

**EVALUATION OF HAPTIC FEEDBACK METHODS FOR
TELEOPERATED EXPLOSIVE ORDNANCE DISPOSAL
ROBOTS**

by

Alex J. Burtness

An essay submitted to The Johns Hopkins University in conformity with the
requirements for the degree of Master of Science.

Baltimore, Maryland

January, 2011

© Alex J. Burtness 2011

All rights reserved

Abstract

This thesis reports on the effects of sensory substitution methods for force feedback during teleoperation of robotic systems used for Explosive Ordnance Disposal (EOD). Existing EOD robotic systems do not feature any type of haptic feedback. It is currently unknown what benefits could be gained by supplying this information to the operator. In order to assess the benefits of additional feedback, a robotic gripper was procured and instrumented in order to display the forces applied by the end effector to an object. In a contact-based event detection task, users were asked to slowly grasp an object as lightly as possible and stop when a grasp was achieved. The users were supplied with video feedback of the gripper and either (1) no haptic feedback, (2) surrogate visual feedback, or (3) surrogate vibrotactile feedback. The force information came exclusively from the current being used to drive the gripper.

Peak grasp forces were measured and compared across conditions. The improvements gained from vibrotactile over no haptic feedback were statistically significant and reduced the threshold at which event detection took place from an average of 8.43 N to an average of 5.97 N. Qualitative information from the users

showed a significant preference for this type of feedback. Vibrotactile feedback was shown to be very useful, while surrogate visual force feedback was not found to be helpful quantitatively nor was it preferred by the users. This feedback information would be inexpensive to implement and could be easily added to existing systems, thereby improving their capabilities to the EOD technician.

Primary Reader: Professor Allison Okamura

Secondary Reader: Dr. Matthew Kozlowski

Acknowledgments

I would first like to thank my advisors, Professor Allison Okamura, Dr. Matthew Kozlowski, and Stuart Harshbarger for their support, patience, and guidance through this process.

I would like to thank the engineers and technicians at Naval Explosive Ordnance Disposal Technology Division (NAVEODTECHDIV). Specifically at TECHDIV, I would like to thank Dr. Kurt Hacker who made the initial arrangements for me to begin this work. Additionally, I would like to thank Byron Brezina for providing valuable information on the development of robotic systems for EOD, and Rob Simmons for information on underwater EOD technologies.

Thanks also to Stephen Phillips for providing resources and contacts, Robert Armiger for supplying access to the Revolutionizing Prosthetics code repository, and John Bartusek for helping to set up the HD-2 arm.

I would also like to recognize the US Naval Academy class of 2010 EOD Officers for their willingness to serve their country in a such an important way during a time of war. I'm finally finished boys. I'll see you in Panama City. Hooyah to LT Eric

Jewell for helping to guide America's next generation of EOD officers.

I would like to thank all the members of the haptics lab who made it such a welcoming place. Special thanks to Amy Blank for helping with statistics. Thanks also to Joe and Kamini for helping to fight the good fight as robotics master's students, and my roommates and lifelong friends Justin Kramer, Eric Wittig, and Mike Head for never once accusing me of "slacking off at Hopkins all day."

I'd like to thank Jessica for her ever present support through three years at the Academy, many late nights of graduate school, and the long separation that we have ahead of us. Finally, I would like to thank my family for always nurturing my curiosity, and putting up with all of the outlandish behavior and experiments that came with it.

However, I save my highest acknowledgments for the men and women in our armed forces who deploy overseas with the mission of rendering safe ordnance and improvised explosive devices. These brave young men and women risk their lives every day in order to protect our soldiers, sailors, marines, and airmen. They deserve nothing but the best research and technology from academia and industry.

Contents

Abstract	ii
Acknowledgments	iv
List of Tables	x
List of Figures	xi
1 Introduction	2
1.1 Motivation	2
1.2 Prior Work	6
1.2.1 Prior Work in Haptic Feedback	7
1.2.2 Prior Work in Explosive Ordnance Disposal Robotics	10
1.3 Thesis Contributions	25
1.4 Organization	25
2 Experimental System	27

2.1	Overview	27
2.2	Input Device	28
2.2.1	Logitech Dual Action Gamepad	30
2.3	Manipulator	32
2.3.1	Three-Jaw Gripper	32
2.3.2	Robotic Arm	39
2.4	Sensors	42
2.4.1	Accelerometer	42
2.5	Feedback Devices	44
2.5.1	Vibrotactor	44
2.5.2	Graphical Feedback System	45
2.6	System Integration	47
2.6.1	Microcontroller	47
2.6.2	Digital Video Camera	49
2.6.3	Force/Torque Sensor	50
2.6.4	Framework and Setup	53
3	Experiment	55
3.1	Preliminary Experiments	55
3.1.1	Accelerometer Test	56
3.1.2	F/T Sensor and Current Sensor Test	57
3.1.3	Control Methodology	59

3.2	Methods	60
3.2.1	Procedure	61
3.3	Results	64
4	Conclusions	68
4.1	Contributions	69
4.2	Future Work	70
4.2.1	Additional Experiments	70
4.2.1.1	Sustained Force Experiment	70
4.2.1.2	Real-World Task	71
4.2.2	Further Areas	71
	Bibliography	73
A	Code	79
A.1	HapGui.m	79
A.2	PositionStep.m	88
A.3	PositionRamp.m	90
A.4	VelocityControl.m	91
A.5	Forcebar.m	92
A.6	CalibrationRun.m	93
A.7	Arduino Code - SerialReadWrite.pde	99

B	Data Sheets	101
B.1	VPM2 Vibrotactor	101
B.2	Arduino Duemilanove	105
B.3	Kistler Piezotron Accelerometer	108
B.4	Smooth-On OOMOO Silicon Rubber	110
	Vita	112

List of Tables

2.1	Three-Jaw Gripper torque/velocity polynomial values for $\theta \in \{0-200\}$	38
2.2	Three-Jaw Gripper torque/velocity polynomial values for $\theta \in \{200-400\}$	39
2.3	Three-Jaw Gripper - torque/velocity identification raw data	40
2.4	Three-Jaw Gripper - torque/velocity identification filtered data with polynomial fit curves	41
2.5	Sensing range and resolution of forces for the ATI Mini45	51
2.6	Sensing range and resolution of torques for the ATI Mini45	51
3.1	Average applied force from each user in Newtons	64
3.2	Table of statistical significant. (1) No feedback, (2) Surrogate Visual Feedback, (3) Surrogate Vibrotactile Feedback	65
3.3	Post-experiment survey average results. (1) - Very Easy, (2) - Easy, (3) - Moderate, (4) - Hard, (5) - Very Hard	65

List of Figures

1.1	Foster-Miller TALON Robot	3
1.2	Results from police survey regarding the ideal cost of robotic systems for bomb disposal, reproduced from [1]	6
1.3	EOD Teleoperator System	12
1.4	Mark I Wheelbarrow (1972)	13
1.5	Mark II Wheelbarrow (1972)	14
1.6	Mark III Wheelbarrow (1972)	14
1.7	Mark V Wheelbarrow (1973)	15
1.8	Mark VI Wheelbarrow (1975)	15
1.9	UK MoD Buckeye	16
1.10	Wheelbarrow OCU	16
1.11	Mark VII Wheelbarrow	16
1.12	Remotec Mark VIII Wheelbarrow (1997)	16
1.13	Mark IX Wheelbarrow	17
1.14	Remotely Operated Vehicle for Emplacement and Reconnaissance	18
1.15	Remotely Actuated Mobile Platform for Render Safe and Disposal	18
1.16	Remote Control EOD Tool and Equipment Transporter	19
1.17	Semi-Autonomous Mobile System for Ordnance Neutralization	20
1.18	Remote Ordnance Neutralization System	21
1.19	iRobot PackBot	22
1.20	Hydroid Remus	23
2.1	Basic teleoperation control loop	28
2.2	Logitech Dual Action Gamepad	31
2.3	Contineo Robotics Three-Jaw Gripper	33
2.4	Mapping of motor encoder counts to gripper position [25, 75, 125, 175, 225, 275, 325, 375]	35
2.5	Graph displaying the prevention of extrapolation error by adding a horizontal asymptote	37
2.6	Northrop Grumman HD-2 Manipulator	43

2.7	Mounted Kistler Accelerometer	44
2.8	Mounted Vibrotractor	45
2.9	Graphic Feedback System	46
2.10	Arduino Duemilanove	48
2.11	Logitech Quickcam	49
2.12	ATI Mini45 Force/Torque Sensor	52
2.13	Grasping object instrumented with the ATI Mini45 F/T Sensor - Pen for scale	53
2.14	System Framework	54
3.1	Accelerometer test output showing three separate grasps, noted in red, of the instrumented object	56
3.2	F/T output during four successive grasps of the instrumented object .	58
3.3	Corresponding GUI output during four successive grasps of the instru- mented object	59
3.4	The setup of the gripper and instrumented object during the experiment	62
3.5	Plot of mean peak forces applied to the instrumented object, averaged for all subjects and all trials for each condition	64
3.6	Post-experiment survey ratings	66

Dedicated to our Nation's fallen EOD warriors

Chapter 1

Introduction

1.1 Motivation

Since the start of the Global War on Terror in 2001, 5,777 United States service members have been killed in overseas operations. Another 41,030 have been wounded in action [2]. It has been estimated that roadside bombs, Improvised Explosive Devices (IEDs), and suicide car bombs have accounted for 50% of the casualties in Afghanistan and 60% in Iraq [3].

In addition to the threat faced by those in the military, over 100 million land mines are currently planted around the world, causing between 15,000 to 20,000 civilian casualties per annum in addition to the countless injuries caused by unexploded ordnance (UXO) [4]. Prior to 2001, there were over 1,000 casualties annually in Afghanistan alone, making it the country with the highest fatality rate due to land



Figure 1.1: Foster-Miller TALON Robot

mines and UXO. The overwhelming majority of those casualties were civilians [4].

Explosive threats pose a serious danger to both militaries and civilian populations who live and work in areas where land mines and UXO are abundant. These threats are dealt with by civilian bomb disposal units and military EOD units. These units have used robotic systems since the 1970s in order to render safe explosive threats from a distance, saving countless lives. In the U.S. Navy alone, 200 Man Transportable Robot Systems, shown in Figure 1.1 from [5], have been destroyed since 2001, each an instance where a technician might otherwise have been injured [6]. However, these systems are fairly rudimentary when compared to some of the high-performance teleoperation systems used in other applications such as minimally invasive surgery, maintenance in space and hazardous material handling.

The systems currently in use tend to command robots in joint space using velocity control toggle switches. Due to reliability and computational constraints, no currently fielded EOD robots use Cartesian or master-slave control. Visual feedback is given to the user on the Operator Control Unit (OCU) from the onboard camera. Some systems display an output of the pose of the robot. A high level of skill is needed in order to efficiently control these types of robots, as the operator has to “learn” the robot’s inverse kinematics and Jacobian matrices. This heavy mental workload is one of several reasons that EOD robots tend to have relatively few degrees of freedom (DOF).

Additional constraints exist, including the need to be compact in size in order to maximize access to confined areas. Varying conditions and hazardous work also put a premium on the need for low-cost maintenance, which also tends to encourage the fielding of low-DOF systems. These systems are also significantly limited in the feedback given to the user. Current systems lack any type of kinesthetic or tactile feedback. At best, information on applied forces must be inferred from auditory information from the motors and internal models of the effects of system inputs.

This final limitation is a significant one, given that a great deal of the work being done is delicate in nature. The actions of accessing an explosive device, rendering it safe, and gathering evidence afterwards could be greatly influenced by the addition of haptic feedback to the operator.

While many of these limitations could be overcome with a substantial increase in

spending, there is a major incentive to keep the cost of these systems low. While industrial robots frequently attain a mean time between failure (MTBF) of 50,000 hours or more [7] [8], the average EOD robot has a MTBF of only 6 to 20 hours [9]. While this number would be excessively low in any field, in Explosive Ordnance Disposal, failures tend to be catastrophic ones.

In addition to issues related to reliability and the hazards of the environment, there is a significant disparity in the level of technology used to create explosive threats and that which is used to dispose of it. As an example, a typical land mine costs between \$3 and \$30 [10]. Costs to remove land mines average around \$800 per land mine, in addition to the potential cost of human life for those who remove them [4]. Likewise, many IEDs can be constructed with exceptionally inexpensive materials, as unexploded ordnance tends to be readily available. Robotic systems, while varying significantly in price, are invariably several orders of magnitude more expensive than the threats they seek to neutralize [11].

Results from civilian police departments [1], seen in Figure 1.2, indicate that the ideal cost for a robotic system should be under \$40,000. This is likely a function both of the likely catastrophic failure rate of the robot and the relatively limited funding available to bomb disposal units. While this cost may be unattainable given the necessary capabilities of an effective EOD robotic system, it is a testament to the importance of cost minimization.

Efforts to overcome current robotic limitations must be constantly cognizant of

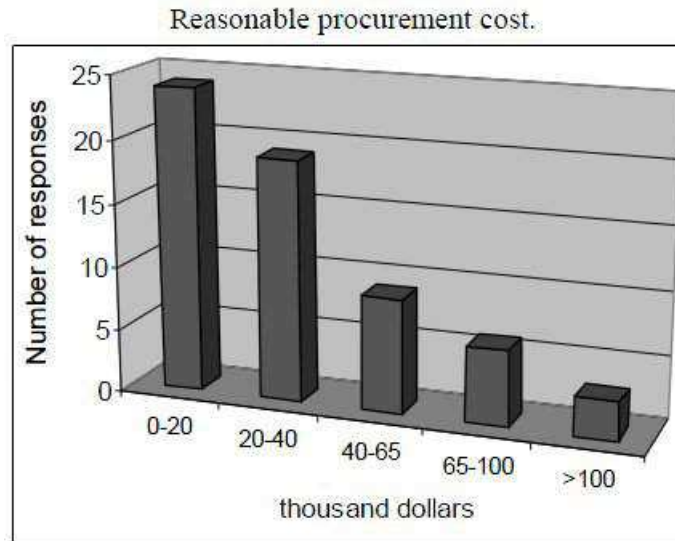


Figure 1.2: Results from police survey regarding the ideal cost of robotic systems for bomb disposal, reproduced from [1]

the cost involved in doing so. While performance and reliability should always be maximized, the system should ultimately be expendable.

With these considerations in mind, there is a significant need to develop cost-effective methods to display haptic information to the user.

1.2 Prior Work

This research builds on previous work from two very different areas: haptic technologies for telemanipulation, and robotic systems for Explosive Ordnance Disposal.

1.2.1 Prior Work in Haptic Feedback

Haptics refers to the sense of touch, and haptic technology invokes devices and software that displays haptic information to users in virtual and teleoperated environments. Haptic feedback is often described as cutaneous (tactile feedback, related to the skin) or kinesthetic (force feedback, related the muscles and joints). The development and efficacy of haptic feedback for teleoperation in various applications is relevant to the research described in this essay.

Some of the earliest haptic feedback systems were designed for teleoperation in hazardous environments, particularly manipulation of radioactive materials and later for space robots and surgery [12]. Originally, haptic feedback to the user was produced due to a direct mechanical connection between the “master” device and the remote “slave” robot. Then, as master and slave devices were physically disconnected and controlled “by wire”, numerous control schemes invoking sensors on the slave and actuators on the master were developed to enable haptic feedback.

Much of the research in haptic feedback for teleoperation has focused on high-performance, low-impedance devices operating in a *bilateral* mode. That is, force and motion information are exchanged between the master and the slave. Challenges in bilateral teleoperation include maintaining stability and transparency in light of uncertainty in the dynamic models of the human operator, and time delays. Stability for teleoperators can be defined as bounded system inputs resulting in bounded system outputs. Transparency is the ability of a teleoperator to make the user feel as if

he is directly manipulating a remote environment, rather than through a teleoperator. Supervisory and shared control are methods of overcome delays and increasing performance without requiring the human constantly in the loop, but lack transparency. In addition, wave variables have been used in bilateral teleoperators to eliminate the destabilizing effects of lag.

While direct haptic feedback based on bilateral teleoperation will likely be useful in EOD systems in the years to come, methods such as sensory substitution are much more applicable to situations requiring robustness in challenging operational environments. The specifics of the EOD environment require lower cost, more robust, solutions to haptic displays than the high-fidelity bilateral systems being developed for other applications. Many of the benefits of sensory substitution methods for force feedback were shown by M. Massimino in [13]. These include the ability to display to the user small changes in forces, and the lack of issues with instability. For tasks involving detecting contact, sensory substitution out-performed kinesthetic feedback as it allowed the users to sense smaller forces. It was also found that tactile displays were effective because they did not overload the subjects' visual system, nor did they induce operator movement or instability.

Sensory substitution methods have seen significant interest recently, due to their potential application in robot-assisted surgical systems. Gwilliam et al. [14] used the da Vinci Surgical System to detect calcified arteries by means of palpation. Results showed graphical feedback of forced increased user performance of both experienced

and novice users over no haptic feedback, while direct force feedback (to the user's hands, via the master manipulator) increased user performance only among experienced users. Likewise, Kitagawa, et al. [15], [16] used the da Vinci to perform suturing tasks and used visual and auditory sensory substitution to display forces to the user. Reiley et al. [17] expanded further proved the effectiveness of visual feedback of force information in improving suture tying with an surgical robot.

While most research using surgical systems has focused on visual sensory substitution of force information, there has also been some work developing and evaluating vibrotactile feedback. In [18], the authors develop a vibrotactile feedback system in which vibrations were applied to a subjects foot. They showed that a linear increase in vibration intensity is perceived as a linear increase in force and that the system improved a user's ability to differentiate tissue softness.

Relevant work has also been done in using vibrations for event detection, an important part of telemanipulation using direct, shared or supervisory control. In [19], accelerations were measured on the slave robot and fed back to the user via a vibrating device. Using both context and sensor-based data, event detection can be done with a very high degree of certainty, given an array of sensors to measure the full state of the robotic system [20]. In [21], the stability and robustness of this technique is increased with the addition of smooth phase transitions between events.

In this research, we estimate force applied by the slave robot (the gripper of an EOD robot) on the environment, and display the sensed information via sensory

substitution. The sensor substitution methods were considered are a visual bar graph, similar to Kitagawa, et al. [15], [16] and vibration feedback via pager motors attached to the master device (a game controller).

1.2.2 Prior Work in Explosive Ordnance Disposal Robotics

Technological innovation has long played an important role in Explosive Ordnance Disposal. During World War II, the Research Department of the US Navy Bomb Disposal School [22] and their counterparts in the United Kingdom, the Unexploded Bomb Committee [23], made remarkable improvements to the technologies available to EOD technicians and Ammunition Technical Officers (ATOs). Many of the solutions that they came up with could not be tested in laboratory conditions, so these groups spent significant amounts of time in the field working on live ordnance [22] [23].

A few of the many innovations that these two groups devised during World War II are listed below:

- Acid Trepanning - A nitric acid solution applied to the steel bomb case in a fine spray to cut a hole in a piece of ordnance. No undesirable effect upon hitting the main charge.
- Freezing Technique - Lowering the temperature of the fuze until the dry cell of the battery no longer produces a current. Freezes the mercury globule in the

mercury tilt switch. Frozen using a dry ice/ alcohol slush.

- Plaskon Resin Injection - Attack on mechanical fuzes by inserting a quick hardening resin [22].
- Magnetic Clock Stopper - A large electromagnet fixed to the side of the bomb through which high current was passed. The resulting magnetic field stopped the ticking of mechanical clocks while it was in place.
- Mine Locater - Early metal detector.
- Fuze Extractor No. 1 (Freddy) - Frame, pneumatic jack, an extractor rod, and a discharger. Used a CO2 cartidge which raised the extractor rod when pierced. Because there were several inches of play before the fuse was extracted, the ATO had several minutes to distance him/herself.
- Radiography - Early X-Ray technology with an adjustable frame which could be fitted to bombs of varying circumferences [23].

