Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 161–166
ISBN 978-83-60810-51-4

Laser trail shape identification technique for robot navigation based on genetic programming

Takeshi Tsujimura, Hiroki Fukushima, Yoshihiro Minato, and Kiyotaka Izumi

Department of Mechanical Engineering,
Saga University,
1 Honjo, Saga-city,
840-8502, Japan
Email: tujimura@me.saga-u.ac.jp

Abstract—This paper proposes a meta-heuristic image processing application for mobile robot navigation. It classifies figures that are drawn on a wall by hand with a laser pointer. Image processing technique extracts optical flow of the laser beam trail, which represents vectors along edges of shapes. Genetic programming learns geometric characteristics of laser trail shapes and creates classification algorithm. Three typical figures, such as a circle, a triangle, and a square, are evaluated and identified in high accuracy. We have investigated the effects of genetic programming parameters on the performance of shape identification. As a result, proposal system makes it possible to command robots by easy and intuitive action of drawing a figure only with a laser pointer.

I. Introduction

ROBOTS are coming to be used not only in the factories but also at home. It is desirable for the aged or children to control them with easier, simpler, and more intuitive method instead of key boards or joy sticks.

Several papers have reported on robot operating methods using a laser pointer [1]-[4]. A laser pointer can indicate the arbitrary points in the real space in easy and simple action. The previous studies applied the laser pointer only to direct the robot destination. It is difficult to instruct robots more complicated actions without any additional devices.

We have proposed the navigation system which guided robots according to the connoted figures drawn by a laser pointer. It classified several typical figures by evaluating incidence of momentary motion of laser spot according to the criterion determined based on geometrical properties of each figure [5]. We cannot be satisfied with its experimental record in shape identification.

We newly applied genetic programming to automatically create the classification algorithm. It learns the procedure of laser beam trail.

This paper proposes the meta-heuristic approach to identify shapes drawn by a laser pointer. We constructed the shape identification system based on classification programs evolved by genetic programming, and evaluated the performance of discrimination between three typical shapes.

II. Concept of Robot Navigation System

A. Robot navigation procedure

The conceptual diagram of the navigation system is shown in Fig. 1. This system acquires motion of the laser spot by a CCD camera, obtains geometric features by calculating optical flows, identifies its shape and gives a robot the command corresponding to the drawn figure. We assign robot commands simple figures such as a circle, a triangle, and a square.

Our proposed system distinguishes figures among those three shapes based on the laser beam trail.

The procedure of robot navigation is shown in Fig. 2.

(1) A human operator draws a figure with a laser pointer on the wall.

(2) A CCD camera catches the image of the laser spot trail. An image processor extracts motion of the laser spot as a series of optical flow vectors.

(3) The pattern recognition algorithm, previously created by genetic programming, distinguishes the drawn figure among candidate shapes.

(4) The system provides robots with the connoted command corresponding to the identified shape.

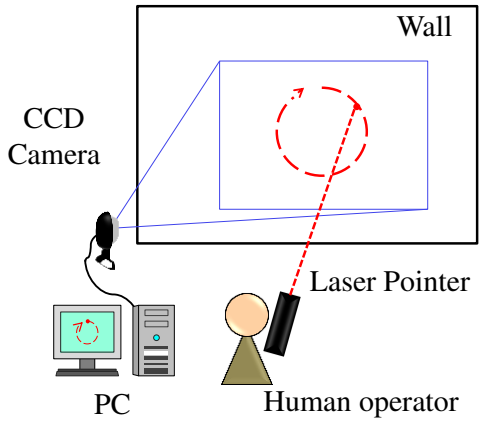

Fig. 1 Concept of robot navigation system

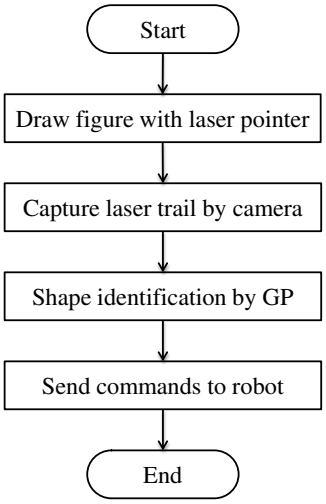

Fig. 2 Procedure of robot navigation

The 3rd stage of the procedure classifies drawn figures. Because a laser pointer is operated by hand, it is difficult for a human operator to draw figures correctly. It is necessary to evaluate such inaccurate shapes.

