Visually-Guided Reaching by Genetic Programming

Jumpol Polvichai *Prabhas Chongstitvatana*

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University, Bangkok 10330, Thailand
email: g37jpv@chulkn.car.chula.ac.th & fengpjs@chulkn.car.chula.ac.th

Abstract

This work describes a hand-eye system that can learn from its experience. The task is to visually-guide the hand to the target avoiding obstacles. The path planning problem is solved by genetic programming. The vision system maps the experiment onto an image plane which is used to do simulation run during the genetic programming process. The genetically created programs are validated by the actual runs on the robot successfully.

1 Introduction

One aim to achieve general intelligence in the field of artificial intelligence is to have a system that can learn. A learning system can extract information from an unknown environment, especially when an accurate model of environment is not available. Agents having a continuous existence can also learn for experience, they should be able to learn how to solve new problems when the circumstances arise.

One technique of learning in artificial intelligence is *genetic programming* method. In genetic programming paradigm genetically breeds populations of computer programs are used to solve problems. The individuals in the population are compositions of functions and arguments. An evolutionary process is driven by the measure of fitness of each individual computer program in handling the problem environment.

Visually-guided robotics task is a good domain for experimentation in learning as it maps perceptual tasks in the world in which the relevant laws and interrelationships between an active agent and the environment are highly reliable.

2 Previous Work

Visual feedback has been used to guide robots in hand-eye coordination tasks since the early days of robotics research. Jones [4] demonstrated the use of visual tracking methods to perform block stacking and loose insertion.

In motion planning for hand-eye coordination task, the system such as HANDEY [9] relies on geometrical modeling of the environment and robots. The system plans collision free paths in configuration space. It does rely on accurate models which makes the system unable to cope with uncertainty.

In Mel [10], MURPHY, a robot motion planning system in presented. The system approaches the problem of visually-guided reaching using connectionist method. MURPHY uses neural-like units to learn the association between visual perception and robot arm motions in which it learns both forward kinematics and inverse differential kinematics. To plan a collision free path , the system uses heuristic search the sequences of arm configurations which move the hand to the target. MURPHY moves the arm randomly and does gradient descent in the distance to the target, backtracking if it fails and tries other paths if a current path is blocked by obstacles. It is this motion planning part that this paper attacks by genetic programming.

A good introduction to genetic programming can be found in Koza [7]. A similar approach to genetic programming is genetic algorithms which is pioneered by John Holland of the University of Michigan (Holland [3]). A lot of background can be found in Goldberg [2]. Koza and Rice [8] applied genetic programming to generate robot programs. Their example is that a mobile robot searching for a box and pushs it to the wall. The application of genetic algorithms to robot arm planning is numerous, for examples: Khoogar, Parker and Goldberg [5] solved inverse kinematics of redundant robots; Khoogar and Parker [6] did the obstacle avoidance; Chan and Zalzala [1] planned minimum-time trajectories.

3 Problem Statement

The system consists of a robot arm and a camera. The camera looks at the arm and the environment (as in figure 1). The task is visually specified : to move the hand to the target while avoiding obstacles.

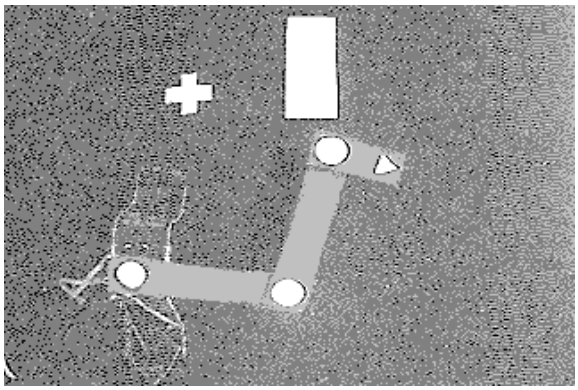

Figure 1 The picture of the robot arm and the environment looking through the camera.

4 Experiments

A vision system is used to identify the target, the obstacles and the robot arm. To simplify the vision algorithms, the robot arm has a distinctive color and two different thresholds are used to distinguish the robot arm from the obstacles and the target. A simple heuristic is used to locate three joints of the robot arm. The shoulder joint is the one near the bottom left of the image. The fingertip has a triangular shape. The wrist is then the joint that is near to the fingertip and, lastly, the elbow is the remainder joint.

The target, the obstacles and the arm are represented directly in the image. Therefore checking the collision and the out of bound condition are done in the image plane. This is, in a way, similar to the analogical representation paradigm.

