using heading and pushing-direction noise (section 5.1, above) in the ball model with magnitudes of 15.0° and 5.0° respectively. The evolved controllers moved more slowly and exhibited spinning behaviour when the ball deviated sharply, see figure 8.7. Transferred to the Khepera the performance was

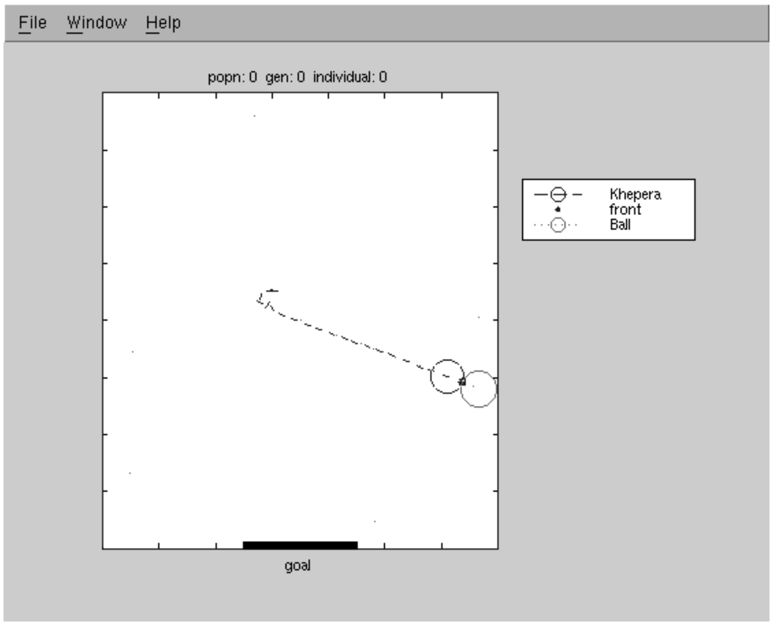


Figure 8.6: Typical ball pushing. Ball-as-box model.

noticeably worse than with the ball-as-box model controllers. [Nolfi *et al.* 94] mention the importance of modelling noise at the real world levels — this is a case in point; the values chosen were based on rough observation and probably give a model as inaccurate as the original one.

The near-success of the reality transfer shows the dynamical modelling to be adequate but incomplete[7].

8.5 Ball Spiraling

The Khepera line of sight being at a tangent to the direction of motion requires the Khepera to periodically face the ball as it circles, e.g. helical orbiting of the ball, it if it is to be able to align the ball between itself and the goal. The fitness function expressing this behaviour is shown in equation 8.3:

[7]which was always appreciated

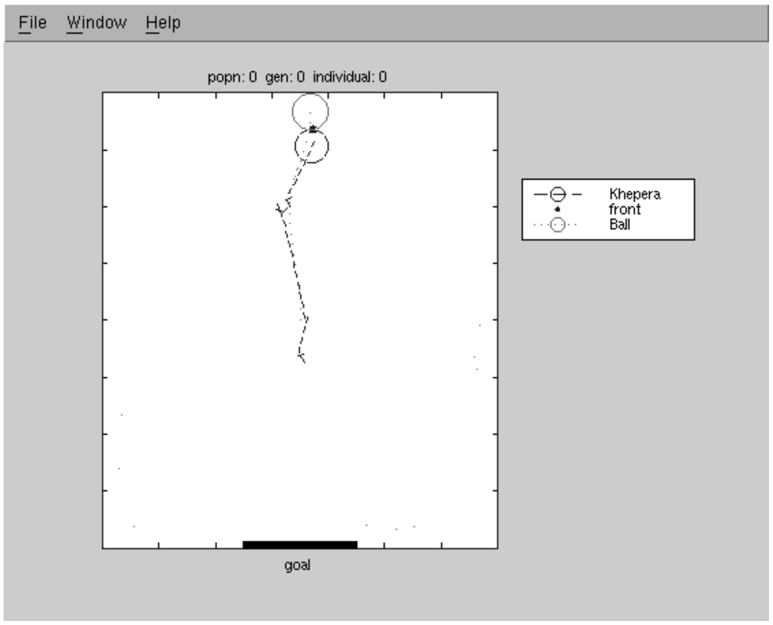


Figure 8.7: Typical ball pushing — Unvalidated ball model. Closer spacing of tracer dashes than in figure 8.6 shows slower movement.

the first term rewards facing the ball from different directions, and the second term rewards doing this quickly.

$$fitness = 0.5\ T\ \frac{s_v}{S} + 0.5\ (T - t_{terminate}) \tag{8.3}$$

S is the number of sectors the direction [$0°$ to $360°$] from ball centre is divided in to.

s_v is the number of sectors visited — one in which the front IR activation was larger than the *limit threshold.*

$t_{terminate}$ is the time at which the trial is terminated: when $s_v = S$ or $t = T$

The starting state, number-of-run parameters, and maximum time steps were as for the ball pushing. Initial runs showed that satisfaction of the fitness function was not unique to the desired behaviour; pushing the box off-centre, causing it to follow a circular path, resulted in procession of the Khepera around the ball with the IR s above the *limit threshold*. An environmental modification was made to remove this as a solution — by making the kinematics of the ball random and sensitive, ball pushing was no longer feasible. Before the desired behaviour was consistently evolved the power of the GP had to be turned up. Eight runs of population size 40 on a 4×1 cellular occupancy grid, run for 50 generations were made. Figure 8.8 describes the run. The average best of run fitness corresponds to approximately visiting all the sectors surrounding the ball in 40 seconds.