B. Concept of shape identification

Conventional studies focused on the corners of shapes to define polygons [6]-[8]. A triangle, for example, was identified by three corners, and classified by the interior angles.

As for figures drawn by hand, it is harder to identify their corners and to evaluate the angles accurately than their edges. That is why we have adopted edges to define shapes instead of corners.

When drawing a figure with a laser pointer, momentary motion vectors of the laser spot lie on the edge of the figure as shown in Fig. 3. In the case of a triangle, momentary motion vectors can theoretically be considered to be composed of three directions. Similarly, a square consists of four directions of vectors, and a circle contains omni-directional vectors.

Therefore, appearance pattern of the motion vectors helps us to identify the shape of a drawn figure.

Genetic programming technique creates a program which indicates the type of shape as output by giving the vectors as input.

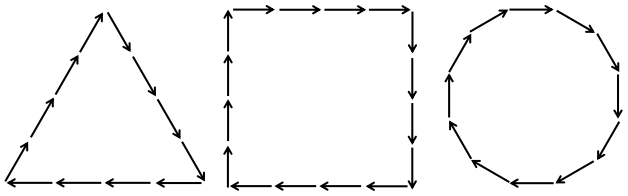

Fig. 3 Momentary motion vectors composing shapes.

III. SHAPE IDENTIFICATION BY GENETIC PROGRAMMING

Shape classification algorithm is established as a classification program created by genetic programming in advance of navigation. Genetic programming technique generates computer programs of dynamically varying size and shape [9]. The program represents classification algorithm. The evolutionary process evaluates how well individual program hits the right answer, and improves it.

This study deals with three shapes: a triangle, a square and a circle. The purpose of the identification algorithm is to provide an index representing each shape as output.

A. Genetic programming design

First of all, we prepared the geometric data regarding the shapes for learning. Each computer program, modified by genetic programming, is evaluated its ability of classification by applying the geometric data. We actually drew triangles, squares and circles with a laser pointer on a wall. Each shape was traced by hand with a 650-nm 1-mW laser pointer. Laser beam traveled 1.5 m from it to a white wall. Laser trail was detected by a 1.3 Mega pixel CMOS camera whose video resolution and frame rate were 640 x 480 and 30 FPS, respectively.

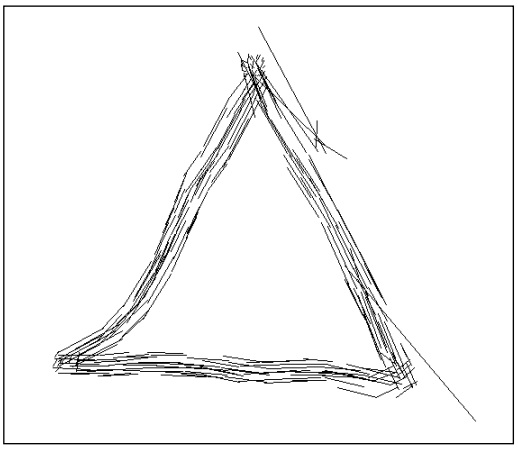

Fig. 4(a) Measured optical flow vectors (triangle)

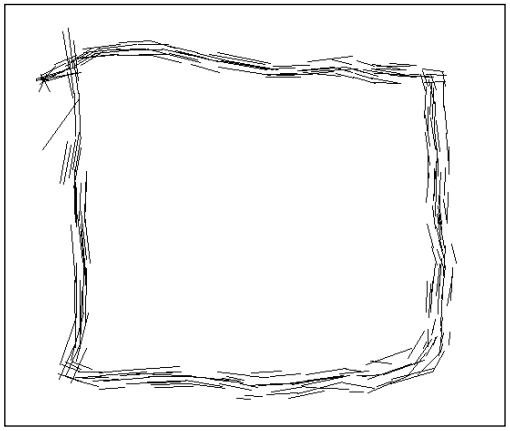

Fig. 4(b) Measured optical flow vectors (square)

