Motion Optimization for the RoboCup 3D Soccer Simulation

[1]Vallade Benoît, [1]Takeshi Sumitani, [1]Tomoharu Nakashima[1]
[2]Takeshi Uchitane and [2]Toshiharu Hatanaka

Department of Computer Science and Intelligent Systems, Osaka Prefecture University
{valladeben@cs, takeshi.sumitani@ci.cs, tomoharu.nakashima@kis }.osakafu-u.ac.jp
[2]Department of Computer Information and Physical Sciences, Osaka University
{hatanaka, t-uchitane }@ist.osaka-u.ac.jp

Abstract

This paper addresses a motion optimization method for RoboCup 3D Soccer Simulation's robots using Particle Swarm Optimization (PSO). A robot's motion is modelled by the motion trajectories of the robot using some parameters. The trajectory equations' parameters are adjusted to obtain the best motion. We use particle swarm optimization algorithms to tune the trajectory parameters .

1 Introduction

Nowadays, one of the major robotic research focuses on humanoid robots and their relationships with human and/or other robots. In the aim to give a common target and problem to all laboratories, the RoboCup has been created. This worldwide project is divided in lots of categories such as rescue and home. Among these categories, the first to have been created was the soccer category. The RoboCup Soccer's objective is to create and improve robots and their behaviours to be able to play to soccer [1] The major research points of this category are the improvement of the robot's motion and their communication. The objective is to be able to play and win against the world-cup championship team by 2050.

The RoboCup Soccer category is further divided into several leagues according to the regulation of robots: Humanoid, middle-size, small-size, standard platform, and simulation. The simulation league focuses on the intelligent aspect of soccer robots as it employs virtual soccer players of computer program that play soccer in a virtual soccer field. There are two sub-leagues in the simulation league depending on the specification of the virtual field: 2D and 3D. While the 2D league has no concept of height as every object is represented as a two-dimensional vector, the 3D league has a more realistic virtual environment: The soccer field is placed in a three-dimensional space and soccer robots are built in the virtual field by joining multiple elemental parts with joints. Thus the soccer robots are more realistic and more complicated than 2D soccer robots. The development of 3D robots faces many problems. One of the major problems is the motion optimization. We need to improve robot motions such as walk, kick, dribble, etc., to obtain stable, quick and efficient movements. But the variety of robot's architecture, the number of parameters needed to configure a motion and the dependencies between the parameters of its motion are the root cause of the complexity of the optimization problem. To be able to answer to these problems we need to create a software able to cope with the different robots configuration and research the best values for the set of parameters in a huge research space.

2 Motion Optimization

A widely used way to optimize robot's motion is to use inverse kinematics. We could create a motion by setting all the engines of the robot, but there are too many parameters. So in order to reduce the number of parameters, we choose some points of the robot and search to optimize their trajectory. After having determined these trajectories by setting a value to the equation's parameters we use mathematics rules to deduce the trajectory of each engine used in this motion. This method is ever used by some other teams; the innovations are to use the PSO as an evolution strategy [2] and to try to create adaptable software.

2.1 Target Trajectories

The first step is to choose the target points on the robot. These points will determine the motion that we want to obtain. For example, in case of a walk motion, the focal points will be the end of the robot's foots and the centre of the hip (see Fig. 1) [3]

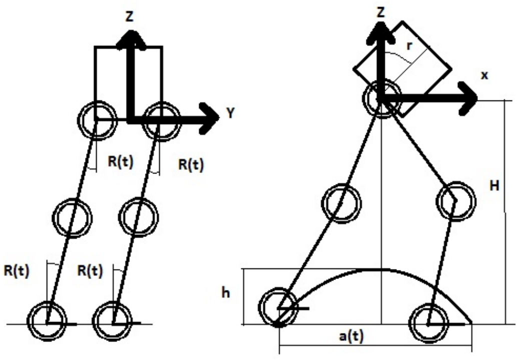

Fig. 1. Choice of target points and trajectories for a walk motion.

