Blending Human and Robot Inputs for Sliding Scale Autonomy*

Munjal Desai
Computer Science Dept.
University of Massachusetts Lowell
Lowell, MA 01854, USA
mdesai@cs.uml.edu

Holly A. Yanco
Computer Science Dept.
University of Massachusetts Lowell
Lowell, MA 01854, USA
holly@cs.uml.edu

***Abstract* – Most robot systems have discrete autonomy levels, if they possess more than a single autonomy level. A user or the robot may switch between these discrete modes, but the robot can not operate at a level between any two modes. We have developed a sliding scale autonomy system that allows autonomy levels to be created and changed on the fly. This paper discusses the system's architecture and presents the results of experiments with the sliding scale autonomy system.**

Index Terms – Sliding Scale Autonomy, Human-Robot Interaction, Mobile Robots, Mixed Initiative, Adjustable Autonomy.

I. Introduction

The continuum of robot control ranges from teleoperation to full autonomy. The level of human-robot interaction, measured by the amount of intervention required, varies along this spectrum. Constant interaction is required at the teleoperation level, where a person is remotely controlling a robot. Less interaction is required as the robot has greater autonomy. Operating in the space between teleoperation and full autonomy is referred to as shared control. Additional definitions of autonomy can be found in Huang, Messina and Albus [1]. Autonomy can also be measured by the amount that a person can neglect a system [2].

Shared control has traditionally operated at a fixed point, where the predefined robot and operator responsibilities remain the same. However, it is easy to imagine situations where it would be desirable to have a system that could move up or down the autonomy continuum. Human operators may wish to override the robot's decisions, or the robot may need to take over additional control during a loss of communications. Research in this area has been called adjustable autonomy, sliding scale autonomy and mixed initiative. For examples of work in this area, see [3-6].

Most autonomous mobile robot systems have discrete autonomy modes modeled according to their application. However, many occasions require a combination of available autonomy modes, which is not possible. In such situations, sliding scale autonomy can be used to provide intermediate autonomy levels on the fly, thus providing a great deal of flexibility and hence allowing optimum usage of the system.

We define sliding scale autonomy as the ability to create new levels of autonomy between existing, pre-programmed autonomy levels. Others have defined sliding scale autonomy as a system with discrete autonomy modes and the capability to shift between them on the fly [7].

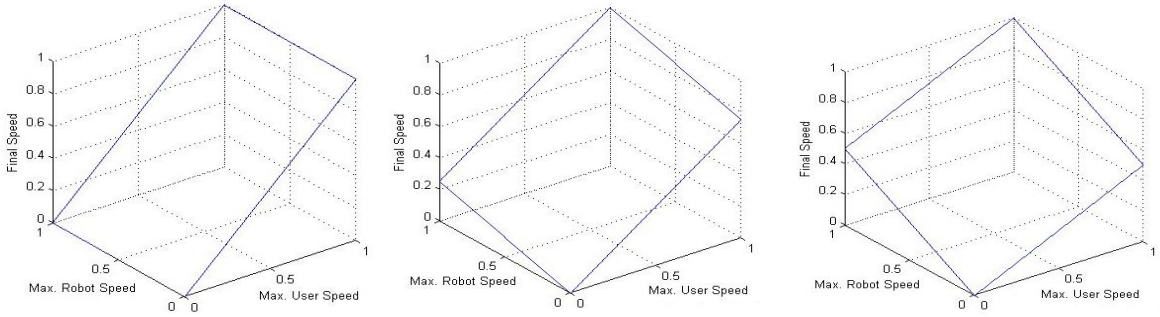
II. Description of the Sliding Scale Autonomy System

Our system was modelled on the INEEL robot control architecture [7], which consists of four discrete autonomy modes:

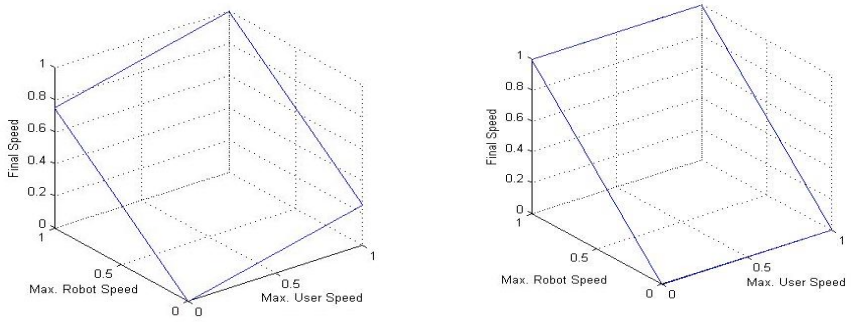
- Teleoperation: In this mode, the user controls the robot directly without any interference from robot autonomy. In this mode, it is possible to drive the robot into obstacles.
- Safe: In this mode, the user still directly controls the robot, but the robot detects obstacles and prevents the user from bumping into them.
- Shared: In this mode, the robot drives itself while avoiding obstacles. The user, however, can influence or decide the robot's travel direction through steering commands.
- Autonomous: The robot is given a goal point to which it then safely navigates.