While many of these advancements certainly saved lives, distance is the only factor that can truly keep an EOD technician safe. Because of this fact, one of the most basic tools that the EOD technician uses is the hook and line, which is an extremely low-tech means to manipulate an object from a distance. In many ways, teleoperated robots have been developed as an extension of this simple solution. Since their inception, robotic systems have been used extensively as a way to render safe explosive threats while maintaining the safe distance of the technician.

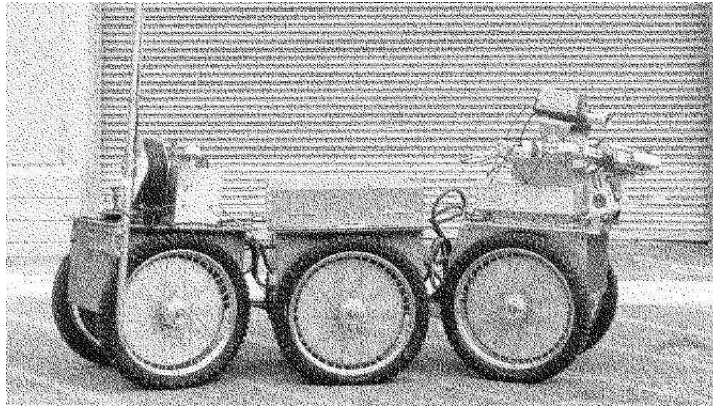


Figure 1.3: EOD Teleoperator System

In the United States, the idea of using robotic systems for EOD was first explored in the 1960s [24]. The EOD Teleoperator System (Figure 1.3, reproduced from [24]) was developed by the EOD Robotics Program and consisted of a master-slave manipulator mounted on a six wheeled vehicle. However, this system was found to be infeasible for EOD use due to its complexity [24].

As a result, the primary development of early fielded EOD robotic systems took place in the United Kingdom. Because of conflicts in Northern Ireland, there was an immediate need to “attach a hook to a car bomb to allow the vehicle to be towed away to a site where it could be safely destroyed. All too often the process of attaching the towing hook triggered the explosion – killing the ATO” [25].

Because of this, Lt. Col. Peter Miller of the Royal Army Ordnance Corps was asked to devise a solution. Miller retrofitted a battery-operated three-wheeled wheelbarrow chassis with a spring loaded hook on a boom to latch underneath a suspect



Figure 1.4: Mark I Wheelbarrow (1972)

car [25]. The controls of this device consisted of four nylon lines. Two steered the front wheel of the device, another reversed the direction of the motor, and the last engaged the spring loaded hook. Both the controls of the robot, and its intended effects were modeled after line and hook methods used by EOD technicians for decades [26].

This design was simple; it was invented, designed, and put into production in 22 days. Named the Wheelbarrow (Figure 1.4), after the platform on which it was created, it was immediately fielded on the front lines in Northern Ireland [26]. Figure 1.4 and all other Wheelbarrow figures are reproduced from [26], unless otherwise noted.

Each failure of a Wheelbarrow was referred to Lt. Col. Miller to solve. As such, several significant improvements were made to the system over a relatively short period of time. The first improvements made to the Mark I, shown in Figure 1.5 were the addition of a second motor to control the steering of the vehicle and a boom that allowed it to drop explosive charges into suspect cars.



Figure 1.5: Mark II Wheelbarrow (1972)

Figure 1.6: Mark III Wheelbarrow (1972)

The Mark III (Figure 1.6) added additional linear actuators which turned the static boom into a robotic manipulator, albeit a simple one. Additionally, an improved chassis was used with a fourth wheel to provide greater stability to the system. Closed circuit cameras were added to a later iteration of the Mark III, as were clamps to hold explosive disrupters [26].

The Mark IV and V (Figure 1.7) saw significant improvements to the kinematic design of the Wheelbarrow, in addition to an improved electronics system. Over the course of two years, Miller and his team produced 22 Mark V's in addition to a handful of each of the earlier iterations of the system. By November of 1973, the Wheelbarrow had been used operationally more than 100 times [26].



Figure 1.7: Mark V Wheelbarrow (1973)



Figure 1.8: Mark VI Wheelbarrow (1975)

Research and development of the Wheelbarrow was taken over by Remotec, Inc. in 1976 and they started to market the Wheelbarrow worldwide. They produced the Mark VII (Figure 1.11) later that year. The purpose of the wheelbarrow has typically been reconnaissance and disruption, much like other early EOD robotic systems such as the UK Ministry of Defense Buckeye, shown in Figure 1.9, reproduced from [26]. Manipulation did not become a major goal for the platform until much later systems such as the Mark IX (Figure 1.13, reproduced from [27]).

The Wheelbarrow is operated by an Operator Control Unit (OCU), shown in Figure 1.10, with toggle switches which control the direction of each joint individually. A separate gain knob controls the speed that each joint moves when commanded.

Parallel to these developments, the United States continued to develop robotic systems for Explosive Ordnance Disposal. Following the EOD Teleoperator System,

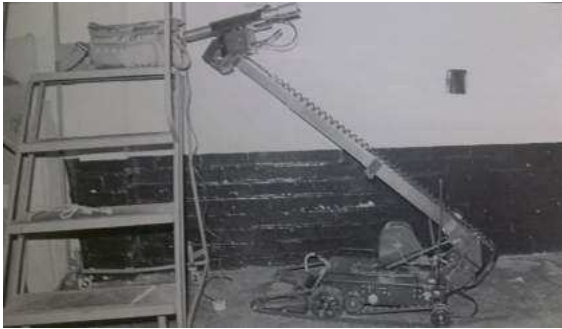


Figure 1.9: UK MoD Buckeye

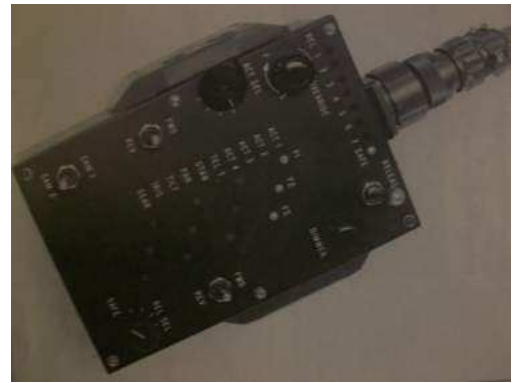


Figure 1.10: Wheelbarrow OCU



Figure 1.11: Mark VII Wheelbarrow



Figure 1.12: Remotec Mark VIII Wheelbarrow (1997)

efforts were made to develop smaller, low cost robotic technologies. The first of these developments was the Remotely Operated Vehicle for Emplacement and Reconnaissance (ROVER) [24]. At \$10,000, the ROVER (Figure 1.14) was a low-cost cable-controlled robotic system. All remaining figures in this chapter are reproduced from [24] unless otherwise noted.

On board, the ROVER had a video camera, simple manipulator, and an interface to fire EOD disrupter tools. Despite its communications and power tether, its portable



Figure 1.13: Mark IX Wheelbarrow

battery pack limited it to an operational endurance of two to four hours. Serious additional limitations were found in the ROVER system and it was discontinued in the mid 1980s. Although it was never operationally fielded, it was a significant learning experience for the EOD community as it demonstrated the efficacy of low-cost, low-DOF robotic systems. Subsequent robotic systems tended to be more akin to the ROVER than the EOD teleoperator system.

As a follow-on to the ROVER, the Remotely Actuated Mobile Platform for Render Safe and Disposal (RAMROD) was developed [24]. The RAMROD, shown in Figure 1.15, was similar in form and cost, but was designed to be weather resistant, field serviceable, and to be able to climb stairs. Similar to the ROVER, shortcomings in the system, as well as the existence of potentially more capable commercially available

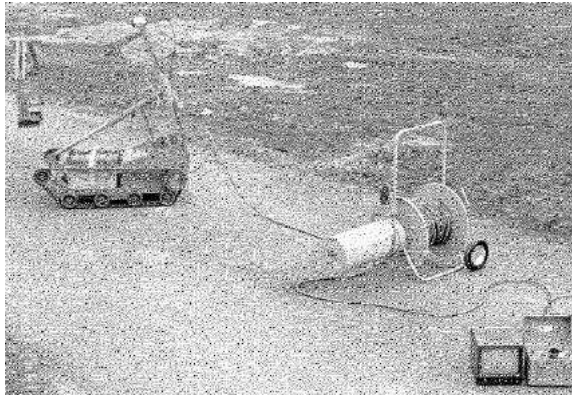


Figure 1.14: Remotely Operated Vehicle for Emplacement and Reconnaissance

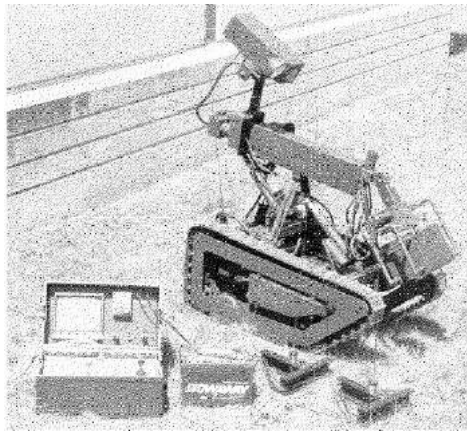


Figure 1.15: Remotely Actuated Mobile Platform for Render Safe and Disposal

systems, prevented the RAMROD from ever being fielded operationally.

The RAMROD program transitioned into a new effort which resulted in the Remote Control EOD Tool and Equipment Transporter (RCT) [24], shown in Figure 1.16. After many years of development, this was the first robotic system to actually be used by troops. While its use overseas was limited to the Gulf War, it was found to be an effective means of dealing with IED threats. However, its effectiveness against



Figure 1.16: Remote Control EOD Tool and Equipment Transporter

conventional ordnance was minimal and its overall unit cost was over \$600,000. It was used by all of the services until it was replaced by the Remote Ordnance Neutralization System (RONS).

The morphology of the RONS (Figure 1.18) is very similar to that of its predecessor, although its feasibility was first proven by the Semi-Autonomous Mobile System for Ordnance Neutralization (SAMSON) [24]. The SAMSON (Figure 1.17) featured the first 6-DOF manipulator arm to be used on an EOD robot. Additionally, it demonstrated the capability of end effector tool exchange, and more advanced manipulation. The RONS, fielded in 1999, built on lessons from the SAMSON and proved capable of assisting EOD technicians in more aspects of the mission than any previous system. The RONS remains in use by all services, with over 320 robots having been produced. It is used most frequently by Air Force EOD technicians because of their specific mission set [28].

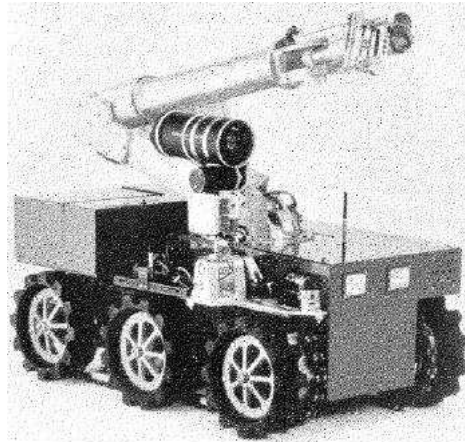


Figure 1.17: Semi-Autonomous Mobile System for Ordnance Neutralization

Much as The Troubles in Northern Ireland provided the imperative to make robotic systems an essential part of the UK EOD tool kit, so did the Iraq War have a significant impact on the role of robotic systems in EOD in America. In both of these conflicts, EOD was at the front lines, and IEDs and car bombs were the weapon of choice. In the UK, this environment led to the creation of the Wheelbarrow. In the US, ongoing efforts to develop a Man Transportable Robot System (MTRS) resulted in the fielding of a combined 3,000 QinetiQ Talon (Figure 1.1) and iRobot PackBot (Figure 1.19, reproduced from [29]) robots from 2005 to present [28]. Each MTRS costs roughly \$140,000, has relatively few degrees of freedom and almost no autonomy. However, both systems perform very well in extreme environments and are optimized for the rigors of field work.

While the MTRS has improved considerably in terms of reliability, survivability, and capabilities from their predecessors, the controls and user interface for these

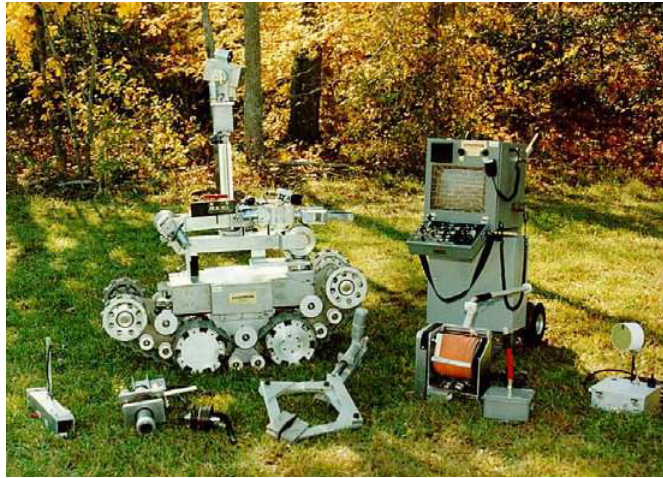


Figure 1.18: Remote Ordnance Neutralization System

systems look remarkably similar to those of the earliest EOD robots. The output of a closed-circuit television camera, easily identifiable on the RAMROD, SAMSON, PackBot, and Talon, is displayed on a small screen of an operator control unit. While newer systems have multiple cameras and some advanced optics technology, the visual display is the only feedback given to the operator.

The size of the OCU increased with later systems, as can be seen with the RONS OCU. This trend was reversed with the MTRS, both of whose controllers are similar in size to a large brief case. The user input on these OCUs are almost universally velocity control toggle switches, with a gain dial to adjust the speed of the joint being moved. The Talon and PackBot departed from this slightly by using continuous input joysticks and “intuitive” hockey puck-sized paddles respectively. Both models can now be controlled with a standard size video game controller which maps each joystick axis to a joint on the robot.



Figure 1.19: iRobot PackBot

A significant portion of the US EOD mission is conducted underwater in combating both naval mines and sunken ordnance. In order to assist in this mission, unmanned underwater vehicles (UUVs) such as the Hydroid REMUS (Figure 1.20, reproduced from [30]) are employed. The REMUS is a 5 ft long, 80 lb submersible that can operate at depths up to 100 ft and is equipped with a large array of sensors for navigating in the water column and locating ordnance.

Due to the constraints of the underwater environment, unmanned underwater systems are employed in manpower intensive operations such as broad area surveillance. Allowing UUVs to take over this slow, intensive work reduces risk to EOD technicians and allows them to focus on intelligence gathering and render safe procedures on ordnance [31].

Current systems fielded by the US Military for underwater EOD operations lack



Figure 1.20: Hydroid Remus

any manipulator and instead focus on intelligence, surveillance, and reconnaissance. While manipulation will likely be a goal in future systems, the largest focus for improvement on these systems is in more capable sensors and increased autonomy and power [31].

There has been some work in academia to develop robotic systems for EOD. Due to the small number of EOD technicians, and hence EOD robots, the results from the majority of studies developing EOD robotic systems have not been implemented, expanded upon, or seen significant citation.

In [32] a system is devised where a large number of low-cost robots execute a Pick Up and Carry Away (PUCA) mission to combat cluster ordnance. The relative benefits of exhaustive and random searches are examined as well as the importance of multiple drop off points. Further development of this system in [33] emphasizes

the importance of low-cost, performance, and simplicity.

Several efforts have been made to create robots for demining. In [34] a low-cost, light weight system for demining is developed. The study lacked significant evaluation of the robotic system and noted that the cost of the robot, at around \$6000, was still an order of magnitude greater than hoped. In [35], sensors are determined to be the greatest limiting factor in creating effective robotic solutions to demining.

iRobot developed a system for kinesthetic gripper force feedback on the Pack-Bot robot in [36]. Forces were displayed to the user with a modified Novint Falcon interface. Results from this study indicated increased performance of delicate manipulation tasks with haptic feedback, but tasks times tended to increase as well. Additionally, the study noted that user performance decreased significantly when using the Falcon without force feedback.

In [37], an impedance-controlled bimanual system for EOD with virtual fixtures to prevent self-collision was proposed. This system was used to satisfactory results, but with significantly increased task completion time over the manual case. Additional work to make this robotic system robust and mobile did not occur.

While other work has taken place to develop robotic systems for EOD, they have primarily been demonstrative and have not significantly influenced fielded systems.

1.3 Thesis Contributions

This thesis describes the following contributions:

- To the best of the author’s knowledge, the first systematic development and assessment of a sensory substitution haptic feedback system for a teleoperated Explosive Ordnance Disposal robot
- A detailed examination of the relative benefits gained from low-cost feedback solutions when applied to grasping tasks
- Experimental evidence demonstrating improved event detection with haptic feedback

1.4 Organization

This thesis is organized into several chapters following this introduction. First, Chapter 2, describes the various pieces of the physical, electromechanical and software systems that were used for the experiments, with particular focus on the integration of these components. This chapter describes the input devices, manipulators, feedback devices, and system integration tools used.

Next, Chapter 3 gives a detailed explanation of the experiment that was conducted, including a defined protocol. Then the data from the experiment is presented, annotated, and followed by statistical analysis.

The thesis concludes in Chapter 4 with a discussion of the contributions of the research and the areas of future work. Following this conclusion, documentation and code are attached as appendices.

Chapter 2

Experimental System

2.1 Overview

Teleoperated robotic systems put the human operator into the control loop of the robot. In all currently fielded Explosive Ordnance Disposal systems, the operator gives velocity commands to the robot in joint space using toggle switches or joysticks. The operator is provided with a live camera feed through the Operator Control Unit (OCU).

In order to improve the usefulness of the telepresence, information about applied forces can also be displayed in order to better provide the user with information with which to make decisions about subsequent commands to give the robot.

For this control loop (Figure 2.1) to be realized, several interworking pieces must be implemented. First, a robotic system must be selected to which the operator can

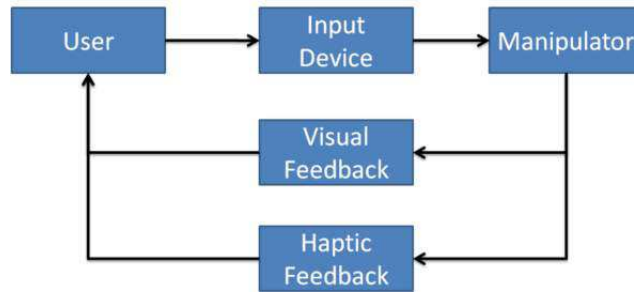


Figure 2.1: Basic teleoperation control loop

give commands. For EOD robots this must include a mobile platform, a manipulator arm and an end effector tool or gripper for interacting with the environment. Additionally, a method must exist for the operator to give commands to the robot. Finally, both visual feedback systems and haptic feedback systems must be designed in order to close the loop.

2.2 Input Device

After examining input devices that are currently used in EOD robots, a video game controller was selected as the single input device used to give commands to the robot. This input device is currently used on MTRS systems as an improvement on its standard interface. Initial plans for this research hoped to generalize these findings by examining several different input devices, but ultimately, time and resources prohibited this.

While using a single input device does not invalidate of the findings of this research,

examining multiple input devices is particularly important for haptic feedback as some haptic feedback modalities act on the user through the input device. Additionally, the effect of any particular feedback modality is likely also a function of the compatibility of the input device to that feedback modality. In order to make our experimental platform the most effective, an input device was chosen that is very similar to what is currently being used in the field and will likely remain a standard feature of near-term EOD robotic systems.

Several additional input devices were examined, including the Cyberglove and Cybergrap, the Novint Falcon, and a master/slave controller. While the Cyberglove and Cybergrasp may have allowed for detailed force feedback of grasp forces, 21 of its 22 sensors would have gone unused, as the gripper that was selected had a single underactuated DOF. Additionally, an effective means was not found to control the manipulator in addition to the end effector without use of the Cyberforce system or an optical tracking system, both of which are unlikely to be fielded operationally in the near term.

