Support Algorithms for Eye-Hand Coordination Robotic Therapy

N. Pernalete[1], F. Tang[1], S.M. Chang[1], F.Y. Cheng[1], P. Vetter[1], M. Stegemann[1], and J. Grantner[2]
[1]California State Polytechnic University, Pomona, CA, USA
[2] Western Michigan University, Kalamazoo, MI, USA

Abstract - *This paper presents the support algorithms used for developing a system intended to improve eye-hand coordination in children diagnosed with this problem, using a robotic mapping from a haptic user interface to a virtual environment. A detailed analysis of patters (e.g., labyrinths, letters and angles) created in a virtual environment, as well as the support algorithms (position, force, velocity, inertia and viscosity) developed is presented in this paper. These algorithms are incorporated into the tasks in order to introduce general computer assistance to the mapping of the user's movements to the computer screen. Users' results are evaluated in Accuracy and Time by an edge-detection based evaluation function described in section 3. A fuzzy-logic based Intelligent Decision System takes the data produced by the evaluation function and suggests the most appropriate test to be executed next by the user.*

1 Introduction

HAPTIC devices provide a proprioceptive input allowing the user to perceive movement and location of 3-dimensional space through force feedback. They record and report 3D spatial positions, velocity vectors which features make them highly useful for training and learning purposes. In [1, 3, 6] several different approaches for teaching handwriting have been proposed. The application will help users with disabilities acquiring the writing and drawing skills by using a haptic device. It uses a robotic mapping from a haptic user interface to a virtual environment. For the tasks, the application consists of a force reflecting haptic interface drive, PHANToM Omni with OpenHaptics Toolkit (version 3.0) [7] and OpenGL (Open Graphics Library). The OpenHaptics is patterned after the OpenGL API, making it familiar to graphics programmers and facilitating integration with existing OpenGL applications. OpenHaptics is composed of two layers: a haptic device API (HDAPI) and a haptic library API (HLAPI). HDAPI provides a low-level access to the haptic device. HLAPI provides an advanced haptic rendering capability. HDAPI and HLAPI are built on top of the PHANTOM device drivers (PDD). Eye-hand coordination is the ability of our vision system to coordinate the information received through the eyes to control, guide, and direct the hands in the accomplishment of a given task. Eye-hand coordination uses the eyes to direct attention and the hands to execute a particular task. Most hand movements we perform require visual input to be carried out effectively. The results of a well-designed study [8] reported that using a PHANTOM robot to study eye-hand coordination with force feedback improves the subject's accurate perspective performance by approximately 9% and contact by another 12%. Grip strength measurements are an important method for occupational therapists to learn about the degree of disability of the hand [9]. Intervention for eye-hand coordination and grip strength becomes necessary to successfully perform activities of daily living such as dressing, feeding, drawing and writing.

Project Funded by The National Science Foundation Research to Aid People with Disabilities (NSF-RAPD) AwardID#0828492

2 Background

Extensive research is being done in the field of haptics to improve hand and arm movements. One approach taken was to expose a subject to a perturbing field to develop an internal model of the field as a relation between experienced limb states and forces [10]. In this study [10], they concluded that after-effects persisted after many trials using a force field. Former studies have shown that haptic technologies can be very instrumental in teaching motor skills and manual crafts based on a principle of rehabilitation in which simple movements can be improved by constant practice. Other work by Bardorfer et al. [11] shows some labyrinths or mazes created in the virtual environment, in which the user has to move the pointer (ball) through it and could feel the reactive forces of the walls. In this paper, we discuss the possibility of improving eye-hand coordination in children diagnosed with this problem, using a robotic mapping from a haptic user interface to a virtual environment. The goal is that by improving their coordination and strength, they will increase their participation in handwriting as well as in activities of daily living. The evaluation function was designed and tested first using college-age subjects with no defined eye-hand coordination problems before it was implemented with children. The current results of this function (% accuracy and time) are being used as inputs to the Intelligent Decision Support System (IDSS), which in turn suggests the next task to

be performed by the user. This system has been designed and implemented for testing with a group of children at the Motor Development Clinic at Cal Poly Pomona.

