

Gesture-based Programming for Robotic Arc Welding

Kevin Dixon, Soshi Iba, John Dolan, Pradeep Khosla
Carnegie Mellon University
Dec. 6, 2002

1. Introduction

This report presents the current status of the CMU-ABB gesture-based programming project. Section 2 describes the components of the multi-modal controller (MMC) system, section 3 presents tests performed using the controller, and section 4 presents a summary of accomplishments to date.

2. Multi-modal controller description

During the period October 2000 to November 2002, CMU has developed a multi-modal controller that allows users to direct robotic arc welding using gesture and speech commands. The system has the following major components: Cyberglove for gesture sensing, Polhemus Fastrak 6-DOF sensor for position sensing, Microsoft speech recognition system, a PC or laptop, WebWare for robot-PC communications, and an ABB robot (both the IRB140 and IRB1400 have been tested). Figure 1 gives a system diagram.

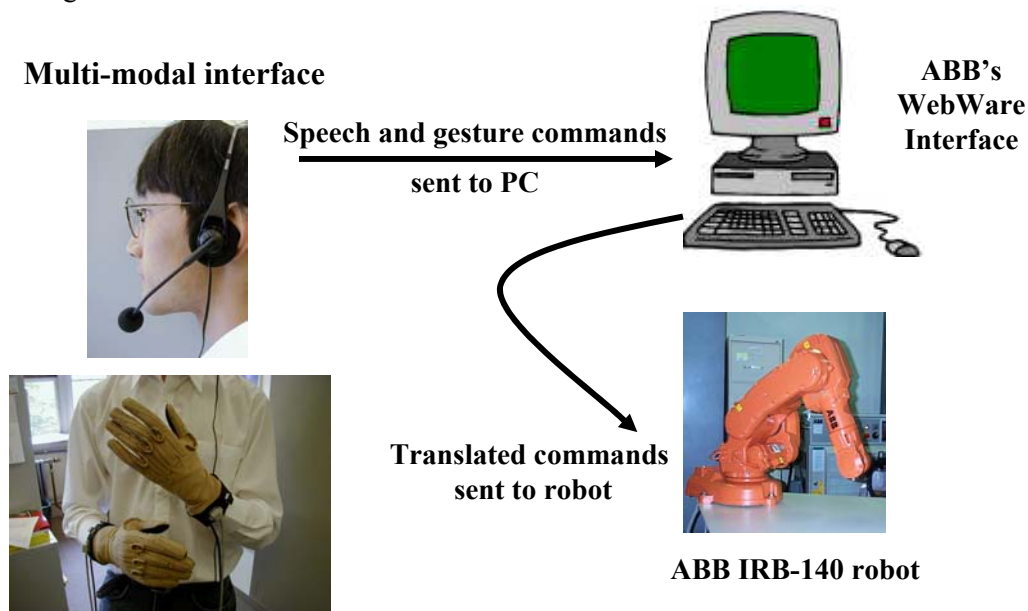


Figure 1. Multi-modal controller system components

The PC or laptop mediates between the multi-modal input devices (glove and speech) and the ABB robot. On the input side, it runs software that translates raw voice and glove inputs into robot controller commands. On the output side, it sends RAPID commands to the robot that contain correct position and orientation information based on the appropriate coordinate transformations.

Sections 2.1-2.3 give an overview of the gesture recognition, speech recognition, and WebWare system components. Section 2.4 summarizes our sensor calibration method for absolute positioning of the robot. Section 2.5 describes the command vocabulary for the multi-modal controller and refers the reader to an appendix containing a detailed description.

2.1 Gesture recognition

One of the key motivations behind this work is to turn a human hand into a multi-functional 3-D parameter-specification device through the use of hand gestures. Hand gestures can potentially be used to specify position, velocity, acceleration, size, direction, angle, angular velocity, etc. Since robotic arc welding is three-dimensional in nature, hand gesture can be an intuitive tool for providing parameters for its programming. For the demonstration, we prepared a set of gestures to specify Cartesian position, direction, velocity, angular rotation, and angular velocity (Table 1).

<i>Gestures</i>	<i>Parameters</i>
<i>Index-Pointing</i> (index finger)	Finger tip position, direction
<i>L-Pointing</i> (index finger & thumb)	Finger tip position, direction
<i>X-Pointing</i> (index finger, middle finger & thumb)	Finger tip position, direction
<i>Waving</i>	Finger tip velocity, Palm direction
<i>Turning</i>	Angular velocity
<i>Grasp</i>	Binary (grasp or not)
<i>Pinch</i>	Binary (pinch or not)
<i>Open</i>	Binary (open or not)

Table 1: Gesture-Parameter relationships

In order to recognize dynamic hand gestures, we used hidden Markov models (HMMs), which are known to work well for temporal and stochastic data. We applied the hidden Markov model toolkit (HTK, by Microsoft) to hand-gesture recognition. Using HTK, which was primarily developed for speech recognition research, we were able to treat hand gestures as words, and a sequence of hand gestures as a sentence. Through gesture sentences, we were able to apply grammatical constraints to gesture recognition (e.g., *Open* comes before and after *Waving*). Prior to the demonstration, three training subjects each spent two hours to record a total of ~4000 executions of gesture sentences (a total of ~19200 gestures) to train the basic gesture models. The large number of training sets was necessary due to variability in gesture execution and the stochastic nature of HMMs.

To achieve user-independent gesture recognition, we used two additional strategies: the use of triphones and adaptation. Phones are primary units of recognition. For example, *z*, *ia*, *r*, *ow* are the phones that describe the word “zero”, and each phone is modeled by individual HMMs. However, such monophonic description does not model transitions between phones. Therefore, triphones are used to model phones including transitions. User adaptation was another important strategy in customizing HMMs to the characteristics of a particular user. The new user is asked to perform few gesture sentences (in our case, we asked the user to execute one sentence twice) to provide supervisory adaptation data to the system. The system then adapts HMMs accordingly by generating model parameter transforms that further reduce modeling errors on given adaptation data.

The above strategies lead to a user-independent gesture recognition system. However, the recognition was not reliable enough on highly dynamic gestures such as *waving* and *turning*, due to the limited amount of training and adaptation data¹. Therefore, those two gestures were not given any functionality in the overall system (Table 2).

¹ Our gesture HMMs are modeled from 3 trainers, and given 10 seconds of adaptation - compared to 630 speakers (TIMIT database), and 15 minutes of adaptation for Microsoft’s Speech API.

<i>Gestures</i>	<i>Words</i>	<i>Phones</i>	<i>Function</i>
<i>Index-Pointing</i>	PTI	pti	Specify Fingertip Position/Direction
<i>L-Pointing</i>	PTL	ptl	Slaving (Position)
<i>X-Pointing</i>	PTX	ptx	Slaving (Rotation)
<i>Waving Forward/Backward</i>	WVF/WVB	wvf1+wvf2 / wvb1 + wvb2	N/A
<i>Turning In/Out</i>	TNI/TNO	tni1+tni2 / tno1 + tno2	N/A
<i>Grasp</i>	GPW	gpw	N/A
<i>Pinch</i>	GPC	gpc	N/A
<i>Open</i>	OPN	gpn	N/A

Table 2: Gesture-Function relationships

Figure 2 shows examples of the three gestures used in the MMC: index-pointing, L-pointing, and X-pointing.



Figure 2. MMC gestures: index-pointing (teach absolute position), L-pointing (position slaving), and X-pointing (orientation slaving)

The gesture recognition system is integrated into the ActiveX control “GestureGlove.ocx”, which provides gesture recognition results and various hand parameters to a parent program (written in VC++ or VBasic) as well as providing a graphical display of the hand.

2.2 Speech recognition

The Microsoft speech recognition system comes free with MS XP and MS Office XP. Once trained with an “accent” model through a roughly 15-minute reading session, the system is highly speaker-independent for speakers with that accent (e.g., American English, Indian English, or Swedish English). Additional speakers merely need to pronounce a 3-sentence sequence in order to adjust the volume of the microphone. In our tests, we used English exclusively, but the product permits recognition of many other languages, including Swedish.

2.3 WebWare interface software

Multi-modal commands and prediction values are the inputs to the system. These inputs are translated into robot movement commands or instructions for the internal program. Ideally, we would issue the movement commands directly to the robot controller (Figure 2). However, there is no facility to issue movement commands directly from an off-board computer.

Consequently, the implementation of the interface is more complex (Figure 3). At the lowest level of the interface, a simple RAPID program moves the robot based on the values of its persistent variables.