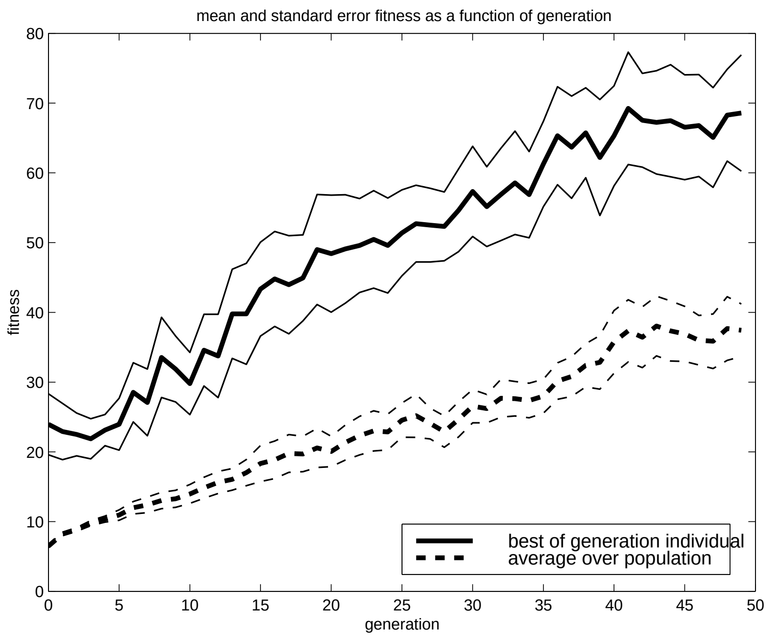


Figure 8.8: Fitness as a function of generation for ball homing.

Six of the eight runs evolved successful controllers, achieving the task using qualitatively different behaviours, three of which are in figure 8.9. During each of these orbits the Khepera is frequently 'looking up' to allow for check-ing of alignment with the ball and goal.

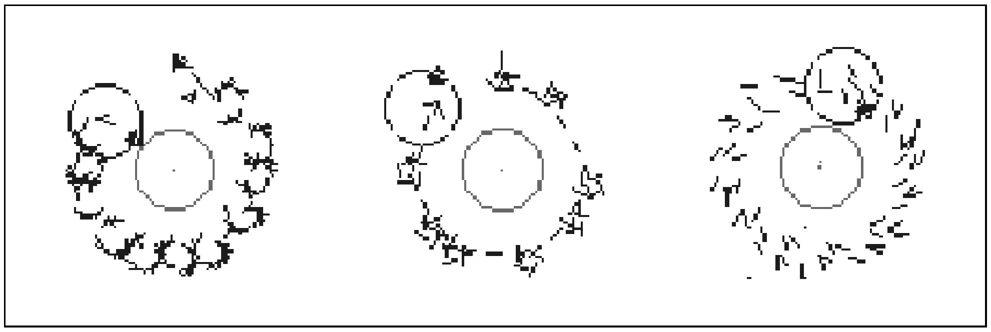


Figure 8.9: Behaviours to allow search for alignment of the ball and goal in the direction of the Khepera: a) helical orbiting, b) circling of the ball with periodical spinning on the spot, c) saw-tooth orbit.

8.5.1 Reality Transfer

The controller from figure 8.9b was chosen for reality transfer; this was successful. Placed within τ of the ball it would settle into the same circling and spinning behaviour. A quantitative comparison was made at the behavioural level by recording the box revolution rate and the number of on-the-spot spins per revolution. This was done for a single trial of ten revolutions in reality and multiple trials in simulation. The results are shown in table 8.5.1, a Z-test confirms the quantities are significantly different. There has been a degradation of performance upon reality transfer.

The solely qualitative transfer of the behaviour has shown the sensor and Khepera-dynamics modelling to be adequate, but imperfect.

8.6 Pushing to Goal

This module arbitrates between the already evolved ball-pushing and ball-spiraling primitives. The fitness function expressing fast pushing of the ball

	in simulation	in reality
spins per ball revolution	19	10 ± 1
time steps per ball revolution	326	230 ± 40

Table 8.2: Degradation of Ball spiraling upon reality transfer

to the goal is shown in equation 8.4. It provides an incremental path for the evolution: encouraging first closer and closer movement of the ball to the goal, then faster scoring of goals once this behaviour is acquired.

$$fitness \;=\; 0.5\ (T - t_{termination}) + 0.5\ T\ (1 - D_{t\_termination}) \tag{8.4}$$

where $t_{termination}$ is T, or the time step at which a goal is scored, should this happen. D is the scaled ($\times 1/500$) centre to centre goal-Khepera separation.