3 Design of Support Algorithms

At first, a test pattern is chosen by the therapist or suggested by the IDSS (Intelligent Decision Support System), and then the application loads it as an image into the haptic workspace. The image properties are shown in Table 1. There are three patterns shown in Fig. 1: Maze, LeLe hand writing pattern, and Complex Labyrinth. The subject is required to move the stylus along and follow the specified trajectory while holding the stylus on the X-Y plane (air). A mapping was also designed for the children to follow the trajectories on a piece of paper (X-Z plane).

TABLE 1. THE IMAGE PROPERTIES

Property	Value
Type	Bitmap Image
Width	512 pixels
Height	327 pixels
Horizontal Resolution	71 dpi
Vertical Resolution	71 dpi
Bit Depth	24

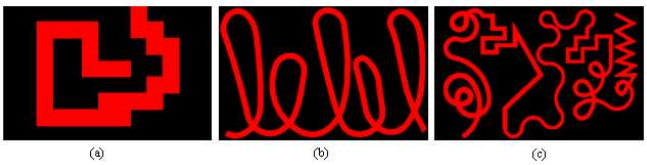


Figure 1. (a) Maze; (b) LeLe hand Writing Pattern; (c) Complex Labyrinth

3.1 Support Algorithms Developed for the Tasks

Various effects are implemented [4, 5, 7] in the form of assistant functions; these are plane constraints, force feedback effect, inertia effect and viscosity effect. Some resemble the handwriting assessments test by using regular pencil and paper.

3.1.1 Regulation of Positions

It is important to know how to haptically render contact with the plane constraints. The plane does not allow passing through such that if the user applies a force against the plane, the plane will give a reverse force and will not allow the user to go through it. The application is built with an X-Y plane workspace and set the Z-coordinate position to zero throughout the task execution in order to keep the slave position (virtual representation of the stylus position) within the workspace. The application created a virtual wall model (see Fig. 2) which is applied to the Z-plane in order to prevent the user from penetrating through the surface. A large coefficient K is used to simulate the force F from the virtual wall, where Zwall represents the point position on the surface corresponding to the real slave position Zp.

$$F = \begin{cases} 0 & -> \ if\ Z_p > Z_{wall} \\ K(Z_{wall} - Z_p) & -> \ if\ Z_p \le Z_{wall} \end{cases} \quad (1)$$

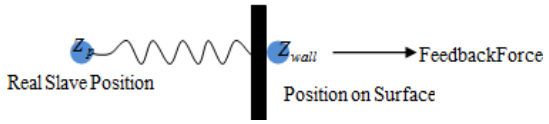


Figure 2. Virtual Wall Model

3.1.2 Force Assistance Functions

This application assists the user by applying the force in the direction of the task to be completed, or attracting the user to the desired path if he/she diverts from it. Force feedback can be used to guide the subjects hand in a predetermined trajectory when he/she is unable to move in response to visual feedback. The force (see Fig. 3) is applied either in the direction of the master (stylus) or opposite to the direction of the master given by:

$$\begin{aligned} Fx &= [\pm Fx,\ 0,\ 0] \quad -> \ For\ assisting\ motion\ in\ X\text{-}plane \\ Fy &= [0,\ \pm Fy,\ 0] \quad -> \ For\ assisting\ motion\ in\ Y\text{-}plane \end{aligned} \quad (2)$$

Where Fx and Fy correspond to the constraint forces exerted in the Master side on X and Y directions. The motion in the Z-plane is constrained at zero, making the force effect always zero in that plane. The force is varied from 0.5 N to 1 N depending on the level of difficulty to be implemented in the execution of the task.

$$\begin{aligned} Fx &= [+Fx,\ 0,\ 0] \quad -> \ if\ X_c < X_1 \\ Fx &= [-Fx,\ 0,\ 0] \quad -> \ if\ X_c > X_2 \\ Fy &= [0,\ +Fy,\ 0] \quad -> \ if\ Y_c < Y_1 \\ Fy &= [0,\ -Fy,\ 0] \quad -> \ if\ Y_c > Y_2 \end{aligned} \quad (3)$$

For the force effect, the system triggers an impulse by commanding a force with a direction and magnitude for a small duration.

3.1.3 Inertia Effect Assistance Function

The inertia effect is mainly used to reduce tremor in the hands of the user having problems with eye-hand coordination and, at the same time, improve their grip strength as this test is conducted both with power and precision grasps. The effect is implemented by increasing the mass, damping coefficient and spring stiffness. The implementation of this function is based on pulling a point mass around by a spring.

