Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

A High-Level Controller for Robot-Assisted Rehabilitation

Duygun Erol[1] , Nilanjan Sarkar[1,2], *Senior Member, IEEE,* Bibhrajit Halder[2]

[1]*Department of Electrical Engineering and Computer Science, Vanderbilt University*
[2]*Department of Mechanical Engineering, Vanderbilt University*
Robotics and Autonomous Systems Laboratory

{duygun.erol, nilanjan.sarkar, bibhrajit.halder} @vanderbilt.edu

Abstract - The goal of our research is to develop a high-level controller to provide reference trajectories automatically to the low-level controller of a rehabilitation robotic device. The high-level controller, which is the supervisory controller, is an event driven, asynchronous discrete event system (DES), described by a finite state automaton. Extensive simulations are performed using the supervisory controller for a rehabilitation task. The results demonstrate the feasibility of the proposed supervisory controller.

Index Terms - robot-assisted therapy, high-level controller for rehabilitation therapy, supervisory controller.

I. INTRODUCTION

Stroke is a highly prevalent condition [1], especially among the elderly that results in high costs to the individual and society [2]. In the last few years, robot-assisted rehabilitation for physical rehabilitation of the stroke patients has been an active research area to assist, monitor, and quantify rehabilitation therapies [3]-[5]. The clinical results from the use of these devices have demonstrated that robotic devices can relieve the therapist of tedious and repetitive activities, and can provide a more standardized delivery of therapy with the potential of enhancing quantification of the therapeutic process.

Given the success of the current robot-assisted rehabilitation work that mainly focuses on low-level control action (e.g., position control, force control, or impedance control), it is expected that the efficacy of robotic assistance in more complex tasks will need to be explored in the future. If a task is simple (i.e., it does not involve multiple subtasks that need coordination) then providing a desired or reference task trajectory for the low-level device controller can be accomplished without much difficulty. However, it will be quite difficult to specify a reference trajectory for low-level controllers for an entire complex task. In such a case, the reference trajectory of a specific subtask will likely depend on what happened in the previous subtask. Subsequent subtasks may be influenced by whether the previous subtask has completed as desired or whether any unanticipated event, such as the patient wanted to stop, the robot had a problem, etc., occurred. It is of course possible to define a low-level reference trajectory for each subtask of a complex task considering all possible scenarios and anticipating all possible events. However, such a scheme could be inflexible, difficult to manage, and difficult to extend and adapt to new task requirements. It is likely to be more useful if a new framework can be developed where the low-level reference trajectory generation can be made an automated function of a separate high-level decision-making process. In this manner, a structured design approach can be utilized to generate low-level reference trajectories for a complex task.

We propose a hybrid supervisory control structure to accomplish the following: i) decomposing a complex task in several subtasks, ii) generating low-level trajectory for each subtask, iii) monitoring the execution of each subtask, and iv) analysing the effect of each event and deciding the planning of the following subtask. Hybrid supervisory control structure has been used in industrial robotics, medicine, and manufacturing [6]-[8]. However, there has been no work carried out to our knowledge on systematic designing of a supervisory controller for rehabilitation purposes. The paper is organized as follows: Section II describes the high-level controller (supervisory controller). A detailed computer simulation study of our proposed supervisory control is presented in Section III. Section IV presents the potential contributions of this work.

II. HIGH-LEVEL CONTROLLER

A hybrid supervisory control structure can be conveniently represented in two parts, "Plant" and "Controller" [9] (Fig. 1). The continuous part, identified as the "Plant" is typically described by differential/difference equations. The "Controller" includes a discrete decision process that is typically a discrete-event system (DES) described by a finite state automaton. Together they can be represented by hybrid automata. In this research the plant consists of various reference trajectories for different subtasks. The "Controller" decides the reference trajectory for a subtask based on events. When a complex task is chosen, the "Controller" retrieves the corresponding subtask chain from the task library and initiates sequential execution of the reference trajectories required to complete the selected task. The reference trajectories are given to the low-level device controller during the execution of the task. In our research, the low-level controller is an artificial neural network (ANN) based PI gain scheduling direct force controller that has the ability to provide optimal assistance for a wide range of people. We have presented the simulation [10] and real-time implementation [11] results of the low-level controller on unimpaired participants.