After having chosen the points, the second step is to choose their target trajectories. These trajectories will create the desired movement. They are represented by a movement equation such as linear movement or oscillators' movement and can be coupled together. To keep the example of the walk motion, here an equation of movement for one of the tree target points of this motion:

$$\dot{\varphi}_i(t) = \omega_i + \sum_{j(j\neq i)}^{3} \omega_{ij} \sin(\varphi_j(t) - \varphi_i(t) + \delta\theta_{ij}) \quad (1)$$

Finally we obtain some equations which are dependant of a set of parameters. And to fully determine these equations we need to put value on these parameters.

2.2 Parameter Optimization

The precedent equations depend on parameters which can be separate in two kinds. There are the parameters whose values are constant, like the size of legs, arms or foot of the robot. They are based on the robot's architecture. The second kind is the parameters which will be tune by the research algorithm. Here an example of set of parameters (for walking motion)

Parameter name	min	max
$\omega_1 (= \omega_2 = \omega_3)$	2.0	10.0
$\omega_{12} (= \omega_{21} = \omega_{13} = \omega_{31})$	-1.0	1.0
$\omega_{23} (= \omega_{32})$	-1.0	1.0
a_{max}	0.0	$2\sqrt{(L1 + L2)^2 - H^2}$
a	0.0	a_{max}
St	0.0	a_{max}
h	0.0	$\frac{L1 + L2}{2}$
H	$\frac{L1 + L2}{2}$	$L1 + L2$
r	0.0	30.0
$roll_{max}$	0.0	10.0
P_{gain}	0.0	5.0
D_{gain}	-5.0	5.0

Table. 1. List of parameters for walking motion.

Each parameter has a value and a definition space. The limits of these definition spaces are given by constant values or expressions which depend on the value of other parameters.

In order to give a value to this set of parameters the second step uses a research algorithm. This algorithm will progress in a research space define by crossing all the definition spaces. It will search the best combination of value for the set of parameters.

The simulator will be use to evaluate each solution. We will launch a simulation for each solution and get data about robot's performance and environment state. The research algorithm will use these data to evaluate the quality of each solution.

2.3 Inverse kinematics

During the third step we use the inverse kinematics in aim to find the movement parameters (angle, velocity, centre... etc) of each engines used in the motion (knee, hips... etc).

We deduce these parameters from the robot architecture, the successive positions of the target points and by using some mathematical rules like Pythagoras, and trigonometry.

3 Software Design

An important aspect of this software is that is still in development and even after being finished it could be subject to improvement (like parallelization, complex target motion). That is why we have decided to set up the design in three modules.

The first module is the PSO module. This one will be the subject of further discussions, so we won’t describe it now.

The second module has many objectives; first it should be able to represent the architecture of the robot. The software will be able to use the xml file representing the robots for the simulator, but we also think to insert a module which allows the user to enter manually via a graphical interface the configuration of the robot. After the robots defined by the system, the user will be able to choose some target points on the robot and their target trajectories and indicate if they are coupled. The equation of these trajectories will be automatically generated as well as the corresponding set of parameters. Afterward the user will manually choose the definition space (max and min) of each parameter. The aim of this paper is that these values can be constant as well as an equation. This implies the set up of an equation parser. This one is for the moment in development but will not be the subject of this paper. As last but main objectives, the third module will have the ability to use the solution returned by the PSO module and to use the inverse kinematics to define the motion parameters of each engine used in the movement and generate a motion configuration file. In fact to move, our actual robocup agents use motion configuration files. It exist one file for each motion that the agent wants to execute and the AI of this one will have to choose between all its possibilities. These configuration files are xml formatted and describe the motion parameters of each engine. The engines are defined by the representation of the robot’s architecture. But the problem is that each robot’s configuration an each choice of target point will implies a different mathematical problem to solve. In consequence we will have to set up an IA able to solve them.

The server module is the last one. Its task is to help the user to easily use the simulator server. This interface should allow the user to launch a simulator server, to get the data from the simulation by using a log file system and finally to be able to kill the server. In the aim to be easily improved to a parallel version of the software, the interface has been design to allow managing more than one server in same time. Another little part of the software has been developed in order to easily management he log file and extract the data. However this last one will not be describe in this paper.