3.1.4 Viscosity Effect

The Viscosity effect is developed to provide a smooth execution of the task for the user by eliminating the fast and irregular movements of the master (stylus) that is added by the user. It provides the effect of moving the drive in a denser medium by a small force to the user's hand at the same time, which in turn, activates the hand muscles while executing the task. The application used OpenGL to draw the ideal path with a force to guide the user. A good approximation of the snap distance can be solved by the following force formula:

$$F = K*X, \text{ where}$$
$$F: \text{Force in Newtons (N).}$$
$$K: \text{Stiffness control coefficient (N/mm).} \quad (4)$$
$$X: \text{Displacement (i.e. snap distance).}$$

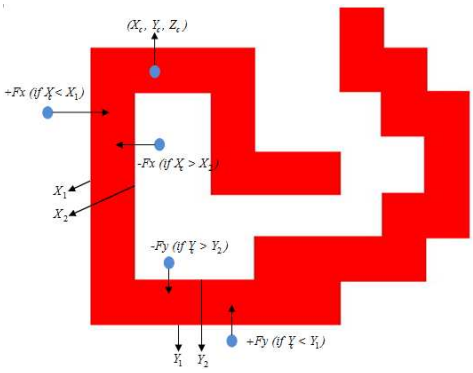


Figure 3. Representation of the Force Assistance Function

The system defines the snap distances to use based on approximating how much force the user needs to exert to pull off from the ideal path. The project is a Win32 console application (see Fig. 4). The data flow of the application is shown in Fig. 5. When the user enters the application, he/she will be asked what test mode he/she would like to run. After loading the test environment, he/she can choose a level for inertia effect, start recording, or quit the application by using the right-click menu.

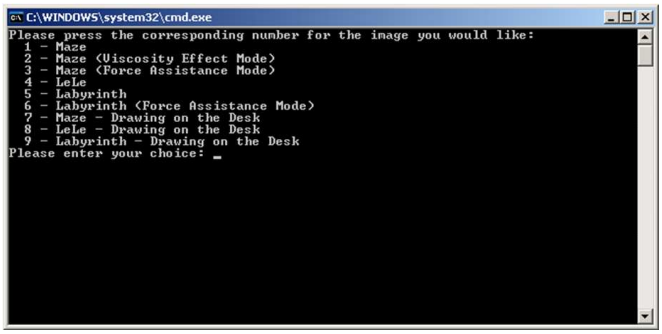


Figure 4. The Menu of the Application

3.2 Description of the Ratio Adjustment

Due to the fact that the units for the evaluation function and haptic device's coordinate system are different from each other, a ratio adjustment is required to get a more accurate evaluating score. Fig. 8 shows the different start point (0, 0) between the evaluation function and the haptic device's coordinate system. Since the evaluation function stores the image pixels in an integer array, there are no negative values for either coordinate. That is not the case with the haptic device's coordinate system, which starts at the center of the image. Since the two coordinate systems cannot be mapped to each other without a unit definition or a ratio, one or several base points are introduced to solve this problem. This point is demonstrated in Fig. 6, which shows a blue base point on the top left hand corner of the image. The red and green points represent different start points (0, 0) for two different coordinate systems (See Fig. 6). The red point is located on the top left hand corner of the image while the green point is located on the center of the image. For demonstration purposes, the ratio between the green and red points' coordinate systems was set to two as shown. Then, the location of the blue point is mapped from one coordinate system (red) to the other (green). In order to map a point from a coordinate system to another, the start point (0, 0) must be identified first. Fig. 6 shows the difference between coordinate systems of red and green points. A blue point is used to demonstrate the mapping process. The first step of the mapping process is to make the start point (0, 0) of the two coordinate systems coincide, as shown in Fig. 7.