In running the experiments, a forward kinematics model is built by instantly various parameters measured by the camera in the beginning of the run. This model is used during the simulated run of each program that is genetically generated

For preparing to use genetic programming, the set of terminals are identified to include six primitive servo motor functions and the system checking functions in the terminal set. Thus, the terminal set T for this problem is T = { s+, s-, e+, e-, w+, w-, HIT?, SEE?, INC?, DEC?, OUT?}. A set of functions for this problem, the function set F consists of F = { IF-AND, IF-OR, IF-NOT}.

The function s+ (shoulder up) drives the shoulder motor up 1 step and s- (shoulder down) drives the shoulder motor down 1 step. The similar meaning applied for e+, e- (elbow) and w+, w- (wrist). All of these functions will return true, if the servo motor can move to its position. If the servo motor can not move, as a result of, hitting an obstacle, moving out of the image, or moving to the limit of the servo motor, the function returns false.

The function HIT? checks whether each link of the robot arm hits the obstacle or not. This function depends on the value of PAIN-variable. When the arm crashes against an obstacle, 10 is added to the PAIN-variable, otherwise, 1 is substracted from the PAIN-variable if it is not equal to zero. The function returns true if the PAIN-variable is not equal to zero, . The function SEE? checks whether the path from the fingertip to the goal has any obstacle or not. The function INC? checks whether the distance between the fingertip and the goal is increasing or not. The function DEC? checks the opposite, the decreaing of the distance between the fingertip and the target. The function OUT? checks if each joint of the robot arm moves out of bound or not.

The function IF-AND is a four-argument comparative branching operator that executes its third argument if its first argument and its second argument are true, or otherwise, executes the fourth argument. The function IF-OR is a four-argument operator that executes its third argument if its first argument or its second argument is true, or otherwise, executes the fourth argument. The function IF-NOT is a three-argument operator that executes its second argument if the negative of its first argument is true, or otherwise, executes the third argument.

5 Genetic Programming Process

There are five stages in one cycle of the genetic programming process in the experiment.

Stage 1 : Creation of an initial population

The first step generates an initial population of computer program that randomly mixes the functions and the terminals. A sample of computer program shows in figure 2. The size of population is 40 computer programs.

```
(IF-AND w+ w+ e+ (IF-AND (IF-NOT (IF-NOT OUT? s- w+) (IF-AND e+ s- e- (IF-OR (IF-NOT w+ s+ e+) s+ e- e-)) (IF-OR (IF-NOT SEE? w- e-) w+ e+ e+)) w- (IF-OR SEE? (IF-OR e- (IF-OR (IF-OR HIT? e+ s+ e-) INC? w- e-) s+ w+) (IF-AND (IF-NOT w- e- e-) w- w- w+) s-) e+)))
```

Figure 2 A sample of computer program which is randomly generated.

Stage 2 : Verification of each computer program

In each generation, iteratively execute each program until one of the termination criterions has been qualified. The termination criterions is described as follows:

- Maximum execution time : not over 1,000 execution times.
- Maximum time : not over 50 seconds.
- Dead condition : There are two conditions.
  1.The arm has the same final position over 50 execution times.
  2.The value of PAIN-variable is over 2,000 units.
- Successful : the arm can reach to the goal position.

Stage 3 : Evaluation of each program

The fitness (or performance) of individual programs is evaluated with respect to the following conditions.

First, start with Fitness = 5,000. In case that the arm is not successful, subtract a multiple of k-constance and the distance between the fingertip position and the goal position from the fitness. If in the final position, the path from the fingertip to the target is blocked by an obstacle, substract 1,000 from the fitness. Next, if the arm is in the dead position, then subtract 4,000 from the fitness. If successful, add 3,000 to the fitness.

Stage 4 : Selection of the best ten programs

Based on the individual fitness of each program, the best ten programs are selected.

Stage 5 : Genetic manipulation

The manipulation process uses genetic operators to create a new population of individual programs. There are four main operations : reproduction, crossover, addition and extension. The reproduction is the operation that copies the best ten programs into the next generation. The crossover is the operation that creates the new ten programs by recombining randomly chosen parts of two of the existing best ten programs. The addition operation generates an additional node from the root node (firgure 3). The extension operations does likewise but from a terminal node (figure 4).

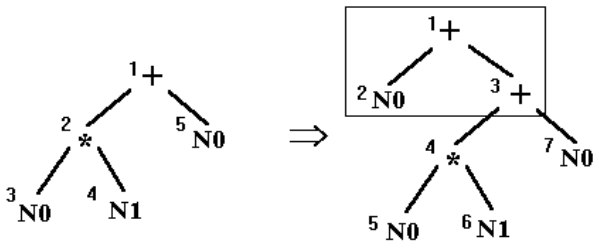

Figure 3 The addition operation.