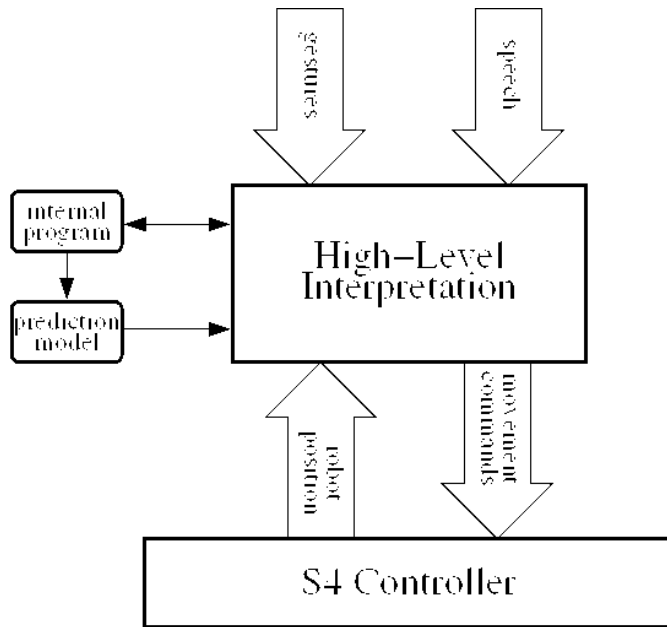


Figure 3. Ideal multi-modal controller interface to robot

This RAPID program is under thirty lines long (almost half of which is devoted to joint-space commands) and is consequently called "skeleton". To affect robot movement, WebWare/RAP allows us to modify the needed variables via a network connection. A custom CMU/ABB API translates movement commands from the high-level multi-modal interpretation module into WebWare instructions to modify RAPID variables.

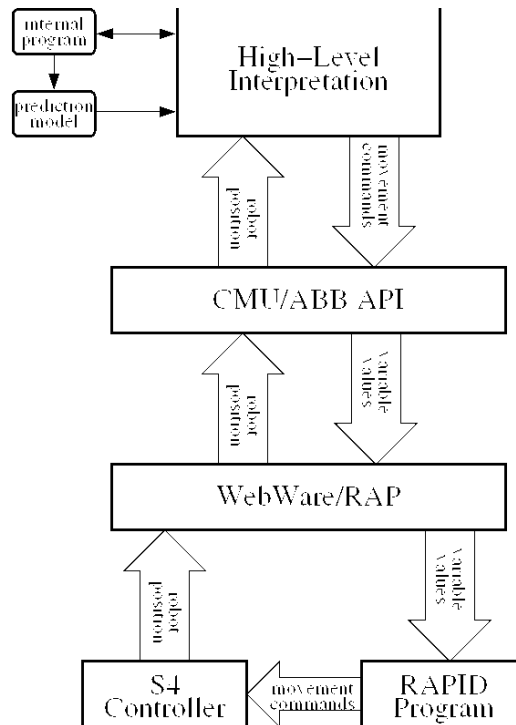


Figure 4. Actual multi-modal controller interface to robot

This circuitous flow of information might be expected to introduce latencies, but the primary source of delay seems to originate inside the robot controller itself. Specifically, there is a buffer of commands internal to the controller that must be filled before the robot will move. This is true for joystick-based commands as well. If this buffer were short-circuited, we believe that the vast majority of delay in the GBP system would be eliminated.

Another problem with the current implementation of the interface relates to the peculiarity of the inverse-kinematic path planning of the ABB controller. Joints 4 and 6 make large motions, sometimes resulting in joint overruns at many *reachable* places in the workspace, well away from an obvious Jacobian singularity. This is the common experience of ABB robot users. This is an ABB-specific problem and one hopes it will be fixed in an upcoming controller update. However, in the current implementation of the interface, this controller problem is aggravated because there is no way to monitor joint values from WebWare. We have created a provisional, hacked solution, but it will be best to solve the problem at the root in the robot controller itself.

Figure 4 shows a screen shot of the GUI developed at CMU to monitor and adjust the WebWare-based interface to the robot controller. It has a variety of options that facilitate testing and calibration.

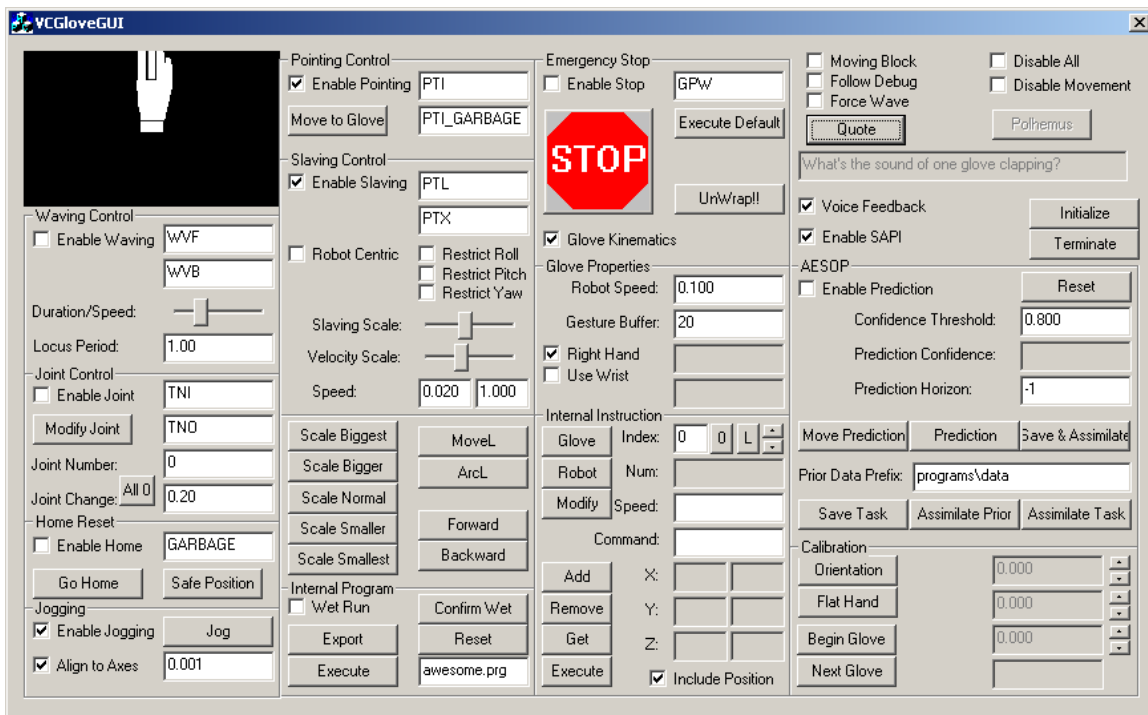


Figure 5. GUI for WebWare interface to robot controller

2.4 Sensor calibration

In order to do absolute positioning of the robot at the glove fingertip position, the Polhemus position sensor and the robot need to know the relationship between their respective coordinate reference frames. In principle, this can be done fairly simply, by attaching the sensor to the robot end-effector, moving it to three known points (e.g. an origin and points along the robot x and y axes with respect to that origin), and finding the corresponding transformation. In practice, since the Polhemus sensor is electromagnetic, metal-mass distortions result in errors that make a single such calibration only locally valid. We therefore use the described calibration method at multiple points in the workspace and

combine these multiple calibrations based on a weighting that is a function of the distances from the currently sensed point to the various calibration origins. The result gives a reduced and more uniform error over the workspace than does a single calibration.

2.5 Command vocabulary

Our design of the command vocabulary was based on the following principles:

- Create commands only for the most common functions (i.e., do not attempt to fully replace the joystick)
- Use the smallest possible number of commands in providing these functions
- Use a flat, rather than a hierarchical, command structure to minimize user cognitive load and needed feedback
- Use gesture only where it is clearly superior to voice → use voice for “executive” functions, gesture for positioning functions

The first and second principles led us to provide only basic motion (joint and Cartesian) and programming (teach, modify, delete, etc.) functions. The third principle caused us to avoid “modes” for the system, in which one would have to “resurface” to a main menu in order to descend a different branch. The final principle resulted in a system that uses only three gestures (pointing, position slaving, and orientation slaving), with one of them (pointing) used for two different purposes: absolute positioning and jogging. Appendix 1 gives a summary command reference, and Appendix 2 gives a detailed description of the command vocabulary.