To create a system with sliding scale autonomy, we identified the characteristics that help define each of these modes. Our system has the ability to change all of the variables for these characteristics on the fly. New autonomy modes are created by blending desired characteristics. If particular settings are determined to be a useful autonomy mode that the operator would like to store for later use, the mode-defining characteristics can be saved in a preset slot for future use.

* This work was supported in part by NSF IIS-0308186, NSF IIS-0415224 and NIST 70NANB3H1116.



(a) 100% User, 0% Robot (b) 75% User, 25% Robot (c) 50% User, 50% Robot



(d) 25% User, 75% Robot (e) 0% User, 100% Robot

Fig. 1: Figures a-e show the varying speed profiles based upon the percentage of speed contribution for the user and the robot. The robot speed and the user speed are combined using the speed contribution to determine the ultimate speed of the robot.

A. System Variables

1) Force Field: There are four force fields, one in each compass direction. These can be independently changed and act like a virtual wall. These values define the distance between the robot and the virtual wall in each of the four directions. Whenever any object comes in contact with a force field, the movement of the robot in that particular direction stops; the robot can still be moved in other directions. The values for force fields range between 0 – 4 robot lengths.

2) User Speed: This defines the maximum speed at which the user can drive the robot. This value ranges from 0 – 1.

3) Robot Speed: This defines the maximum speed that the robot can set. Even though this maximum value is set by the user, the actual speed value is decided by the robot. For example, if the robot is moving in an obstacle filled area, it will keep the actual speed to a low value even if the robot speed value is set to 1. This value ranges from 0 – 1.

4) Speed Contribution: This can be used to switch from the user having full control of the final speed to the robot having full control of the speed or any point in between, without changing the user speed and robot speed values. This can be very helpful when the user wants to be able to quickly transfer/blend control without actually changing the user speed and robot speed values. Fig. 1 shows the speed profiles that result from varying speed contributions.

5) Speed Limiter: Because of inertia and traction, the robot does not always stop dead when its force field touches an object; this is especially true when it is going at high speeds. The speed limiter can be used to control the user's contribution to speed by deciding when to start slowing down and at what rate. The value ranges between 0 – 1.

For example, when the robot is traveling in a narrow hallway with the user speed set to 1 and a speed limiter value greater than 0, if the user commands the robot go forward at full speed, the speed limiter will slow down the robot based upon the current distance to the hallway walls and the force field that has been set.

Similarly, if there is an obstacle in the path of the robot, the robot will start to slow down at a rate dependent on the value of the speed limiter. The robot will then come to a stop when the force field comes in contact with the object.

6) Obstacle Avoidance : This has a value between 5 – 15 robot lengths, with increments of 1. The number indicates the distance of a point on the straight path the robot is currently on that the robot is supposed to reach. Once initiated, the robot automatically drives itself to the specified point. If there are any obstacles encountered, the robot will try to avoid them and recalculate the path to the desired point. Fig. 2 shows the calculation of the robot's path.

If the robot is not able to reach the point due to an excess number of obstacles and crosses the limit (limit

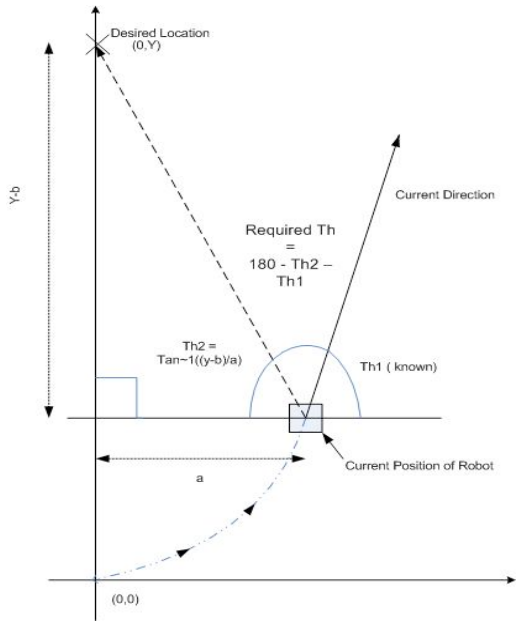


Fig. 2: Calculation for Obstacle Avoidance.