The ball is always placed on the centre spot of the pitch, requiring a ‘dribble’ of 40cm to score. Khepera starting positions and runs per starting position were as in section 8.5. The maximum-time-steps, T, was set to 500 as an approximation of the time a reasonable attempt on goal might take given the spiraling velocity, etc. Forty generations were run with tournament size 4, cellular occupancy and other-parameters as in section 8.4 [8]. The evolved, intended, behaviour of spiraling for alignment, then pushing forward is shown in figure 8.10. It can be seen that two helical orbits are made during the push to goal, as a result of goal detector unreliability causing the goal to ‘disappear’. This is an uncharacteristically efficient push to goal; most involved 5 or 6 helical orbits of the ball and correspondingly approximately

[8] increasing selective pressure (tournament size 4) to give more aggressive search whilst maintaining diversity (cellular occupancy)

three minutes to score.

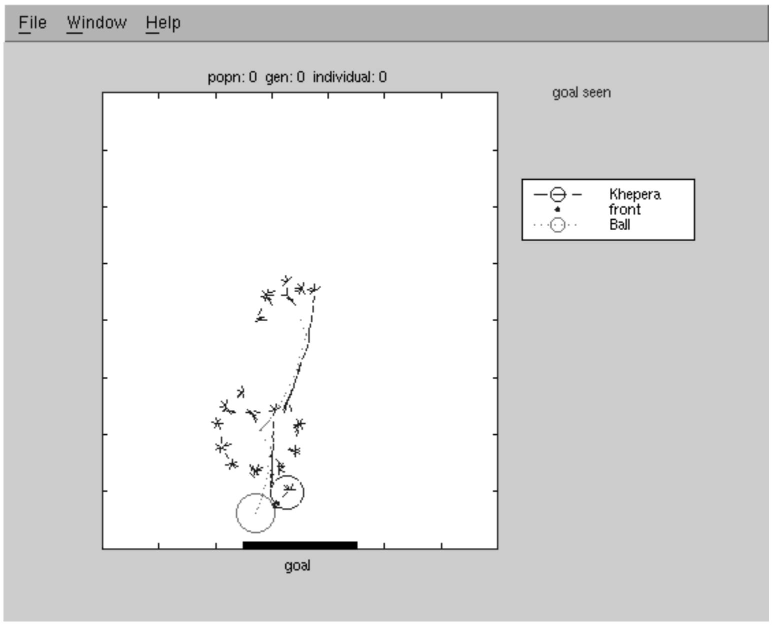


Figure 8.10: Hierarchical goal scoring behaviour — spiraling to align ball, goal and Khepera, then pushing to goal.

8.6.1 Reality Transfer

The evolved behaviour does *not* work on the the real robot because the goal is never detected whilst spiraling. This anomaly can be traced to the rotating motion of the Khepera whilst ball-spiraling and is presumably caused by motion blurring of the goal. Such an effect was never considered[9] during the vision modelling stage and the evolved controller has relied upon an approximation (or rather oversight) that is not valid. This failure highlights the implicit assumptions in, and approximate nature of *any* model. Such 'quirks' of real world systems are bad news for the simulation approach to evolutionary robotics.

[9]quite understandably!

8.7 A Monolithic Push-to-goal Controller

Having traced the cause of reality-transfer failure, it would be possible to re-evolve ball spiraling with an additional constraint on the rotational velocity of the ball-spiraling Khepera. However, the behaviour in simulation showed the approach to be rather inefficient, suggesting that the decomposition of pushing to goal, at least in this manner, is not a good idea; perhaps having no 'natural' sub-modularity and best left as a monolithic behaviour.

An attempt at evolving a pushing-to-goal primitive, to address this hypothesis, was made. The terminal set comprised real constants, IR and goal detector sensors. The fitness function was as in section 8.6 and the power of the GP was turned up to cope with the expected decrease in tractability of the problem: population size 400, 100 generations, cellular occupancy and maximum tree depth 10 were used. The evolution was successful and the behaviour of the evolved controller is shown in figure 8.11. Its strategy is to push the ball in a circle to the right when the goal is not visible, then when the goal comes into view spin away and back in to contact with the ball so that the goal is out of sight and the ball is pushed in an arc to the left. The motion of the Khepera brings the goal into view again and the process is repeated: the aggregate effect is movement of the ball in a straight line towards the goal. It can consistently score from all starting positions.