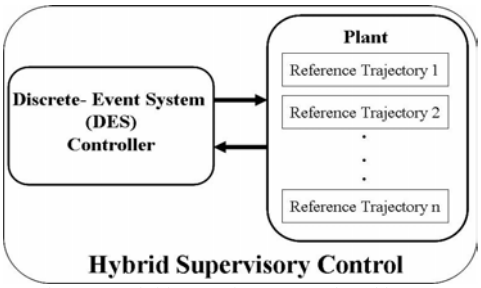

Figure 1: Hybrid Supervisory Control Architecture

The supervisory control structure can be realized using hybrid automata. A hybrid automata is defined as $H = (Q, X, \varepsilon, Init, G)$. Q is a set of discrete variables/states. In our supervisory control structure different reference trajectories represent different states. X is a set of continuous states. This set represents the states of the low-level continuous part of the system by characterizing the status of the patient's arm movement, which can be obtained from the kinematic and dynamic analysis of the analysis of the continuous states of the arm device. ε is a set of events. The examples of events include therapist wants to change the reference trajectory, robot joint angles are out of limit, patient reached the goal position, etc. Init is an initial condition of the system. G is the set of guard conditions. The guard conditions include all the necessary conditions that must be satisfied before execution of the next subtask.

The "Controller" can be realized using a DES. The DES is a nondeterministic finite automaton, $D = (\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$. Here $\tilde{P}$ is the set of discrete states; $\tilde{X}$ is the set of plant symbols generated based on the event; and $\tilde{R}$ is the set of control symbols (symbols generated by the controller). $\psi : \tilde{P} \times \tilde{R} \to 2^{\tilde{P}}$ is the state transition function. For a given DES plant state and a given control symbol, state transition function specifies a set of possible new DES plant states. The output function, $\lambda : \tilde{P} \times \tilde{P} \to 2^{\tilde{X}}$, maps the previous and current states to a set of plant symbols. The state transition function is defined as a mapping from $\tilde{P} \times \tilde{R}$ to the power set of $\tilde{P}$, $2^{\tilde{P}}$, since for a given state and an input symbol the next state is not uniquely defined. The output function is defined similarly. An example is given to explain the use of the "Controller" for our application. Consider a situation where the patient had pain and wanted to stop during the execution of the task. This is considered as an event. This event generates a plant symbol, $\tilde{x}$ for DES. Based on $\tilde{x}$, the DES produces a control symbol, $\tilde{r}$. The state transition function generates a set of possible new DES plant states using $\tilde{x}$ and $\tilde{r}$. One of the possible plant states generated by ψ when the patient had pain is to stop the robot. The output function λ, gives a set of plant symbols based on the previous state and current state.

III. RESULTS

We implemented the supervisory control structure in a computer simulation using the Simulink toolbox of Matlab [12] for a widely used rehabilitation task called incline board. In this task, patients are asked to move to a desired point and come back to their starting point on an incline board. During the execution of this task, unexpected situations may occur such as patient has pain, robotic device joint limits are out of range, and therapist wants to decrease the robotic assistive device speed.

We presented unexpected situations that may occur from the patients, therapists, or robotic device side (Table I) as events in the supervisory control framework. If new unexpected situations occur, which have not been considered in the framework before, then they can be included as a new event to the supervisory controller. Four states are defined in the supervisory controller, *state1*, *state2*, *state3,* and *state4* (Fig. 2). *x* represents the position of the patient. The desired reference trajectories are given to the low-level controller when these states are active. Here the velocity profile is the reference trajectory for the low-level controller. When *state1* is active, which means the patient is performing the task under normal conditions (i.e., no unexpected situation occurred), the desired velocity trajectory is given to the low-level controller. When patient has pain, or therapist thinks it is not safe for a patient to continue the task or robot's joint angles are out of range, *state2* is active and the zero velocity profile is given as the reference trajectory to stop the robot's movement. On the other hand, if therapist wants to decrease the desired velocity because he/she thinks that the patient cannot track the desired velocity trajectory, then *state3* becomes active to provide a reduced velocity trajectory to the low-level controller. The patients are required to move to a goal position and come back to the starting position to complete the entire incline board task. If the patient reaches the goal position (*x*=goal_position), then *state4* is activated. When *state4* is active, negative velocity trajectory is given to the low-level controller to make the patient come back to the starting position.