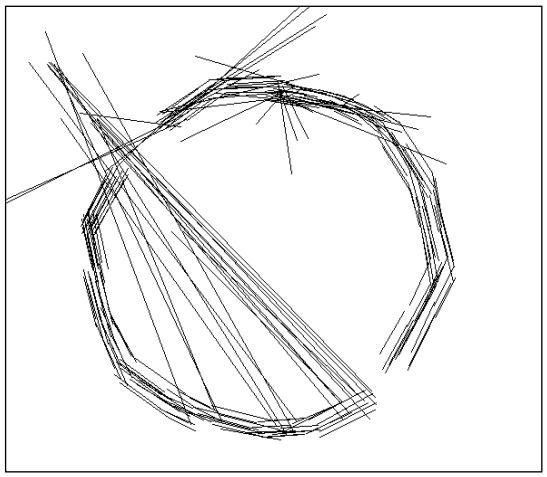

Fig. 4(c) Measured optical flow vectors (circle)

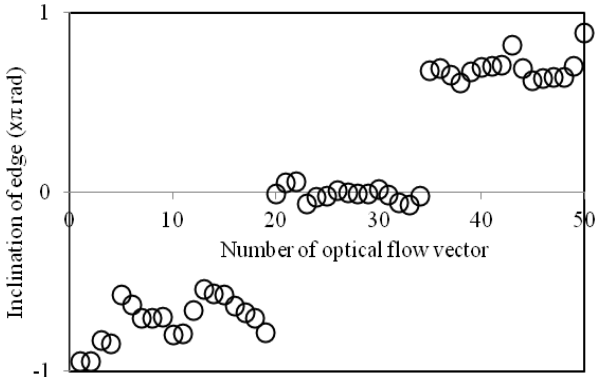

Fig. 5(a) Inclination of edge (triangle)

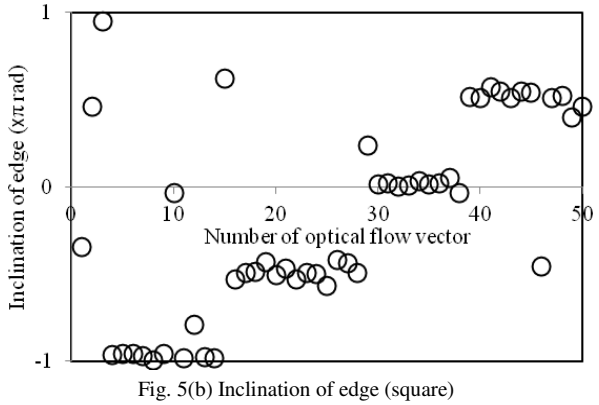

Fig. 5(b) Inclination of edge (square)

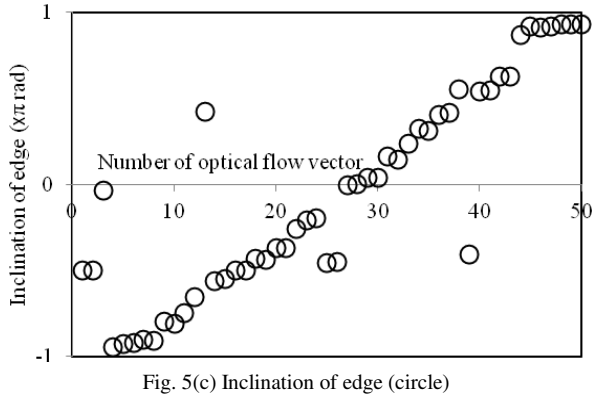

Fig. 5(c) Inclination of edge (circle)

The camera was placed 1 m apart from the wall. The length of an edge of the drawn triangles and squares was around 0.5 m. The diameter of the drawn circles was also 0.5 m.

Image processing was conducted by an Intel Core i7 @2.93GHz CPU. Image processing provides optical flow vectors concerning the figures as shown in Fig. 4, where each figure is constituted by 50 optical flow vectors. We therefore obtained 50 inclination data of each figure's edge as shown in Fig. 5.

An instance of a shape is composed of 50 inclination angles of the optical flow vectors. It also contains the numerical index of shape, index_i. The index represents the type of the shape. In this paper, a circle, a triangle, and a square are expressed by 0, 1, and 2, respectively.