In our initial demonstration in February 2001, we used a wider range of gestures, including ones for resizing or shifting programs. Such uses of gesture may be appropriate for modification of existing programs either at the welding station or during off-line programming, but they did not seem necessary for a demonstration of a basic arc-welding programming capability. Refer to the multi-modal controller prospects section (section 4) below for additional discussion of the role of gesture in further development of the controller.

3. Multi-modal controller tests

3.1 CMU user study

In January-February 2002, we did a user study to compare the ease of use of the multi-modal controller and the teach pendant. We ran 10 subjects, naïve users with no familiarity with welding or robot control, through the experiment. Each was asked to perform two positioning tasks with each interface, the first consisting of four points on a horizontal surface, the second consisting of four points on a horizontal surface and four points on an adjoining vertical surface. Half the subjects started with the teach pendant interface; the other half started with the CyberGlove interface. The CyberGlove interface allowed only position and orientation slaving, not absolute positioning by pointing. The tasks were timed, and the performances with the two interfaces were compared.

There are several things to note about this experiment:

- The controller was not nearly as stable as it was in the recent November 2002 experiments. In particular, various out-of-bounds and over-acceleration conditions in CyberGlove control caused a reset that took at least 10 seconds to recover from.
- Moving the robot vertically downward from its initial position in position slaving mode caused joint 4 to make large rotations that often caused an “joint out-of-bounds” error and a reset. This appears to be an artifact of the smallness and inverse kinematics of the IRB140.

- Speech recognition was not implemented, so the experiment controller responded to a subject's voice commands by selecting the appropriate mode from a graphical user interface. This undoubtedly slowed subjects down somewhat.
- CyberGlove control of orientation was not correctly implemented and therefore non-intuitive. Some users were able to adapt based on visual feedback, but others could not control orientation well.
- Scaling was not used.
- By coincidence, the three least skillful subjects (4, 6, and 10) all used the CyberGlove interface first. This would be averaged out for a larger number of subjects, but not for a total of five for each interface.

Due to these various factors, the experimental results are not as instructive as they might otherwise be. Nevertheless, a report based on the experiment is attached as Appendix 3. The basic conclusion of the report was that the CyberGlove interface generally allowed faster average times than the teach pendant (on task 2) after acclimatization (in this case, after performing task 1). As stated, the results are not conclusive and are tainted by the above considerations, so a new user study with the improved multi-modal interface would be instructive.

All of the above-mentioned problems had either been corrected or were not an issue (e.g., IRB1400 vs. IRB140 for the second point) by the November 2002 demo in Laxå.

3.2 ABB demonstration and user study

During the week of 11-15 Nov. 2002, Kevin, Soshi, and John tested and demonstrated the multi-modal controller in Laxå. With the help of experienced welders and teach pendant users Mikael Svensson, Peter Quick, Karl Bergsten, and Magnus, we explored the following issues, in accordance with the plan for the week developed by Martin Strand: positioning efficiency, usability, creation of complete programs, arc-welding, and predictive programming. Each of these aspects is documented below.

3.2.1 Positioning efficiency

There were two basic issues here: the rapidity with which users could move the robot to a series of positions and the accuracy of the Polhemus sensor for positioning.



Figure 6. Karl performing positioning tests

3.2.1.1 Positioning tests

We set up an experiment with 9 positions throughout the workspace using the wooden Tinkertoys, shafts and wheels that can be put together in various ways (see Figure 6). We then timed Peter, Karl, and Kevin teaching these 9 positions in three different ways: slaving with the multi-modal controller (MMC), pointing plus slaving with the MMC, and using the teach pendant. In slaving, the robot imitates human hand motion with a scaling factor; in pointing, the robot goes to the absolute position pointed to by the user, with some sensor error. Mikael monitored the positioning accuracy and ensured that it was comparable for all three methods.

	Slave	Point+Slave	Pendant
	(13/14Nov)	(13/14Nov)	(13/14Nov)
Peter	3:35/2:06	3:40/3:27	2:50/2:13
Karl	6:55/3:05	4:15/3:55	3:00
Kevin	3:00	3:20	3:10

Table 3. Times for 9-position Tinkertoy task

We conclude the following from these data:

- The MMC interface is immediately usable. For this simple task, Peter’s MMC times were comparable to his teach pendant times in each case. Karl’s times after familiarization were comparable to his times with the teach pendant. In addition, their times after familiarization were comparable to those of an expert MMC user, Kevin.
- Familiarization improves MMC performance rapidly. Peter and Karl significantly reduced their times in successive runs on 13 and 14 Nov for the slave (by 46% and 55%, respectively) and modestly for point+slave mode (by 6% and 8%, respectively).
- Fine positioning is more time-consuming than gross positioning, at least for this simple task. Point+slave times were greater than slave times in all but Karl’s 13 Nov trial.

3.2.1.2 Sensor calibration and accuracy

We used the calibration method described in section 2.4, teaching an initial frame near the middle of the workspace and then several additional frames at points where errors seemed unacceptably large. In order to gain a measure of the error and to compare single-frame to multiple-frame calibration error, we took a mesh with 10-cm squares of sensor readings at three different z-levels (25, 40, and 60 cm) above the work table. Figure 7 shows the single-frame calibration in the lower plot and the multiple-frame calibration in the upper plot for $z = 40$ cm. Figure 8 shows the same thing for $z = 60$ cm.

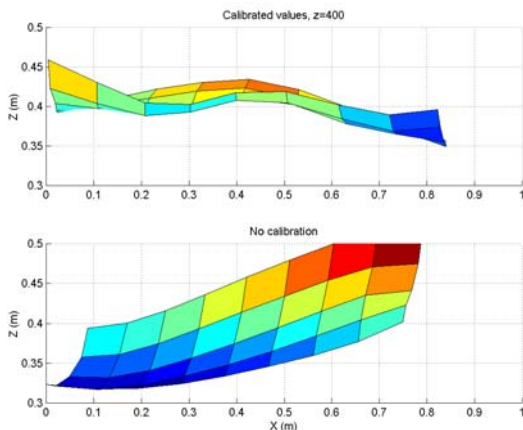


Figure 7. Multiple- (top) and single-frame (bottom) calibration readings for $z = 40$ cm

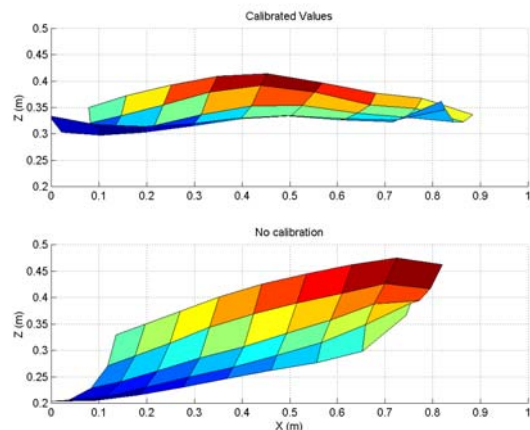


Figure 8. Multiple- (top) and single-frame (bottom) calibration readings for $z = 60$ cm

In each case, the goal of the multiple-frame calibration is to make the mesh plot as close to a horizontal sheet as possible. For $z = 60$ cm, Figure 9 shows the improvement in error resulting from the multiple-frame calibration (average error 49 mm) with respect to the single-frame (average error 78 mm).

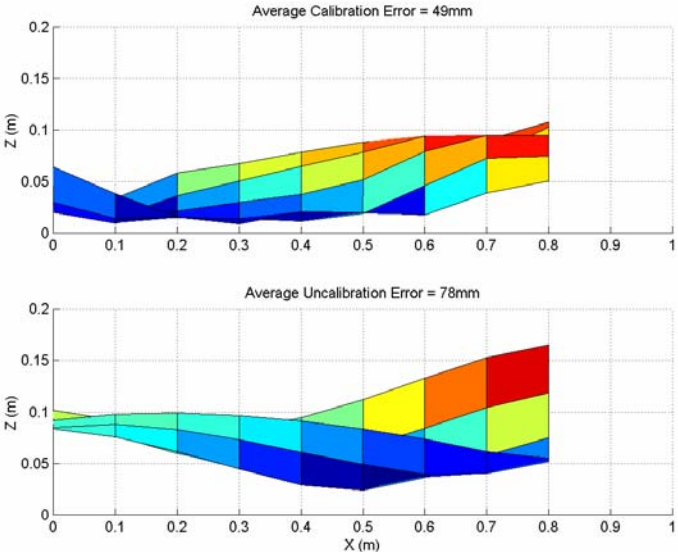


Figure 9. Multiple- (top) and single-frame (bottom) calibration errors for $z = 60$ cm

We also tested the effect of electrical noise from MIG/MAG arc welding at a neighboring workstation on the sensor. Figure 10 shows this effect. The standard deviation of the sensor noise with no arc welding was 0.1 mm, whereas with arc welding on at a nearby station (3m away) it was 1.0 mm (the figure title has these switched). While the effect of the welding is obvious in the plot, the size of the noise is tolerable, rather than catastrophic. We also tested TIG welding, and saw no effect on the sensor.