TABLE I: Events of the Supervisory Controller

Event	Description
E1	Patient do not feel comfortable during the execution of the task
E2	Patient has pain during the execution of the task
E3	Therapist wants to stop the task execution
E4	Therapist wants to decrease the speed of the robot during the execution of the task
E5	Robot's joint angles are out of limit during the execution of the task
E6	Patient wants to continue task execution after he/she wants to stop
E7	Therapist wants to increase the speed of the robot during the execution of the task
E8	Patient continues task execution when the speed of the robot is decreased

The evaluation of the proposed supervisory control framework was demonstrated by three cases. In all three cases, the patient was asked to move to a desired point which was 0.1m on the incline board and then came back to the starting position after he/she reached the desired position.

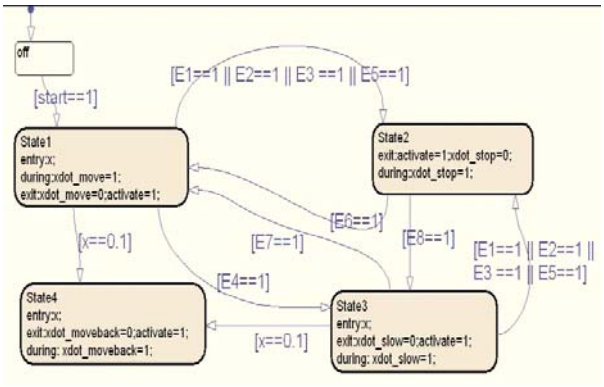

Figure 2: Stateflow Model Developed for Incline Board Task

In the first case, we simulated a situation where the patient did not feel comfortable during the execution of the task, and wanted to stop for a while, and then wanted to continue the task with a reduced velocity. If the unexpected situations were not considered, the velocity trajectory (Fig. 3-dashed line) would be given to the low-level controller, which was not responding to unexpected situations during the execution of the task. Alternatively, the reference trajectories considering all possible scenarios (unexpected situation) could be programmed and stored in a trajectory library. When a task was started, the desired reference trajectory could be retrieved from the trajectory library and executed in a pre-programmed order. As can be seen from the Fig. 3-solid line, each reference trajectory (from A to B, from B to C, etc.), could be programmed and stored in the trajectory library. However, designing all possible reference trajectories, retrieving the necessary reference trajectories, and providing the retrieved reference trajectory at the required time would be difficult to manage. Using our proposed supervisory control architecture, when an unexpected situation (event) occurred, the state transition function generated the possible state, which provided the required reference velocity trajectory to the low-level controller automatically.

When the task started, *state1* was active and the desired velocity trajectory from A to B was given to the low-level controller (Fig. 3). When E1 (patient do not feel comfortable) was triggered, *state2* was activated to stop the robot's movement and the zero velocity low-level trajectory from B to C was given to the low-level controller (Fig. 3) at the time E1 activated. Then, patient wanted to continue task execution with a reduced velocity to complete the rest of the task, and an event E8 was triggered. When E8 was triggered, *state3* became active and the velocity trajectory from C to D was given as a reduced velocity trajectory to the low-level controller (Fig. 3). When patient reached the goal position, which is 0.1m, *state4* is activated to provide the velocity trajectory from D to E to the low-level controller. Fig. 4 shows the position of the patient when each event was triggered and it can be seen from Fig. 4 that the patient completed the entire task.