Consequently, the instance, f_i, is described as

$$f\_i = (slant1i, slant2i, \ldots., slant50i, index\_i)$$

where slantNi [N=1, 2, …., 50] represents the inclination angle of the N-th optical flow vector, and i is number of individual drawn figure.

We obtained 50 instances for each of three shapes.

We next prepared the set of primitive functions as shown in Table 1. The set of terminals is also prepared as

$$(1 - k/18)\,\pi \qquad [k=0, 1, \ldots., 36]$$
$$slant1, slant2, \ldots., slant50$$

Each computer program is a composition of functions from the function set and terminals from the terminal set.

TABLE I FUNCTION SET

Function	explication
if(A,U,V)	If A is TRUE then U else V
add(U,V)	U + V
sub(U,V)	U - V
mul(U,V)	U • V
div(U,V)	U / V
equal(U,V)	If (|U-V|<π/18) then TRUE else FALSE
not_equal(U,V)	If (|U-V|<π/18) then FALSE else TRUE
>(U,V)	If (U>V) then TRUE else FALSE
<(U,V)	If (U<V) then TRUE else FALSE
S>(U,V)	If (sin(U)>sin(V)) then TRUE else FALSE
S<(U,V)	If (sin(U)<sin(V)) then TRUE else FALSE
C>(U,V)	If (cos(U)>cos(V)) then TRUE else FALSE
C<(U,V)	If (cos(U)<cos(V)) then TRUE else FALSE
and(A,B)	If both A and B are TRUE then TRUE else FALSE
or(A,B)	If either A or B is TRUE then TRUE else FALSE
not(A)	If A is TRUE then FALSE else TRUE

The fitness measure of the problem was determined as accuracy rate of the programs that correctly answered the type of shape. The closer the fitness is to 1, the better the program is.

Every computer program in the population returned a numerical value, which indicated the type of shape in this problem. The hits measure counted the number of fitness case for which the numerical value returned by the program came within a designated tolerance of the correct value described in the instance.

We then determined the values of parameters to control the runs. The population size and the maximum number of generations were 3000 and 300, respectively.

The termination criterion was triggered either by running the maximum number of generations or by the satisfaction of success predication by at least one program in the population. The success predicate for this problem was that a program hit the right answer with regard to the entire instances for learning.

B. Experiments

A run of genetic programming started with the creation of a population of 3000 random computer programs. They were generated from the available functions from the above function set and the available terminals from the terminal set. We compared three types of function set shown in Table 2. Basic arithmetic operations were available in CASE A. CASE C evaluated the inclination angle of optical flow vectors in trigonometry.

TABLE II AVAILABLE FUNCTION

Function	CASE A	CASE B	CASE C
if(A,U,V)	○	○	○
add(U,V)	○	×	×
sub(U,V)	○	×	×
mul(U,V)	○	×	×
div(U,V)	○	×	×
equal(U,V)	○	○	○
not_equal(U,V)	○	○	○
>(U,V)	○	○	×
<(U,V)	○	○	×
S>(U,V)	×	×	○
S<(U,V)	×	×	○
C>(U,V)	×	×	○
C<(U,V)	×	×	○
and(A,B)	○	○	○
or(A,B)	○	○	○
not(A)	○	○	○

○: available, ×: unavailable

The randomly generated individuals found in the initial generation of the population had blind random search of the space of computer programs representing possible solutions according to the flowchart shown in Fig. 6. The Darwinian reproduction operation was applied a designated number of times to single individuals selected from the population on the basis of their fitness. Both the genetic crossover and mutation operations were applied to the designated number of pairs of parents selected on the basis of their fitness to breed a new population of programs. The crossover and mutation rate were individually altered from 0 to 1.0.

We designated the best-so-far individual as the result of a run of genetic programming.