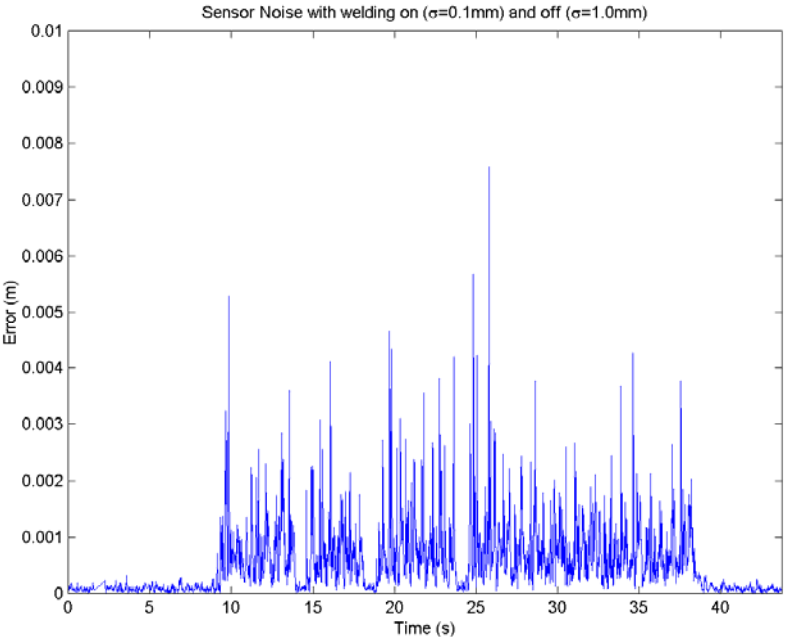


Figure 10. Effect of MIG/MAG arc welding noise on electromagnetic Polhemus sensor

3.2.2 Usability

Based on user tests and comments, the satisfactory and unsatisfactory aspects of the system with respect to usability are listed below.

Satisfactory

- All system users termed the MMC “overall impressive”
- As described and quantified in section 3.2.1.1, the system was immediately and efficiently usable with very little training or description
- Average absolute positioning error ca. 5 cm even with metal mass distortion
- The system’s absolute positioning capability was especially attractive to Mikael, who believed it could save ABB money in large (by physical size and number of welds) robot programming jobs
- Reliable speech recognition
 - 15-minute training for one Swedish speaker of English
 - 92-95% recognition of command vocabulary with 0% false positives for additional Swedish speakers
 - 92-95% recognition rates were achieved in noisy conditions, with the controller welding fan turned on part of the time
- Reliable gesture recognition for multiple users with ca. 3-minute training
- “Hand as tool” provided intuitive, tool-centric orientation

Unsatisfactory

- Speech slowed things down compared to the teach pendant
- Mikael expressed preference in absolute positioning for a swan-neck device imitating the welding pistol and buttons to replace voice commands such as “Teach”
- Slaving was delayed and “nervous” (jerky), at least for near-one-to-one scaling, and made fine positioning difficult (note that this was significantly improved during the week by making the smallest scaling ratio 1[robot]-to-32[user] – see section 3.2.4 below)
- Occasional unwanted robot motion due to a combination of gesture transitions and CPU latencies due to voice feedback – improved during the week by users getting used to starting gestures at stable positions and speeding up the voice feedback
- Occasional robot controller reboot problem that is probably RAP-related

3.2.3 Creation of complete programs

This capability was already tested in the experiment described in section 3.2.1.1 above. Users used the “Forward” and “Backward” commands to traverse and adjust a taught program step-wise, and the “Run” command to execute it in full. Users agreed that the basic capabilities to create a program were already in place. They made the following additional suggestions:

- In instances where multiple program points have been taught with a vertical or other constant orientation, and one of these points has subsequently had its orientation adjusted, allow a “Keep orientation” command in order for subsequent already-taught points to “assume” this orientation without also having to be adjusted
- Add “Straight” command to make tool point straight down in order to achieve a “default” orientation position without the need for explicit adjustment
- Allow jogging in the exact direction of pointing, rather than snapping to one of the world axes
- Add orientation jogging for small angular adjustments
- Include the ability to adjust the external axis
- Make scaling 1-10 instead of the current commands (“Scale bigger”, “Scale smaller”, etc.)

3.2.4 Arc welding

Karl performed multiple welds of two plates in both a “rooftop” and a V shape using the MMC. Due to the small number of points involved, he used position and orientation slaving, rather than absolute pointing, to do this. Over the course of multiple trials, Karl’s ability to position accurately with the MMC was steadily improved by his suggestion of reducing the smallest scale factor until it reached 1(robot)-to-32(user). With this scaling, he was able to achieve welding quality equal to that achievable with the teach pendant. Peter also successfully performed at least one high-quality V-weld with the MMC.

Karl’s times for two consecutive V-welds performed on 14 Nov are given below. The MMC required significantly more time in each case. Several contributing reasons for this are:

- Karl stated that “orientation was easier with the joystick”. This was probably a combination of a lack of familiarity with the hand-as-tool MMC orientation adjustment, the fact that the teach pendant allows one to restrict rotation to a single axis and the MMC currently does not, and the increased ease of doing this when the workpiece is aligned with the robot world frame axes, as it was in this case.
- Karl also said it was difficult to move strictly along the seam axis with the MMC, whereas it was simple with the teach pendant. This is an artifact of the alignment of the workpiece with the robot world frame axes. When this is not the case, the MMC should have an advantage over the teach pendant in moving along arbitrary axis directions.

	Multimodal	Pendant
Karl	3:03/2:25	1:29/1:08

Table 4. Times for V-welding

3.2.5 Predictive programming (learning by observation)

The MMC includes a capability to store information about earlier taught programs, compare the program currently being taught to these programs, and predict which of the earlier taught programs is most like the current one. The predictor is capable of recognizing displaced, rotated, and differently scaled programs, as long as the aspect ratio remains the same. We tested the predictor with a simple rectangular workpiece by teaching the four corner points, then rotating the workpiece in the workspace and beginning to teach them again. After X points, a prediction for the remaining Y points was available. The prediction is sensitive to the accuracy of the orientation of the welding pistol with respect to the workpiece. While the users confirmed that this makes sense from a standpoint of usability, the inability to perfectly achieve the original orientation with respect to the rotated workpiece resulted in prediction errors of 1-2 cm(?).

3.2.6 Evaluation summary

The following are the most salient results from the November 2002 evaluation of the multi-modal controller:

- Positioning efficiency
 - Slaving and pointing are useful control methods not available with the teach pendant
 - Despite metal mass distortion, the Polhemus electromagnetic sensor is usable for absolute positioning, with average error of 5 cm using a simple multi-frame calibration method that can probably be improved upon

- Simple refinements during one week of tests significantly improved control accuracy (e.g. 1-to-32 scaling): further tests with users would likely bring additional improvements, up to a certain point
- Elimination of controller delay would make slaving much faster
- Usability and creating programs
 - Gesture and speech are reliable and usable; however, expert teach pendant users say they prefer a pen plus buttons
 - For simple tasks, the MMC immediately allows expert joystick users (welders) to work almost as quickly as with the teach pendant → Non-expert users (still welders) would probably learn multimodal control more quickly than the teach pendant
 - A large percentage of the most significant teach pendant functions are already present in the multimodal controller
- Arc welding
 - Users could perform high-quality linear welds with the MMC
- Predictive programming
 - Prediction works for a simple example, with errors dependent on accuracy of weld-pistol orientation with respect to workpiece

4. Summary

To date, the ABB-CMU gestural programming project has achieved the following:

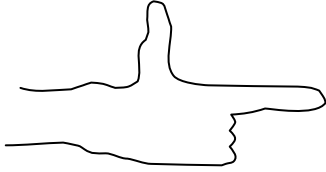
- Created a multi-modal arc-welding robot controller using gesture and speech that is:
 - Easily learned
 - Comparable in time and quality performance to the teach pendant for *expert teach pendant users*
 - Capable of ~5-cm-average-error absolute positioning despite metal mass distortion
 - Robust to a modest (Laxå) level of industrial electrical and audio noise
- Developed a multi-modal speech/gesture vocabulary that:
 - Is minimally complex (sparse and non-hierarchical for ease of memorization)
 - Is capable of a large percentage of essential teach pendant functions
 - Attempts to use gesture and speech for tasks in which they are respectively most effective
- Developed reliable, adaptive gesture-recognition techniques requiring minimal user training
- Developed program predictor that predicts the intended program in the presence of translation, rotation, and scaling