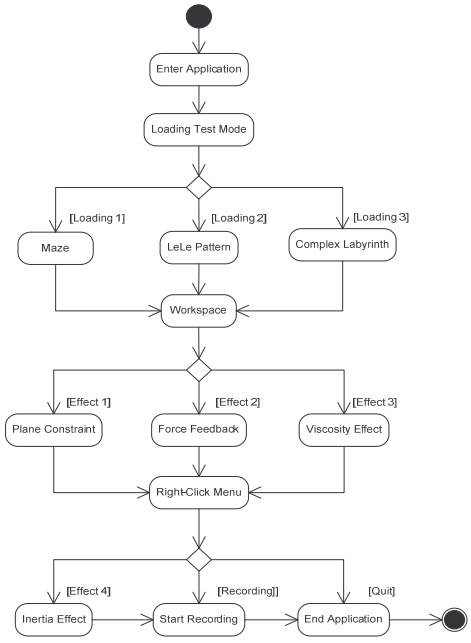


Figure 5. The Data Flow of the Application

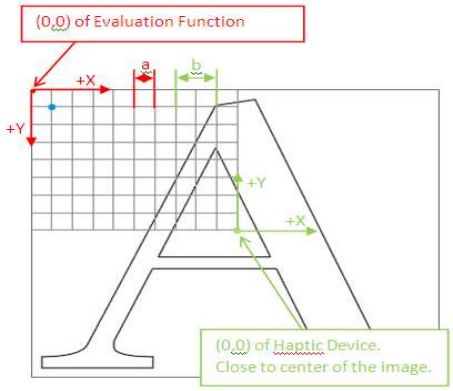


Figure 6. Two Coordinate Systems as Red and Green, Indicating the Evaluation Function and the Haptic Device Respectively. This Figure Shows the Conversion Between them.

Next, the ratio is obtained by given the location of a point in two different coordinate systems. Fig. 7 shows an example, given the locations (-4.5, 3.5), (-9, -7) of the same point in two different coordinate systems. As shown, the ratio is then set to two. The x and y coordinate ratios coincide in this case, but it is possible to set them differently. By using this mapping procedure, the coordinates recorded by the haptic device can be easily mapped into coordinates used in the evaluation function (pixel based) to get a more accurate evaluating score. The following formula is used to calculate the ratio x and y:

$$\text{Ratio } X = |A_{x1} - A_{x2}| / |B_{x1} - B_{x2}| \quad (5)$$
$$\text{Ratio } Y = |A_{y1} - A_{y2}| / |B_{y1} - B_{y2}|$$

Where

A and B represent pixel and haptic device's coordination respectively.

(x_1, y_1) represents the First Point in two coordination system.

(x_2, y_2) represents the Second Point in two coordination system.

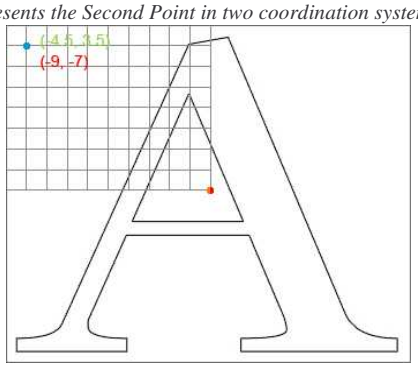


Figure 7. The Start Point (0, 0) of Red Point Is the Same as the Blue One; The Coordinate of the Blue Point in the Coordinate System of Red Point Now is (-9, -7); The Coordinate of the Blue Point in the Coordinate System of the Green Point Now is Still (-4.5, 3.5)

3.3 Description of the Evaluation Function

In order to implement the Intelligent Decision Support System [4], an appropriate evaluation function is designed and tested to determine the accuracy (% of valid points) and time (seconds) of the tasks executed by users. There are three steps taken in the design of this function: First: is to count the percentage of valid points, Second: is to find out how many points are located on the ideal path, and Third: is to measure the completion time based on the average. Appropriate values are carefully chosen for penalties.

3.3.1 Percentage of Valid Points

Since the result is captured by time intervals, the path itself is comprised of several points. Edge detection [2] is used to seek out the valid boundary. Results show in Fig. 8 (Left) and (Right-Red Line). After the boundary is obtained, the percentage of valid points is then calculated.

3.3.2 Number of Points Located on the Ideal Path

The ideal path is determined manually as the input images are complex and irregular in shapes shown in Fig. 8 (Right-Green Line) and (Right-Black Line).