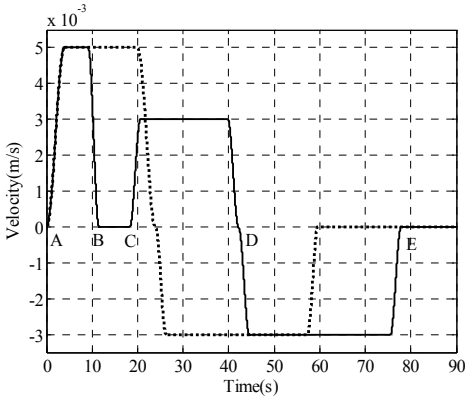

Figure 3: Low-Level Velocity Trajectory for Case 1

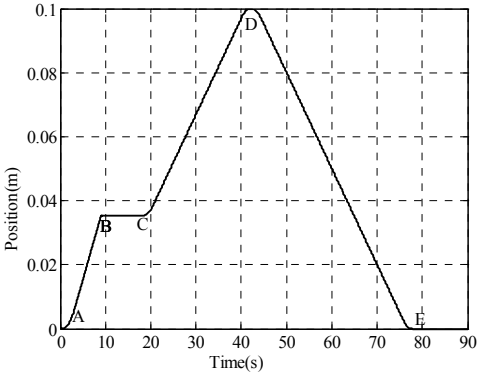

Figure 4: Position Profile for Case 1

In the second case, we simulated a situation where the patient felt pain and did not want to continue the rest of the task (E2 is triggered). When task execution starts *state1* was active. The desired velocity trajectory was given to the controller from A to B. When E2 was triggered *state2* was activated to stop the robot's movement by providing a zero velocity trajectory from B to C to the low-level controller (Fig. 5). It can be observed from Fig. 6 that the patient could not reach the desired position, thus could not complete the entire task.

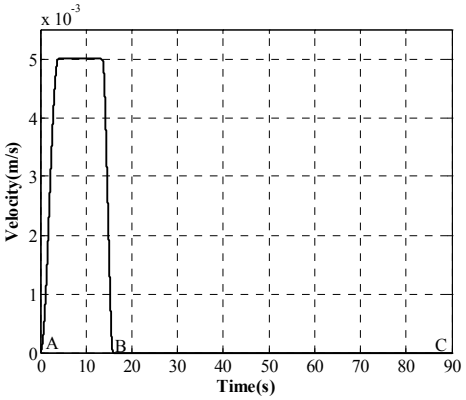

Figure 5: Low-Level Velocity Trajectory for Case 2

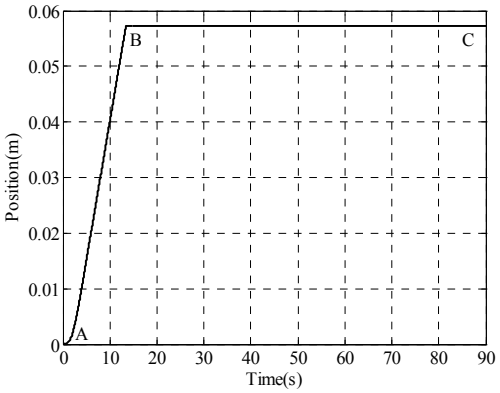

Figure 6: Position Profile for Case 2

In the third case, we simulated a situation where the therapist decreased the speed of the robotic device during the execution of the task because patient could not track the desired velocity (E4 is triggered). When the task started, *state1* was active and the desired velocity trajectory from A to B was given to the low-level controller (Fig. 7). When event E4 was triggered, *state3* was activated and the reduced velocity (0.003m/s) trajectory from B to C was given to the low-level controller (Fig. 7). Patient reached the goal position 0.1m (Fig. 8) and *state4* was activated to provide the velocity trajectory from C to D to the low-level controller to complete the rest of the task.