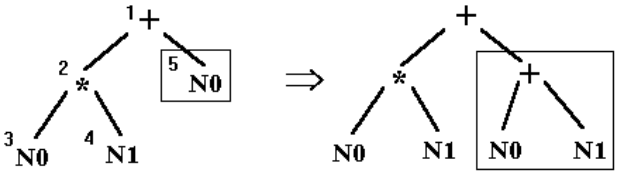

Figure 4 The extension operation.

In each generation, if there are no successful programs and the best ten programs of each generation have the same final position over 4 times, then mutation comes into action. The mutation operator performs random changes of the command code in every individual program in the population. The probability of this changing is 1/3.

6 Results and discussion

The experiments are run for 100 generations. In each generation the number of successful plan is record. The average performance is calculated from 12 experiments. The best program is then validated by running it on the real robot with the actual feedback (HIT?, SEE?, INC?, OUT?) from the camera. It performs successfully.

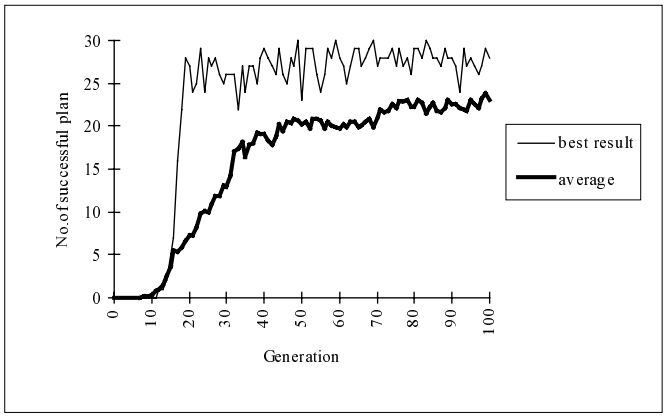

Figure 5 Performance plotted against generation for these experiments. The fat line shows the average performance of the population, and the thin line represents the performance of the best result of each generation.

In figure 5, the learning cure shows that the performance improves rapidly from the generation 10 to 40 and then level off by still increasing until the 100th generation. A sample of experiment as show in figure 6, the run of the best program on the actual robot is same as the run in the simulation. Some of experiments, the run on the actual robot is not exactly same as the run in the simulation. This is caused by the discrepancy in position predicted by the kinematics model and the actual position of the robot arm. The reason for this error comes from the non-linearity of the camera field of view. In theory, Instead of using the simulation run for GP, the actual run can be performed. But this will take unacceptable amount of time. An actual run of

a typical program takes about 120 second. It will take 40 x 120 x 100 s., or about 130 hours to complete an experiment.

7 Conclusions

This work has shown that the genetic programming method can be used to solve the path planning part in visually-guided reaching. The system improves its performance by learning from experience. The vision system maps the environment directly into the internal representation. This representation is use successfully as a simulated environment for genetic programming.

The ongoing work is on eliminating the current forward kinematics and constructing the actual *mental image* of arm configurations. This will enable the system to better reflect the real world in its simulation run phase.

Acknowledgements

This research was supported by the National Science and Technology Development Agency.

References

[1] K. K. Chan and A. M.S. Zalzala, "Genetic-Based Minimum-Time Trajectory Planning of Articulated Manipulations with Torque Constraints," IEE Colloquium on "Genetic Algorithms for Control Systems Engineering"(Digest No. 1993/130), pp. 4/1-3, 1993.

[2[D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. MA: Addison-Wesley, 1989.

[3] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[4] V. Jones, "Tracking: An Approach to Dynamic Vision and Hand-Eye Coordination," Ph.D thesis. University of Illinois, Urbana Champaign, 1974.

[5] A. R. Khoogar, J. K. Parker and D. E. Goldberg, "Inverse Kinematics of Redundant Robots using Genetic Algorithms," Proceedings.1989 IEEE International Conference on Robotics and Automation, pp. 271-276, vol. 1, 1989.

[6] A. R. Khoogar and J. K. Parker, "Obstacle Avoidance of Redundant Manipulators Using Genetic Algorithms," IEEE Proceedings of SOUTHEASTCON'91, pp. 317-320, vol. 1, 1991.

[7] J. R. Koza, "Genetically Breeding Populations of Computer Programs to Solve Problems in Artificial Intelligence," Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, pp. 819-827, 1990.