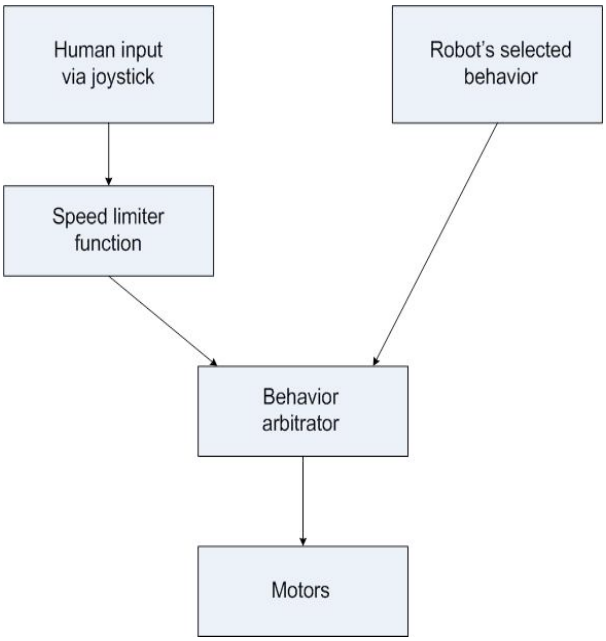


Fig. 3: Information Flow Diagram. The architecture for the sliding scale autonomy system.

= the value of Obstacle Avoidance) in either X-axis or the Y-axis, the robot exits this mode.

In order for the robot to move with an obstacle avoidance setting, the robot speed and speed contribution must be greater than 0. Once the destination point is reached or the limits are crossed, the obstacle avoidance ends, and the robot will revert to driving based upon the current combination of robot and human inputs.

B. *System Description*

The human user provides input to the robot using a GamePad which has six degrees of movement and six buttons on it. The human inputs to the SSA system are translation and rotation speeds, given by moving the game controller. After some initial processing to eliminate noise from hand jitter, the human input is given to the speed limiter function to ensure that the current speed is safe enough for the current environment.

At the same time, the robot determines its desired behavior. The robot's behaviors can be determined using any robot architecture, ranging from reactive control to a hybrid architecture. As with the human input, the robot's selected behavior is expressed as translation and rotation speeds. The robot's speeds are not passed through a speed limiter, as the robot should be programmed to drive at safe speeds using sensor readings to locate nearby obstacles.

The human and robot outputs are then passed to the behavior arbitrator. Using the speed contribution, the final translation and rotation values are computed, taking the force field values into account as well. Thus at this level, the arbitration is between these translation and rotation values and not the arbitration of goals. However, we plan to investigate methods for blending goals in the future.

The system architecture is diagrammed in figure 3.

C. *Presets*

Presets provide a way to store and load system variable values. Once a desired behavior has been found, a user can simply save the current values into a preset and load it at a later point in time using the joystick. Thus, the user is not required to change the system variable values every time.

III. RESULTS

We performed tests to see how the system behaves when there is an object in front of it, by either enabling or disabling the user speed, robot speed, speed limiter and obstacle avoidance variables. Here enabling means setting a value above 0 and disabling means setting the value to 0. The experiments were repeated several times and the diagrams shown in figures 4 through 11 represent the general behavior.

Figure 4 shows the front force field enabled with no contribution to the behavior from the robot. The user can drive the robot at a constant speed until the obstacle is reached.

The experiment in figure 7 was repeated several times by changing the distance between the robot and obstacle, the user speed, the speed limiter and front force field. In each case, the robot would always start to significantly slow down and stop at the specified distance from the object.

Figure Note: In all figures below, the dashed rectangle is the force field and the solid rectangle inside it is the robot. Solid path lines indicate higher speeds. Dashed path lines indicate lower speeds, with the speed decreasing with smaller dashes.

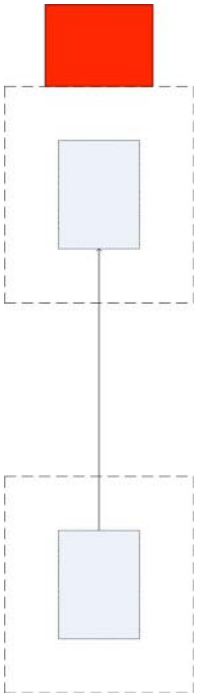


Fig. 4: User speed enabled, robot speed disabled and speed limiter disabled. The robot moves at a constant speed and stops when the force field touches the object.