3.3.3 Completion Time

It is calculated based on the sampling time and the number of samples obtained after execution. Penalties for both accuracy and time are carefully chosen after careful testing and discussion. Fig. 8 (Left) shows the original image for which users are to follow the black line. After getting the results from users after executing the haptic tasks, they are simply loaded into the evaluation function. In order to evaluate the results correctly, the image must be of a fixed size. Different image sizes may cause incorrect mapping from the result to the values in the evaluation function. There are two required steps before applying the evaluation function. First, a black-white based image is required in this step. It is used for the program to easily identify the edge. In the user interface, the image can be displayed as colorful as possible.

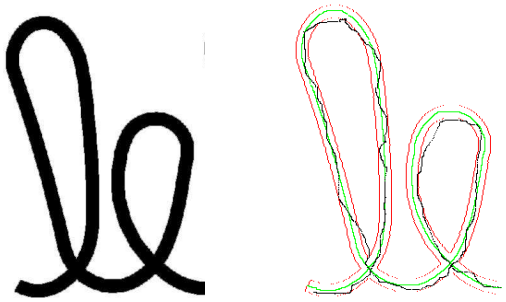


Figure 8. (Left) Original Image; (Right) Image Combinations of Edge (Red Line), Ideal Path (Green Line) and Result (Black Line)

In the "PGM.h", it contains a class "PGM", which is relative to any possible function that used in this program for portable gray map image (refer to PGM image), such as loading the PGM image, detecting the edge, or writing the result to a PGM image. By using this class to load the original image displayed in Fig. 8 (Left) and employing its function "boundaryDetect()",

it is able to generate the edge image shown in Fig. 8 (Right-Red Line). This function also outputs a file whose extension is ".bdy". Notice that the edge file generated by the above function is an edge-like image instead of a real edge image. The reason for not employing the actual edge detection algorithms is the decreasing number of the calculation to know the exactly edge. This function simply eliminates the middle part of a line and leaves the head and tail pixels (see Fig. 9).

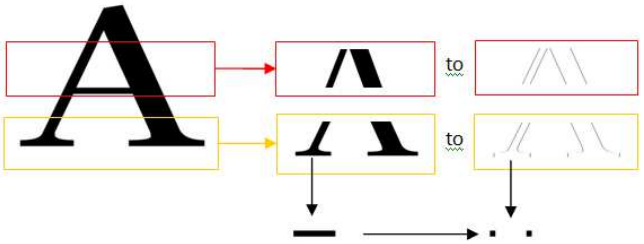
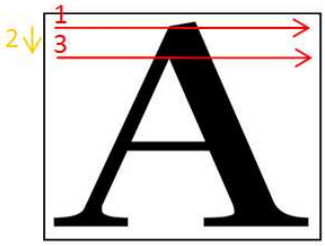


Figure 9. Showing the difference from actual edge

Fig. 10 shows how the function "boundaryDetect()" works. First, it scans the image left to right. Once it reaches the right-most pixel, it will go to the second step. The second step will start at the next line from the left-most pixel and repeat the first step. It continues to do so until the function hits the bottom line's right-most pixel. The output of this function will provide a set of pairs of points to describe a valid area.



1. Do step 1 until hit the right-most pixel for line 1.
2. Go to next line (line 2), and repeat step 1.
3. Do step 1 until hit the right-most pixel for line 1.

Continue to do the above 3 cycles, until hit the bottom line's right-most pixel.

Figure 10. Two steps of "boundaryDetect()"

Second, since the image has an irregular shape, some manual modification is required to generate the ideal path of the image shown in Fig. 8 (Right-Green Line). The image of the ideal path can be imported to the PGM class to create a file whose extension is ".pts". After these two steps are completed, the evaluation function can be applied to the result. The class, "evaluation", is a collection of all necessary evaluation functions. After the edge (bdy file) and ideal path images (pts file) are produced by the class "PGM", the class "evaluation" is ready to be employed. There are four steps needed to be done before evaluating the result in this class. The first three steps are:

1) Loading the boundary file, by using the "loadBdyFile()" function
2) Loading the points file of ideal path, by using the "loadPtsFile()"function, and
3) Loading the results, by using the "loadIdealFile()" function.