The Novint Falcon, while possibly effective in controlling the manipulator, was not assessed to have a particularly good mapping to the workspace of the full manipulator arm. While several possibilities existed for overcoming this, time was the primary factor ruling out this input device. Finally, a passive mini-master manipulator could have been built in order to send joint commands to the manipulator, however, both time and funding prevented this from becoming immediately feasible,

although this type of control has a reasonable chance of being fielded on future EOD robotic systems.

2.2.1 Logitech Dual Action Gamepad

Because benefits can be gained from using systems that operators are already familiar with [38], several current robotic platforms are controlled with video game controllers, rather than bulky operator control units. Therefore, the Logitech Dual Action Gamepad (Figure 2.2) was used in our setup in order to provide the operator with a control input with which he likely already had extensive experience.

Each joint on the gamepad controller was linked to a separate joint axis on the robot. When possible, the mapping between the robot and controller joints was constructed in a logical way based on how the movements would affect the manipulator frame of reference. For example, left and right motions of the left joystick were mapped to counter clockwise and clockwise rotations of the torso joint, respectively. Each axis operated in velocity control mode using a scaled input from the analog joysticks (The motivation for this choice is given in Section 3.1.3).

The gamepad was connected to the computer using a USB port and was read using a serial protocol. By utilizing the JavaJoystick.m MATLAB object from the Revolutionizing Prosthetics library [39], the gamepad was initialized and controlled. Its twin joysticks were read using encapsulated functions, and the X and Y axes for each joystick yielded a continuous output of -1 to 1. Button values were placed into an



Figure 2.2: Logitech Dual Action Gamepad

array after each update, with a 0 referring to a an unpressed button and a 1 referring to a pressed button.

Each of the 4 DOF of the manipulator was mapped to an axis of the gamepad when in arm-control mode. The speed of each joint was proportional to the distance each joystick was displaced. A press to the uppermost left button of the gamepad toggled gripper-control mode which allowed the left axis to be used to open and close the gripper, while still having the right joystick retain control of the distal joints of the manipulator. Button 2 was used as an emergency stop button during manipulation with the robotic arm, and was used to end each trial during experimentation.

2.3 Manipulator

The manipulator used in this research consisted of a prototype Three-Jaw gripper and 4-DOF robotic arm.

2.3.1 Three-Jaw Gripper

While most EOD robotic systems utilize a two-jaw gripper or parallel gripper, there is an effort to transition towards robotic systems that are more anthropomorphic [24]. The majority of tools and interfaces are built with the human hand in mind, so it is a logical choice to use grippers that are similar in form and function. While this may eventually lead to robotic grippers with DOF on the order of the human hand, it is more likely that transition will first occur by introducing grippers that are conformal in nature and possess coupled kinematics similar to the human finger which still take advantage of anthropomorphic morphology, but lack the complexity of higher-DOF grippers.

The Three-Jaw Gripper (Figure 2.3, reproduced from [40]) built by Contineo Robotics is inspired by the human hand but is designed to be much more simple. It contains 9 DOF, but is actuated with a single motor. The excess DOF are underactuated. This design feature allows each finger to naturally conform around a grasping surface as each link in the kinematic chain makes contact with an object. This design also turns the “palm” of the gripper into a natural grasping surface, increasing

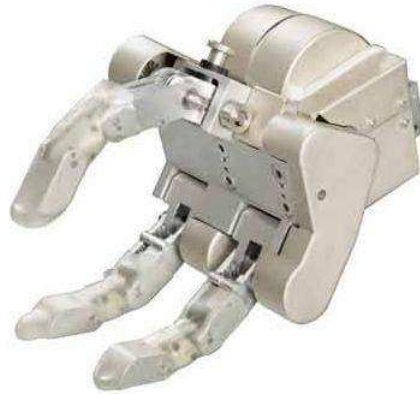


Figure 2.3: Contineo Robotics Three-Jaw Gripper

the stability of a given grasp through further kinematic coupling. There is natural compliance built into each finger joint so that the stalling of a single finger will not immediately stall the remaining fingers.

The gripper used for these experiments is an early prototype of a family of conformal grippers which are currently in the final stages of development and scheduled to be released within the year.

The motor is built with a current sensor, tachometer, and encoder. The motor itself consists of a brushless motor driving a frictional planetary gear with a cycloidal drive output. The output is then sent through a compound spur gear train which drives the fingers on the gripper. The final drive ratio is approximately 1000:1.

In an attempt to develop technology that uses as little additional hardware as possible, we used the current sensor in order to determine the torque being output by the gripper. In order to better understand the necessary torques required for the

gripper to achieve a particular state, the system parameters were identified.

While a mapping of motor torques to accelerations can be achieved by analytically describing the dynamics of the system, the significant nonlinearity, gearing, backlash, and compliance would greatly reduce the accuracy of such a technique. As such, empirical methods were pursued in order to discover the parameters of the system. The equation governing the relationship between current and output torque was assumed to be of the following form:

$$\mathcal{I} = \phi(\theta, \ddot{\theta}) + \beta(\theta, \dot{\theta}) + \tau_{\text{applied}} \quad (2.1)$$

Where \mathcal{I} is the current driving the motor, θ is the absolute position of the motor, ϕ represents the torque needed to accelerate the motor, β represents the torque needed to close the gripper at a constant velocity, and τ is the current being supplied to apply torque on an object. Both ϕ and β were assumed, and experimentally confirmed, to be dependent on θ as well as $\ddot{\theta}$ and $\dot{\theta}$ respectively.

An experiment was performed where the gripper was opened and closed numerous times, with a variety of speeds. Each open and close command took place over a range of 400 counts of the encoder on the motor shaft, with 0 being completely open and 400 being completely closed (Figure 2.4). The variable θ was assigned to represent the position of the gripper in encoder counts, although strictly speaking it did not represent either the “angle” of the gripper or the motor shaft. This assumption can be made without loss of accuracy as the mapping of motor position to gripper position

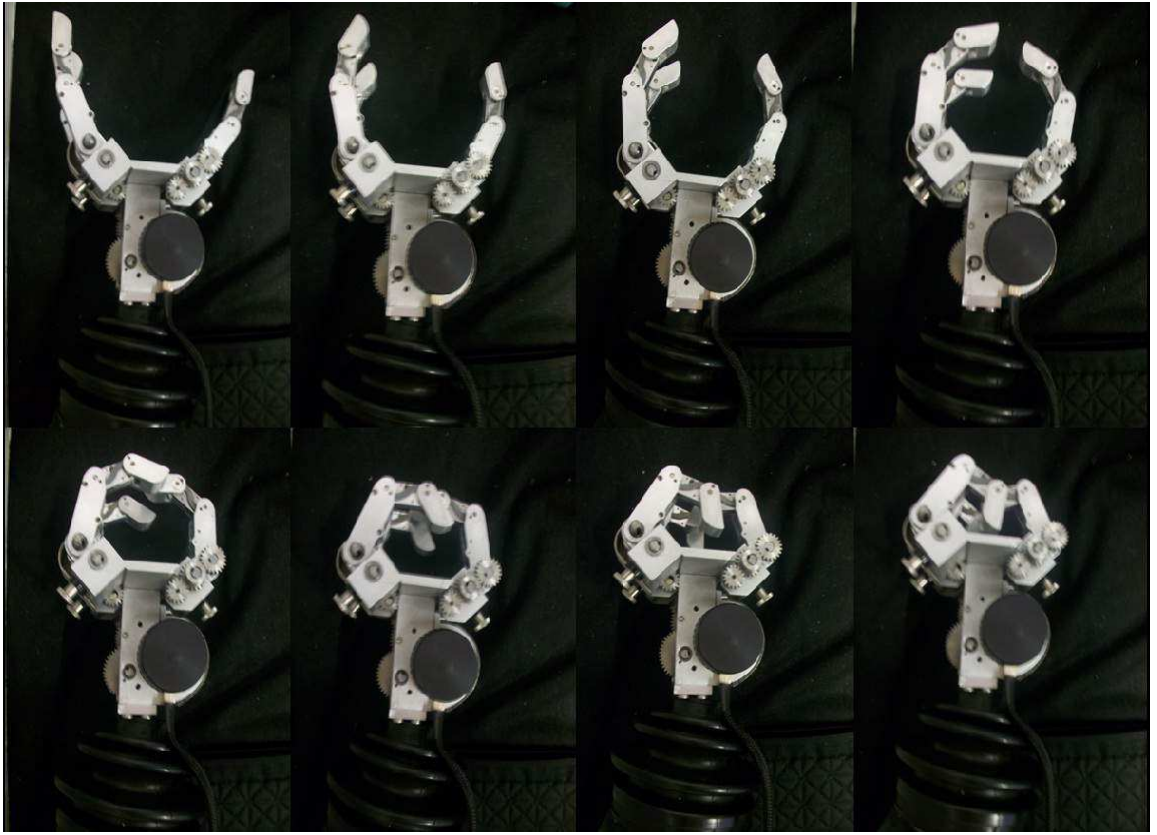


Figure 2.4: Mapping of motor encoder counts to gripper position [25, 75, 125, 175, 225, 275, 325, 375]

is an arbitrary one.

The function β was assumed to depend on θ and $\dot{\theta}$. Because velocity terms can easily be found without acceleration, but not vice-versa in the discrete case, β was isolated by opening and closing the gripper at different speeds and then removing unapplicable data points. Any data with acceleration was removed, thereby eliminating ϕ . Additionally the gripper was not supplying any torque to an object. As such, the function β was isolated.

$$\mathcal{I} = \beta(\theta, \dot{\theta}) \quad (2.2)$$

Each of the remaining terms were sampled relatively easily, but the function was further simplified from a multi-input/single-output system to a single-input/single-output system by assuming that the function was constant with respect to θ over a relatively small range of θ . This reduced the complexity of the function to the point where a least squares solution could map inputs ($\dot{\theta}$) to outputs (\mathcal{I}). An n^{th} order polynomial was constructed to model the relationship between β and $\dot{\theta}$.

$$\beta_{\theta}(\dot{\theta}) = \sum_{i=0}^n p_i \dot{\theta}^i \quad (2.3)$$

As previously stated, this polynomial was assumed to be constant over a relatively small range of θ . As such, the data was separated into different batches around each θ range (Figure 3.3). It was experimentally found that eight separate batches of θ , consisting of 50 counts each, led to functions which resembled the functions from bordering batches of data, but did not necessarily resemble the functions derived from data two batches away.

Polynomials were then constructed (Table 2.4) that mapped \mathcal{I} to $\dot{\theta}$ in a least squares sense for a given θ range. A 6th-order polynomial was found to minimize interpolation error unless the number of data points was sufficiently small, in which case a 3rd-order polynomial was used in order to prevent overfitting the data.

Because of the inability of polynomial curve fitting to extrapolate to data outside

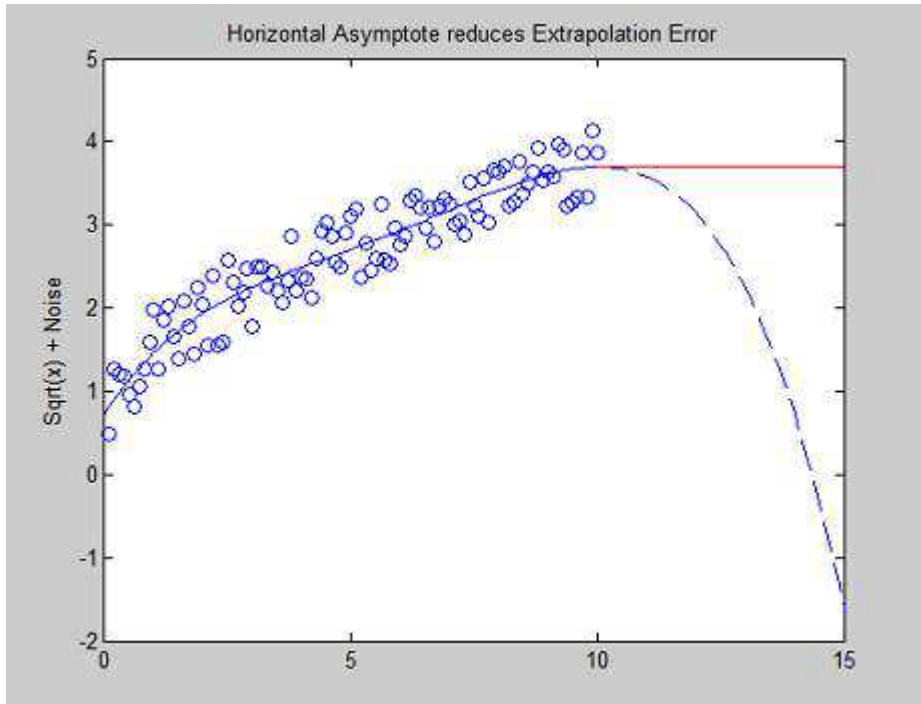


Figure 2.5: Graph displaying the prevention of extrapolation error by adding a horizontal asymptote

of the region for which it was created, a horizontal asymptote (Figure 2.5) was created starting at the final data point and continuing on to higher values of $\dot{\theta}$. While this assumption of a horizontal asymptote is not perfect, it is a significant improvement over using the polynomial values to predict extrapolated data, and significantly increased the robustness of the system.

After the required amount of current to drive the gripper with constant velocity was modeled, a set of data was taken using various terms for the acceleration of the gripper. Again, the data was sorted into batches based on θ . The measured velocity for each data point was used in order to subtract off the current being used to drive the gripper at that velocity. According to the model, the remaining torque should

	0° – 50°	50° – 100°	100° – 150°	150° – 200°
p0	6.413E-05	-3.369E-10	-2.6441E-10	-2.161E-10
p1	-0.0170	1.698E-07	1.40E-07	1.207E-07
p2	1.589	-3.286E-05	-2.8685E-05	-2.602E-05
p3	7.585	0.00306	0.00284	0.00270
p4		-0.144	-0.141	-0.139
p5		3.726	3.760	3.807
p6		0.556	0.530	0.558

Table 2.1: Three-Jaw Gripper torque/velocity polynomial values for $\theta \in \{0-200\}$

have been due to the acceleration of the gripper as there was no torque applied. The data was fitted to an k^{th} order polynomial relating current to $\ddot{\theta}$

$$\phi_{\theta}(\ddot{\theta}) = \sum_{i=0}^k p_i \ddot{\theta}^i \quad (2.4)$$

It was found that data did not imply a simple non-linear function between $\ddot{\theta}$ and current, but rather that the effects of static friction were significantly more of a determining factor than inertial effects when the gripper was already in motion. As such, the model was revised to be of the following form:

$$\mathcal{I} = \beta(\theta, \dot{\theta}) + \tau_{applied} + \psi \quad (2.5)$$

Where ψ was a function modeling the effects of static friction. With this corrected model, the method for determining the function β did not change as ψ only had non-zero values where acceleration was present.

	200° – 250°	250° – 300°	300° – 350°	350° – 400°
p0	-3.1353E-10	-2.1861E-10	-2.691E-10	1.515E-05
p1	1.648E-07	1.194E-07	1.370E-07	-0.00675
p2	-3.340E-05	-2.506E-05	-2.681E-05	1.0638
p3	0.00325	0.00252	0.00253	14.600
p4	-0.155	-0.1276	-0.121	
p5	3.916	3.520	3.334	
p6	0.556	0.503	0.471	

Table 2.2: Three-Jaw Gripper torque/velocity polynomial values for $\theta \in \{200-400\}$

2.3.2 Robotic Arm

Several manipulators were examined for use, including the WAM arm from Barrett technology, the TALON manipulator, the Packbot manipulator, and the HD-2 manipulator from Northrop Grumman. While the WAM arm provided the most capabilities and even allowed for upper arm kinesthetic feedback, it bears the least resemblance to currently fielded EOD systems and an examination of the effects of greater DOF and dexterity on EOD teleoperation performance likely could be its own study.

While both the TALON and the Packbot are heavily used systems, both would have been somewhat difficult to come by and offered fewer options for reading data from the robotic system into a laptop for processing. The HD-2 arm (Figure 2.6) on the other hand, while not currently commercially available, was acquired on loan from Contio Robotics and was readily controllable using a MATLAB GUI. This GUI was able to be integrated with a GUI made for feedback purposes in order to simplify the setup.