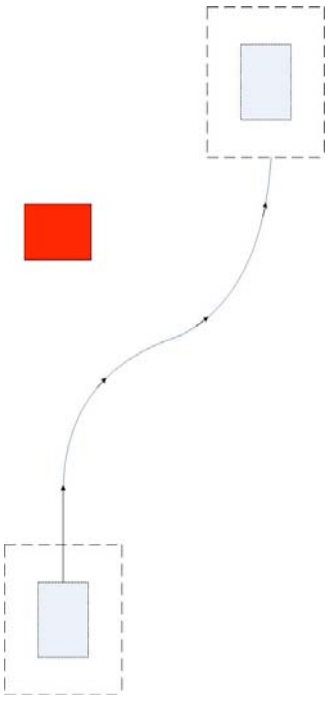


Fig 5: User speed disabled, robot speed enabled and speed limiter disabled. As the obstacle is approaching, the robot turns to open space.

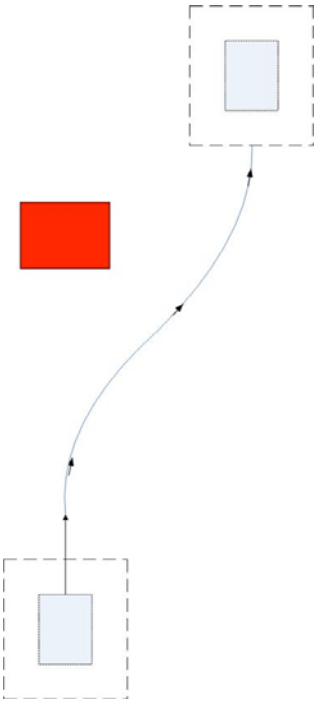


Fig 6: User speed enabled, robot speed enabled and speed limiter disabled. The robot passes closer to the obstacle due to the user's influence

Fig. 7: User speed is enabled, robot speed is disabled and the speed limiter is enabled. When the object comes into the robot's view, the speed limiter function kicks in and starts to decrease the robot's speed.

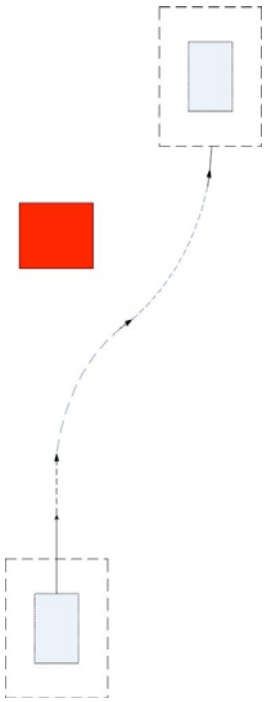


Fig. 8: User speed is enabled, robot speed is enabled and the speed limiter is enabled. The speed limiter kicks in to decrease the user's contribution. Later, the robot starts to turn because of the object in front. As there is nothing in the robot's view during the turn, the speed limiter lets the user's contribution go up.

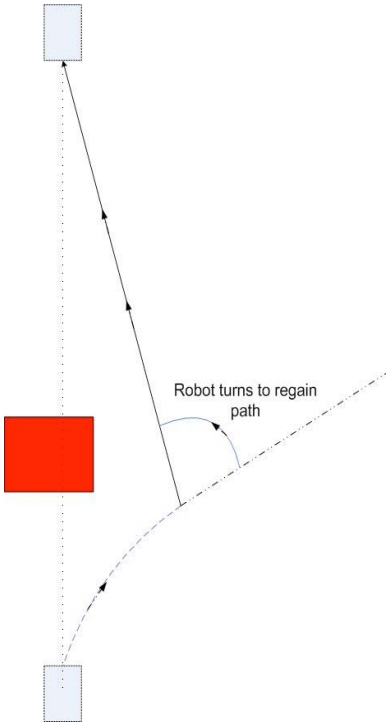


Fig 9: With the obstacle avoidance variable set, the robot steers around an obstacle, regaining the user's forward path once past the obstacle.