Appendix 1

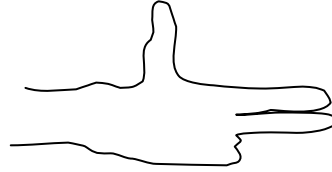
Motion and control commands

“Stop” or “Halt” – emergency stop


“Disable” or “Enable” – disable or enable multimodal control



Position slaving



Orientation slaving

“Jog” with –  move the robot along one of the world frame xyz axes

“Joint [1-6]” – select a joint

“Joint plus” and “Joint minus” - move the selected joint

“Scale Bigger”, “Scale Biggest”, “Scale Smaller”, “Scale Smallest”, or “Scale Normal” – change scaling

“Move” – move to the most recently taught point

“Home” – move to home position

“Safe” – move to safe position

“Unwrap” – unwrap from extreme joint position

Programming commands

“Teach” or “Teach weld” – append current robot position to program

“Teach” or “Teach weld” with  – append pointed-to position to program

“Run” – executes entire program

“First” or “Last” – move to first or last program position

“Forward” or “Backward” – move to next or previous program position

“Modify” – modify the current program position

“Delete” or “Delete all” – delete current program position or all positions

“Insert” – insert the current robot position before the current program position

“Wet” or “Dry” – turning welding on or off

“Current position” – get the current program position

“Number of positions” – get the number of program positions

“Learn program” – adds current program to LBO model/database

“Move prediction” – moves to the next predicted program position

Appendix 2

Multi-modal Controller Command Summary

1 Basic functions

- Motion functions
 - Joint control
 - Cartesian control
 - ◆ Slave mode
 - Position (x, y, and z) in the world frame (with an xyz offset)
 - Orientation (roll, pitch, and yaw) either in a frame local to the hand/welding tool or in the world frame
 - ◆ Jogging: move robot incrementally along x, y, and z axes in the world frame
 - Scaling
- Programming functions
 - Pointing function: teach (append) current hand position (and possibly) orientation in the world frame
 - Teach (append) current robot position
 - Modify, insert, or delete (or delete all) point(s)
 - Program prediction
 - ◆ Add the current program to the prediction database
 - ◆ Move to the next position in the predicted program
 - Test/run/edit program
 - ◆ Run
 - ◆ Wet/Dry (set welding on/off)
 - ◆ First/Last (move to first or last program position)
 - ◆ Forward/Backward (move to next or previous program position)
 - Feedback
 - ◆ Current position (verbal notification of current position index)
 - ◆ Number of positions (verbal notification of current number of program positions)

2 Commands

The following commands are voice, gesture, or combined voice and gesture commands. In order to avoid false triggering by normal speech, all voice commands must be preceded with the word “Command”. Voice commands in the following text are in quotes and omit the word “Command” for brevity’s sake. So, for example, the voice command “Teach” must be uttered by the user as “Command Teach”. We are currently providing various types of audio feedback to note that commands have been received, but this feedback is adjustable based on aggregate user preferences.

2.1 Motion commands

- *“Joint [1-6]”, where the second word specifies the joint number, followed by “Joint plus” or “Joint minus” moves one of the six robot joints.* There is no default joint selected at the beginning of the program, so a “Joint plus” or “Joint minus” command at that point results in a verbal prompt for the desired joint number. The user responds by speaking a number between 1 and 6. These commands substitute for the joint-based joystick mode. They may not be used as much as the Cartesian mode, but are probably needed for completeness, and possibly for fine adjustment and recovery from singularity and other awkward positions.

- *A gesture with index finger and thumb extended slaves the robot to the hand's position.* This substitutes for the Cartesian position joystick mode. The robot moves along the world frame's xyz axes, using the xyz offset that is present each time the position gesture is first used, so that the robot only begins to move when the hand is moved from the position where the gesture is initially presented.
- *A gesture with index finger, middle finger, and thumb extended slaves the robot to the hand's orientation.* This substitutes for the Cartesian orientation joystick mode. The reference frame can be either the world frame or the hand frame presented when the orientation gesture is first used. In the latter case, the extended fingers correspond to the welding tool axis, and the top of the hand corresponds to the plane 1) containing the upper part of the tool and 2) perpendicular to the plane containing the entire, bent tool. With this convention, rocking the hand causes the tool to rotate about its tip in the plane within which it is contained.

The response to gesture in positional and orientational slaving is similar to imitative teleoperation, but for the position gesture, orientation changes in the hand are ignored, and vice versa. When a gesture is being maintained, the robot is slaved either positionally or orientationally to the hand position. When no gesture is maintained, the user can move his hand to a new reference position. This allows the user to reposition his hand and make multiple motions that add up to longer distances or greater angles than can be covered in a single motion.

- *A pointing gesture with the index finger extended coupled with the "Jog" command moves the robot incrementally along one of the world frame xyz axes.* The pointing gesture is resolved into one of the six possible positive and negative world-frame xyz directions, and its magnitude depends on the current scaling. This is probably best for relatively small adjustments to position.
- *"Scale Bigger", "Scale Biggest", "Scale Smaller", "Scale Smallest", and "Scale Normal" scaling commands.* These voice commands scale the joint, slave, and waving motions. A particular scaling command applies to the type of motion command most recently given (joint, slave, or motion), with slave being the default at the beginning of the program. The default for slave motions is 2-to-1, although this is settable. "Scale Bigger" increases the scale factor by some amount, and repeated "Scale Bigger" commands continue increasing the scale factor up to some limit that can be reached directly with the "Scale Biggest" command. "Scale Smaller" and "Scale Smallest" behave similarly by reducing the scale factor. "Scale Normal" returns scaling for all motions to its default.
- *"Move" moves the robot to the most recently taught point in the program.* This command is intended to be used to move a robot into position after a "Teach" or "Teach weld" command has been given (see command descriptions below). The "Teach" command simply appends the hand position to the program, without moving the robot to the point. Therefore, it's possible to use multiple "Teach" commands without the robot moving. The "Teach" command allows one to have the robot follow along to check the "Teach" commands, if desired.
- *"Home" moves the robot to a pre-programmed position above the workpiece area.* This position is a potential initial and final position for any program. It allows the user to remove the robot from the workpiece with a single command, and to have a pre-programmed position that doesn't require detailed multiple motion commands.
- *"Safe" moves the robot to an out-of-the-way position.* For example, this could rotate the robot 90 degrees away from the work area to allow the user to use pointing to teach several points without the robot interfering. All motion commands are disabled by "Safe" in order to prevent the robot from hurting the user. In order to restore responsiveness to motion commands, the user must give the command "Enable" (see section 3.4).
- *"Unwrap" incrementally moves the robot out of an extreme joint position.* This is used when "Move", "Home", "Safe", or other commands are impossible for the controller due to extreme joint positions. After several uses of this command, it should be possible to use other motion commands.

2.2 Programming commands

- *“Teach” or “Teach weld” appends either the current robot position or the current hand pose to the robot program.* If the user is not pointing, these commands append the current robot position to the program. If the user is pointing, these commands append the current hand pose to the robot program. Audio feedback helps the user distinguish between these two cases. To point, the user points to a desired position with his index finger. The current hand pose is then defined as the current position (index-fingertip location) and orientation in world-frame coordinates of the user’s hand. The difference between “Teach” and “Teach weld” is that “Teach” writes a MOVE (move linearly without welding) command to the program, while “Teach weld” writes an ARCL (move linearly while welding) command to the program.
- *“Run” executes the entire program without stopping, starting with the first program position.*
- *“First” moves the robot to the initial program position.* It is useful when one wants to run the entire program, but the robot is somewhere in the middle of the program, and moving stepwise to the beginning of the program would require multiple “Backward” commands (see below).
- *“Last” moves the robot to the final program position.* “Last” moves the robot directly to the final program position.
- *“Forward” moves to the next program position.*
- *“Backward” moves to the previous program position.*
- *“Modify” substitutes the current robot position for the current program position.*
- *“Delete” deletes the current program position and sets the current program position to the next program position.* The exception is when you delete the last program position, in which case the current program position is set to the formerly next-to-last program position, or to null if the last program position is the only program position.
- *“Delete all” deletes all program positions.*
- *“Insert” inserts a point before the current program position.* “Insert” could insert a point before or after the current program position. Because “Teach” appends positions after the final program position, we chose to make “Insert” insert points before the current position, so that it’s possible to use it to insert a point before the initial program position.
- *“Wet” turns welding on; “Dry” turns it off.*
- *“Current position” causes the system to tell the user the index (initial position is 1) of the current program position.* The “current program position” is defined either as the last taught program position or the last position reached by a “Run”, “First”, “Last”, “Forward”, or “Backward” command. The current program position is undefined if the program has zero points. In that case, none of the above commands does anything. “Home” and “Safe” move the robot to non-program positions and therefore do not change the current program position.
- *“Number of positions” causes the system to tell the user the current number of program positions.*
- *“Learn program” adds the current program to the predictive learning-by-observation (LBO) model.*
- *“Move prediction” after the system has said it has a prediction causes the robot to move to the next predicted program point but does not add it to the program.*

2.3 Stop commands

- *“Stop” or “Halt” stops motion of the robot by interrupting the robot controller.* It should be used for emergency stopping.
- *“Disable” and “Enable” respectively deactivate and reactivate the multimodal control program,* so that speech and voice commands will not be recognized. This allows a user to freely speak and move without fear that he will accidentally cause undesirable robot motions.