Figure 2 shows is the global algorithm that the system will follow:

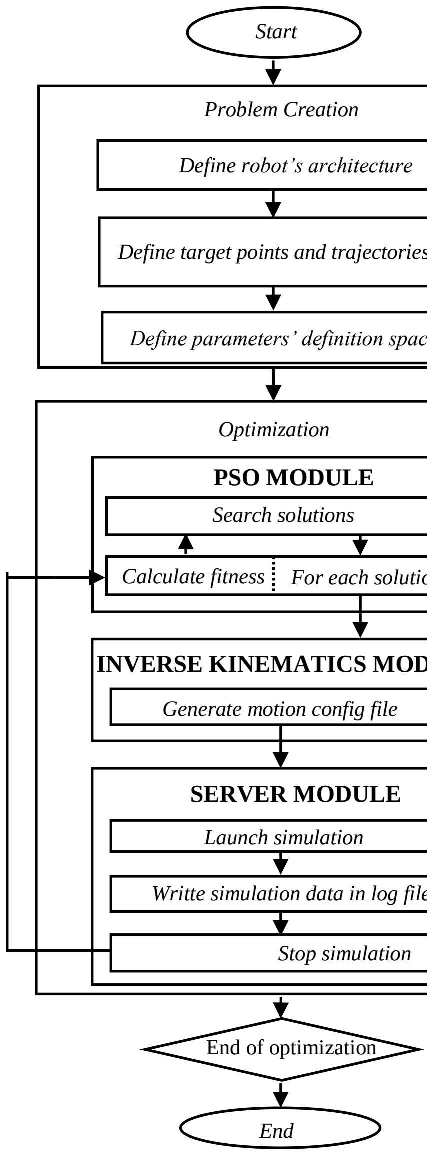
Start
Problem Creation
Define robot’s architecture
Define target points and trajectories
Define parameters’ definition space
Optimization
PSO MODULE
Search solutions
Calculate fitness
For each solution
INVERSE KINEMATICS MODULE
Generate motion config file
SERVER MODULE
Launch simulation
Writte simulation data in log file
Stop simulation
End of optimization
End

Fig. 2. Global algorithm.

4 Particle Swarm Optimization Module

4.1 Standard algorithm

The particle swarm optimization (PSO) is a research algorithm whose objective is to find the best combination of values for a set of parameters.

The particle swarm optimization is based on the idea of a flock of fishes which move in a space [4] The aim of each fish is to find the place where there is the best food (this representing the quality of a solution calculated with the fitness function). The fishes always remember the last best place that they have find and they communicate together to share information about the best place find by the whole flock. The fish will move to a new position in function of these two information. Finally the whole flock should finish by find the best place of the space.

Algorithm Components. As said just before the PSO is based on a swarm of particles which are inserted in a research space. The space is created by crossing the definition spaces of the set parameters. Each particle knows its position in this space, the quality of this position, and remembers about the best position that the particle has ever found and the best position that the swarm has ever found. In addition it knows their both quality. To finish each particle have a velocity which is use to move in the space. The position and velocity of each particle have to be contained in the research space.

Initialisation. At the beginning we have to initialise each value of the particles. So we generate a random position in the space and a random velocity. The best position of the particle will take the value of its first position. The best position of the swarm will be determined by sharing information between all the particles' best position.

Evolution.

Number of Iterations. After the initialisation, we launch the research. The research will continue until a stop criterion is complete. These criterions can be the time, the number of iteration or an optimum value has been reach (with a certain error).

Update Position and Velocity. At each iteration, we calculate the new position of the particles and the quality of this position. The quality of the particles is measured by a pre-specified objective function. Then we check if the best position of the particle have been improved and share the information with the whole swarm. Finally we update the velocity of each particle.

The new position of a particle is calculated in function of the old position and the velocity (see Fig. 3).

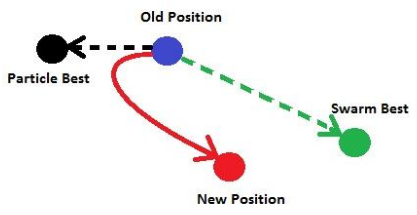

Fig. 3. Update of the particle position.