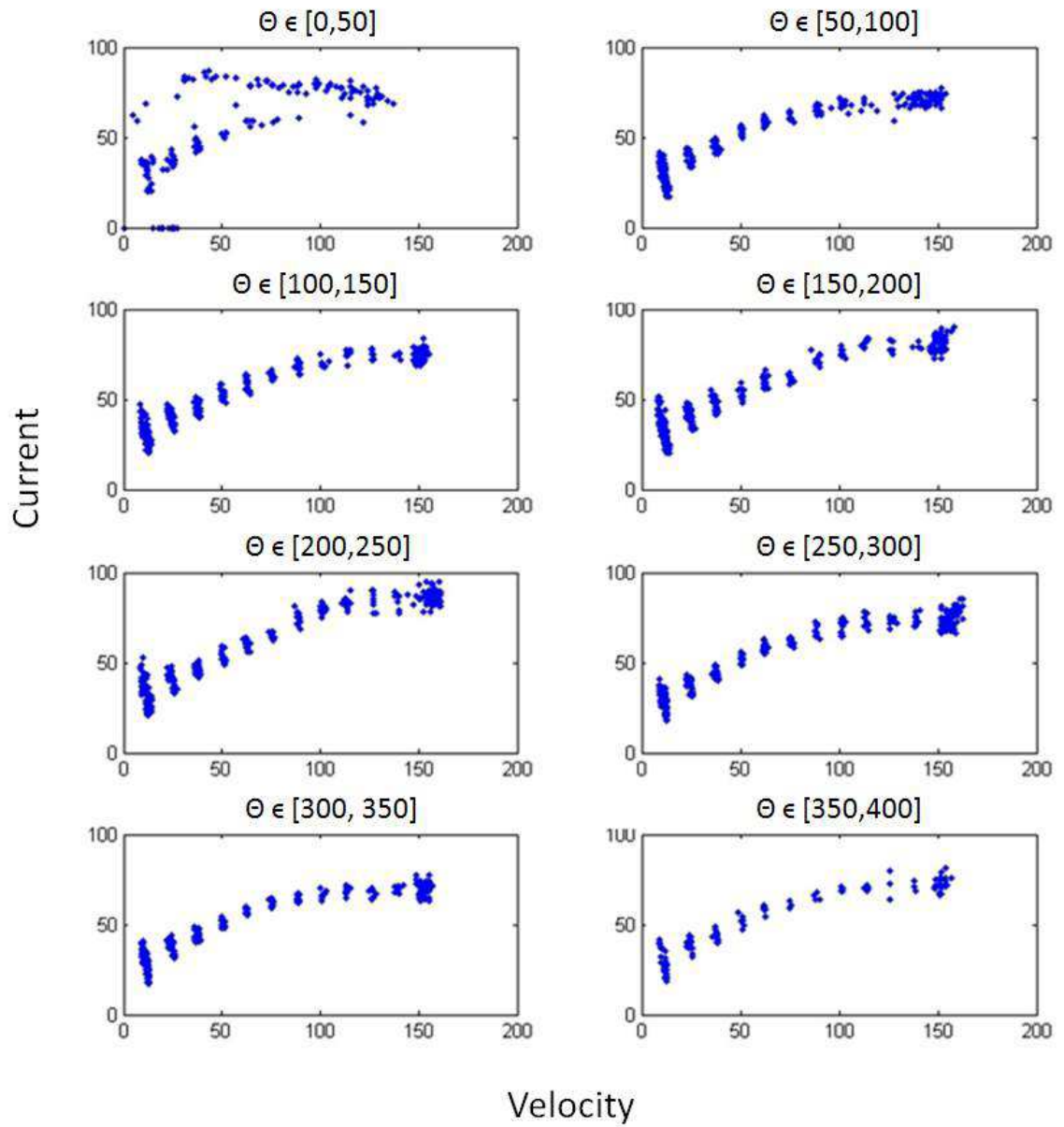


Table 2.3: Three-Jaw Gripper - torque/velocity identification raw data

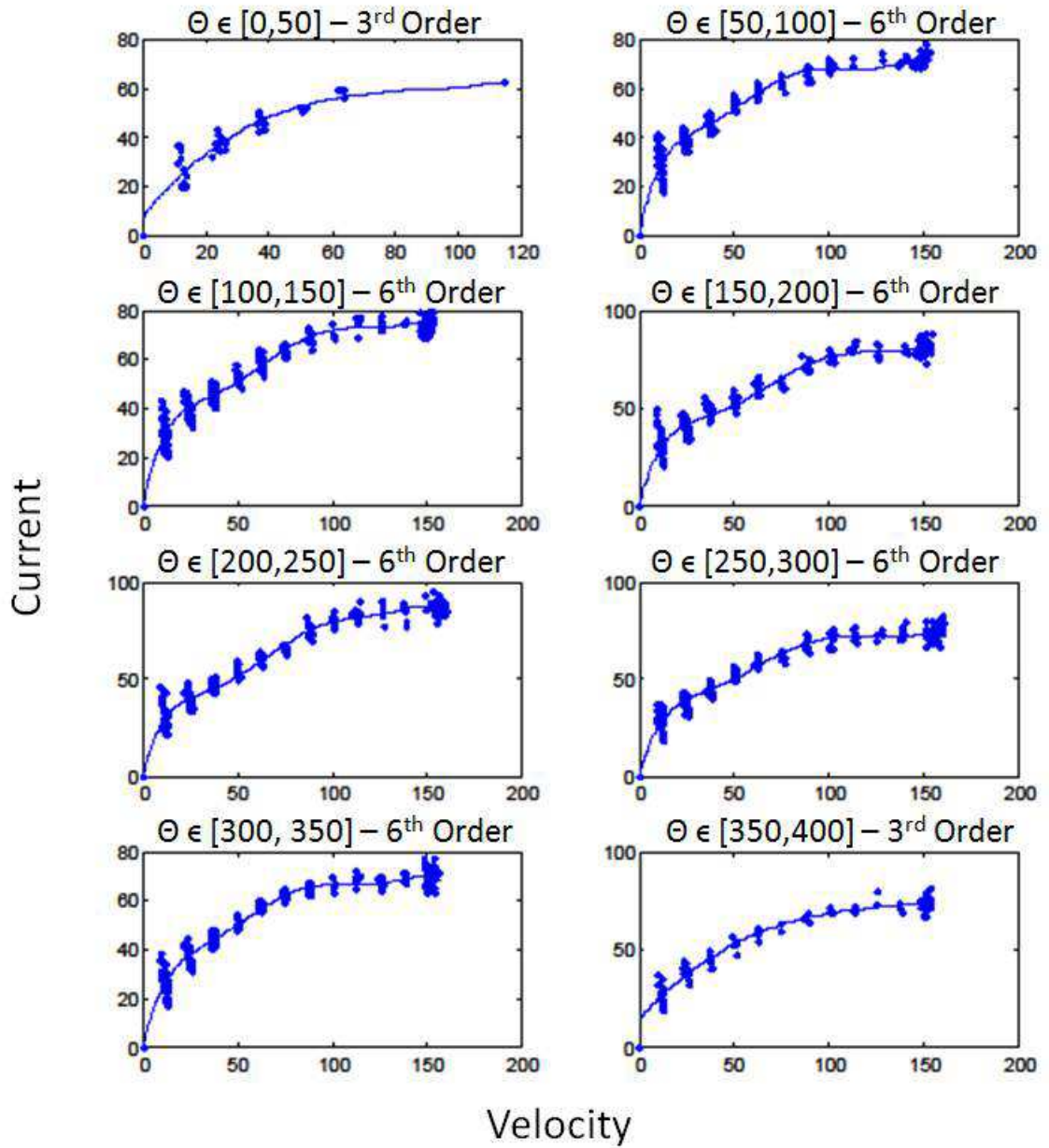


Table 2.4: Three-Jaw Gripper - torque/velocity identification filtered data with polynomial fit curves

The HD-2 Manipulator is a 4 DOF (typically 5, but the Contineo Gripper Prototype lacked wrist roll) manipulator which measures 52 inches when fully extended. It has a lift capability of 125 lb close to the body and 40 lb at full extension.

Due to limitations described in Chapter 3, the arm was not used for experimentation. It was, however, an important part of the system setup as it gave insight into the difficulties of controlling an EOD manipulator and gripper in joint space with an input device with fewer DOF than the robot.

2.4 Sensors

Several sensors were considered in order to acquire haptic information. Initially, Polyvinylidene fluoride pressure sensors, strain gages, accelerometers, and various other sensors were examined in order to measure applied pressure and the state of the robot. However, due to time and equipment limitations, it was decided to use the current information from the motor and accelerometer data, thereby measuring both applied forces and vibrational effects.

2.4.1 Accelerometer

In order to sense high-frequency vibration of the gripper, the Kistler Piezotron 3 DOF Accelerometer was used. In sensing early contact, vibrational effects from the discontinuity of contact are more important than applied forces. The underactuation



Figure 2.6: Northrop Grumman HD-2 Manipulator

and compliance of the manipulator essentially places several cascaded low pass filters between the finger tips and the base of the gripper. To account for this, while keeping the sensor out of the potential grasping area, the accelerometer was mounted on the distal phalanx of the single opposable finger as shown in Figure 2.7.

The values from the sensor were passed through a power supply/signal conditioner, and then read through an A/D input on an Arduino Duemilanove microcontroller.

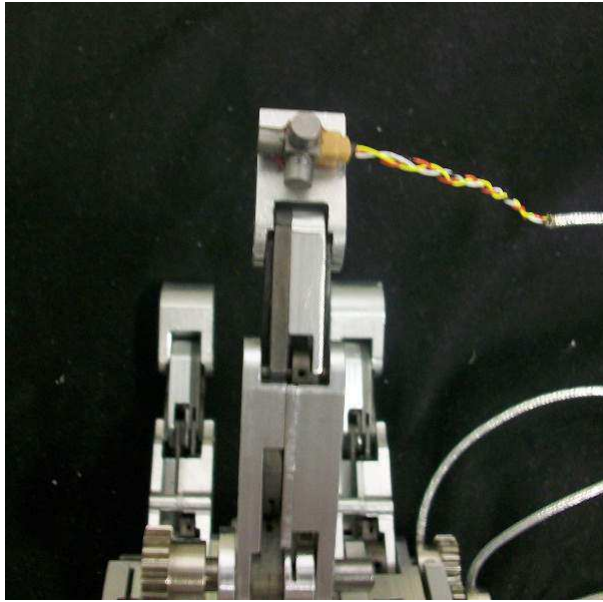


Figure 2.7: Mounted Kistler Accelerometer

2.5 Feedback Devices

Taking inspiration from several proven methods in robotic minimally invasive surgery, both surrogate visual feedback force feedback and surrogate vibrotactile force feedback were provided to the user.

2.5.1 Vibrotactor

The VPM vibrotactor was selected as a vibrotactor as it was small enough to be easily mounted to the input device as shown in Figure 2.8. Additionally, it drew little enough current such that it could safely be driven directly through the pulse width modulation channel of the microcontroller without any additional amplifying

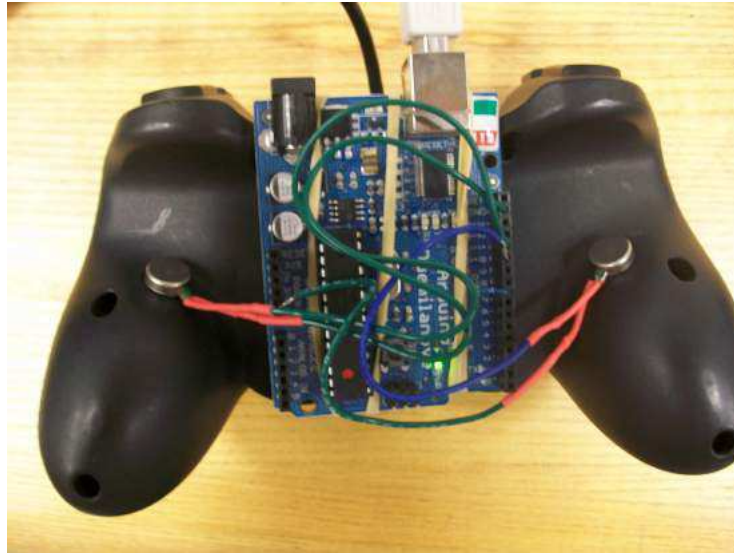


Figure 2.8: Mounted Vibrotractor

circuit. Such a vibrotactor is also known as a pager motor, due to their ubiquitous use in (originally) pagers and (now) cell phones to indicate an incoming message/call without a significant audible signal.

2.5.2 Graphical Feedback System

A graphic feedback system (Figure 2.9) was designed using MATLAB in order to guide the user through experimentation and also to provide camera information and visual force information. The system was built using the MATLAB GUI Development Environment and provided users with the camera feed, the visual force bar, the full state of the system (position, velocity, current), and the control frequency for purposes of debugging and ensuring that the system was working properly.

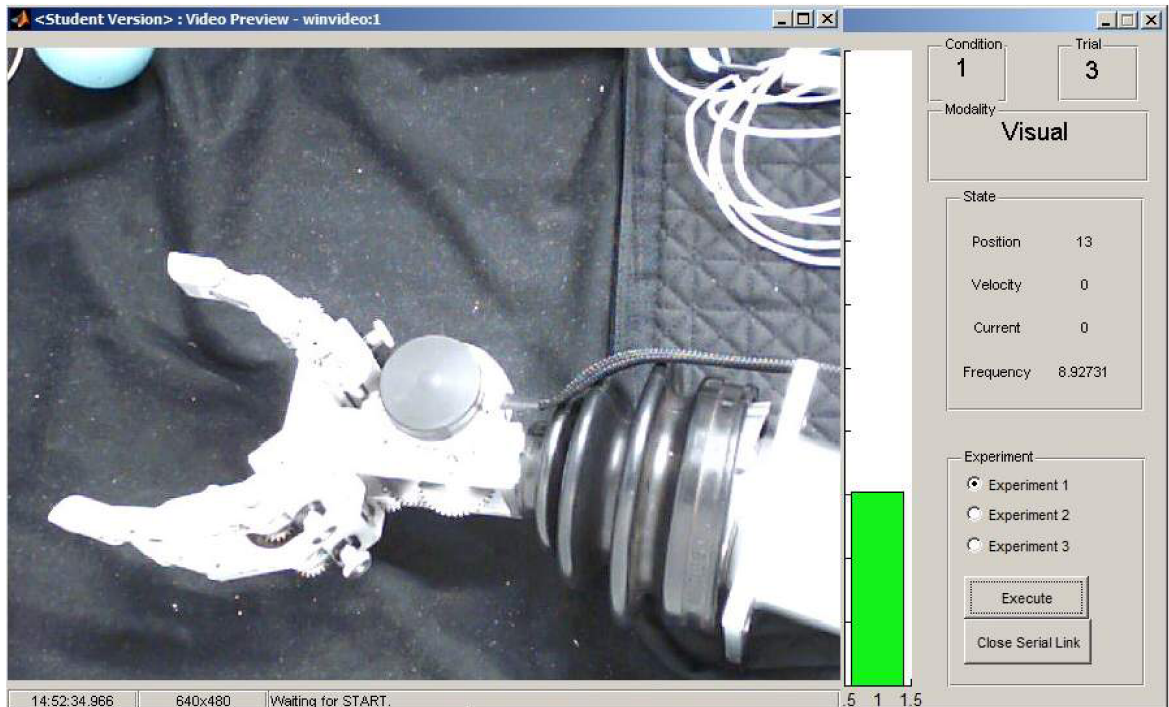


Figure 2.9: Graphic Feedback System

During experimentation, the feedback modality (no feedback, vibrotactile feedback, or visual feedback) was displayed to the user on the screen, as was the condition number and trial number. While the graphical user interface was originally designed to display information from the camera directly, it was found that the burden of computation in addition to communicating with hardware slowed the operating frequency (and hence the camera refresh rate) to an unacceptable level. By opening a separate window for the camera and placing where the camera feed should have been displayed, the performance of the system improved substantially.

2.6 System Integration

Due to the number of interworking parts used in experimentation, system integration was extremely important and also the most time consuming part of this research. Several important pieces were needed to operate the system successfully. First a microcontroller was needed in order to read sensor information through its A/D channel. It was also used as a simple interface for sending commands to the vibrotactor. Next a camera was needed to display visual information to the user. Finally, an accurate F/T sensor on an instrumented object was needed in order to accurately measure the forces being applied for purposes of data logging, as the information from the sensors on the robot were inaccurate and did not measure forces directly.

2.6.1 Microcontroller

In order to integrate the components of the system, an A/D converter was needed to read sensor information into MATLAB. Additionally, a variable voltage source was required in order to drive the vibrotactors.

The Arduino Duemilanove, shown in Figure 2.10 (reproduced from [41]), was chosen for its low cost and ease of use. It contains 14 digital input/output pins, including 6 that are capable of pulse width modulation, 6 analog input pins, 3.3 and 5V reference signals, and serial connection pins. The Arduino can be powered with a 9V battery

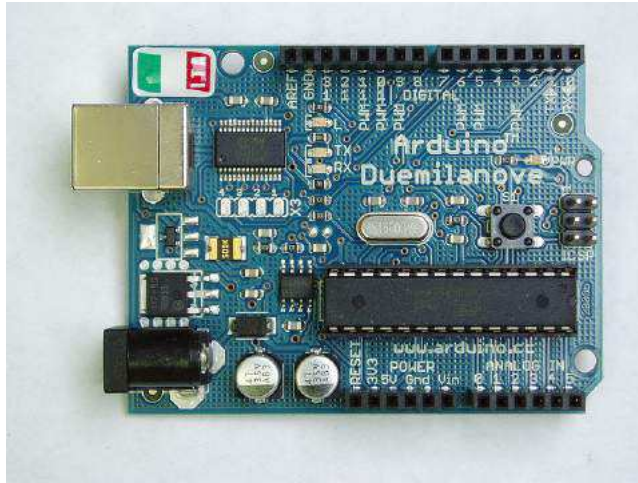


Figure 2.10: Arduino Duemilanove

or a USB connection and operates at a clock frequency of 16MHz and has 32KB of flash memory thanks to the ATmega328 chip that it employs. The Arduino is coded in C/C++ using the Wiring Library using an Integrated Development Environment.

The Arduino was used as an I/O device and also as an A/D converter. The Arduino was mounted to the back of the Dual Axis Gamepad. Commands were sent to the vibrotactors using the pulse width modulation pins. Commands were received from MATLAB as serial messages ranging from 0-100. These messages were scaled to binary levels (0-255) and supplied to the vibrotactor using the D/A channel. These levels corresponded to 0-5V respectively.

When MATLAB required data from the sensors wired to the Arduino, it sent the message 'p' (for ping) through the serial port. This result in the Arduino returning all of the applicable sensor data from the A/D converter in addition to a time stamp.



Figure 2.11: Logitech Quickcam

2.6.2 Digital Video Camera

Similar to the OCU's on which EOD technicians currently view output from cameras mounted to robots, the user viewed the scene through a camera feed rather than looking at it directly. In general, the displays on EOD robots do not tend to be high-fidelity systems. Additionally, the view from any given angle is typically occluded by either the robotic gripper or the robotic arm. As such, a camera was selected based on price only, without considering quality. The Logitech Quickcam (Figure 2.11) was suitable and readily available. It was read into MATLAB through a USB port and displayed to the user.

With both the camera display and communication through each part of the system running through MATLAB, the frame rate suffered significantly over what it might have been had the system been in a stand-alone custom program. Efforts were made

to increase the control frequency to an acceptable rate (~ 10 Hz) but some delay and digitization was desired in order to create a reasonable facsimile EOD telemanipulation.

An example of the video quality can be seen in Figure 2.9. Again, in order to replicate working environments, one finger of the robot was intentionally occluded and no attempt was made to fix the resolution of the camera feed.

Although the user was not able to directly view the object, the user was allowed to listen to auditory cues from the object and gripper motor. While the option of blocking the user's aural channel was considered, most EOD robots have a microphone and speaker system on the OCU through which the user can receive auditory information about the state of the robot. Thus, we allowed the natural aural feedback to remain. More accuracy could have been achieved by recording and playing the audio in sync with the video.

2.6.3 Force/Torque Sensor

In order provide measurements on the amount of force being exerted by the user during the experiments, an accurate force/torque (F/T) sensor was needed. The ATI Mini45 F/T sensor, shown in Figure 2.12, reproduced from [42], was chosen for its balance of package size and durability, as well as its relatively high sensitivity in all six degrees of freedom. The sensing range and resolution for forces and torques are displayed in Tables 2.5 and 2.6, respectively.

	F _x	F _y	F _z
Sensing Range	145 N	145 N	290 N
Resolution	1/16 N	1/16 N	1/16 N

Table 2.5: Sensing range and resolution of forces for the ATI Mini45

	T _x	T _y	T _z
Sensing Range	5 N-m	5 N-m	5 N-m
Resolution	1/752 N-m	1/752 N-m	1/1504 N-m

Table 2.6: Sensing range and resolution of torques for the ATI Mini45

To effectively utilize and protect the sensor, it was built into an instrumented object (Figure 2.13) onto which the robotic end effector could grip. In order to provide a sufficiently linear correlation between grasp position and grasp force, the force sensor was placed between two compliant objects. Each object was hemispherical and had a radius of 32mm. Each hemisphere was composed of Smooth-On OOMOO-25 Silicon Rubber (see Appendix B.4).

The process of curing the rubber involves mixing equal volumes of two compounds (A and B) together for 5 minutes, and then pouring into a mold and letting cure for 75 minutes. In order to produce hemispheres that follow Hooke's Law for a fairly large degree of compression, efforts were made to decrease the hardness of the resulting silicon compound. By violating the 1:1 ratio, it was found that the hardness of the compound could be controlled with a high degree of repeatability. The following ratios were tested: 1:3, 2:3, 1:1, 3:2, and 3:1.



Figure 2.12: ATI Mini45 Force/Torque Sensor

It was found that the hardness of the silicon was proportional to the content of compound A. In all cases other than the control, the curing times were significantly higher than the recommended 75 minutes. This was particularly true of the combinations with a high content of compound B. For the 1:3 combination, the cure time was on the order of 10 hours.

It was qualitatively found that the 1:3 silicon compound was selected and had significantly lower hardness than either the control compound or the 3:1 compound.

The F/T sensor was placed between the two silicon hemispheres and separated from them by an acrylic disk. Screws were set into the hemispheres, passed through the acrylic, and were secured to the F/T sensor. A slit was cut out of the left hemisphere in order to allow the data cable to exit the instrumented object properly.



Figure 2.13: Grasping object instrumented with the ATI Mini45 F/T Sensor - Pen for scale

2.6.4 Framework and Setup

The various parts of the system was linked together as shown in Figure 2.14. Two laptops were required, as 5 Universal Serial Bus (USB) ports were required in addition to a Personal Computer Memory Card International Association (PCMCIA) card.

Data logging took place on both laptops. On the first laptop, MATLAB logged the following information about the gripper and the input device: time, position, velocity, current, calculated pressure, user input, feedback mode. On the second laptop information from the F/T sensor was recorded. Although 6 measurements were available from the sensor, only the force in the Z direction (aligned with the

principle axis of the object) was used for analysis.