3 Discussion

The joint control and slaving functions described above are voice- or gesture-based substitutes for the existing teach-pendant joystick controls. They therefore more or less transfer existing functions to an alternative, more intuitive input medium. The pointing function, on the other hand, is a currently unavailable, highly intuitive shorthand for expressing robot position and orientation. The pointing function has a clear advantage in expressive power, but two disadvantages: 1) it is only possible with absolute position sensing and hence relatively high cost; and 2) even then, the accuracy may be insufficient to teach positions non-interactively – for example, a 1-cm error in the z-axis when teaching a point on a plate could mean that the robot tries to punch through the plate. Constant errors like this, resulting in the need to disengage the robot brakes and reset, would limit the usefulness of the system. A possible solution to this potential problem is to record points with a fixed standoff along the tool axis, so that collisions are unlikely. Fine adjustment could then be used to achieve the exact desired position.

4 Scenarios

Since slave mode and pointing mode are two different ways to achieve the same result, we give two scenarios below, the first using slave mode, and the second using pointing mode. In practice, these two modes can be used interchangeably during the same programming process as the user sees fit.

Consider the welding of an L-shaped plate. The plate requires specification of three points that describe the welding seams. The robot's initial position is nowhere near the plate. The user first wants to teach an initial "home" position above the workpiece.

4.1 Scenario 1

The user extends his index finger and thumb and moves his hand in the world-frame x, y, and z directions in order to slave the robot to the home position. He opens his hand, repositions it, and makes the pointing gesture again several times with the default 2-to-1 scaling, but grows tired of multiple small moves, so he gives the command "Scale Bigger", and is able to quickly move the robot into place. He issues the "Teach" command to teach the initial (home) position. (Alternatively, he finds the pre-programmed home position suitable and simply gives the "Home" command.)

The user again makes the index-finger-and-thumb pointing gesture and slaves the tip of the tool to the first weld position on the L-piece, resetting the position reference as needed. The tip is now close to where it needs to be, but the orientation is off, so he makes the two-finger pointing gesture with extended thumb. He then adjusts the orientation by rotating his hand, recognizing that the pitch axis corresponds to the axis about which the assumed crimp in the tool was bent down. In this mode, the tool rotates about its TCP. Scaling can be used here, too, and is probably most useful if the orientation response is too sensitive (note: scaling makes sense for rotation about a single, even a non-principal, axis, but maybe not for compound motion). Once the orientation adjustment is complete, the user opens and relaxes his hand.

Once the desired robot pose is achieved, the user gives the "Teach weld" command. He then follows a similar procedure with the remaining two points on the L-piece: position and orientation gestures to move the robot into position, and "Teach weld" or "Teach" to record the point in the program. The third point, after which welding is complete, should also be stored using the "Teach weld" command. To return to the home position, the user uses the position/orientation gestures and "Teach" once more.

To test the program, the user says "Run", and the robot automatically visits each point in the program without pausing at each position. Everything went by a little fast, so he says "First" and decides to use the "Forward" and "Backward" commands to visit the program positions one at a time. He gives the "Forward" command, and the robot moves to the second program (and first weld) position. He notices that the first weld position is a little too far from the workpiece, so he adjusts it by using slave position/orientation gestures until the robot is in position. He then gives the "Modify" command to substitute the current robot position for the previous first weld position. To continue checking the

program, he says “Forward” again, and the robot moves to the second weld position. Just to make sure that the first weld position is right, he says “Backward” and checks it once more. It looks good, so he issues three more “Forward” commands, checking weld positions two and three and arriving at the end of the program. He decides that this final home position isn’t necessary, after all, so when the robot stops there, he says “Delete” and the point is removed from the program.

To run the final program starting with the initial position, the user again gives the “Run” command.

4.2 Scenario 2

The user positions his hand at the desired home position, with his index finger pointing along the desired tool axis, and gives the “Teach” command. Although it’s not required, he gives the “Move” command in order to move the robot to the last taught (and current program) position, i.e., the home position. (Alternatively, he finds the pre-programmed home position suitable and simply gives the “Home” command.)

He positions (and orients) his hand at the first weld position and gives the “Teach weld” command, then does the same for the second weld position. After positioning his hand at the third position, he gives the “Teach weld” command again. Finally, he puts his hand at the home position and issues another “Teach” command.

To test the program, the user gives the “First” command, and the robot moves to the first (home) position. He then says “Forward” to move to the first weld position. Due to sensor errors, the first weld point is not taught as accurately as it should be, so he adjusts it using slave position/orientation gestures. He then gives the “Modify” command to substitute the current robot position for the previous first weld position. He gives the “Forward” command to proceed to each successive point, making similar adjustments as needed using slaving and the “Modify” command. He decides that the final home-position point isn’t necessary, after all, so when the robot pauses there, he says “Delete” and the point is removed from the program.

To run and check the final program starting with the initial position, the user gives the “Run” command.

Appendix 3

Multimodal Control of Industrial Robots

Anirudh Shah
April 12, 2002

Abstract

This paper describes our current research on understanding how users interact with different interfaces to control industrial robots. In this paper, we introduce the CyberGlove interface, which uses speech and hand-based gestures to program an industrial robot. Our research group is currently developing this interface. In our experiment we propose to compare the ease of learning of the CyberGlove interface to that of the conventional Teach Pendant in controlling industrial robots.

1 Introduction

One of the major challenges faced in programming industrial robots is the steep learning curve faced by the robot programmer. The non-technical background of the programmer makes the learning curve even steeper. The current methodologies that are in use are the culprits for this steep curve. We will discuss one of the more popular of these methodologies (i.e., the Teach Pendant, a handheld device consisting of a joystick and button pad). Due to cost and space constraints, the display tends to be cramped, and the interface cryptic and non-intuitive. Current methods of robot control are highly specified, inflexible, and non-intuitive. As our interaction with robots increases, an intuitive user interface to the robot becomes essential. Hand-gestures provide a concise, rich, and instinctive form of communication. Unlike a joystick or a keyboard, hand gestures provide a more natural interface to the programmer. There is a high level of redundancy that can easily be used to convey geometric and temporal data. These features make it a particularly attractive tool to control a robot

Our research goal is to present a more intuitive robot-programming interface and to demonstrate that our interface is indeed easier to use than the Teach Pendant. This paper gives a brief overview of the underlying technology involved and a detailed analysis of the experiment conducted to support our claim. In section 2, we discuss the state of the art and the underlying technology and give a brief outline of the implementation of the CG interface. In section 3, we introduce the concept of multimodal control of industrial robots and why it is more intuitive and easier to learn than conventional methods. We also give a sketch of our experiment and the assumptions that we made. In section 4 we describe the procedure of the experiment, the different methodologies that we used, and the challenges that we faced in designing the experiment. In section 5, we present the results from our experiment and give a detailed analysis of the results.

2 Underlying Technology

- **The Teach Pendant:** One of the most popular interfaces currently used in industry is the Teach Pendant. It consists of a handheld device with a joystick and a visual interface such as a touch screen or an LCD panel. The joystick controls the primary motion of the robot. It gives the user the ability to move the entire arm from one point to another, as well as individual control over the joints. It also gives control of robot arm orientation.
- **The CyberGlove:** There are three primary components of our interface; the CyberGlove, the speech and gesture recognition engines, and the software interface that glues the different parts together and communicates with the robot controller. Their functionalities can be summarized as follows:
 - The CyberGlove: Made by Immersion 3D Interaction, it is an off-the-shelf product. It has 22 sensors per hand: two flex sensors on each finger, four abduction sensors, thumb

cross-over, palm arch, wrist flexion and abduction sensors. It also has sensors to measure the flexion of the distal joints on the four fingers.