[8] J. R. Koza and J. P. Rice, "Automatic Programming of Robots using Genetic Programming," AAAI-92, Proceedings, Tenth National Conference on Artificial Intelligence, pp. 194-201, San Jose, California, July 12-16, 1992.

[9] T. Lozano-Perez, J. L. Jones, E. Mazer and P. A. O'Donnell, HANDEY A Robot Task Planner. Cambridge, Massachusetts, MIT Press, 1992.

[10] B. W. Mel, Connectionist Robot Motion Planning, A Neurally-Inspired Approach to Visually-Guided Reaching. Academic Press, Inc, 1990.

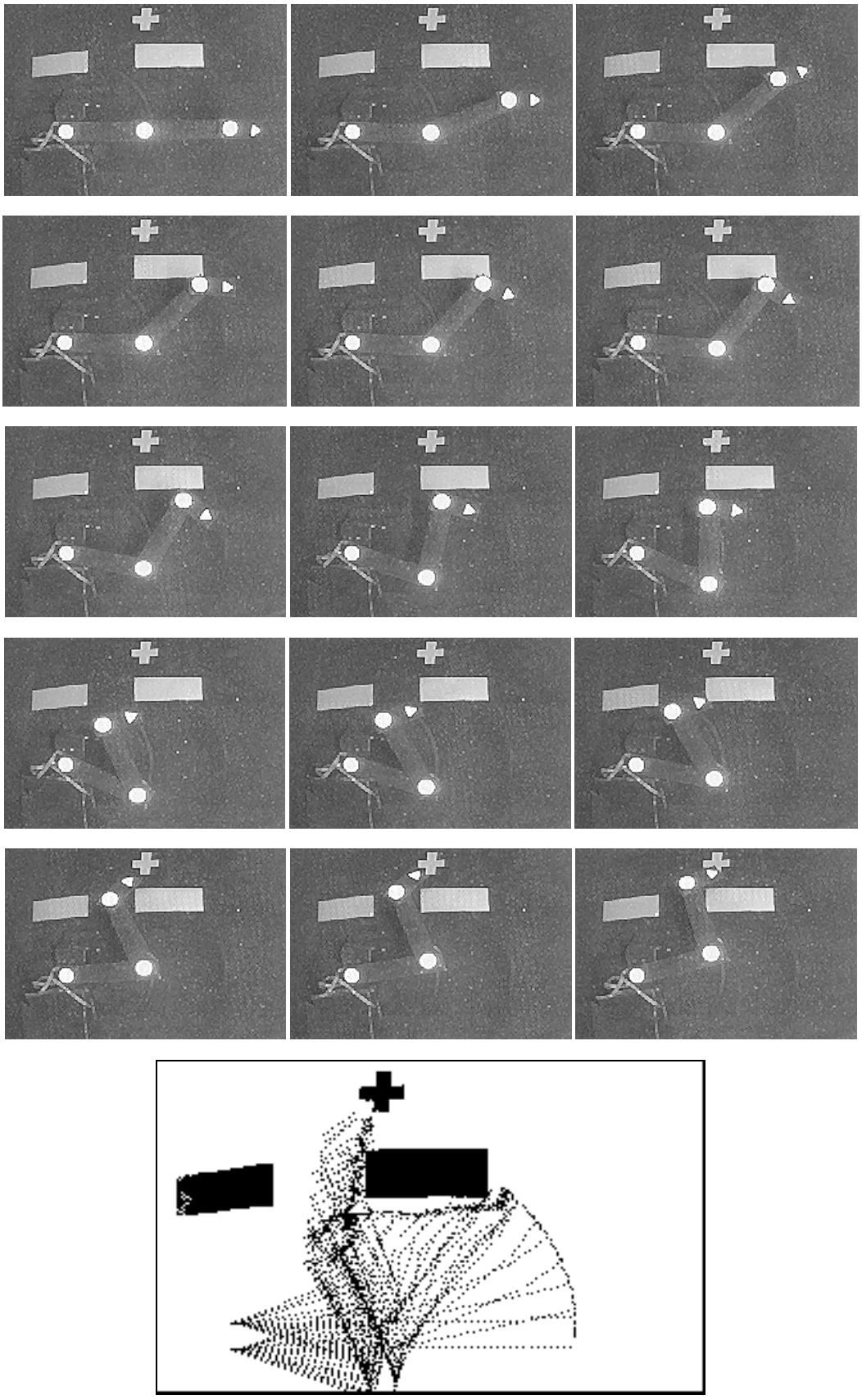

Figure 6 A sample of experiment which show the movement of the arm on the actual robot (above) and in the simulation (below).