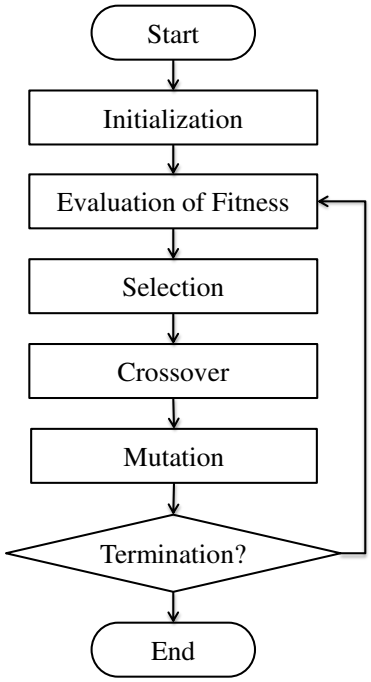

Fig. 6 Flowchart of genetic programming

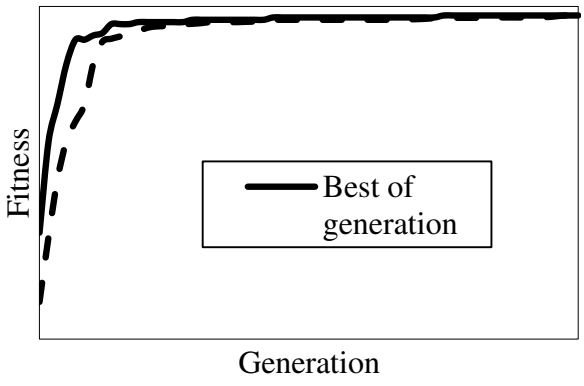

Fig. 7 Fitness curve of shape identification

Figure 7 presents the fitness curves on condition that population size is 3000, maximum tree depth is 20, maximum program size is 20, crossover rate is 1.0, mutation rate is 0.9, and using function set CASE C.

It displays the typical process of evolution where the vertical axis shows the accuracy rate of modified programs in learning, and the horizontal axis expresses the generation. The solid line indicates the fitness of the best-of-generation program, and the broken line represents the average of fitness for the population as a whole. The solution program is evolved rapidly until 50 generations as fitness exceeds 0.9.

We next investigated the effects of genetic programming parameters on the performance of shape identification. Accuracy rate of the eventually evolved programs is evaluated in terms of available functions and maximum program size. Maximum program size represents the limit of the number of producted program rows.

The final, i.e. in the 300th generation, fitness of three types of function set is compared with regard to maximum program size in Fig. 8, provided that population size is 3000, maximum tree depth is 20, crossover rate is 1.0, mutation rate is 0.9. Figure 8(a) expresses that the best-of-generation program provided the desirable performance when the program size was more than 30, and the function set CASE A was inferior to the others. Figure 8(b) indicates little relevance of maximum program size to the average of the whole programs. CASE C is considered to the best choice among three, whose fittness is not more than 0.97.

The effects of crossover and mutation rates are then evaluated under the condition that maximum number of generation is 300, population size is 3000, maximum tree depth is 20, maximum program size is 20, and using function set CASE C. Figure 9(a) reveals that the final fitness regarding the best-of-generation was as high as 0.9 or more when mutation rate was more than 0.2. Even if mutation rate was lower, the final fitness took high as far as crossover rate is more than 0.4. Figure 9(b) indicates the final average fitness of whole programs was up to 0.9. The larger crossover rate was, the higher the fitness increased.

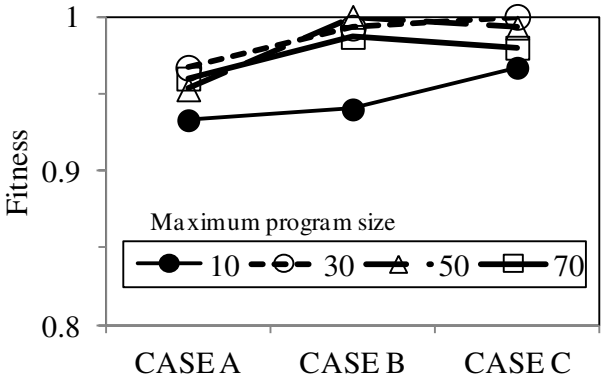

Fig. 8(a) Relationship between final fitness and function set (best of generation)

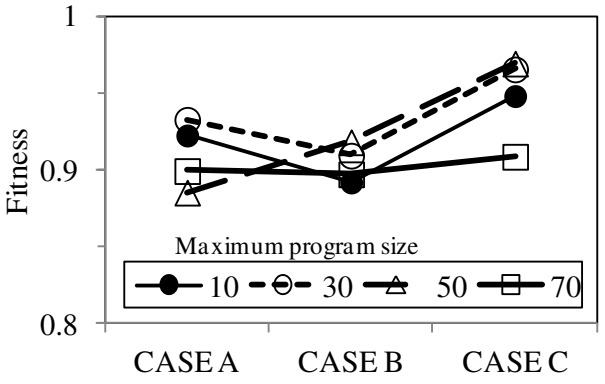

Fig. 8(b) Relationship between final fitness and function set (average)