After loading these files, the first and second evaluating rules will be applied by the function "startEval()", by using two separate function calls namely "step1()" and "step2()". The function, "step1()", simply seeks these valid points by comparing all points from the results to a valid area described by a ".bdy" file and records these valid points to an array for the step2 function to use. The accuracy percentage is calculated from the number of total valid points divided by the number of total points. A sample result from this function is as follows: *Total Number of Points is 4132; Number of Valid Points is 3648; Accuracy Percent is 88.2865%.* Next, the function, "step2()", uses the valid points from the previous step to determine if a particular point is considered to be close to the ideal path. The tolerance pixel is introduced in this function. Fig. 11 (Left), shows the tolerance pixel, which can be set to 5, 7, or 10 pixels. Fig. 11 (Right) shows how the evaluation function determines if a point is considered to be close to the ideal path. With careful consideration, selecting this tolerance pixel will help determine the user's accuracy in executing the haptic tests. A sample result from this function is as follows (with the tolerance pixel set to 5px).

Total Ideal Valid Points: 2724; Total Valid Points: 3648; Percentage of Points which are considered close to the Ideal Path: 74.6711%

After some testing and analysis, this function will help determine both the user's accuracy and the elapsed time, which will serve as the basis for the Intelligent Decision Support System planned later in this project.

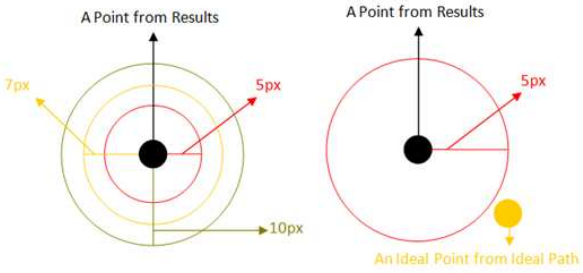


Figure 11. (Left) The tolerance pixels; (Right) The tolerance pixels is 5pixels. If there is an ideal point close to a point from results within 5 pixels, then it consider not close to ideal path.

3.4 Verification of the Evaluation Function

After applying the evaluation function to the users' results, it is necessary to check that the results are not made to cheat on the evaluation function, producing an inaccurate result. For example, in Fig. 12, the evaluation function will come out with a high score that shows the user successfully completed the task and is qualified to proceed to the next level. However, it should not be categorized as high performance, as the user only finished part of the task. The red thick line indicates the valid

area which we assume a user will always try to follow in order to get a high score. The white thin line indicates an example of a user's result that is made to cheat on the evaluation function.

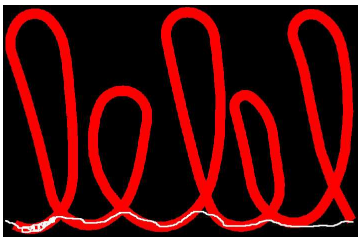


Figure 12. Violation Result

To make sure the ratio adjustment is appropriate for this project; a graphical user interface was developed to measure the transformed results (original coordinates to pixel coordinates). As Fig. 13 shows, the GUI provides four main functions, "Pre Loading", "Start Reading", "Evaluating", and "Exit". The "Pre Loading" function, reads the original image, then the edge file is loaded through the boundary detect algorithm. It also imports the ideal path and stores this data into files for further usage. The "Start Reading" function converts the haptic coordinate to the pixel coordinate, performs the evaluation function, and displays an image for results with the edge file. The "Evaluating" function provides a way to re-evaluate the results while the tolerance range is selected. Each function can run separately, however, the "Start Reading" function will require the data (edge file) from the "Pre Loading" function. The data (edge file) is usually fixed while the program is being tested by the subjects. Therefore, the program can use previous calculated data in the "Start Reading" function. Fig. 14 shows the flow graph of the main program for the evaluation function.

4 Experiments and Analysis of Results

The subjects were instructed to start the tasks at the workspace and simply guide the PHANToM to perform the various tasks to be executed. Position data in X, Y and Z coordinates, and the time and trajectories taken to execute the tasks were recorded for further analysis done by the evaluation function. A group of children (11 in experimental group and 10 in control group) participated in a collaborative research project between the Motor Development Clinic, the Department of Kinesiology and Health Promotion, and the College of Engineering at Cal Poly Pomona. They were given three tests during the first and last week of the Motor Development Clinic session that included: a handwriting test, Developmental Test of Visual Perception and the Motor Free Visual Perceptual Test. Testing and practice using the Robotic Haptic Device occurred before or after the child's motor development session for 30 minutes once a week. The clinic's environment for the pre and post testing was different although it was the same location and test administrators. For the pretest, children were alone in the testing rooms, and for the post testing, the Motor Development Clinic was in session. It is speculated that the additional noise and distractions could account for the lack of increase in scores from pre to post test. However, for the experimental group there was a slight increase in handwriting and over all Visual Motor Integration (see Table 3). Fig. 15 and 16 show the Evaluation Function's results (%Accuracy, and Time in seconds) for LeLe with and without Inertia effect employed. Children executed the tasks manipulating the PHANToM's stylus in the X-Y plane (air) while observing the computer's monitor. The overall observation is that children did improve in both Accuracy and time.