- Gesture recognition: Our team member Soshi Iba developed this engine. It uses Hidden Markov Models to spot and recognize gestures captured by the CyberGlove.
- Speech recognition: This is off-the-shelf software made by Microsoft.

3 Experimental Setup and Procedure

3.1 Experimental Setup

We conducted our experiment on an ABB Robotics IRB 140, a small stationary industrial robot. The standard interface provided was the Teach Pendant. We utilized the Teach Pendant as our basis for comparison with the ease of use of the CyberGlove. Our primary goal was to measure the learning curve involved in acquiring a basic grasp of the interface and the ability to perform rudimentary tasks. Our primary measurement was the number of simple tasks a user was able to complete given a fixed amount of time.

3.2 Procedure and Methodologies

The subjects were given a brief tutorial on the interface. The same amount of time was allotted for each tutorial (Teach Pendant and CyberGlove). The subject was then given three minutes to familiarize him or herself with the interface. The experiment was broken up into two tasks designed to mimic arc welding. The first consisted of a sheet of paper with four distinct targets or checkpoints and a marker attached to the robotic arm. The subject had to use the interface to move the arm and make a mark at each of the checkpoints using the marker. All of the checkpoints were in the same XY plane. This task had to be completed within five minutes. The second task was similar but more time-consuming, this time with eight checkpoints and a time limit of nine minutes. To further enhance this test, the checkpoints were in different XYZ planes. This exposed the subject to the different techniques involved in positioning the robot arm in three-dimensional space, much like what would be found in real welding.

The pool of subjects was divided into two equal groups. Both conducted the experiments using the two interfaces but in reverse order. Group one began the experiments with the Teach Pendant, while group 2 began with the CyberGlove.

3.3 Experimental Procedure Flow:

1. A brief tutorial lasting five minutes is given for the first interface
2. The subject is given three minutes to get accustomed to the interface
3. Perform the two tasks with the first interface
4. Answer a questionnaire about the tasks and the interface
5. A brief tutorial lasting five minutes is given for the second interface.
6. The subject is given three minutes to get accustomed to the second interface
7. Perform the two tasks with the second interface
8. Answer a questionnaire about the tasks and the interface

4 Challenges and Problems

Due to the proprietary nature of the robot controller and the restricted access to the controller's Application Programming Interface, the software interface for the CyberGlove could not command complete control over the robot. One of the largest problems with the interface was over-acceleration and sudden jerky movements of the joints. This often caused the robot joints to go out of bounds when the subject moved his or her hand too quickly or too far. When this occurred, the robot had to be manually reset to its starting position. The Teach Pendant does not have such problems as it has access to the internal controls of the robot that perform bound-checking. These factors have been taken into account in the analysis of the experimental results.

5 Results

The experiment was conducted with ten subjects. Our primary metric was the time consumed per checkpoint. In order to account for the reset time with the Cyber Glove, we subtracted twenty seconds from the overall time each time the arm went out of bounds. We divided the subject pool into two groups: Group1 (ID: 1, 3, 5, 7, 9) performed the experiment using the Teach Pendant first and then the Cyber Glove. Group2 (ID: 2, 4, 6, 8, 10) performed the experiment with the Cyber Glove first and then with the Teach Pendant.

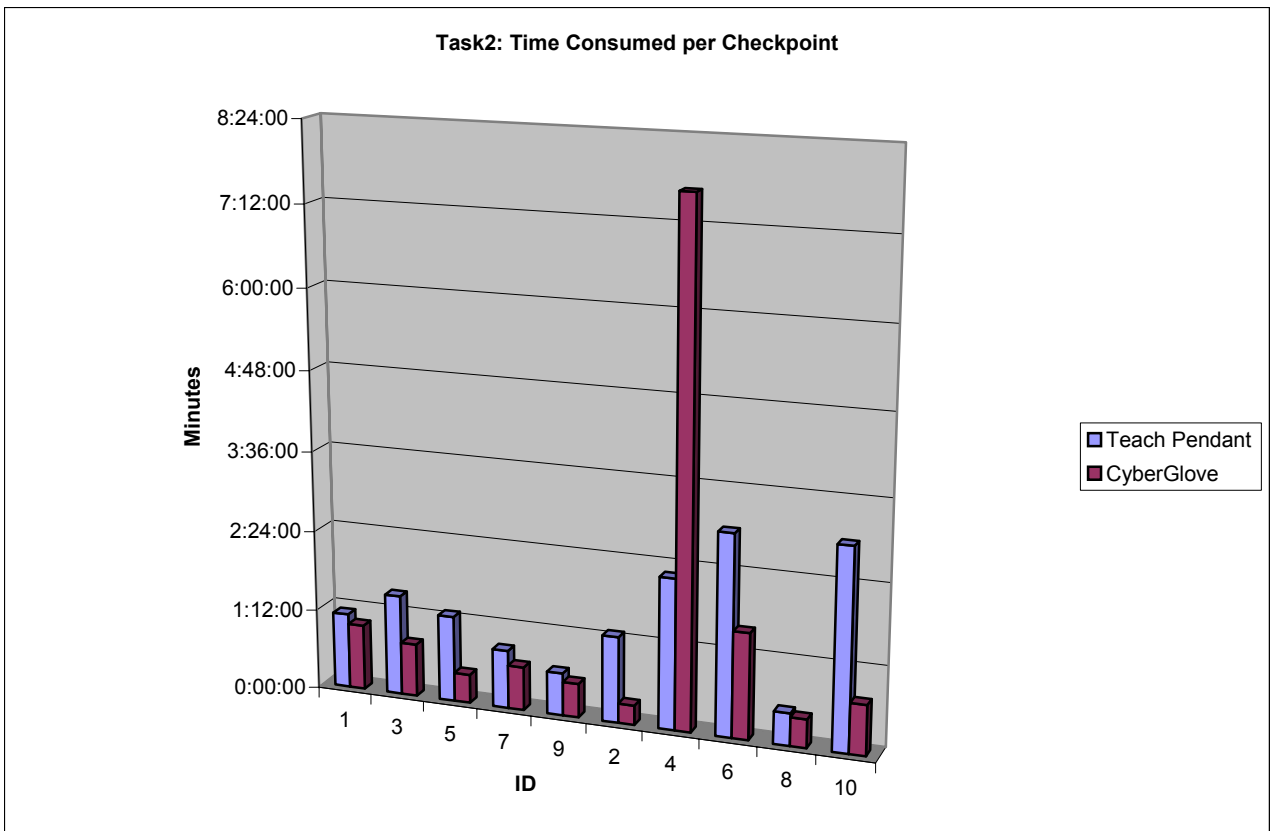
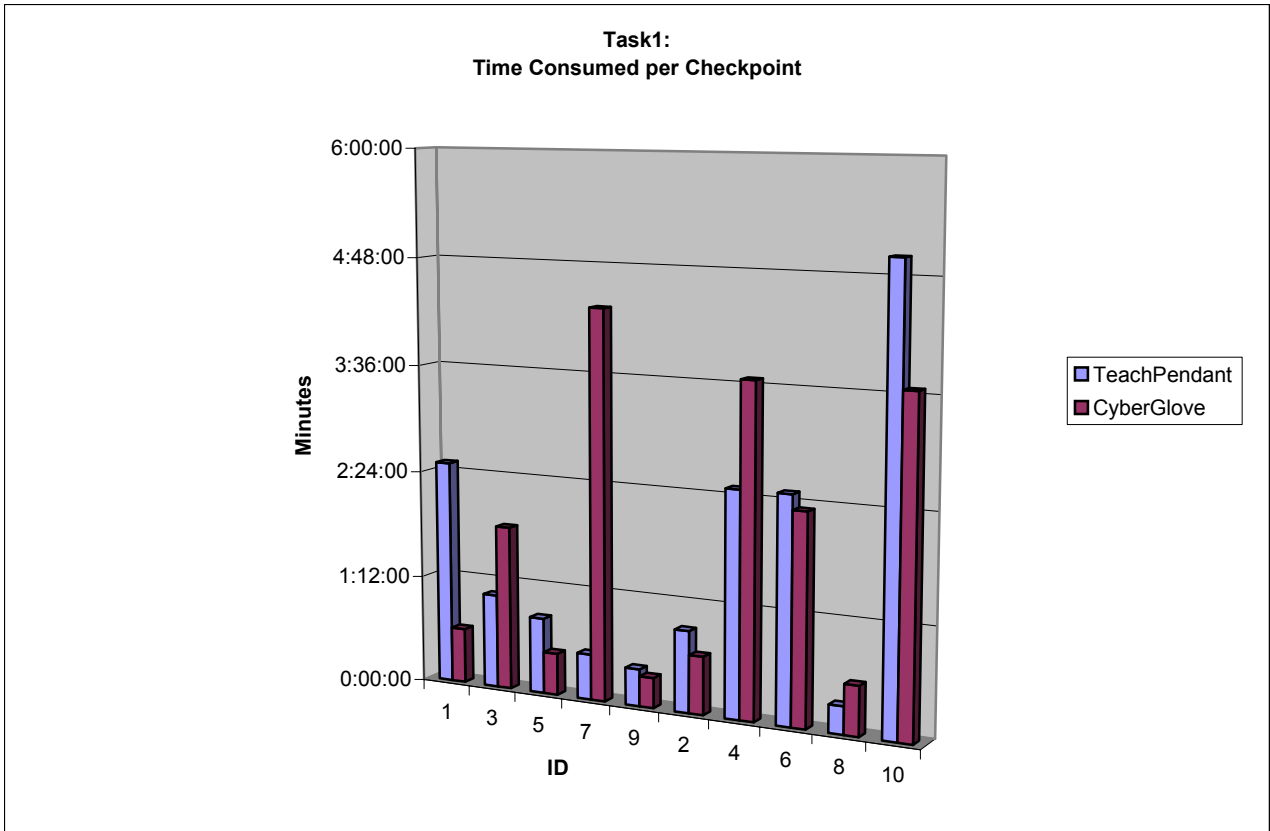