Its scoring rate is approximately five times that of the hierarchical controller in section 8.6, which could be a result of the more powerful evolutionary run used. The setup in section 8.6 was run using the same GP parameters to show that the better performance was not due to this. Figure 8.12 describes averages over runs of both approaches. It can be seen that the hierarchical controller fitness has already converged in the initial population but to a lower fitness than the monolithic controller; this represents the upper limit of performance using the constraints conferred by the designer's decomposi-

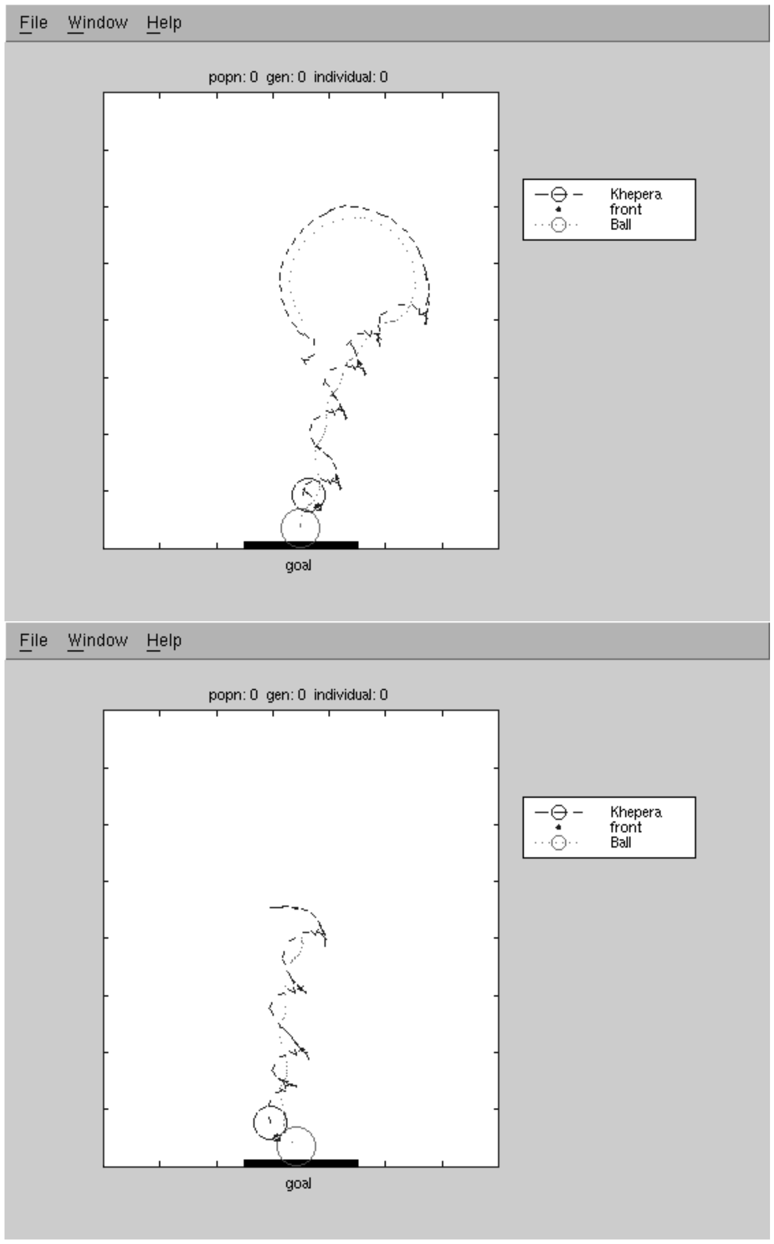


Figure 8.11: Monolithic goal-scoring controller behaviour — circling with ball to locate goal then pushing to goal: a) starting facing way from the goal, b) starting facing perpendicular to the goal.

tion. It should also be noted that the fittest hierarchical controllers did not combine the ball-spiraling and pushing in the way intended, spiraling and pushing quasi-continuously to goal. Comparison of the number of controller tree evaluations[10] before convergence, considering the evolution required for the arbitrator's primitives also, gives approximately 10^6 against 10^5 in favour of the hierarchical approach.