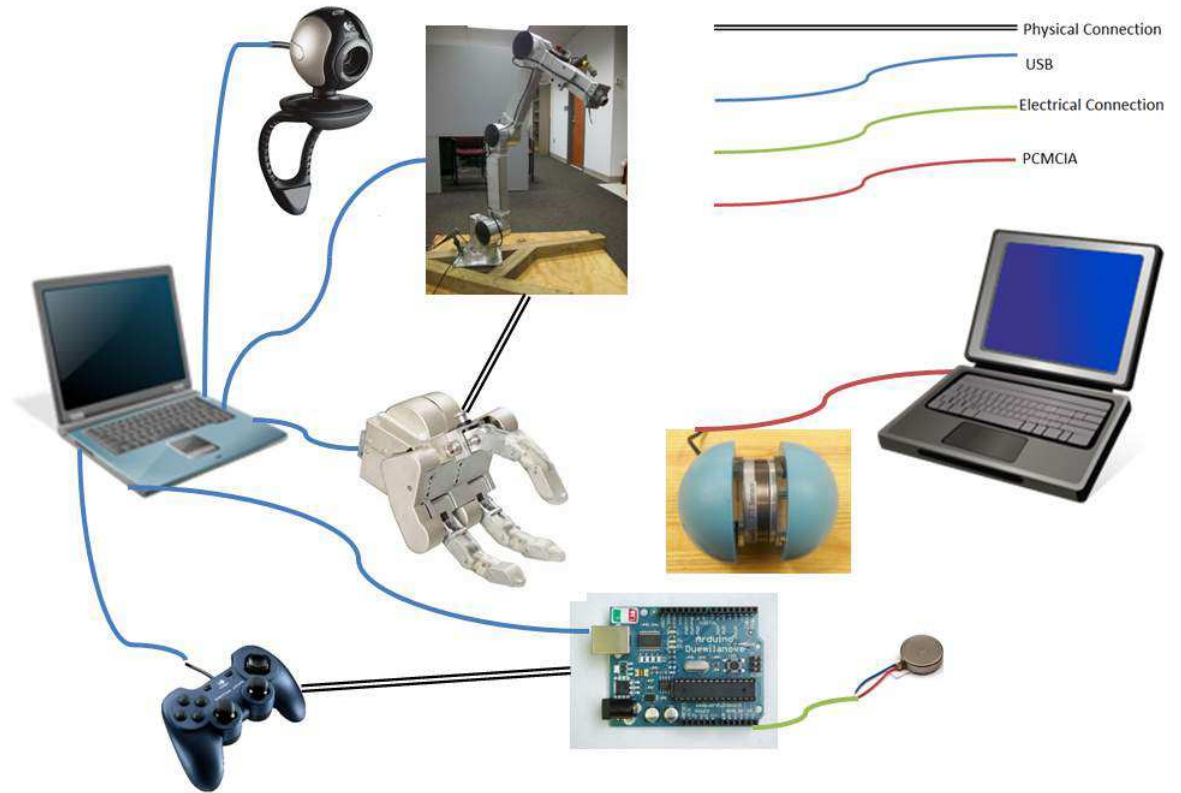


Figure 2.14: System Framework

Chapter 3

Experiment

Using the apparatus described in Chapter 2, an experiment was performed to test the users' ability to teleoperate the gripper to grasp the instrumented object with minimal force.

3.1 Preliminary Experiments

Several preliminary experiments took place in order to determine which experiments and methods would be most appropriate. First, the output from the gripper-mounted accelerometer was measured. Next the F/T sensor and current sensor output were measured in tandem to reveal their similarities or differences. Finally, early tests determined which control scheme would be the most effective for controlling the robot with the input device.

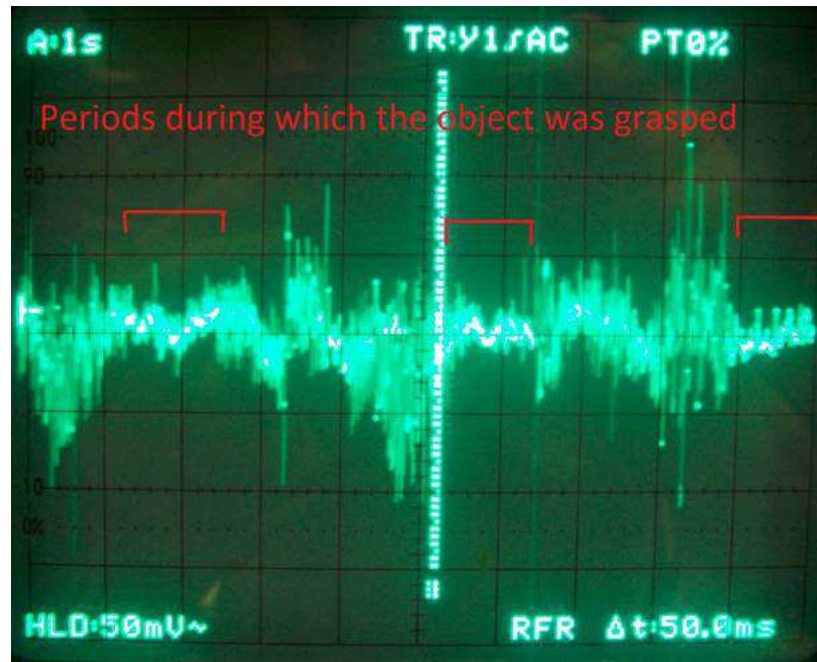


Figure 3.1: Accelerometer test output showing three separate grasps, noted in red, of the instrumented object

3.1.1 Accelerometer Test

Although accelerometer data has been used successfully in prior work [19], it was done with smooth, low backlash systems with no gearing. In contrast, our system was heavily geared and had significant backlash. An experiment was done to determine how this would affect the accelerometer output.

With the accelerometer mounted to the gripper, its output was read into an oscilloscope while the gripper moved through a series of poses. The gripper closed onto the compliant object, squeezed, and opened several times. The output at first seemed to indicate that the vibrations from the gearing masked the effects of grasping the object completely (Figure 3.1).

Upon further inspection and filtering, however, it was found that event detection could take place with the accelerometer, not by looking for increases in the signal where contact took place, but rather, where the signal is damped by the low-pass filter effect of the compliant object. Applying a frequency analysis of the signal (e.g., applying a Fast Fourier Transform) likely could have made the data more readily usable, but an adequate means of reading the signal and applying the transform was not immediately available. Although the data showed that the accelerometer was technically usable, it was decided that the sensor should not be used, as the identification of contact would not be consistent between compliant objects and rigid ones.

3.1.2 F/T Sensor and Current Sensor Test

In order to test the effectiveness of the current sensor data, its output was compared against the F/T sensors. Data from both was logged with both which the gripper contacting, squeezing, and releasing the object four times in succession. Each squeeze was intended to be harder than the last.

As can be seen, the data from the F/T sensor (Figure 3.2), provided information clearly showing each grasp as it took place. As intended, the strength of each grasp increased from the one before it.

This information can be compared to the output from the calculated torque information using the current sensor (Figure 3.3). First, it is obvious that there are

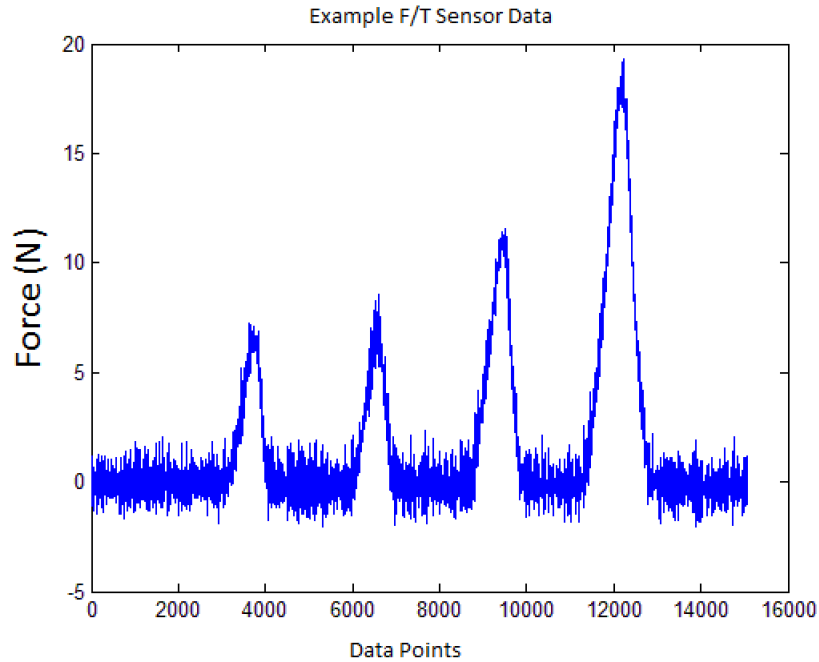


Figure 3.2: F/T output during four successive grasps of the instrumented object

five apparent contacts rather than four. The fourth is an artificial contact brought about by improper modeling of the acceleration and friction effects of the motor. This tends to occur when the user applies a “step input” by pushing the joystick all the way forward rapidly. When the user operates the gripper more slowly and gradually, the false contacts rarely take place.

Ignoring the false contact shows that the gripper made contact with the object four separate times and that each contact was with more force than the last. However, the proportions are not the same as they were from the F/T sensor. As such, the output from the current sensor is not able to provide the user with direct information about the forces being applied. Additionally, the fourth contact saturates the output,

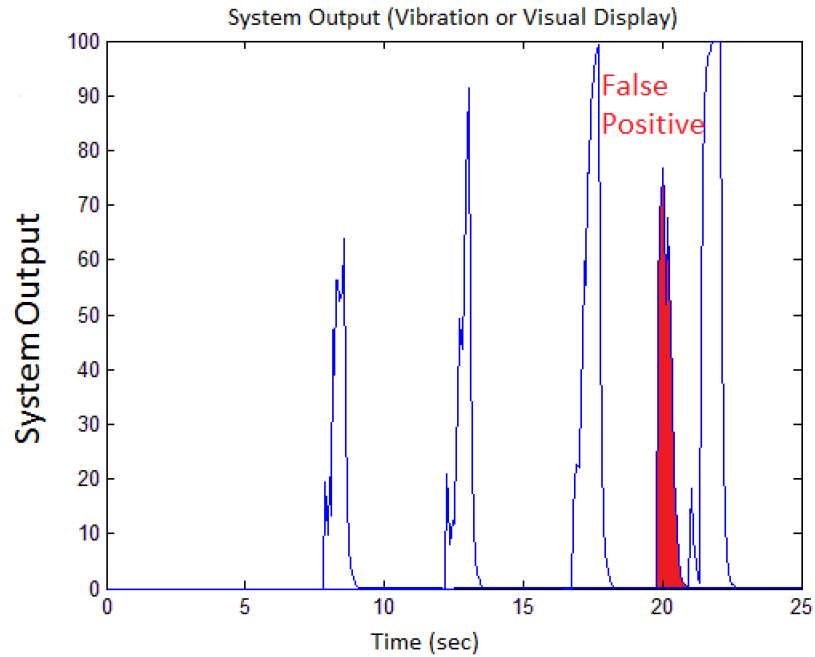


Figure 3.3: Corresponding GUI output during four successive grasps of the instrumented object

preventing it from providing additional information.

3.1.3 Control Methodology

While all current EOD robots use velocity control in joint space, several other control options were examined. First, a position control scheme was created where the desired position, θ_d was determined by the position of the joystick. Pushing the joystick as far forward as possible set θ_d to 400, while pulling the joystick back entirely set θ_d to 0. The desired position was then approached using a PID controller.

The second control scheme again used position control, but rather than the joystick specifying θ_d directly, it supplied the velocity of θ_d . Pushing the joystick all the way

forward supplied a ramp input to the same PID controller. This scheme would have felt similar to velocity control, but would have differed in that the former when given a command of 0 would have continued to try to reach its desired position whereas the latter would have immediately stopped.

Finally, velocity control was examined and was ultimately found to provide much better performance than the other schemes. This is in part due to low sampling rate, which caused instability during position control. This instability could be decreased by adjusting gains, but the responsiveness of the system suffered as a result. Ultimately, velocity control was used for all further experimentation.

3.2 Methods

Three experiments were originally intended. The first would test the effects of haptic feedback on the user's ability to detect contact with an object. The second would test the effects of haptic feedback on the user's ability to accurately apply a given level of force to an object. The third tested the system qualitatively in a situation similar to the operational environment.

Because of the results of the F/T and Current Sensor test, the second experiment was not seen as applicable as the force information gained from the current sensor was indirect at best. Time was the limiting factor for the third experiment, though it is a priority for future work. Thus, the experiment described below is for the contact

detection task.

3.2.1 Procedure

This experiment measured the effectiveness of the system in decreasing peak and sustained forces applied by the user in a contact/grasping task. The user was given control of the gripper via the joystick on the gamepad. The user was then instructed to close the gripper as lightly as possible until two opposing fingers came into contact with an object instrumented with the F/T sensor. The object and manipulator were placed such that the fingers of the gripper closed around the principal axis of the instrumented object (Figure 3.4). The user was allowed to use the following forms of information in order to detect when contact had occurred: visual information through the camera display, a surrogate force feedback visual display, or a surrogate vibrotactile force feedback display.

Five subjects were recruited between the ages of 24 and 28. Three were male, two were female. Four were right-hand dominant, one was left-hand dominant. None of the subjects had any neurological disorders, injuries to their dominant hand, impaired vision, or any other circumstance which might affect their ability to successfully perform the task. The users gave informed consent. The protocol was approved by the Johns Hopkins University Institutional Review Board.

Before the trials began, each user took a brief pre-experiment survey. The subject was shown the experimental setup, including the robotic gripper. Then the subject



Figure 3.4: The setup of the gripper and instrumented object during the experiment was seated such that he or she could not see the gripper and instrumented object except through the camera setup. After explaining the types of feedback to expect, the subject was allowed to freely test the system with all feedback modes present for an unlimited period of time. Following this, the subject notified the researcher that he or she was ready to begin the experiment. The subject was instructed to press the “2” button in order to start the experiment.

After pressing “2” for the first time, the GUI randomly selected the first feedback modality to be given to the user. This information was displayed on the GUI so that the subject would know what to expect.

The following exchange then took place:

Experimenter: “Close the gripper.” The subject would then proceed to close the gripper.

Subject: “Done.” The subject would respond as such when they believed they were in contact with the object. The experimenter would check and then respond.

Experimenter: “Good. Open the gripper and press 2” OR “No contact. Close the gripper more.”

Pressing the “2” button during a trial would end that trial and begin the next trial. The same procedures were used in each trial. The trial number was displayed on the GUI. After every 5 trials, the GUI would randomly select one of the remaining feedback modalities.

At the end of the experiment, the user was asked to take a brief post-experiment survey. The user ranked the performance of the task under each feedback condition from the following options:

- (1) Very Easy
- (2) Easy
- (3) Moderate
- (4) Hard
- (5) Very Hard

Following this, the user was asked to comment on which strategies he or she used for each task and any further comments.

	Sub1	Sub2	Sub3	Sub4	Sub5	Average	StDev
None	7.403	12.1938	7.231	3.4742	11.8576	8.43192	2.875024
Visual	6.2792	5.285	6.1068	7.699	7.8926	6.65252	0.914624
Vib	7.2942	3.0082	7.0538	3.899	8.628	5.97664	2.018432

Table 3.1: Average applied force from each user in Newtons

3.3 Results

The peak force from each trial was extracted. A mean was taken for each set of 5 data points corresponding to an individual using a single feedback modality. The results are displayed in Table 3.1. The average peak for over all users for each modality was then taken and is displayed in Figure 3.5. As can be seen, both the vibrotactor and the visual feedback modes assisted the user in discerning the contact event.

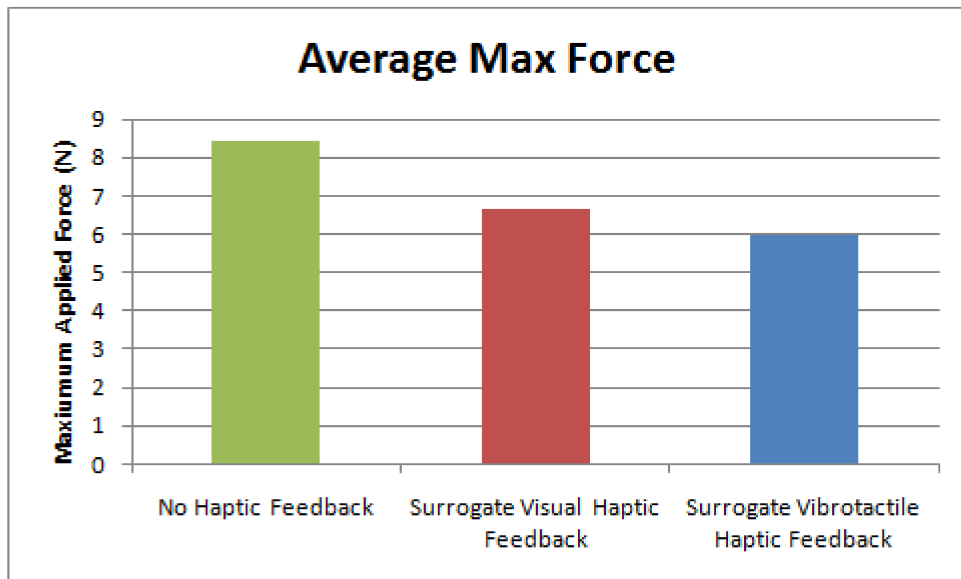


Figure 3.5: Plot of mean peak forces applied to the instrumented object, averaged for all subjects and all trials for each condition

In order to determine whether users' performance under the different feedback modalities were statistically significantly different, ANOVA was used with a α_{FW} of .05. Box's epsilon-hat adjustment was used to correct for violations of sphericity. The statistical results are shown in Table 3.2.

Post-hoc comparisons with a family-wise alpha level of 0.05.
Results in per-comparison alpha level of 0.0167

Comparison	F	df1	df2	P	significant?
1 vs. 2	3.5325	1	24	0.0724	n
1 vs. 3	7.1293	1	24	0.0134	y
2 vs. 3	0.8336	1	24	0.3703	n

Table 3.2: Table of statistical significant. (1) No feedback, (2) Surrogate Visual Feedback, (3) Surrogate Vibrotactile Feedback

The improvements in performance of vibrotactile feedback over no feedback were statistically significant, while the performance due to the visual feedback were not.

In addition to the experimental data, users' preferences as stated on their surveys was collected. This data was placed into Table 3.3 and the average results for each modality are displayed in the Figure 3.6.

As the data shows, improvements were made by giving the user haptic feedback. The vibrotactile feedback provided larger improvements for this task and was also

	Sub1	Sub2	Sub3	Sub4	Sub5	Average	StDev
None	4	3	3	2	3	3	0.4
Visual	3	2	3	4	3	3	0.4
Vib	1	1	2	3	1	1.6	0.72

Table 3.3: Post-experiment survey average results. (1) - Very Easy, (2) - Easy, (3) - Moderate, (4) - Hard, (5) - Very Hard

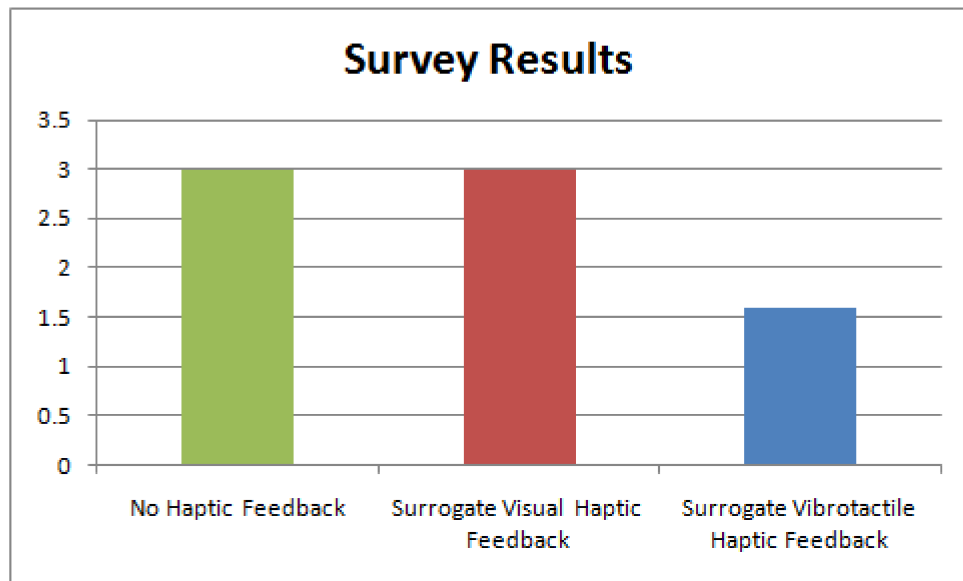


Figure 3.6: Post-experiment survey ratings

statistically significant, with a p value of .0134. The detection threshold was reduced on average from 8.43 N to 5.97 N in the no feedback, and vibrational feedback cases, respectively. Although these gains were substantial, responses from users were equally informative.