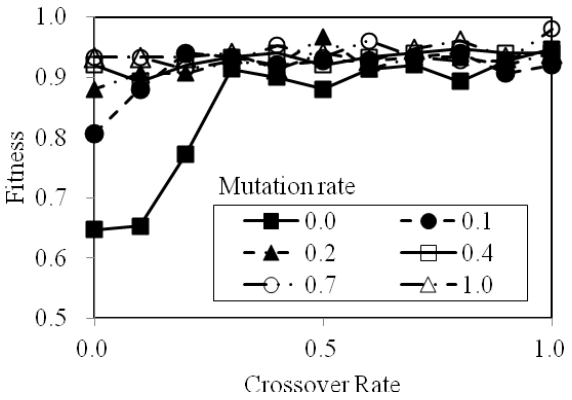

Fig. 9(a) Relationship between final fitness and crossover rate (best of generation)

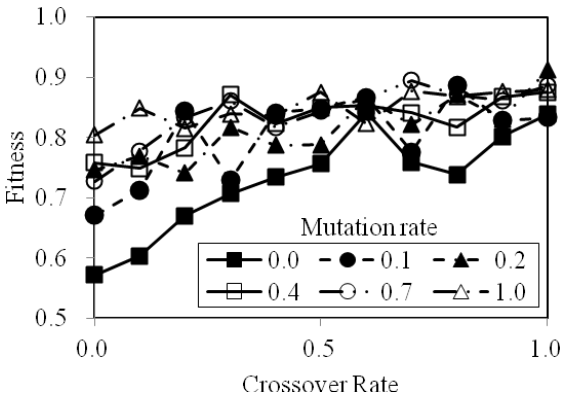

Fig. 9(b) Relationship between final fitness and crossover rate (average)

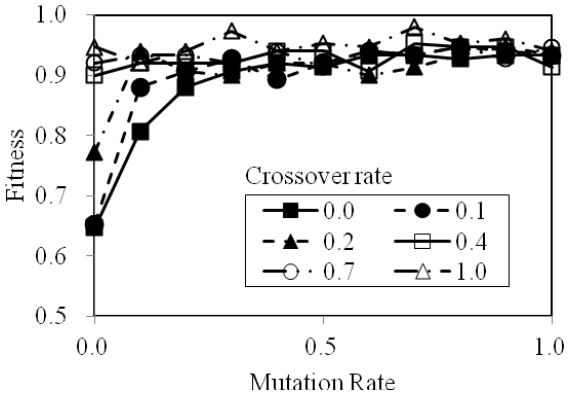

Fig. 10(a) Relationship between final fitness and mutation rate (best of generation)

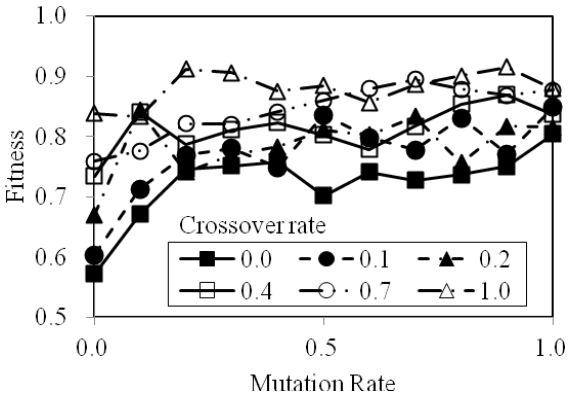

Fig. 10(b) Relationship between final fitness and mutation rate (average)

Figure 10(a) suggests that the final fitness regarding the best-of-generation was around 0.9 independent of crossover rate unless choosing mutation rate smaller than 0.1. Figure 10(b) shows that the final average fitness was not influenced so much by mutation rate but by crossover rate.

The best-of-run program in the 300th generation is indicated as a classification tree in Fig. 11, where population size is 3000, maximum tree depth is 20, maximum program size is 20, crossover rate is 1.0, mutation rate is 0.9, and using function set CASE C.