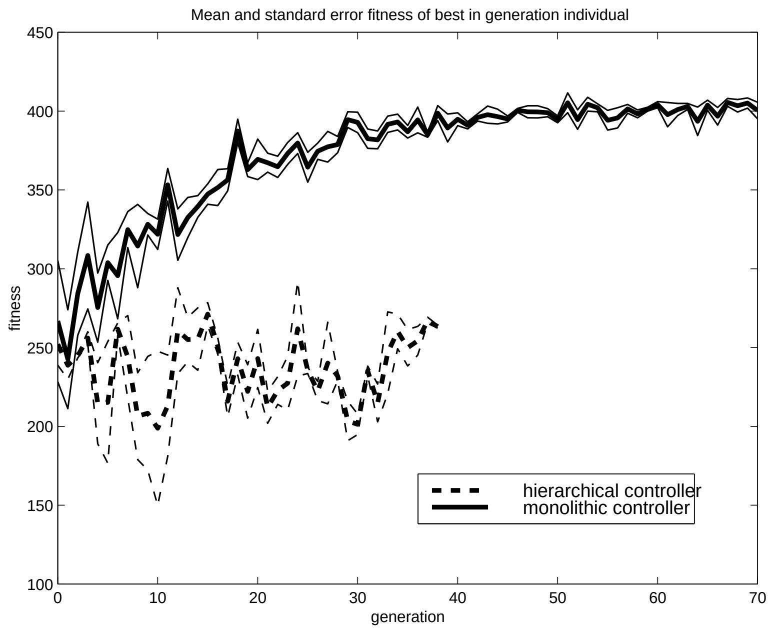


Figure 8.12: Fitness as a function of generation for pushing-to-goal: Monolithic and hierarchical controller.

Reality transfer of the monolithic controller is successful in part. The Khepera can score when up to approximately a quadrant of the circular search path has to be made to achieve alignment of the Khepera, goal and ball; in which case it proceeds to push the ball to the goal in the manner seen in simulation. This means that the Khepera can start facing up to $90°$ from the goal ($40cm$ away) and be able to score. Full circling happens

[10]The product of population size,generations and time steps

occasionally but the diameter of orbit is usually larger than the size of the stadium. The failure of transfer of the circling behaviour can be attributed to the unmodelled complexity of the ball, compared with box, motion; the reliability of the ball dynamics in simulation allow behaviour that is not possible with the less reliable, and systematically different, ball dynamics in the real world.

An arbitrator between ball-homing and pushing-to-goal behaviours, giving the complete goal-scoring behaviour, was not evolved because of time limitations.

8.8 Discussion

Successful, consistent, evolution of all controllers in simulation is shown. A common feature of the (hierarchical controllers) synthesis is the low populational requirements of the GP runs necessary for evolution: the population-size, generations product is typically order(s) of magnitude smaller than other work in the field, evolving comparative behaviours for Kheperas. The GP populational requirements agree with [Lee *et al.* 97], the basis of this work, and strongly suggest that Lee's controller architecture is particularly efficient and suitable for genetic programming. The inclusion of a high-bandwidth vision sense has *not* increased the populational requirements necessary for *controller* evolution. The controllers have coped with the unreliable virtual sensors evolved, in some cases exhibiting novel behaviours and methods to do this; the quasi-nondeterministic reactive controller in section 8.3 in particular.

The crux of any approach to evolutionary robotics is its ability to 'cross the reality gap'. The approaches used can be seen to be a success in this respect, with qualitative and occasional quantitative transfer for the most part. The results show that with careful capture and representation of actual sense and action it is *likely* that evolved controllers will transfer to reality. Conversely, the partial transfer of behaviours involving motion of the ball show that without sufficient modelling reality transfer should not be expected. The quantitative reality transfer of ball-homing shows that it is possible to sufficiently accurately predict virtual sensor output, through empirical characterisation on a set of test images, for purposes of simulation. The complete failure of goal-pushing upon reality transfer, due to motion-blur, highlights a caveat to simulatory evolutionary robotics: that there will always be unmodelled and unknown aspects of the real robot and environment, and that these can significantly effect performance.

The chronicle of getting the behaviours to evolve should make apparent that fitness function design is not just about the fitness function! In practice achieving the desired behaviour is a balance between environmental properties, training set, and fitness function, with all of these interacting to define the true 'fitness landscape' and 'solution' behaviour: Section 8.4 shows how an environmental property (of contact with the walls giving termination) can reduce a necessarily componential fitness function to a monolithic one. Section 8.3 shows the importance of a representative training set to evolving the general, rather than a specific[11] sub-set of, desired behaviour. Section 8.5 shows how artificial properties of the environment, in this case the 'impossiball' modelling of the ball, can be introduced to shape the behaviours evolved.

Decomposition has been shown to have advantages in terms of increasing tractability: the task of goal-scoring has been partially solved requiring evolutionary resources linear to to the number of modules. Through decomposition individual evolutionary runs have been no more demanding than that of simple behaviours such as collision avoidance and light following, seen in early evolutionary robotics work. In these terms decomposition seems a candidate for overcoming the barrier of tractability to solving real world, rather than

[11]and often optimised to the exact situation evolved for

‘toy’, problems. However, an experiment with a monolithic controller shows that decomposition may impose sub-optimal strategies upon the controller. For the task of pushing-to-goal a monolithic controller was far superior to a hierarchical one.