In the comments section of the survey, the subjects as a group correctly identified where their performance improved. However, as individuals, subjects only correctly identified their best performance case 60% of the time. They did, however identify their worst performance 100% of the time, although some users marked two modes as equally difficult.

The most frequent response was that the visual feedback was either “ignored” or “not helpful” or “distracting”. The majority of the subjects claimed to rely heavily on

the motion of the object in all cases, and used the vibration or visual bar as a check. As this object was both compliant and lightweight, it suggests that this feedback may be even more helpful in the event that a heavy, rigid object, such as a piece of ordnance, needs to be manipulated.

Chapter 4

Conclusions

Teleoperation systems that operate without haptic feedback are significantly limited in what they can accomplish when compared to high-performance haptic feedback systems. The benefits of even limited amounts of feedback have been shown in literature, and in the experiments described here, to be substantial. Despite this, no currently fielded EOD robotic systems display any type of force information to the user. This seriously impedes the ability of an EOD technician to work on a piece of ordnance remotely and limits them to gross manipulation and pick-and-place tasks. Cost-effective, robust solutions to this problem are particularly applicable to the EOD environment.

4.1 Contributions

In the first chapter, the perceived benefits of haptic feedback for EOD are described. While actual force feedback can increase user performance, the use of sensory substitution, in the form of a visual, audio, or tactile display can provide the information to the user without dynamically affecting the use of the input device. The history of robotics in Explosive Ordnance Disposal is described as well. While this review is not a comprehensive description of the numerous projects that have taken place over the years, it is, to the best of the author's knowledge, the most thorough examination of the topic in a single document.

In the second chapter, the experimental setup is described. This setup consisted of a robotic gripper, manipulator arm, camera setup, microcontroller and various other components. The chapter focused on the successful integration of these components.

Finally, in the third chapter the experimental methods and results are described. The experiment tested users' ability to detect a contact event with two types of surrogate haptic feedback. Vibrotactile feedback was found to reduce the threshold at which the user detected contact from 8.43 N to 5.97 N on average. Additionally, in a survey given to users at the end of the experiment, users were found to prefer this type of feedback over the other conditions. Surrogate visual feedback did not substantially increase user performance and was not well received by the subjects.

4.2 Future Work

This research can be continued to further explore the best design practice and performance measures for haptic feedback for teleoperated EOD robots.

4.2.1 Additional Experiments

As part of this work, we designed two additional experiments that have not yet been performed, as they required additional sensors.

4.2.1.1 Sustained Force Experiment

Having established the effects of haptic feedback on user contact forces, another experiment should measure the ability of the system to assist the user in repeating grasps of a constant and sustained force. The user would be given control of the manipulator without needing control of the robotic arm. The gripper would be placed in a position to contact the principal axis of the instrumented object when it was closed. The user would then be trained to know when a certain force threshold had been hit. The information given from the current sensor is not sufficient for this task as the system is not backdrivable, and the gearing, rather than the current, holds the sustained force. Thus, this experiment would only be possible if other sensors, such as strain gages, could be used to sense applied force.

The subject would slowly close the gripper while observing a given feedback

method. Feedback would be given from the computer when the subject reached a particular force threshold. This training would be repeated five times. The feedback methods that the user is allowed to observe would be the following: visual, surrogate visual, and surrogate vibrotactile.

Following training, the subject would be asked to achieve the given force threshold using only the information given by the selected feedback modality.

4.2.1.2 Real-World Task

Another important experiment would replicate a real-world task that an EOD technician would likely undertake. Using this task, qualitative data would be gained correlating the gains from the first two tasks, to real world gains on actual mission performance. A domain expert is critical for this experiment in order to provide information on specific benefits gained through haptic feedback.

4.2.2 Further Areas

There are several additional areas where future work may be beneficial. First, the experiments focused primarily on near-term haptic technologies for the EOD environment. Further research needs to be done to explore the effectiveness of a larger variety of feedback modalities, particularly those that have potential to be significant enabling technologies in the future. Bilateral manipulator arms and kinesthetic feedback through the manipulator should be examined in full, as should the full range of

near-term sensory substitution technologies that were not examined in this work.

Additionally, research should be done to generalize the results for manipulators that are truly dexterous and possess degrees of freedom on the order of the human hand. Doing so would allow for further applications of this technology to later iterations of EOD robots, which will likely feature components that are anthropomorphic and dexterous.

Finally, the setting which EOD robots are fielded add significant further constraints to teleoperation and the design of useful feedback systems. Specifically, the effects of digitization and delay are both significant and may have important implications for the effectiveness of various types of feedback modalities. These system properties should be measured and techniques should be explored in order to minimize their detrimental effects. In addition, there are significant constraints on the size of mobile OCUs and operator input devices.

It is imperative that work such as this continues so that EOD technicians can have the best possible equipment in the field.

Bibliography

- [1] H. G. Nguyen and J. P. Bott, “Robotics for Law Enforcement: Applications Beyond Explosive Ordnance Disposal,” in *Proceedings of SPIE International Symposium on Law Enforcement Technologies*, 2000.
- [2] Department of Defense, “Casualty report,” 2011, <http://www.defense.gov/news/casualty.pdf>.
- [3] C. Wilson, “Improvised Explosive Devices (IEDs) in Iraq and Afghanistan: Effects and Countermeasures,” 2007.
- [4] A. M. Bottoms and C. Scandrett, Eds., *Applications of Technology to Demining*. Monterey, California: Society for Counter Ordnance Technology, 2002.
- [5] GlobalSecurity.org, www.globalsecurity.org.
- [6] Rear Admiral Michael P Tillotson, USN, *Navy EOD: Then and Now - Frogmen, UDTs, SEALs and Explosive Ordnance Disposal Teams from World War II to*

- Iraq*, United States Navy Memorial, 2010, comments as a Panelist at the United States Navy Memorial on 13OCT10.
- [7] R. Potter, “How to compete with offshore low labor costs: Employ highly skilled labor at 30 cents per hour,” <http://www.robotpackaging.com>.
- [8] Kawasaki Robotics, www.kawasakirobotics.com.
- [9] J. Carlson and R. Murphy, “How UGVs physically fail in the field,” *IEEE Transactions on Robotics*, vol. 21, no. 3, June 2005.
- [10] Public Broadcasting System, “The cost of land mines,” <http://www.pbs.org/saf/1201/features/landmines2.htm>.
- [11] J. Krasner, “Robots going in harms way,” *The Boston Globe*, March 2007, <http://www.boston.com/business/technology/articles/2007/03/12/robotsgoinginharmsway>
- [12] G. Niemeyer, C. Preusche, and G. Hirzinger, *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. New York, NY: Springer, 2008.
- [13] M. J. Massimino, “Improved force perception through sensory substitution,” *Control Engineering Practice*, vol. 3, no. 2, pp. 215–222, February 1995.
- [14] J. C. Gwilliam, M. Mahvash, B. Vagvolgyi, A. Vacharat, D. D. Yuh, and A. M. Okamura, “Effects of haptic and graphical force feedback on teleoperation palpation,” in *International Conference on Robotics and Automation*, 2009.

-
- [15] M. Kitagawa, “Indirect feedback of haptic information for robotassisted telemanipulation,” Master’s thesis, The Johns Hopkins University, September 2003.
- [16] M. Kitagawa, D. Dokko, A. M. Okamura, and D. D. Y. and, “Effect of sensory substitution on suturemanipulation forces for robotic surgical system,” *The Journal of Thoracic and Cardiovascular Surgery*, vol. 129, no. 1, pp. 151–158, 2003.
- [17] C. E. Reiley, “Evaluation of augmented reality alternatives to direct force feedback in robot-assisted surgery: Visual force feedback and virtual fixtures,” Master’s thesis, The Johns Hopkins University, April 2007.
- [18] R. E. Schoonmaker and C. G. L. Cao, “Vibrotactile force feedback system for minimally invasive surgical procedures,” in *IEEE Conference on Systems, Man, and Cybernetics*, October 2006.
- [19] K. Kuchenbecker, J. Gewirtz, W. McMahan, D. Standish, P. Martin, J. Bohren, P. Mendoza, and D. Lee, “Verrotouch: High-frequency acceleration feedback for telerobotic surgery,” in *Lecture Notes in Computer Science*, vol. 6191, 2010, pp. 189–196.
- [20] M. R. Tremblay and M. R. Cutkosky, “Using sensor fusion and contextual information to perform event detection during a phase-based manipulation task,” in *International Conference on Intelligent Robots and Systems*, August 1995.

-
- [21] J. M. Hyde, M. R. Tremblay, and M. R. Cutkosky, “An object-oriented framework for event-driven dextrous manipulation,” in *4th International Symposium on Experimental Robotics*, June 1995.
- [22] J. D. Bartleson, *History of U.S. Navy Bomb Disposal*. Virginia Beach, Virginia: U.S. Navy Explosive Ordnance Disposal Association, 1992.
- [23] A. B. Hartley, *Unexploded Bomb*. New York, New York: W W Norton and Company Inc, 1958.
- [24] Byron Brezina et al., “Analysis of Alternatives Advanced Explosive Ordnance Disposal Robot System,” 2008.
- [25] The Times, “Lieutenant-Colonel ‘Peter’ Miller: Inventor of the Wheelbarrow remote control bomb disposal device that saved countless lives,” 2006.
- [26] P. Birchall, *The Longest Walk: The World of Bomb Disposal*. London: Arms and Armour Press, 1997.
- [27] Defense Industry Daily, www.defenseindustrydaily.com.
- [28] Byron Brezina. Telephone correspondence. Interview conducted on 12DEC10.
- [29] iRobot, www.iRobot.com.
- [30] Hydroid Incorporated, www.hydroidinc.com.
- [31] Rob Simmons. Telephone correspondence. Interview conducted on 23NOV11.

- [32] C. Debolt, Ed., *Applications of Technology to Demining*. Society for Counter Ordnance Technology, 2002, ch. The BUGS "Basic UXO Gathering System" project for UXO clearance & mine countermeasures.
- [33] T. N. Nguyen, C. O'Donnell, and T. B. Nguyen, "Multiple autonomous robots for UXO clearance, The Basic UXO Gathering System (BUGS) Project," vol. 3.
- [34] "Lightweight robot for demining," *Applications of Technology to Demining*, vol. 2, no. 1, 2002.
- [35] J. D. Nicoud, "Vehicles and robots for humanitarian demining," vol. 24, pp. 164–168.
- [36] M. Buehler, "RTK - Remote Touch Kit: Final Technical Report and Test Results," September 2010, Report prepared for iRobot.
- [37] A. Kron, G. Schmidt, B. Petzold, M. I. Zah, P. Hinterseer, and E. Stenbach, "Disposal of explosive ordnance by use of a bimanual haptic telepresence system," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, May 2004, pp. 1968 – 1973.
- [38] Panel on Human Factors in the Design of Tactical Display Systems for the Individual Soldier, National Research Council, *Tactical Displays for Soldiers: Human Factors Considerations*. National Academies Press, January 1997.
- [39] The Johns Hopkins University Applied Physical Laboratory Revolutionizing

Prosthetics 2009 Team, “Revolutionizing Prosthetics 2009 MATLAB Repository.”

[40] Contineo Robotics, www.contineo-robotics.com.

[41] M. Banzi and D. Cuartielles, www.arduino.cc.

[42] ATI Industrial Automation, www.ati-ia.com.

Appendix A

Code

A.1 HapGui.m

```
1 function varargout = HapGui(varargin)
2 % HAPGUI M-file for HapGui.fig
3 %     HAPGUI, by itself, creates a new HAPGUI or raises the existing
4 %     singleton*.
5 %
6 %     H = HAPGUI returns the handle to a new HAPGUI or the handle to
7 %     the existing singleton*.
8 %
9 %     HAPGUI('CALLBACK',hObject,eventData,handles,...) calls the local
10 %    function named CALLBACK in HAPGUI.M with the given input arguments.
11 %
12 %     HAPGUI('Property','Value',...) creates a new HAPGUI or raises the
13 %     existing singleton*. Starting from the left, property value pairs are
14 %     applied to the GUI before HapGui_OpeningFcn gets called. An
15 %     unrecognized property name or invalid value makes property application
16 %     stop. All inputs are passed to HapGui_OpeningFcn via varargin.
17 %
18 %     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 %     instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help HapGui
24
25 % Last Modified by GUIDE v2.5 19-Jan-2011 12:26:50
```

```

26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                   'gui_Singleton',  gui_Singleton, ...
31                   'gui_OpeningFcn', @HapGui_OpeningFcn, ...
32                   'gui_OutputFcn',  @HapGui_OutputFcn, ...
33                   'gui_LayoutFcn',  [], ...
34                   'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if narginout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46
47 % --- Executes just before HapGui is made visible.
48 function HapGui_OpeningFcn(hObject, eventdata, handles, varargin)
49 % This function has no output args, see OutputFcn.
50 % hObject    handle to figure
51 % eventdata  reserved - to be defined in a future version of MATLAB
52 % handles    structure with handles and user data (see GUIDATA)
53 % varargin   command line arguments to HapGui (see VARARGIN)
54 handles.arduino = 1;
55 if handles.arduino
56     s1 = serial('COM4'); %define serial port for the sensor board input
57     s1.BaudRate=9600; %define baud rate
58     fopen(s1); %open serial port
59     handles.s1 = s1; %Establishes a global variable for accessing the serial port
60 end
61
62 handles.run = 1; %Establishes the global variable "run"
63 handles.numTrials = 0; % Keeps track of which individual trial the user is on
64 handles.conditionNum = 1;
65 handles.maxConditions = 3;
66 handles.maxTrials = 5; % How many trials the user will do with each setting
67 handles.w = rand(3,1);
68 disp('Arduino... READY');
69 disp('Updating Paths...');
70 cd C:\Users\owner\Desktop\TEMPBurtness\RP2009\VRE\Common
71 addpath_Common
72 disp('Paths Updated - READY');
73 disp('Initializing Manipulator and Controller...');
74 J = JavaJoystick;
75 M = MainDrive;

```

```

76 handles.M = M;
77 handles.J = J;
78 handles.pressure = 0;
79 %handles.vid = videoinput('winvideo', 1, 'YUY2_320x240');
80 %vid = handles.vid;
81 %vid.ReturnedColorSpace = 'grayscale';
82 disp('Initialization Complete');
83 who
84 tic
85 handles.command = 0;
86 guidata(hObject, handles);
87 % UIWAIT makes HapGui wait for user response (see UIRESUME)
88 % uiwait(handles.figure1);
89
90
91 % --- Outputs from this function are returned to the command line.
92 function varargout = HapGui_OutputFcn(hObject, eventdata, handles)
93 % varargout cell array for returning output args (see VARARGOUT);
94 % hObject handle to figure
95 % eventdata reserved - to be defined in a future version of MATLAB
96 % handles structure with handles and user data (see GUIDATA)
97
98 % Get default command line output from A handles structure
99
100
101 % --- Executes on button press in VisualForce.
102 function VisualForce_Callback(hObject, eventdata, handles)
103 % hObject handle to VisualForce (see GCBO)
104 % eventdata reserved - to be defined in a future version of MATLAB
105 % handles structure with handles and user data (see GUIDATA)
106
107 % Hint: get(hObject,'Value') returns toggle state of VisualForce
108
109
110 % --- Executes on button press in vibroForce.
111 function vibroForce_Callback(hObject, eventdata, handles)
112 % hObject handle to vibroForce (see GCBO)
113 % eventdata reserved - to be defined in a future version of MATLAB
114 % handles structure with handles and user data (see GUIDATA)
115
116 % Hint: get(hObject,'Value') returns toggle state of vibroForce
117
118
119 % --- Executes on button press in Execute.
120 function Execute_Callback(hObject, eventdata, handles)
121 % hObject handle to Execute (see GCBO)
122 % eventdata reserved - to be defined in a future version of MATLAB
123 % handles structure with handles and user data (see GUIDATA)
124
125 M = handles.M;

```

```
126 J = handles.J;
127 Kspeed = .08; %Increases the gain of the speed
128 lastButton = 1;
129 trial = 1;
130 w = rand(3,1);
131 data = cell(3,5);
132 dataSamp = 1;
133 currentData = zeros(10000,7);
134 tStart = tic;
135
136 for i = 1:3
137     for j = 1:5
138         data(i,j) = mat2cell(zeros(10000,2));
139     end
140 end
141
142 while handles.conditionNum <= handles.maxConditions
143     while handles.numTrials <= handles.maxTrials% While you're not done with all 5 trials
144         while handles.run == 1 %While you're not done with this specific trial.
145             %% Selecting a controller.
146             load VOID
147             %load polynomialValues;
148
149             if handles.conditionNum > 1 && handles.numTrials == 0
150                 handles.numTrials = 1;
151             end
152
153             if handles.numTrials == 0
154                 VibForce = 1;
155                 VisForce = 1;
156             end
157             handles.w
158             handles.run
159             maxWind = 0;
160             if handles.numTrials > 0;
161                 [maxW, maxWind] = max(w);
162                 if maxWind == 1
163                     VibForce = 0;
164                     VisForce = 0;
165                     set(handles.modeText,'String', 'None')
166                 elseif maxWind ==2
167                     VibForce = 0;
168                     VisForce = 1;
169                     set(handles.modeText,'String', 'Visual')
170                 else
171                     VibForce = 1;
172                     VisForce = 0;
173                     set(handles.modeText,'String', 'Vibration')
174                 end
175             else
```

```
176     set(handles.modeText,'String', 'ALL')
177 end
178
179 J.getdata;
180 if J.buttonVal(2) == 1
181     set(handles.controlBox,'Value', 0)
182     handles.run = 0;
183     M.normalizedVelocity = 0;
184     pause(.5);
185 elseif J.buttonVal(1) ==1
186     set(handles.controlBox,'Value', 1)
187 elseif J.buttonVal(3)==1
188     set(handles.controlBox,'Value', 1)
189 else J.buttonVal(4)==1
190     set(handles.controlBox,'Value', 1)
191 end
192
193
194 %% Running the controller
195 if get(handles.controlBox,'Value')
196     velocityControl;
197 else
198     M.normalizedVelocity =0;
199 end
200
201 %% Get Data
202 set(handles.trialText,'String', num2str(handles.numTrials));
203 set(handles.conditionText, 'String', num2str(handles.conditionNum));
204
205 if exist('velocity', 'var') == 0
206     handles.velocity = 0;
207 end
208 if exist('accleration', 'var') == 0
209     handles.acceleration = 0;
210 end
211
212 motorData = M.get_data;
213 command = handles.command;
214 handles.oldCommand = command;
215 oldCommand = handles.oldCommand;
216 command = J.axisVal(4);
217 handles.command = command;
218 %disp(oldCommand)
219 %disp(command)
220
221 handles.oldVelocity = handles.velocity;
222 handles.velocity = M.motorActualVelocity;
223 delay = toc;
224 tic;
225 jerk = -(command - oldCommand);
```

```

226 handles.acceleration = (handles.velocity -handles.oldVelocity)/(180*delay)*.1+ handles.accele
227 % disp(handles.acceleration)
228 if handles.acceleration > 2
229     handles.acceleration = 2;
230 end
231 if handles.velocity > 50
232     handles.acceleration = 0;
233 end
234 set(handles.positionText, 'String', M.motorActualPosition);
235 set(handles.velocityText, 'String', M.motorActualVelocity);
236 set(handles.currentText, 'String', M.motorActualCurrent);
237 set(handles.frequencyText, 'String', 1/delay);
238
239 thetaTest =[40  120  200  280  360];
240 [x,bestTheta] = min(abs(M.motorActualPosition-thetaTest));
241
242 % Figuring out which boxes have been checked
243 %VisForce = get(handles.VisualForce,'Value');
244 %VibForce = get(handles.vibroForce,'Value');
245
246 pressure = M.motorActualCurrent;
247 J.getdata;
248 yData = J.axisVal(4);
249 if length(motorData) > 5
250 if -yData*Kspeed > 0 && motorData(6) > 10
251     estAccelTorque = 0;
252     if handles.acceleration
253         estAccelTorque = 0; %polyval(PolynomialValuesAccel, handles.acceleration)
254     end
255     %estVelTorque = polyval(cell2mat(PolynomialValuesDesiredVelocity(bestTheta)), -J.axisVal(
256     estVelTorque = polyval(cell2mat(PolynomialValuesActualVelocity(bestTheta)), handles.veloc
257     if sign(jerk)
258         estJerk = 50+75*jerk;
259     else
260         estJerk = 0;
261     end
262     pressure = get(handles.Gain, 'Value')*(pressure - estAccelTorque - estVelTorque - estJer
263 else
264     pressure = 0;
265 end
266 end
267
268 if pressure > 100
269     pressure = 100;
270 end
271
272 if pressure < 0
273     pressure = 0;
274 end
275