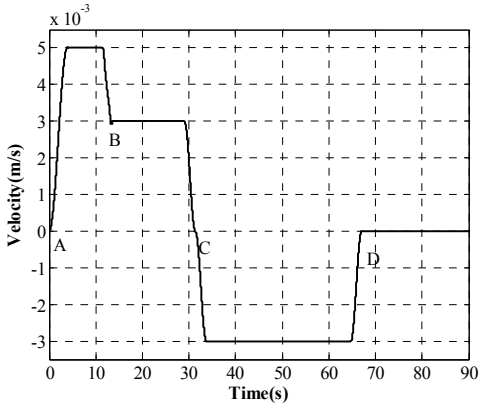

Figure 7: Low-Level Velocity Trajectory for Case 3

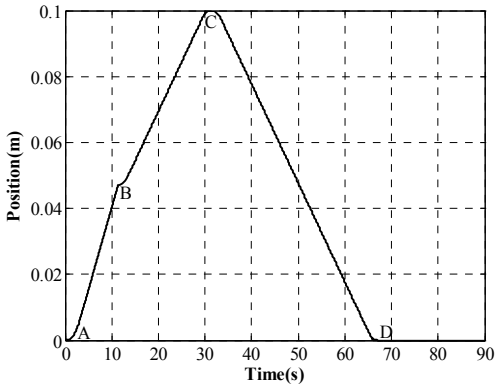

Figure 8: Position Profile for Case 3

IV. CONCLUSION

In this work a high-level (supervisory) controller for robot-assisted rehabilitation has been presented. The proposed supervisory controller generates low-level reference trajectories in an automated manner, considering possible unexpected situations that may occur during the performance of the rehabilitation task. This is the first time to our knowledge that supervisory control theory is implemented for rehabilitation purposes. The supervisory controller provides a framework, which is flexible, manageable, and easily extendable to new task requirements and unexpected situations. We simulated different situations that may occur during the performance of a rehabilitation task in order to present the feasibility of the proposed controller. Further work in this area would involve experimentation of the proposed high-level controller with the low-level controller [10][11] for various rehabilitation tasks in real-time.

V. ACKNOWLEDGMENTS

We gratefully acknowledge the help of Dr. Thomas E. Groomes who is the Medical Director of Spinal Cord and Traumatic Brain Injury Program, and therapist Sheila Davy of Vanderbilt University's Stallworth Rehabilitation Hospital for their feedback during this work.

REFERENCES

[1] American Heart Association: Heart and Stroke Statistical Update, http://www.Americanheart.org/statistics/stroke.htm, 2000.

[2] D.B. Matchar, P.W. Duncan, "Cost of stroke," Stroke Clin Updates 1994; 5, pp. 9-12.

[3] H. I. Krebs, N. Hogan, M.L. Aisen, B.T. Volpe, "Robot–aided neurorehabilitation", IEEE Trans. on Rehab. Eng., 1998, 6,pp.75-87.

[4] P. S. Lum, C. G. Burgar, S. P.C., M. Majmundar, and M. Van der Loos, "Robot-assisted movement training compared with conventional therapy techniques for the rehabilitation of upper limb motor function following stroke.," Arch. Phys. Med. Rehabil., vol. 83, pp. 952-9, 2002.

[5] L. E. Kahn, P. S. Lum, and D. J. Reinkensmeyer, "Selection of robotic therapy algorithms for the upper extremity in chronic stroke: insights from MIME and ARM Guide results," Proceedings of the 8th ICORR, Kaist, Daejeon, Republic of Korea, pp. 208-10, 2003.

[6] H.I. Suh ,H.J. Yeo, J.H. Kim, J.S. Ryoo, S.R. Oh, C.W. Lee, B.H. Lee, "Design of a supervisory control system for multiple robotic systems", Proc.of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp: 332 – 339, 1996.

[7] R.A. Williams, B. Benhabib, K.C. Smith, "A hybrid supervisory control system for flexible manufacturing workcells", Proc. of IEEE ICRA,pp: 2551 – 2556, 1994.

[8] S. C. Lauzon, A. K. L. Ma, J. K. Mills, B. Benhabib, "Implementing a discrete-event-system-based supervisory controller for a flexible manufacturing workcell", IEEE International Conference on Robotics and Automation, pp: 1429 – 1434, 2005.

[9] X.D. Koutsoukos, P.J. Antsaklis, Hybrid Systems Control. ISIS Technical Report ISIS-2001-003 2001.

[10] D. Erol, V. Mallapragada, N. Sarkar, G. Uswatte, E. Taub, "A New Control Approach for Robot Assisted Rehabilitation", ", Proc. of IEEE Int. Conf. on Rehab. Robotics, pp: 323 – 328, 2005.

[11] D. Erol, V. Mallapragada, N. Sarkar, G. Uswatte, E. Taub, "Autonomously Adapting Robotic Assistance for Rehabilitation Therapy", The first IEEE / RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006.

[12] Stateflow, Mathworks Inc, http://www.mathworks.com/products/stateflow/?BB=1.