The decomposition chosen was not a good one, highlighting the difficulties of decompositional design, with the fittest hierarchical controllers in section 8.7 combining the behaviour primitives in an unintended way; since the behaviour primitives were as they were because of this intended use the decomposition has failed at the level of design. It is noted that the searching, circling behaviour used by the monolithic push-to-goal controller was evolved in initial attempts at ball spiraling but was rejected as not being of the desired behaviour. The fitness function was further constrained to remove such a strategy as a solution. This shows that the initial implementation, specifying what should be done — looking towards the ball from different directions, rather than trying to constrain how it should be done — ball spiraling, was the way to proceed. The project has only begun to touch upon the issue of decompositional design, factors effecting choice were listed in section 8.2 but much work would need to be done to establish if, when, and how decomposition should proceed. Work on this project has led the author to the following two, rather vague, guidelines:

- Check for tractability of the task; if it's evolvable then fine. If not, or fitness function design is a problem then recursively sub-divide the problem according to these two heuristics.

- In isolation the components of decomposition should be behaviours that a monolithic controller should, or ideally must, do. Defining how the behaviours should be produced, in the example of box-spiraling for instance, may be indicative of an inefficient or over-constrained decomposition.

An example of a decomposition meeting the second heuristic is goal-scoring being divided into ball-homing and pushing-to-goal: to score the robot *must* move to the goal *and* push the ball to goal[12]. Here the designer is imparting true domain knowledge to the evolution instead of a, possibly flawed, intuition — as in the case of assuming ball-spiraling and ball-pushing to be a good/best strategy for scoring.

[12]Unfortunately lack of time prevented evaluation of the efficacy of this decomposition

Chapter 9

Conclusion

Successful simulation, evolution and transfer to reality of a goal-scoring Khepera using visual perception has been shown. This was done through:

- Implementation of vision at an object level, removing the need for pixel-level modelling in simulation, and allowing insertion of domain knowledge in the form of useful perceptual constructs into the evolutionary algorithm.
- Careful representation of the virtual sensor, IRs and Khepera dynamics through sampling of real world responses and recalling these in simulation, recreating a 'virtual reality'.
- Using a monolithic controller, which was seen to perform considerably better than a hierarchical one.

The results are relevant to issues of combining engineering design and evolution. Experiments on decomposition showed that there are potential gains in tractability to be had at the risk of restricting the search space to sub-optimal regions of the fitness landscape; some guidelines for when and how to decompose behaviours are given. The virtual sensor approach showed that it is possible for designers to determine useful perceptual constructs. Work on simulation and reality transfer showed that with thoughtful modelling it is *likely* that the reality gap will be crossed: a failure to cross the reality gap highlighted the inescapable possibility that a significant quirk of reality may have been overlooked. The novel technique of *virtual sensors* was developed and was seen to be advantageous in most aspects assuming one has these virtual sensors, further discussion is given below. The independent evolution of these proved time-consuming and it is unclear whether or not the problem of vision was moved in to a more difficult domain; given that a comparison was not possible with limitations of time. The work is relevant in that behaviours were evolved at the current limit of behavioural complexity in evolutionary robotics. The techniques used show promise in terms of scalability, partially validated in this respect by working on a complex task. Further work on task decomposition and virtual sensors is needed to assess their true worth.

Genetic programming of controllers at the behavioural level through the fitness function, environment and training set was interesting to work with. As evolutionary run-time approaches standard compiler times this starts to resemble a behavioural-level language. The desired results are not always achieved first time but the required parameter adjustments are usually obvious from observation of the deviant behaviour.

9.1 Virtual Sensors

Once the virtual sensors had been evolved they were a success — with evolution of controllers using vision sensors requiring the same populational resources as for the lower bandwidth IR sensors. The ease of evolution and fairly optimal behaviour of evolved controllers shows the evolution could cope with the perceptual constructs imposed by the designer and discredits the arguments for *active perception* in such a problem domain; where it is seemingly fairly obvious what the robot should sense. Their use greatly simplified the modelling of vision and allowed the look-up table approach to be used, offering considerable speed-up over the ray tracing methods required in mathematical modelling.

If one considers the computation required to evolve the virtual sensors then the approach is probably not such a success in terms of increasing tractability. However the lack of comparison with a pixel-level vision simulation prevents any conclusions being drawn; the problem of visual perception for this task may just be very hard. The application of the approach to more complex visual environments/tasks is reliant on the synthesis of sensors for such environments/tasks. It is tempting to assume that by applying more sophisticated techniques from the vast body of machine vision or supervised learning techniques this will always be possible. This may be presumptuous but there is no evidence to suggest such methods are less powerful than the alternative approach of evolving *active perception*.

9.2 Further Work

The project has touched on a lot of issues and most of these remain open issues. Task decomposition and its generic technique of hybrid approaches demand attention and systematic work, e.g. [Perkins & Hayes 97], is only just starting in this area. A more systematic study, ideally comparative, of virtual sensors would be welcome.

The techniques used could be extended to more complicated tasks and environments, removing the sight screens would be an initial step in this direction. Generalisation to multiple or varied environments is another.

The promising controller architecture of [Lee *et al.* 97] could be extended beyond being solely reactive. Recurrency through side-effect memory read-and-write functions was considered, with time constraints preventing implementation.