5 Current Work

The simulation of the IDSS described in [5] with the results obtained from experiments is being successfully implemented. The accuracy and time for each trial run was determined using a defuzzification process. Using these values and the current state of the subject, the program decides the next state for the subject's test.

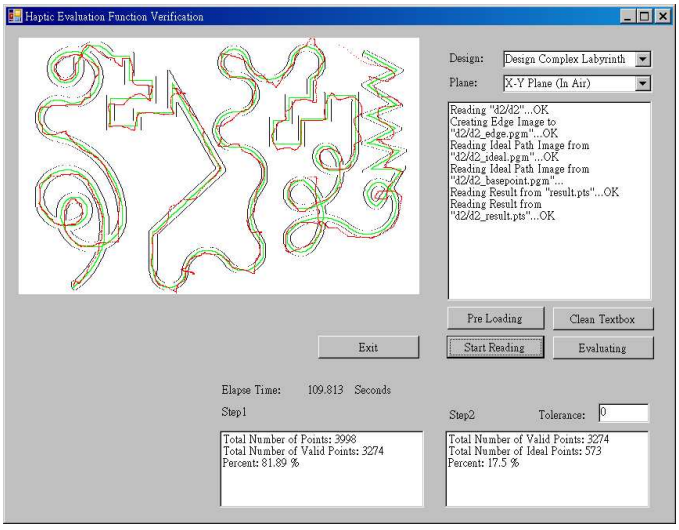


Figure 13. Evaluation Function Result for Complex Labyrinth

6 Conclusions

The project has shown that the use of a robotic haptic interface with assistance functions can help reduce the execution time and increase accuracy for the tests chosen to assess eye-hand coordination problems. The edge-detection based evaluation function proved to be successful in determining these data for the IDSS to make the suggestion on the next task. While the change shown in Table 2 is not significant, it begins to establish the need for more testing and work using the Robotic Haptic Device for children with a disability. The study presented in [4] used only children with identified eye-hand

coordination disability, while this study's children had a number of disabilities. The Cal Poly Motor Development is non-categorical, which means the identification of the disability is not important and all children have some type of motor skills delay. It may be worthwhile in future studies to break the children into categories to determine if there is a difference in the groups.

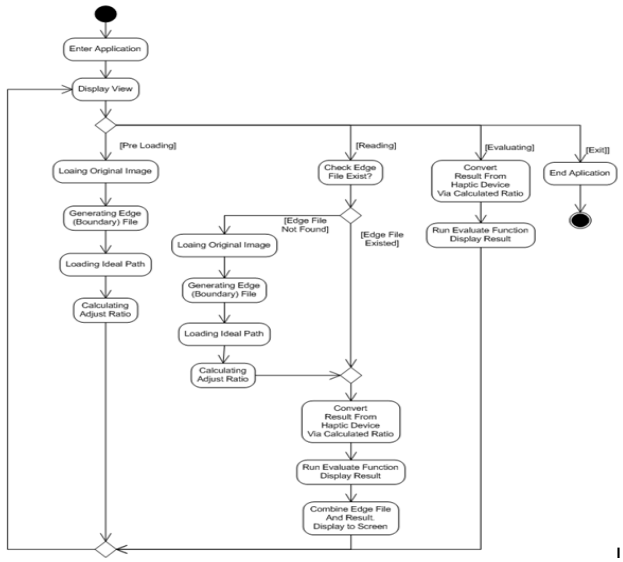


Figure 14. The Flow Graph of the Evaluation Function's Main Program

TABLE 2. THE RESULTS COMPARING THE TWO GROUPS OVER-ALL AVERAGES

Group	Handwriting	Total score of Visual Motor Integration Test	Motor Reduced Performance - subtest	Visual Motor Integration - subtest
Control	No change	-1.7 decrease in performance	No change	-2.4 decrease in performance
Experimental	+.4 increase in performance	+1.7 increase in performance	-2.7 decrease in performance	No change