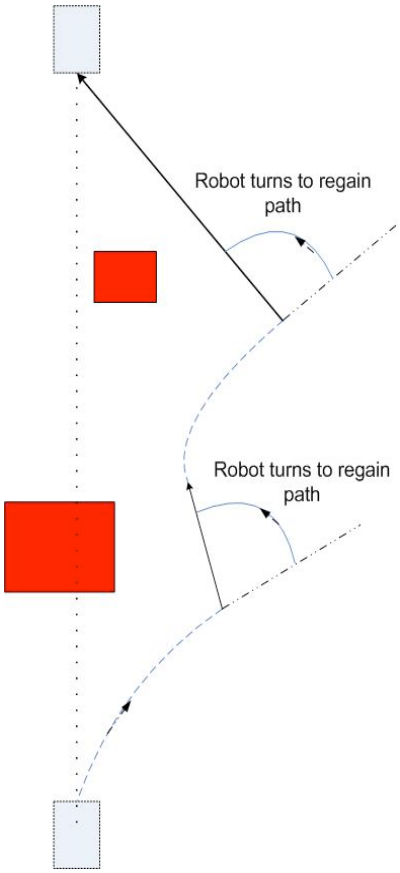


Fig 10: With the obstacle avoidance variable set, the robot steers around two obstacles, regaining the user's forward path once past the obstacles.

The experiments in figures 5, 6 and 8 show that the robot takes a right turn because there was an object on the left side. In figure 5, the robot is driving autonomously with no user input. In figure 6, the human's forward input and robot's obstacle avoidance inputs are blended, causing the robot to drive a bit closer to the obstacle than in the autonomous case in figure 5. Figure 8 also combines the human and robot inputs, but also includes the speed limiter, slowing the robot as it approaches and turns around the obstacle.

Figures 9 and 10 show the use of obstacle avoidance. In figure 9, the robot steers around the single obstacle and brings the robot back to the user's desired path. Figure 10 shows that the robot needs to make two turns to return to the user's desired path.

In all of the experiments, the robot's behavior was to look for open space. We believe that any robot behavior generating rotate and translate would return similar results.

IV. Discussion and Future Work

The sliding scale autonomy system has shown that it has the ability to dynamically combine human and robot inputs, using a small set of variables. These

variables were selected by examining current autonomy levels and determining how they differed from one another. We expect that we will add new variables as the work continues.

While the human now sets all of the variable values, we are investigating how we could allow the robot to change the variables as well, thus creating a system where the robot can also change the autonomy level. We believe that this type of system could be particularly useful when a robot needs assistance. Instead of stopping and requiring user intervention when the robot is unable to determine what to do, the robot could start to shift some autonomy to the user as it begins to recognize that the situation is becoming more difficult. This should prevent the usual problem of a human operator needing to take on full control of the robot in the worst possible situations.

Our investigations will also explore how the system needs to change as the robot's program changes to a hybrid architecture. We will look for ways to combine human and robot goals in addition to translation and rotation speeds.

The system uses a GamePad plugged into the robot's serial port to set system variables and drive the robot. We are currently developing a new interface for the system that uses a wireless GamePad along with a PDA to view and change the system variables. The PDA will also display video.

Improving the methods for the arbitration of user and robot goals will lead to improved human-robot interaction. Our sliding scale autonomy system shows some promising results towards this goal.

ACKNOWLEDGEMENTS

Thanks to Andrew Chanler for helping to interface the robot with the joystick.

REFERENCES

[1] Huang, H.-M., Messina, E., and Albus, J. (2003). "Toward a generic model for autonomy levels for unmanned systems (ALFUS)." *PerMIS 2003*.

[2] Goodrich, M. A., Crandall, J. W., and Stimpson, J. L. (2003). "Neglect tolerant teaming: issues and dilemmas." In *Proceedings of the 2003 AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*.

[3] Kortenkamp, D., Schreckenghost, D., and Martin, C. (2002). "User interaction with multi-robot systems." In *Multi-Robot Systems: From Swarms to Intelligent Automata (Proceedings from the 2002 NRL Workshop on Multi-Robot Systems)*, A. C. Schultz and L. E. Parker, eds. Kluwer Academic Publishers, pp. 213 – 220.

[4] Kortenkamp, D., Keirn-Schreckenghost, D., and Bonasso, R. P. (2000). "Adjustable control autonomy for manned space flight systems." In *Proceedings of the IEEE Aerospace Conference*.

[5] Bruemmer, D.J., J.L. Marble, and D.D. Dudenhoeffer (2002). "Mutual initiative in human-machine teams." *IEEE Conference on Human Factors and Power Plants*, Scottsdale, AZ, September.

[6] Bruemmer, D.J., D.D. Dudenhoeffer, J.L. Marble, M. Anderson, and M. McKay. "Mixed initiative control for remote characterization of hazardous environments for speed contribution." *HICSS 2003*, Waikoloa Village, Hawaii, January 2003.

[7] Bruemmer, D.J., D.D. Dudenhoeffer, and J.L. Marble. "Dynamic autonomy for urban search and rescue." *Proc. 2002 AAAI Mobile Robot Workshop*, Edmonton, Cananda, August 2002.