```



```

276 disp(handles.run)
277
278 pressureOld = handles.pressure;
279 handles.pressure = pressureOld*.5 + pressure*.5;
280 pressure = handles.pressure;
281
282 %Display data as asked by the system.
283 if VisForce ==1;
284     axes(handles.VisBar)
285     hold on;
286     forcebarTest(pressure)
287 end
288
289 if VibForce == 1 && handles.arduino
290     fwrite(handles.s1, pressure);
291 end
292
293 currentData(dataSamp,:) = [toc(tStart), pressure, yData(1,1), M.motorActualCurrent(1,1), M.mo
294 dataSamp = dataSamp + 1;
295
296 drawnow; % Command needed to have the plot reset
297 guidata(hObject,handles);
298     end %End of trial
299     if handles.numTrials >0
300         data(handles.conditionNum, handles.numTrials) = mat2cell(currentData);
301     end
302     handles.numTrials = handles.numTrials+1;
303     set(handles.controlBox,'Value', 1)
304     handles.run = 1;
305     dataSamp = 1;
306     tStart = tic;
307     currentData = zeros(10000,7);
308     end % End of condition
309     handles.numTrials = 1;
310     w(maxWind) = 0;
311     handles.conditionNum = handles.conditionNum +1;
312 end %End of experiment
313 handles.numTrials = 1;
314 handles.conditionNum = 1;
315 save('testData.mat', 'data')
316
317
318
319 % --- If Enable == 'on', executes on mouse press in 5 pixel border.
320 % --- Otherwise, executes on mouse press in 5 pixel border or over VisualForce.
321 function VisualForce_ButtonDownFcn(hObject, eventdata, handles)
322 % hObject    handle to VisualForce (see GCBO)
323 % eventdata  reserved - to be defined in a future version of MATLAB
324 % handles    structure with handles and user data (see GUIDATA)
325

```

```

326
327 % --- Executes on button press in closeSerial.
328 function closeSerial_Callback(hObject, eventdata, handles)
329 % hObject    handle to closeSerial (see GCBO)
330 % eventdata  reserved - to be defined in a future version of MATLAB
331 % handles    structure with handles and user data (see GUIDATA)
332
333 fclose(handles.s1);
334
335 guidata(hObject, handles);
336
337
338 % --- Executes on button press in stop.
339 function stop_Callback(hObject, eventdata, handles)
340 % hObject    handle to stop (see GCBO)
341 % eventdata  reserved - to be defined in a future version of MATLAB
342 % handles    structure with handles and user data (see GUIDATA)
343 handles.run = 0;
344 M = handles.M;
345 M.normalizedVelocity = 0;
346 guidata(hObject,handles);
347
348
349 % --- Executes on slider movement.
350 function slider1_Callback(hObject, eventdata, handles)
351 % hObject    handle to slider1 (see GCBO)
352 % eventdata  reserved - to be defined in a future version of MATLAB
353 % handles    structure with handles and user data (see GUIDATA)
354
355 % Hints: get(hObject,'Value') returns position of slider
356 %        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
357
358
359 % --- Executes during object creation, after setting all properties.
360 function slider1_CreateFcn(hObject, eventdata, handles)
361 % hObject    handle to slider1 (see GCBO)
362 % eventdata  reserved - to be defined in a future version of MATLAB
363 % handles    empty - handles not created until after all CreateFcns called
364
365 % Hint: slider controls usually have a light gray background.
366 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
367     set(hObject,'BackgroundColor',[.9 .9 .9]);
368 end
369
370
371
372 % --- Executes during object creation, after setting all properties.
373 function Gain_CreateFcn(hObject, eventdata, handles)
374 % hObject    handle to Gain (see GCBO)
375 % eventdata  reserved - to be defined in a future version of MATLAB

```

```
376 % handles    empty - handles not created until after all CreateFcns called
377
378 % Hint: slider controls usually have a light gray background.
379 if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
380     set(hObject,'BackgroundColor',[.9 .9 .9]);
381 end
```

A.2 PositionStep.m

```

1 %% positionStep.m
2 % A script file by Alex J Burtness
3
4 % An input from the joystick is converted to a specific desired position
5 % for the motor. All the way forward corresponds with completely closed.
6 % All the way back corresponds with completely open. Gently handling is a
7 % must.
8
9 %function [] = positionStep()
10
11 Kspeed = .02;
12 Ktorque = .1;
13 dpMax = 30;
14 dpOK = 3;
15 %velocity = M.motorActualVelocity;
16 torque0 = 100;
17 torque = torque0;
18
19 % Moving to thhe intial position corresponding with Joystick in center.
20
21 % Running the controller
22 J.getdata;
23 clc;
24 joystickOutput = J.axisVal;
25 yData = joystickOutput(4);
26 positionD = 200-200*yData;
27 motorData = M.get_data;
28 currentPosition = M.motorActualPosition;
29 deltaPosition = positionD-currentPosition;
30 if abs(deltaPosition) < dpOK
31     velocity = 0;
32     M.normalizedVelocity = velocity;
33 else %if abs(deltaPosition) < dpMax
34     velocity = Kspeed*(deltaPosition)%+velocity*3/4;
35     M.normalizedVelocity=velocity;
36     torque = torque0;
37     %else
38     % torque = torque + deltaPosition*Ktorque
39 end
40
41 if torque > 100
42     torque = 100;
43 end
44
45 M.alexPosition(velocity, torque);
46
47 clc;
48

```


A.3 PositionRamp.m

```
1 %% positionRamp.m
2 % A functionfile by Alex J Burtness
3
4 % This controller runs similarly to the position controller, but converts
5 % the input from the joystick into a velocity for the desired position.
6 % After doing so it uses a position controller to reach that desired
7 % position.
8
9 Kspeed = .02;
10 Ktorque = .1;
11 Kposition = 50;
12 dpMax = 30;
13 dpOK = 3;
14 velocity = 0;
15 torque0 = 100;
16 torque = torque0;
17 positionD = M.motorActualPosition;
18
19 J.getdata;
20 clc;
21 joystickOutput = J.axisVal;
22 yData = joystickOutput(4);
23 positionD = positionD - yData*Kposition;
24 motorData = M.get_data;
25 currentPosition = M.motorActualPosition;
26 deltaPosition = positionD-currentPosition;
27 if abs(deltaPosition) < dpOK
28     velocity = 0;
29     M.normalizedVelocity = velocity;
30 else %if abs(deltaPosition) < dpMax
31     velocity = Kspeed*(deltaPosition);
32     M.normalizedVelocity=velocity;
33     torque = torque0;
34     %else
35     % torque = torque + deltaPosition*Ktorque
36 end
37
38 if torque > 100
39     torque = 100;
40 end
41
42 M.alexPosition(velocity, torque)
43
```

A.4 VelocityControl.m

```
1 %% velocityControl.m
2 % A function file by Alex J Burtness
3
4 % The motor will run in velocity control quite smoothly.
5
6 Kspeed = .4; %Increases the gain of the speed
7 J.getdata; %Asks the GamePad to update data
8 joystickOutput = J.axisVal;
9 yData = joystickOutput(4); %Velocity is controlled with the Right Joystick
10 M.normalizedVelocity = -yData*Kspeed; %Setting the normalized velocity will start the velocity
11 clc
```

A.5 Forcebar.m

```
1 %% forcebarTest.m
2 %
3
4 function [] = forcebarTest(value)
5     if value <= 0
6         value = 1;
7     elseif value > 100
8         value = 100;
9     end
10    hsvflip = flipdim(hsv(300),1); %Selects the HSV color map, but upside down.
11    hsvmid = hsvflip(length(hsvflip)*.66:length(hsvflip),:);
12    value = ceil(value);
13    cla
14    bar(value)
15    %axis([.9,1,0,100])
16    colormap(hsvmid(value,:));
17
18 end
```


A.6 CalibrationRun.m

```

1
2 %% CalibrationRun.m
3 % A script file by Alex J Burtness
4 %
5 % Created: 15OCT10
6 % Last Update: 19OCT10
7 %
8 % THIS CODE NEEDS ADDITIONAL STITCTION MODELING AND COMMENTS
9 %
10 % This file automatically runs the hand through a series of motions in
11 % order to determine the amount of toruqe that is needed to move with
12 % constant velocity. Having done that, it finds how much torque is needed
13 % to accelerate the motor.
14
15 Torque = 0;
16 OmegaDesired = 0;
17 OmegaActual = 0;
18 Theta = 0;
19 torqueF = 0;
20 torqueR = 0;
21
22 indexF = 1;
23 indexR = 1;
24 %% Going to the correct starting point
25     M.normalizedVelocity = -.1;
26 while M.motorActualPosition > 40
27     M.get_data;
28     M
29 end
30 M.normalizedVelocity = .1;
31
32 while M.motorActualPosition < 40
33     M.get_data;
34     M
35 end
36 M.normalizedVelocity = 0;
37
38 %%
39 for omega = .05:.05:1;
40
41 tic
42 time = toc;
43 M.normalizedVelocity = omega;
44
45 while M.motorActualPosition < 365
46     motorData = M.get_data;
47     torqueF(indexF) = M.motorActualCurrent;
48     omegaDesiredF(indexF) = omega;

```

```

49     omegaActualF(indexF) = M.motorActualVelocity;
50     thetaF(indexF) = M.motorActualPosition;
51     indexF = indexF +1;
52 end
53 M.normalizedVelocity = 0;
54
55 tic;
56 time = toc;
57 M.normalizedVelocity = -omega;
58 while M.motorActualPosition > 35
59     M
60     motorData = M.get_data;
61     torqueR(indexR) = M.motorActualCurrent;
62     omegaDesiredR(indexR) = -omega;
63     omegaActualR(indexR) = -M.motorActualVelocity;
64     thetaR(indexR) = M.motorActualPosition;
65     indexR = indexR +1;
66 end
67
68 M.normalizedVelocity = 0;
69
70 %Torque = [Torque; [torqueF; torqueR]]
71 %omegaDesired = [omegaDesired; [omegaDesiredF,omegaDesiredR]];
72 %omegaActual = [omegaActual; [omegaActualF,omegaActualR]];
73 %Torque = [Torque;mean(torqueF);mean(torqueR)]
74 %Omega = [Omega;omega;-omega]
75
76 end
77
78 %% Get Acceleration Data
79
80 % Go to correct starting point
81     M.get_data
82     M.normalizedVelocity = -.2;
83 while M.motorActualPosition > 40
84     M.get_data;
85 end
86 M.normalizedVelocity = .2;
87
88 while M.motorActualPosition < 40
89     M.get_data;
90 end
91
92 maxAccel = 4;
93 dAccel = .2
94 desiredVelocity = 0;
95 indF = 1;
96 indR = 1;
97 for acceleration = .05 :dAccel:maxAccel
98     acceleration

```

```
99     M.normalizedVelocity = 0;
100     pause(.1)
101     tic
102     while M.motorActualVelocity < 180 && M.motorActualPosition < 360
103         dTime = toc;
104         tic
105         desiredVelocity = desiredVelocity + acceleration*dTime;
106         M.normalizedVelocity = desiredVelocity;
107         pause(.01)
108         M
109         M.get_data;
110         accelDataF(indF) = acceleration;
111         accelVelF(indF) = M.motorActualVelocity;
112         accelPosF(indF) = M.motorActualPosition;
113         accelTorqueF(indF) = M.motorActualCurrent;
114         indF = indF +1;
115     end
116     M.normalizedVelocity = 0;
117     M.get_data;
118     desiredVelocity = 0;
119     pause(.1)
120     tic
121
122     while M.motorActualVelocity < 180 && M.motorActualPosition > 40
123         dTime = toc;
124         tic
125         desiredVelocity = desiredVelocity - acceleration*dTime;
126         M.normalizedVelocity = desiredVelocity;
127         pause(.01)
128         M.get_data;
129         M
130         accelDataR(indR) = -acceleration;
131         accelVelR(indR) = -M.motorActualVelocity;
132         accelPosR(indR) = M.motorActualPosition;
133         accelTorqueR(indR) = M.motorActualCurrent;
134         indR = indR +1;
135     end
136     desiredVelocity = 0;
137     M.normalizedVelocity = 0;
138 end
139
140 %% Data Manipulation
141
142 close all;
143
144 figure(1)
145 xlabel('blah')
146 title('blah')
147 divisions = 8;
148 divisor = 400/divisions
```

```

149
150 omegaDesiredFTheta = [omegaDesiredF', ceil(thetaF'/divisor)];
151 omegaActualFTheta = [omegaActualF', ceil(thetaF'/divisor)];
152 torqueFTheta = [torqueF', ceil(thetaF'/divisor)];
153
154
155 omegaDesiredRTheta = [omegaDesiredR', ceil(thetaR'/divisor)];
156 omegaActualRTheta = [omegaActualR', ceil(thetaR'/divisor)];
157 torqueRTheta = [torqueR', ceil(thetaR'/divisor)];
158
159 % Organize data into groups of theta with velocity and acceleration
160 for i = 1:divisions
161     omegaDesiredFThetaCut = omegaDesiredFTheta(find(omegaDesiredFTheta(:,2)==i),1);
162     omegaActualFThetaCut = omegaActualFTheta(find(omegaDesiredFTheta(:,2)==i),1);
163     torqueFThetaCut = torqueFTheta(find(torqueFTheta(:,2)==i),1);
164
165     omegaDesiredRThetaCut = omegaDesiredRTheta(find(omegaDesiredRTheta(:,2)==i),1);
166     omegaActualRThetaCut = omegaActualRTheta(find(omegaDesiredRTheta(:,2)==i),1);
167     torqueRThetaCut = torqueRTheta(find(torqueRTheta(:,2)==i),1);
168
169
170
171 omegaActualFnoA = 0;
172 omegaDesiredFnoA = 0;
173 torqueFnoA = 0;
174 maxAccel = 3;
175 % Organize data into groups without acceleration and outliers
176     for j = 2:length(omegaDesiredFThetaCut)
177         if abs(omegaActualFThetaCut(j-1) - omegaActualFThetaCut(j)) < maxAccel && torqueFThetaCut(j) < maxAccel
178             omegaActualFnoA = [omegaActualFnoA, omegaActualFThetaCut(j)];
179             omegaDesiredFnoA = [omegaDesiredFnoA, omegaDesiredFThetaCut(j)];
180             torqueFnoA = [torqueFnoA, torqueFThetaCut(j)];
181
182         end
183     end
184
185     order = 6;
186     if length(torqueFnoA) < 100
187         order = 3;
188     end
189     vandyActual = [];
190     vandyDesired = [];
191     for j = 0:order
192         vandyActual = [vandyActual, (omegaActualFnoA).^j];
193         vandyDesired = [vandyDesired, (omegaDesiredFnoA).^j];
194     end
195
196     PolynomialValuesActualVelocity(i,1) = mat2cell(flipud(pinv(vandyActual)*torqueFnoA'))
197     PolynomialValuesDesiredVelocity(i,1) = mat2cell(flipud(pinv(vandyDesired)*torqueFnoA'))
198     pseudoOmega = 0:1:max(omegaActualFnoA);

```

```

199     pseudoOmegaDesired = 0:.01:max(omegaDesiredFnoA);
200     pseudoTorqueActual = polyval(cell2mat(PolynomialValuesActualVelocity(i,1)), pseudoOmega);
201     pseudoTorqueDesired = polyval(cell2mat(PolynomialValuesDesiredVelocity(i,1)), pseudoOmega);
202
203     figure(1)
204     subplot(divisions/2, 2, i)
205     plot(omegaActualFThetaCut,torqueFThetaCut,'.')
206     axis([0,200,0,100])
207
208     figure(2)
209     subplot(divisions/2,2,i)
210     plot(omegaActualFnoA, torqueFnoA, '.')
211     hold on
212     plot(pseudoOmega, pseudoTorqueActual)
213
214
215
216 end
217
218 %% Accel Data Manipulation
219
220
221 %figure
222 %plot3(accelDataF, accelVelF, accelTorqueF)
223 thetaTest = (1:divisions)*400/divisions - 400/(divisions*2)
224 ind = 1;
225 order = 6;
226 for i = 1:length(accelPosF)
227     [x,j] = min(abs(accelPosF(i)-thetaTest))
228     if accelVelF(i) < 30 && accelTorqueF(i) > 0
229         accelLowVel(ind) = accelDataF(i)
230         accelTorqueLowVel(ind) = accelTorqueF(i)
231         accelTorqueLowVelNoVel(ind) = accelTorqueLowVel(ind) - polyval(cell2mat(PolynomialVal
232         ind = ind +1;
233     end
234
235     accelTorqueNoVelAct(i) = accelTorqueF(i) - polyval(cell2mat(PolynomialValuesActualVelocit
236 end
237 %figure
238 plot(accelDataF(1:length(accelPosF)),accelTorqueNoVelAct)
239 vanderLowVelNoVel = [];
240
241 for i = 0:order
242     vanderLowVelNoVel = [vanderLowVelNoVel, accelLowVel'.^i]
243 end
244 PolynomialValuesAccel = flipud(pinv(vanderLowVelNoVel)*accelTorqueLowVelNoVel')
245 pseudoAccel = 0:.01:max(accelLowVel)
246 pseudoAccelTorque = polyval(PolynomialValuesAccel, pseudoAccel)
247
248 figure

```

```
249 plot(accelLowVel, accelTorqueLowVel, '.'')
250 hold on;
251 plot(pseudoAccel, pseudoAccelTorque)
252 plot(accelLowVel, accelTorqueLowVelNoVel, 'o')
253
254 save VOID PolynomialValuesActualVelocity PolynomialValuesDesiredVelocity PolynomialValuesAcce
```