The look-up table approach, with its faithful reproduction of *the* environment, has implications for a new simulator and robot design methodology. Coupled with automatic data collection techniques, robots could be evolved in situ, capturing their environment then evolving controllers specifically for this. Such highly specific controllers are likely to be easier to evolve than for the generic environment and can be highly optimised to the environment. This may be some time off!

Bibliography

[Angeline 94] Peter John Angeline. Genetic programming and emergent intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994.

[Boyle & Thomas 88] R. Boyle and R. Thomas. *Computer Vision: A first Course*, chapter Perception. Blackwell Scientific Publications, 1988.

[Brooks 85] Rodney A. Brooks. A robust layered control system for a mobile robot. Lab memo 864, MIT AI Lab, September 1985.

[Brooks 90] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[Brooks 91a] R. A. Brooks. Intelligence without reason. A.I. Memo No. 1293, April 1991.

[Brooks 91b] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Brooks 92] Rodney A. Brooks. Artificial life and real robots. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of* *the First European Conference on Artificial Life*, pages 3–10. MIT Press, 1992.

[Bryant 92] Randal E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. Technical Report CS-92-160, Carnegie Mellon University, School of Computer Science, July 1992.

[Clark 97] A Clark. *Being There.* The MIT Press, 1997.

[Cliff *et al.* 92] D. Cliff, P. Husbands, and I. Harvey. Evolving visually guided robots. Technical Report Cognitive Science Research Paper CSRP220, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, England, UK, 1992.

[Cliff *et al.* 93] Dave Cliff, Inman Harvey, and Phil Husbands. Explorations in evolutionary robotics. *Adaptive Behaviour*, 2:73–110, 1993.

[Floreano & Mondada 94] D. Floreano and F. Mondada. Active perception, navigation, homing, and grasping: An autonomous perspective. In Ph. Gaussier and J.-D. Nicoud, editors, *From Perception to Action Conference, Los Alamos.* IEE Computer Society Press, 1994.

[Floreano & Mondada 96] D. Floreano and F. Mondada. Evolution and mobile autonomous robotics. In E. Sanchz and M. Tomassini, editors, *Towards Evolvable Hardware. Lecture Notes in Computer Science.* Berlin:Springer-Verlag, 1996.

[Floreano & Mondada 97] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, 26(3):396–407, 1997.

[Floreano & Nolfi 97] D. Floreano and S. Nolfi. Adaptive behavior in competing co-evolving species. In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life*, pages 378–387. MIT Press, 1997.

[Floreano 97] Dario Floreano. Reducing human design and increasing adaptability in evolutionary robotics. In Takashi Gomi, editor, *Evolutionary Robotics.* AAI Books, Ontario (Canada), 1997.

[Gathercole 98] C. Gathercole. *An Investigation of Supervised Learning in Genetic Programming.* Unpublished PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1998.

[Gomi & Griffith 96] Takashi Gomi and Ann Griffith. Evolutionary robotics – an overview. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996.

[Harvey 92] Inman Harvey. Species adaptation genetic algorithms: A basis for a continuing saga. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial*

Life, pages 346–354, Cambridge, MA, 1992. MIT Press/Bradford Books.

[Harvey *et al.* 93] I. Harvey, P. Husbands, and D. Cliff. Issues in evolutionary robotics. In H. Roitblat J.-A. Meyer and S. Wilson, editors, *From Animals to Animats 2: Proc. of the Second Intl. Conf. on Simulation of Adaptive Behavior, (SAB92)*, pages 364–373, Cambridge MA, 1993. MIT Press/Bradford Books.

[Hautop-Lund *et al.* 97] Henrik Hautop-Lund, John Hallam, and Wei-Po Lee. Evolving robot morphology. invited paper. In *Proceedings of the 4th International Conference on Evolutionary Computation.* IEEE Press, 1997.

[I. Harvey 97] D. Cliff A. Thompson N. Jakobi I. Harvey, P. Husbands. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[Jakobi 94a] N. Jakobi. Evolving sensorimotor control architectures in simulation for a real robot. Unpublished M.Sc. thesis, School of Cognitive and Computing Sciences, University of Sussex, 1994.

[Jakobi 94b] N. Jakobi. Evolving sensorimotor control architectures in simulation for a real robot. Unpublished M.Sc. thesis, University of Sussex, 1994.

[Jakobi 97a] N. Jakobi. Evolutionary robotics and the radical envelope of noise hypothesis. *Journal of Adaptive Behaviour*, 6, 1997.

[Jakobi 97b] N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life*, pages 378–387. MIT Press, 1997.

[Jakobi 98] Nick Jakobi. Evolving motion-tracking behaviour for a panning camera head. In *Proceedings of the 5th International Conference on Simulation of Adaptive Behaviour*. MIT Press, 1998.

[Jakobi *et al.* 95] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *In Proceedings of 3.rd European Conference on Artificial Life (ECAL'95)*. Springer-Verlag, 1995.