This optimum program was eventually applied to classify 150 figures. They had separately been prepared for verification. Experimental results proved that the program successfully classified 144 figures into three candidates of shapes. The accuracy rate of the definitive program was consequently estimated at 0.960.

Our previous study [5] tried to classify the figures drawn with the laser pointer by means of a statistical analysis. The same data were used for shape identification as this paper regarding circles, triangles, and squares. They were distinguished based on the histogram pattern of momentary motion vectors of laser spot. Recognition rate of the statistical method was 0.867. Results supports the proposed meta-heuristic method is superior to the conventional one.

```
(if)
   +--(not)
   |   +--(S<=)
   |       +--(-2.44)
   |       +--(slant8)
   +--(if)
   |   +--(not_equal)
   |   |   +--(slant20)
   |   |   +--(-0.87)
   |   +--(1.92)
   |   +--(slant27)
   +--(if)
      +--(S<=)
      |   +--(slant35)
      |   +--(if)
      |       +--(false)
      |       +--(slant4)
      |       +--(if)
      |           +--(C>=)
      |           |   +--(slant50)
      |           |   +--(-2.44)
      |           +--(0.35)
      |           +--(2.97)
      |
[partially omitted]
      |
      +--(0.00)
```

Fig. 11 optimum classification tree based on best-of-run program

IV. Conclusion

This paper proposed the shape identification system of hand written figures. We focused on configurations of edges to distinguish shapes. It detects motion of laser spot by a CCD camera, and the image processing technique extracted optical flow vectors of figures drawn by a laser pointer.

Genetic programming was applied to create shape identification program based on the optical flow vectors. We conducted the experiments of generating classification algorithm of laser trail shape. Effects of genetic programming parameters were investigated to find the optimum conditions.

The definitive program successfully identified the hand written shapes in accuracy of 0.960. Results suggest that the proposed system helps us to control robots with easier and more intuitive ways than ever.

The authors are continuing investigations on the performance of laser trail shape identification by genetic programming, and are verifying the possibility of the proposed method to classify other kinds of shapes among more candidates than three.

References

[1] T. Suzuki, A. Ohya, S. Yuta, “Operation direction to a mobile robot by projection lights”, in Proc. of IEEE Workshop on Advanced Robotics and its Social Impacts, pp. 160-165, 2005.

[2] C. C. Kemp, C. D. Anderson, H. Nguyen, A. J. Trevor, and Z. Xu, “A Point-and-Click Interface for the Real World: Laser Designation of Objects for Mobile Manipulation”, in Proc. of 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 241-248, 2008.

[3] Y. S. Choi, C. D. Anderson, Jonathan D. Glass, and C. C. Kemp, “Laser Pointers and a Touch Screen: Intuitive Interfaces for Autonomous Mobile Manipulation for the Motor Impaired”, in Proc. of the 10th international ACM SIGACCESS conference on Computers and accessibility, pp. 225-232, 2008.

[4] A. Stopp, T. Baldauf, R. Hantsche, S. Horstmann, S. Kristensen, F. Lohnert, C. Priem and B. Ruscher, “The Manufacturing Assistant: Safe, Interactive Teaching of Operation Sequences”, in Proc. of the 11th IEEE Int. Workshop on Robot and Human interactive Communication, pp. 386-391, 2002.

[5] Y. Minato, T. Tsujimura, and K. Izumi, “Sign-at-ease: Robot navigation system operated by connoted shapes drawn with laser beam,” in Proc. of SICE Annual Conference 2011, pp. 2158-2163, 2011.

[6] R. M. Guest, M. C. Fairhurst, “A Clustering Approach to Corner Point Analysis in Hand Drawn Images”, in Proc. of 16th International Conference on Pattern Recognition (ICPR'02), Vol. 3, pp.940-943, 2002.

[7] A. Wolin, B. Eoff, and T. Hammond, “ShortStraw: A Simple and Effective Corner Finder for Polylines”, in Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling , pp. 33–40, 2008.

[8] C. Harris and M. Stephens, “A combined corner and edge detector”, in Proc. of the Fourth Alvey Vision Conference, pp. 147-151, 1988.

[9] John R. Koza, Genetic Programming II, MIT Press, 1992.