Figure 1 (top) and Figure 2 (bottom)

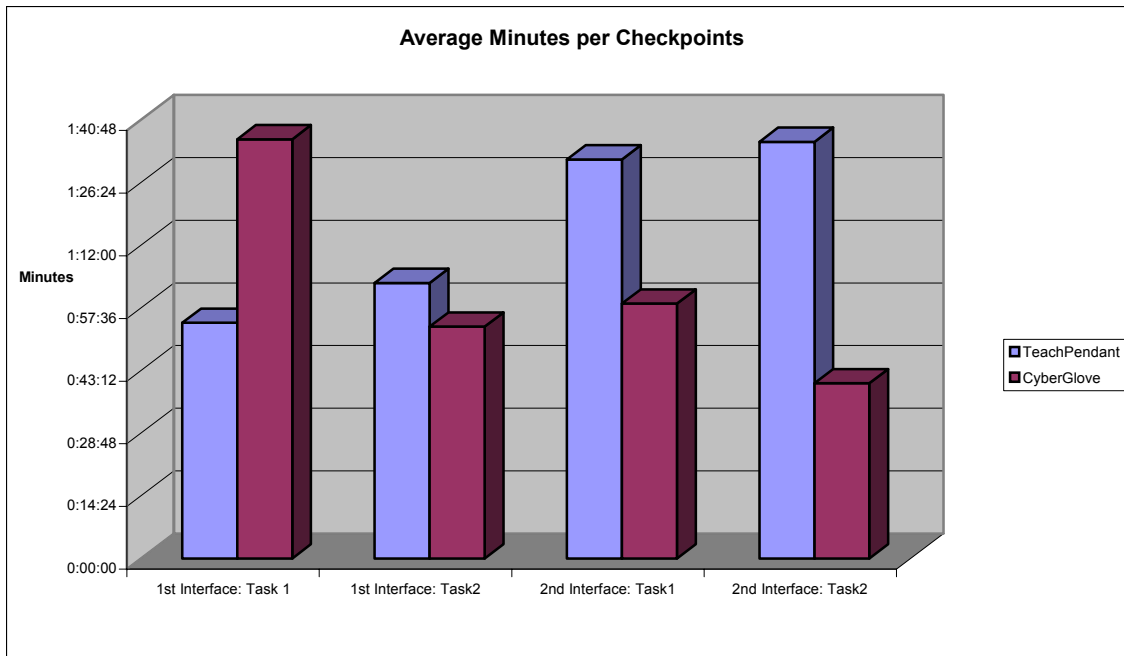


Figure 3: In this chart the 1st column represents the group that completed the respective interface first. This chart does apples to apples comparison.

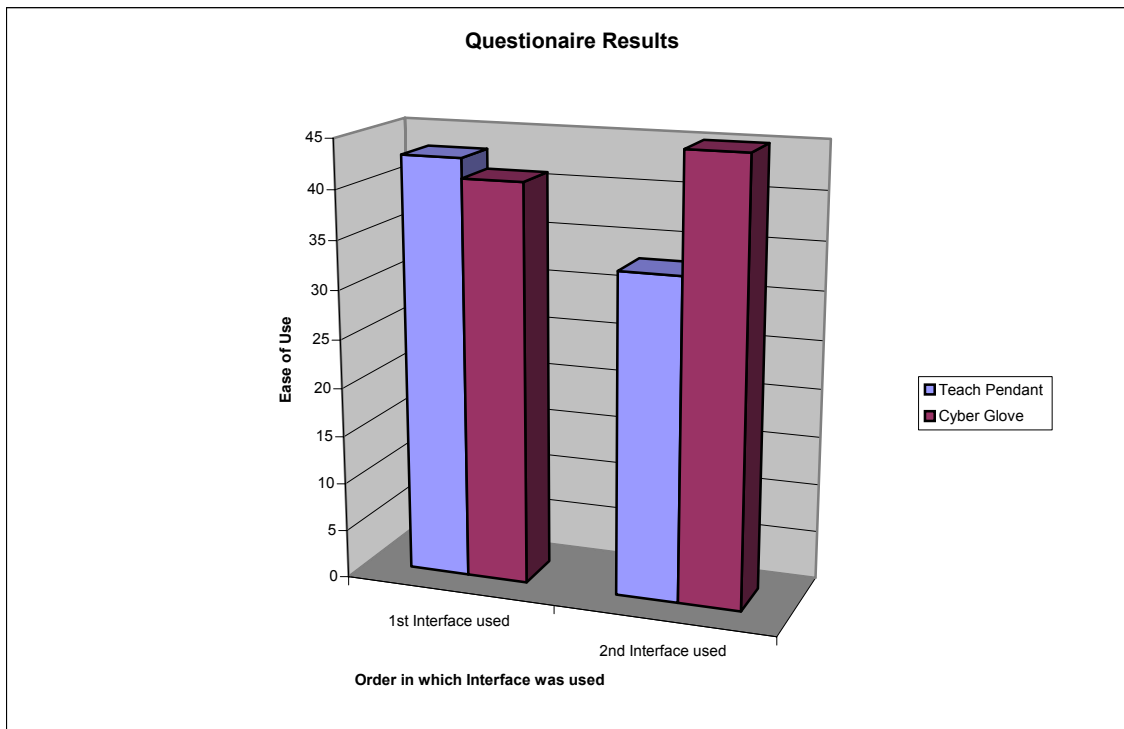


Figure 4

6 Discussion and analysis of the results

The results indicate that the subjects had a hard time with task 1, probably because they had never used such a machine interface before. However, typically their performance on task 2 (either using CyberGlove or Teach Pendant) was better than task 1. One reason for this might be that the subjects acquired a working knowledge of how the robot interacts with the interface. Although the second task adds a third dimension to the experiment, the interface is no longer alien to the user and they slowly develop a basic understanding of the interface. We can observe this trend in the results from Figures 1 and 2.

From Figures 1 and 2 we can also see that Group 1, who started with the Teach Pendant, could easily switch over to the CyberGlove and achieve even better results. One of the implications of this is that the CyberGlove is an interface that is easy to adapt to, and users who are familiar with the Teach Pendant can easily switch over to the CyberGlove. However, the reverse is not true. Once the user gets accustomed to the fluid interface of the CyberGlove, he or she finds it difficult to switch over to the cumbersome Teach Pendant. The fact that the second group had generally better times with the CyberGlove demonstrates this fact.

On the whole, users could in fact achieve better overall times with the CyberGlove. However, due to the technical difficulties that were encountered when controlling with the CyberGlove, the learning curve was slightly higher for the CyberGlove. On further refinement of the interface with the robot, the CyberGlove interface will make an excellent replacement for the Teach Pendant.