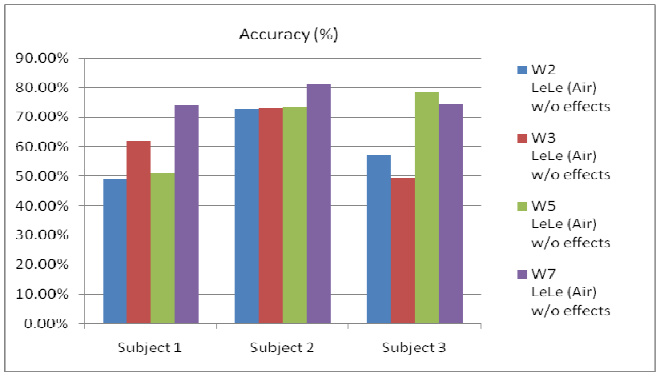


Figure 15. Result for LeLe Pattern without Effects (Accuracy)

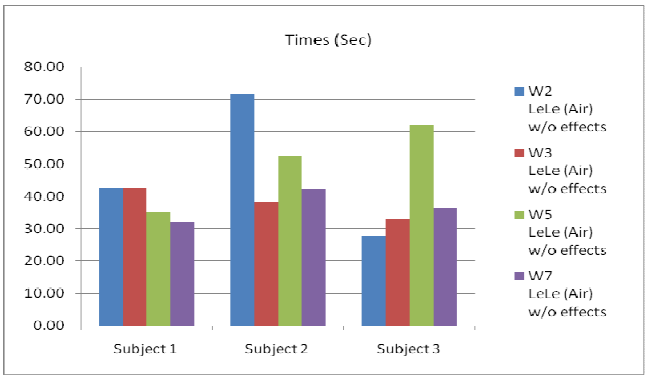


Figure 16. Result for LeLe Pattern without Effects (Time)

7 References

[1] B. Plimmer, A. Crossan, S. A. Brewster, R. Blagojevic, "Multimodal collaborative Handwriting Training for Visually-Impaired People", Chi 2008, Florence, Italy, ACM, pp. 393–402.

[2] J. Canny, "A Computational Approach To Edge Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, 1986, pp. 679–698.

[3] M. A. Eid, M. Mansour, A. El-Saddik, R. Iglesias, "A Haptic Multimedia Handwriting Learning System", ACM Multimedia EMME Workshop 2007, pp.103–108.

[4] N. Pernalete, R.Gottipati, V.Kolipakam, J.Tipple, S.Edwards,R.Dubey, "Eye-Hand Coordination Assessment/Therapy Using a Robotic Haptic Device", The 9th IEEE International Conference on Rehabilitation Robotics ICORR, Chicago, June 2005.

[5] N. Pernalete, R. Gottipati, S. Edwards, D. Janiak, J. Haskin, R.V. Dubey, "Integration of an Intelligent Decision Support System and a Robotic Haptic Device for Eye-Hand Coordination Therapy", The 10th IEEE International Conference on Rehabilitation Robotics ICORR, Netherlands, June 2007.

[6] N. Vishnoi, C. Narber, Z. Duric, N. L. Gerber, "Guiding hand: a teaching tool for handwriting", ICMI 2009, pp. 221–222.

[7] OpenHaptics Toolkit Programmer's Guide.

[8] R. Arsenault, C. Ware. "Eye-hand co-ordination with force feedback", ACM CHI 2000 Proceedings.

[9] S. Edwards, D. Buckland, J. McCoy-Powlen, "Developmental and Functional Hand Grasps", New Jersey, Slack, 2002.

[10] F. Mussa-Ivaldi A., J. Patton L., "Robots can teach people how to move their arm," International Conference on Robotics and Automation. pp. 300-305, 2000.

[11] A. Bardorfer, M. Munih, A. Zupan, A. Primožič, "Upper Limb Motion Analysis Using Haptic Interface," IEEE/ASME Transactions on Mechatronics, vol. 6, no. 3, pp. 3721-3726, September 2007.