[Jakobi ng] N. Jakobi. Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics. In Phil Husbands and Inman Harvey, editors, *Advances in Artificial Life: Proc. 4th European Conference on Artificial Life*. MIT press (1997 forthcoming), (1997 forthcoming).

[Johnson *et al.* 94] Michael Patrick Johnson, Pattie Maes, and Trevor Darrell. Evolving visual routines. In Rodney A. Brooks and Pattie Maes, editors, *ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 198–209, MIT, Cambridge, MA, USA, 6-8 July 1994. MIT Press.

[Kitano *et al.* 97] H. Kitano, M. Asada, Y. Kunyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of The First International Conference on Autonomous Agent (Agents-97))*. The ACM Press, 1997.

[Koza 92a] John R. Koza. Evolution of subsumption architecture using genetic programming. In F. J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 110–119. MIT Press, 1992.

[Koza 92b] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, Cambridge: MA, 1992.

[Koza 94] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge Massachusetts, May 1994.

[Lee 97] W-P. Lee. *Evolving Robots: from Simple Behaviours to Complete Systems.* Unpublished PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1997.

[Lee *et al.* 97] Wei-Po Lee, John Hallam, and Henrik Hautop-Lund. Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. In *In Proceedings of IEEE 4th International Conference on Evolutionary Computation.*

IEEE 4th International Conference on Evolutionary Computation, IEEE Press, 1997.

[Matarić & Cliff 95] Maja J. Matarić and Dave Cliff. Challenges in evolving controllers for physical robots. Technical Report CS-95-184, Brandeis University Computer Science, November 1995.

[Matarić 90] Maja J Matarić. A distributed model for mobile robot environment-learning and navigation. AI Lab technical report 1228, MIT, May 1990.

[Miller & Cliff 94] Geoffrey F. Miller and Dave Cliff. Co-evolution of pursuit and evasion i: Biological and game-theoretic foundation. *Adaptive Behavior*, 1994.

[Montana 93] David J. Montana. Strongly typed genetic programming. BBN Technical Report #7866, Bolt Beranek and Newman, Inc., 10 Moulton Street, Cambridge, MA 02138, USA, 7 May 1993.

[Nolfi *et al.* 94] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In R. A. Brooks and P. Maes, editors, *Proceedings of the IV International Workshop on Artificial Life*, pages 190–197, Cambridge, MA, 1994. MIT Press.

[Nolfi ss] Stefano Nolfi. Evolving non-trivial behaviors on real robots: a garbage collecting robot. *Journal of Robotics and Autonomous System, "Robot learning: The new wave" special issue*, (in press).

[Perkins & Hayes 96] Simon Perkins and Gillian Hayes. Robot shaping - principles, methods and architectures. Technical report, Department of Artificial Intelligence, University of Edinburgh, March 1996.

[Perkins & Hayes 97] Simon Perkins and Gillian Hayes. Incremental acquisition od complex behaviour using structured evolution. In *International Conference on Neural Networks and Genetic Algorithms, Norwich*, 1997.

[Pollack & Ringuette 90] Martha E. Pollack and Marc Ringuette. Introdcing the tileworld: Experimentally evaluating agent architectures. In *AAAI-90*, pages 183–189, 1990.

[Resnick 97] Mitchel Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds.* The MIT Press, 1997.

[Reynolds 94a] Craig W. Reynolds. The difficulty of roving eyes. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, pages 262–267, Orlando, Florida, USA, 27-29 June 1994. IEEE Press.

[Reynolds 94b] Craig W. Reynolds. Evolution of corridor following behavior in a noisy world. In *Simulation of Adaptive Behaviour (SAB-94)*, 1994.

[Reynolds 94c] Craig W. Reynolds. An evolved, vision-based behavioral model of obstacle avoidance behaviour. In Christopher G. Langton, editor, *Artificial Life*

III, volume XVII of *SFI Studies in the Sciences of Complexity*, pages 327–346. Addison-Wesley, Santa Fe Institute, New Mexico, USA, 15-19 June 1992 1994.

[Smith 97] T. Smith. Adding vision to khepera: An autonomous robot footballer. Unpublished M.Sc. thesis, School of Cognitive and Computing Sciences, University of Sussex, 1997.

[Smith 98] T. Smith. Blurred vision: Simulation-reality transfer of a visually guided robot. In J.-A. Meyer and P. Husbands, editors, *Proceedings of EvoRob'98*. Springer-Verlag, 1998.

[Tackett 93] Walter Alden Tackett. Genetic generation of "dendritic" trees for image classification. In *Proceedings of WCNN93*, pages IV 646–649. IEEE Press, July 1993.

[Teller & Veloso 95] Astro Teller and Manuela Veloso. A controlled experiment: Evolution for learning difficult image classification. In *Seventh Portuguese Conference On Artificial Intelligence*, volume 990 of *Lecture Notes in Computer Science*, pages 165–176, Funchal, Madeira Island, Portugal, October 3-6 1995. Springer-Verlag.