The velocity is itself calculated in function of the best Position known by the particle and the best position known by the swarm [5].

Each time the value of the position and the velocity are updated parameter by parameter.

Fitness Function. The quality of a position is calculated with the fitness function. This function is defined by the user depending on the problem. For example in the case of a walk motion optimization, the quality can be calculated depending on the distance travelled in a given time, the error of trajectory of the robot and its stability.

Confinement. If the new position go out the research space the particle become useless because its position is not valid. That is why the particle swarm optimization algorithm uses a confinement procedure which moves the lost particle on the closest edge of the research space and put its velocity to 0.

Limitation. But the standard particle swarm optimization algorithm as its own limits. The fact is it is only possible to use it to search a solution for a set of parameters whose limits of definition space are constant.

4.2 Dynamic search space PSO

In our case, the parameters of movement equation are often linked each other. For example during a walk motion the maximum high authorized for the foot will depends on the actual value of the hips high. It means that each particle will have its own search space depending on its position.

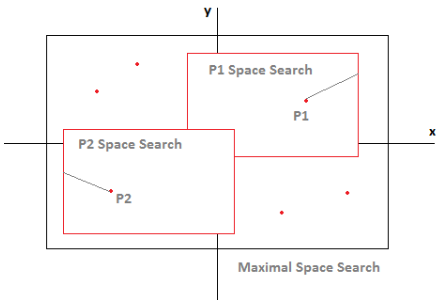

Fig. 4. Search space representation in a two-parameter optimization problem.

Two solutions are possible; the first is to use the precedent algorithm with a space equal to the maximal space search. We define it by putting the maximal value of the parameters in the expressions which determine the limits of the definition space. But use this method implied to effectuate the research in lots of positions which are not valid (they correspond to a totally impossible movement). It is a loss of time. The second one is to use one search space search for each particle. This is the subject of this part.

One Search Space by Particle.

Problem. As said before a research space if defined by crossing the definition spaces of the set of parameters. But in this case some definition space's limits are defined by mathematical expression which depends on the value of some parameters of the set. So the parameters are linked each other.

The problem is that when we want confined the particles in the search space; we change the value of the position parameter by parameter. But each time that we change the value of one parameter, the position of the particle will change. And so on for its search space. If we effectuate the change without any regards for the order of the links between the parameters there is a possibility that the procedure enter in an infinite loop.

To avoid this problem, the best way is to confine the particle following a precise order which correspond to the link order of the parameters.

Tree Representation. To be able to find the good order we can use a tree representation of the links between the parameters. Each parameter has two limits, which depends of a subset of parameters.

In this tree, the nodes will represent the parameters. A parameter is father of all the parameters it depends of. A parameter which depend of nothing is a leaf directly link to the root. The root node doesn't correspond to any parameter. It only exists to be able to create a single tree even with parameters without dependences. Here an example of tree for four parameters A, B, C, D and where to calculate B you need to know the value of C and D.

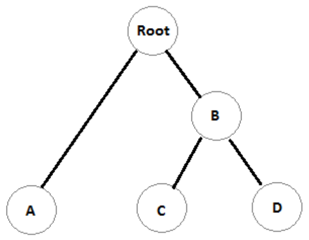

Fig. 5. Tree of parameter dependences.

Modification of the Standard PSO. Finally in the aim to use the PSO algorithm, we need to change the confinement procedure. If the particle is not in its search space, we need to put the particle on the closest edge of the search space. But we have to follow the link order.

The following figure show in red the good path to determine the parameters value after confinement.

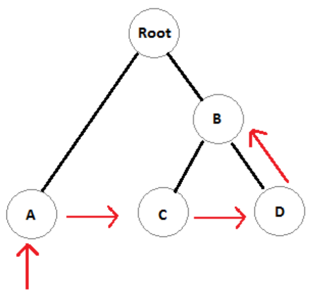

Fig. 6. Confinement parameters order.

We start from the “A” node, and each time that we move on a new node we determine the new limits of the definition space with the parameter value calculate before and put the parameter value of the actual node to the closest limit. And go to the next node until we arrive to the root node.

4.3 Experience

In order to test this module of and to compare the efficiency between the standard and the new algorithm, we have set up an experience. For a same problem composed of a set of variables, which take their value in a defined interval, and of a fitness point, we will launch a set of simulation for both algorithms. And we will record the number of iterations needed to find the fitness point.

The problem. To For this experience we chose a two dimensional problem. It means that the problem is composed of two variables x and y. We defined a fixed fitness point, the point of coordinate (5.5 ; 0.01). In this problem, during the new algorithm test, the second variable (y) space search limits will depend on the value of the first variable (x). The limits of the variables of the standard algorithm test will be defined as the bigest space possible (biggest value for x enter in the limits equation of y). In the next graph we can visualize the space limits:

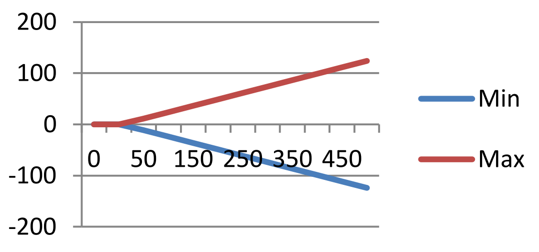

Fig. 7. Example of the space limits (case B).

We will effectuate the experiments in three situations A, B and C. Each situation corresponds to a different variable space search (0<x<1000) , their limits value are presented in the following table :

	New Algorithm		Standard Algorithm	
	Min	Max	Min	Max
A	$-(0.05*x) + 0.25$	$+(0.05*x) - 0.25$	-49.75	49.75
B	$-(0.25*x) + 1.25$	$+(0.25*x) - 1.25$	-240.5	240.5
C	$-(0.5*x) + 2.5$	$+(0.5*x) - 2.5$	-497.5	497.5

Table. 2. List limits of definition for the variable y.

For each experience we will launch a set of simulations with 50 particles and 500 iterations (which is enough to solve the problem).

The results. For each simulation we have check how many iterations the algorithm needed to find the fitness point. Here are the results of the experiences:

	A	B	C
Standard algorithm	279.4	296.5	298
New algorithm	282.3	254.5	250.3

Table. 3. Average number of iterations needed to find the fitness point.

We remark that the standard algorithm become less efficient on a big research space (B and C) and so in these cases we have a gain of time. The gain is not so big but the problem was easy, 2 dimensions and only 50% of impossible positions. But our software will have to deal with more complicated. In addition fitness calculations need a simulation of the robots. This simulation can take 10 sec as 1 or 2 minutes in function of the need. So each iterations gain is a big victory. Finally the improvement of the algorithm still allows to parallelise it [6].

5 Conclusion

To conclude this paper, even this software project still in development some base the design have been choose and some important module have ever been implemented. The aim is to allow to easily generating optimized motions for the robot of y our choice and by the way we can use it to try different kind of architecture.

PSO is a very easy algorithm which is highly parallelizable. The experiences showed the efficiency of the new algorithm. The server module is very important because it will also help to parallelize our software by launching lots of simulation in a same time but also by following the experience of several agents in a same simulation.

References

1. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer, MIT Press (2000.)
2. Cord Niehaus, Thomas Röfer, Tim Laue: Gait Optimization on a Humanoid Robot using Particle Swarm Optimization
3. Uchitane, T., Hatanaka, T.: Applying Evolution Strategies for Biped Locomotion Learning in RoboCup 3D Soccer Simulation, Proc. of 2011 IEEE Congress on Evolutionary Computation, pp. 179-185. (2011)
4. Kennedy, J., Eberhart, R..: Particle Swarm Optimization, Proc. of IEEE International Conference on Neural Networks, pp.1942-1948. (1995)
5. Clerc, M.: Particle Swarm Optimisation, John Wiley & Sons (2010)
6. J.F.Schutte, J.A.Reinbolt, R.T.Haftka, A.D.George: Parallel global optimization with the particle swarm algorithm, (2004)