A.7 Arduino Code - SerialReadWrite.pde

```
1  #include "WProgram.h"
2
3  /* SerialReadWriteTest
4
5  A sketch by Alex J Burtness
6  Created: 30APR10
7  Last Update: 19OCT10
8
9  This file opens a serial connection and waits for a
10
11  */
12
13  //GLOBAL CONSTANTS
14  int C2 = 11;
15  int C3 = 10;
16  int aPin = 0;
17  float pressure = 0;
18  float accel = 0;
19  float pi = 3.14159;
20  float e = 2.71828;
21
22  //GLOBAL VARIABLES
23  long randomvalue = 0; // random value
24  long countervalue = 0; // counter value
25  int serialvalue; // value for serial input
26  int started = 0; // flag for whether we've received serial yet
27  long time;
28
29
30  //SETUP (RUN ONCE)
31  void setup()
32  {
33      pinMode(C2,OUTPUT);
34      Serial.begin(9600);
35  }
36  //LOOP (RUN WHILE(1))
37  void loop()
38  {
39      if(Serial.available()) // check to see if there's serial data in the buffer
40      {
41          serialvalue = Serial.read(); // read a byte of serial data
42          if (serialvalue == 'p')
43          {
44              time = micros(); // Parsing through data
45              accel= AccelReading();
46              pressure = PressureReading();
47              Serial.print(time);
48              Serial.print('/', BYTE);
```

```
49     Serial.print(accel);
50     Serial.print('/',BYTE);
51     Serial.print(pressure);
52     Serial.print('/', BYTE);
53     Serial.print('\n', BYTE);
54 }
55 else // If there is no info in the buffer, read the data
56 {
57     serialvalue = int(serialvalue*2.15 + 40); // Pressure value from MATLAB
58     analogWrite(C2,serialvalue); // Vibrotactor
59     analogWrite(C3,serialvalue); // Vibrotactor
60 }
61
62 }
63
64 //if(started) { // loop once serial data has been received
65 // Serial.println(serialvalue); // echo the received serial value
66 // Serial.println(); // print a line-feed
67 // delay(100); // pause
68 //} */
69 //}
70 }
71 //DEFINE METHODS
72 float PressureReading() // Used to mimic a sinusoidal pressure signal
73 {
74     time = micros();
75     float pressure = .5+ .5*sin(time*pi/4000000);
76     return pressure;
77 }
78
79 float AccelReading() // Used to mimic a periodic decaying signal
80 {
81     time = micros() % 3000000;
82     float accel = abs(pow(e,-time/100000)*sin(time*pi/10000));
83     return accel;
84 }
85 }
```


Appendix B

Data Sheets



Motor Specification

Flat Type Vibration Motor

VPM2

1. STANDARD OPERATING CONDITION

NO	ITEM	RATED CONDITION
2-1	STANDARD VOLTAGE	DC3.0V
2-2	OPERATING VOLTAGE GANGE	DC 2.5 ~ 3.5V
2-3	ROTATING DIRECTION	CW , CCW
2-4	OPERATING TEMPERATURE RANGR	-10℃ ~ +60℃
2-5	STORAGE TEMPERATURE RANGE	-30℃ ~+70℃

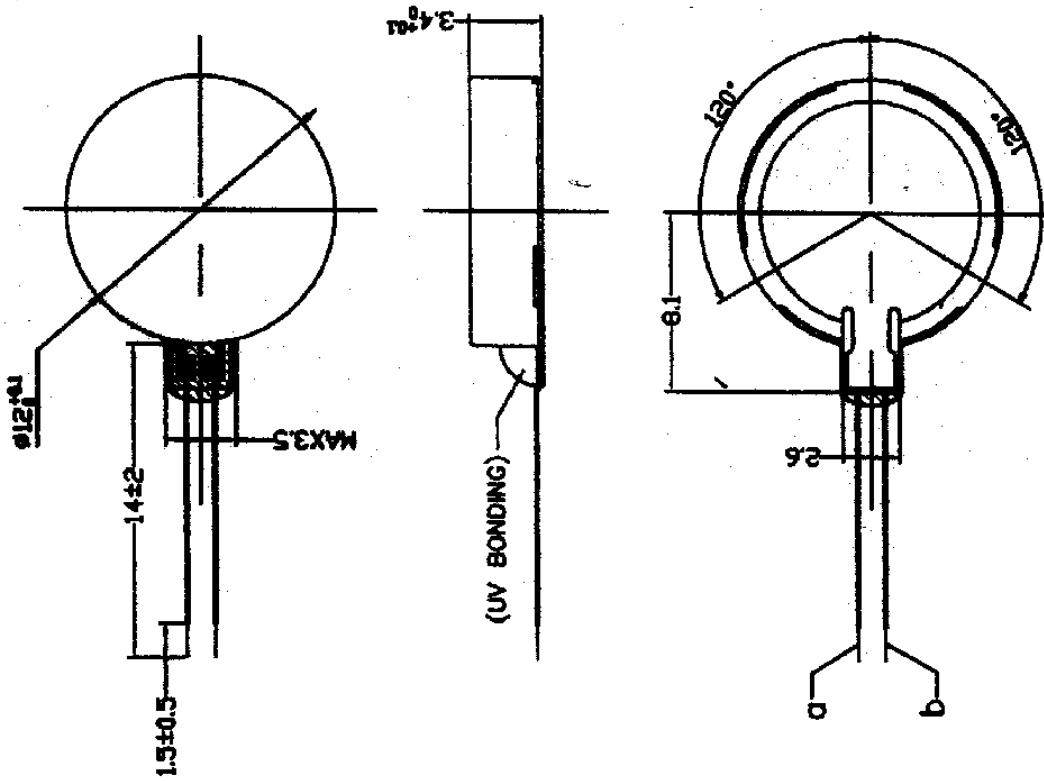
2. MEASURING CONDITION

NO	ITEM	RATED CONDITION
3-1	TEMPERATURE	5℃ ~35℃
3-2	HUMIDITY	35% ~75%RH
3-3	POWER SUPPLY, VOLTAGE SOURCE	DC POWER SUPPLY OR BATTERY3.0V
3-4	POSTURE OF MOTOR	STATE OF STANDARD MEASUREMENT

3. ELECTRICAL CHARACTERISTIC

NO	ITEM	UNIT	SPEC	CONDITION
4-1	STANDARD SPEED	rpm	12,000±3,000	STANDARD VOLTAGE : DC 3.0V
4-2	STANDARD CURRENT	mA	80 MAX	STANDARD VOLTAGE : DC 3.0V
4-3	MIN.STARTING VOLTAGE	V	2.3MAX	ON/OFF-1CYCLE,TTL5CYCLES UNDER DC3V
4-4	TERMINAL RESISTANCE	Ω	32 ± 20%	EACH BRUSH CONTACTS EACH POLE OF COMMUTATOR
4-5	STARTING CURRENT	mA	120MAX	MOTOR LOCKING
4-6	INSULATION RESISTANCE	MΩ	10MIN	MEASURING BETWEEN CASE AND TERMINAL.

TOLERANCE



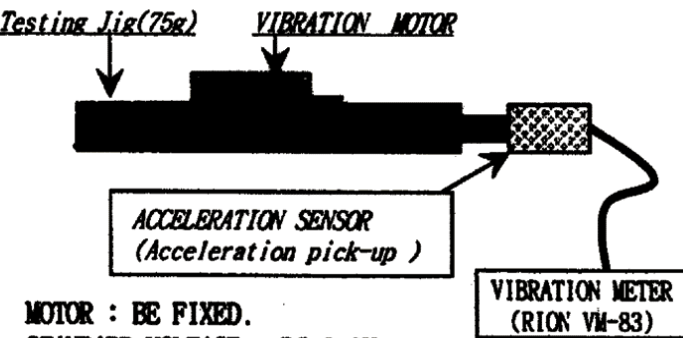
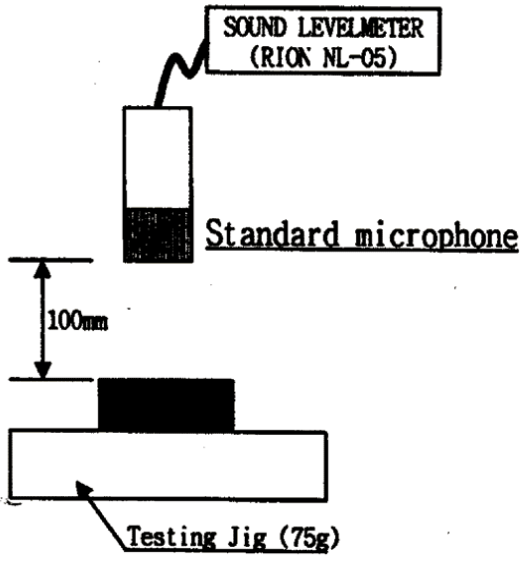
NOTE:

1. Please be careful not to drop or make over 5 kgf force.
2. Applying general tolerance grade3 for unmarked tolerance.
3. Lead wire spec: UL1571#32.
4. Weight of motor ASS'Y about 1.23g.
5. Lead-wire : a-red (+).
Lead-wire : b-blue(-).
6. Other matters are subject to drawer's final confirmation.



VPM2

4. MECHANICAL CHARACTERISTICS

NO	ITEM	UNIT	SPEC	CONDITION
4-1	VIBRATION STRENGTH	G	(1.0)	 <p>Testing Jig(75g) VIBRATION MOTOR</p> <p>ACCELERATION SENSOR (Acceleration pick-up)</p> <p>VIBRATION METER (RION VM-83)</p> <p>MOTOR : BE FIXED. STANDARD VOLTAGE : DC 3.0V . STIPULATION AT ROTATING SPEED .</p>
4-2	MECHANICAL NOISE	dB	50 MAX	 <p>SOUND LEVELMETER (RION NL-05)</p> <p>Standard microphone</p> <p>100mm</p> <p>Testing Jig (75g)</p> <p>MOTOR : BE FIXED . STANDARD VOLTAGR : DC 3.0 V BACK GROUND NOISE 25dB MAX.-A SCALE MICRO PHONE SHOULD BE VERTICAL.</p>
4-3	SHAFT PULL STRENGTH	gf	500 MIN	DEMOLITION TEST BY PUSH-PULL GAUGE.
4-4	BRACKET DEFLECTION STRENGTH	gf	500MIN	DEMOLITION TEST BY PUSH-PULL GAUGE.

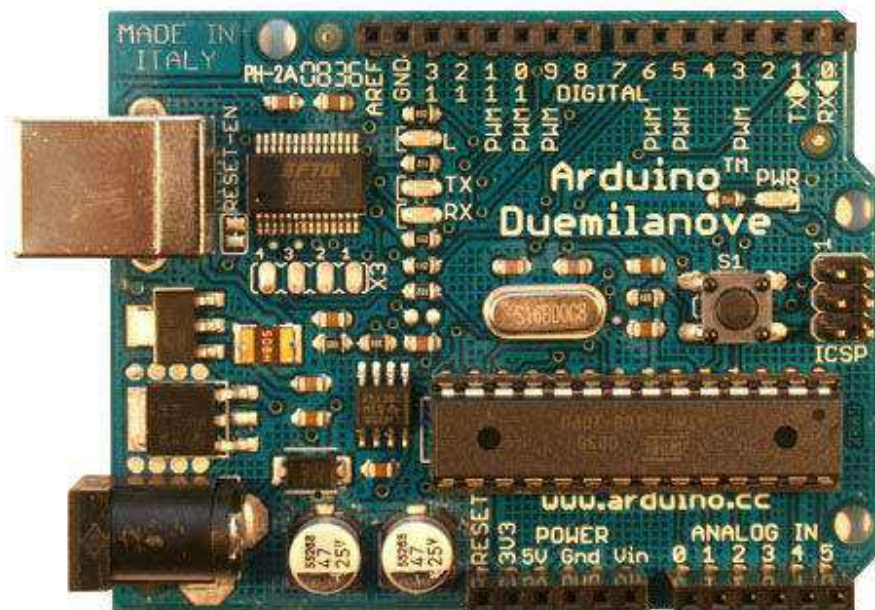
Arduino

 search

Buy | Download | Getting Started | Learning | Reference | Hardware | FAQ

Blog » | Forum » | Playground »

Arduino Duemilanove



Overview

The Arduino Duemilanove ("2009") is a microcontroller board based on the ATmega168 (datasheet) or ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

"Duemilanove" means 2009 in Italian and is named after the year of its release. The Duemilanove is the latest in a series of USB Arduino boards; for a comparison with previous versions, see the [index of Arduino boards](#).

Schematic & Reference Design

EAGLE files: [arduino-duemilanove-reference-design.zip](#)

Schematic: [arduino-duemilanove-schematic.pdf](#)

Summary

Microcontroller	ATmega168
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA

Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Arduino Duemilanove can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); the ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the [EEPROM library](#)); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Duemilanove can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Duemilanove has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the [Wire library](#).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega168 ports.

Communication

The Arduino Duemilanove has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega168 and ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the [FTDI drivers](#) (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial](#) library allows for serial communication on any of the Duemilanove's digital pins.

The ATmega168 and ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a [Wire](#) library to simplify use of the I2C bus; see the [documentation](#) for details. To use the SPI communication, please see the ATmega168 or ATmega328 datasheet.

Programming

The Arduino Duemilanove can be programmed with the Arduino software ([download](#)). For details, see the [reference](#) and [tutorials](#).

The ATmega168 or ATmega328 on the Arduino Duemilanove comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Duemilanove is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega168 or ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Duemilanove is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Duemilanove. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Duemilanove contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

USB Overcurrent Protection

The Arduino Duemilanove has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Duemilanove PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Listen to the name

This is how you can pronounce the board's name in proper Italian, download the sound file in the format that better suits you: [WAV](#), [OGG](#), [MP3](#), [FLAC](#), [WMA](#)

Piezotron® Accelerometer

Type 8694M1

Miniature, Wide Frequency Response, Voltage Mode Triaxial Accelerometer

Light 2,5 gram weight triaxial accelerometer that simultaneously measures vibration in three, mutually perpendicular axes (x, y and z). Designed primarily for measurement applications requiring a high frequency response capability in all three axis.

- Low impedance voltage mode
- Small size and lightweight, less than 2,5 grams
- Quartz sensing element
- Conforming to CE

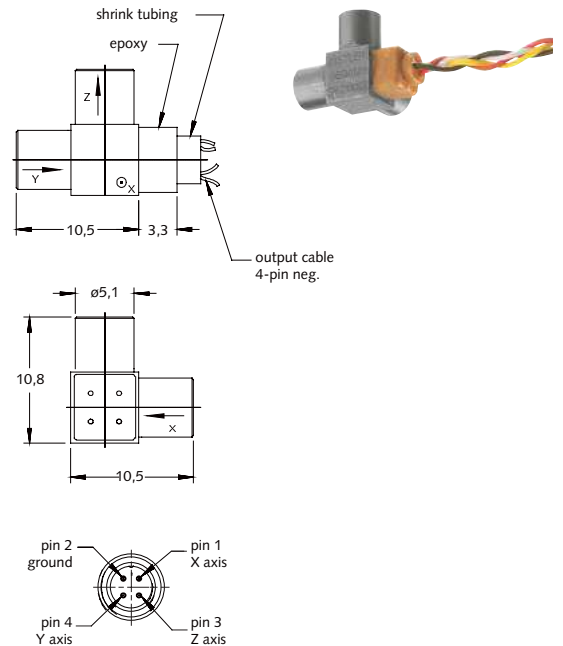
Description

The triaxial accelerometer Type 8694M1 consists of three individual sensor elements mounted in an orthogonal configuration with each containing a preloaded quartz-crystal measuring assembly, a seismic mass, and a miniature hybrid Piezotron electronics. The signal conditioning circuit converts the charge developed in the quartz elements as a result of the accelerometer being subjected to a vibration, into a useable high level voltage output signal at a low impedance level.

Since the Type 8694M1 is a triaxial accelerometer, each sensor axis requires individual excitation power and signal processing. Kistler's 5100 Piezotron coupler series includes a wide selection of single and multichannel units that include both gain and frequency tailoring. Industry standard voltage mode IEPE (Integral Electronic Piezo-Electric) power supply/couplers can also be used with the accelerometer.

Application

The accelerometer Type 8694M1 is well suited for measuring dynamic acceleration, vibration and shocks in applications where minimum mass, small mounting size, and high resonant frequency are essential. The dynamic characteristics of very light test objects are practically not influenced by the accelerometer's small mass. The triaxial accelerometer is ideal for measuring acceleration vectors in space, vibration measurement on thin-walled structures, aircraft and automotive structures and general vibration measurements.



Mounting

The accelerometer Type 8694M1 can be attached to the test surface by using wax, or adhesive. Reliable and accurate measurements require that the mounting surface be clean and flat. The operating instruction manual for the accelerometer Type 8694M1 provides detailed information regarding mounting surface preparation.

Adhesive mounting is recommended for the widest transfer of frequency information, but double-sided adhesive tape or wax may also be used. When using the anodized adaptor, Types 8439 or 8440, the accelerometer will be ground isolated from the test object.

The recommended adhesives, to be placed between the accelerometer and the object or a ground isolated mounting pad, include:

- Petro wax, Type 8432
- Loctite 430: general use between metals
- Loctite 495: general use between other materials.
- 3M Scotch Weld 1838: high temperatures, above 165 °C

Note: Removal of this substance is extremely difficult and care should be exercised when removing the accelerometer.

Technical Data

Specification	Unit	Type 8694M1
Acceleration range	g	±500
Acceleration limit	gpk	±1 000
Threshold nom. (noise 100 µVrms)	grms	0,025
Sensitivity, ±5 %	mV/g	4
Resonant frequency mounted, nom.	kHz	80
Frequency response, ±5 %	Hz	10 ... 20 000
Amplitude non-linearity	%FSO	±1
Time constant, nom.	s	0,5
Transverse sensitivity, nom.	%	<5

Environmental

Random vibration, max.	grms	±2 000
Shock limit (1 ms pulse)	gpk	±2 000
Temperature coefficient of sensitivity	%/°C	-0,05
Operating temperature range	°C	-196 ... 135
Storage temperature range	°C	-195 ... 150

Output

Bias, nom.	VDC	4
Impedance	Ω	25
Voltage full scale	V	±2
Current	mA	±2

Source

Voltage	VDC	12 ... 30
Constant current	mA	4
Impedance, min.	kΩ	100

Construction

Sensing element	Type	quartz-compression
Case/base	material	titanium
Degree of protection case/connector (EN 60529)		IP66
Connector	Type	4-pin neg. int.
Ground isolated		with pad
Mass	grams	2,5
Mounting	Type	adhesive/wax

1 g = 9,80665 m/s², 1 Inch = 25,4 mm, 1 gram = 0,03527 oz, 1 lbf-in = 0,113 N-m

Included Accessories

- Mounting wax

Type
8432

Optional Accessories

- Mounting adapter with M3 thread
- Mounting adapter with 4-40 UNC thread

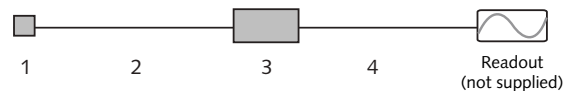
Type
8439
8440

Ordering Key



Measuring Chain

- | | Type |
|---|---------|
| 1 Low impedance sensor | 8694M1 |
| 2 Sensor cable, 4-pin pos. to (3x) BNC pos. | 1576... |
| 3 Power supply/signal conditioner | 51... |
| 4 Output cable, BNC pos. to BNC pos. | 1511 |



OOMOO® Series

1A:1B Mix By Volume Silicone Rubber



www.smooth-on.com

PRODUCT OVERVIEW

No Vacuuming – No Scale – Easy To Use . . . OOMOO® 25 & 30 are easy to use silicone rubber compounds that feature convenient one-to-one by volume mix ratios (**no scale necessary**). Both have low viscosities for easy mixing and pouring . . . **vacuum degassing is not necessary**. Both products cure at room temperature with negligible shrinkage. OOMOO® 30 has a 30-minute pot life, with a six-hour cure time. OOMOO® 25 is a faster version, with a 15-minute pot life and 75 minute cure time.

For The Novice Mold Maker - OOMOO® silicones do not have great tear strength. They are good for making simple one- or two-piece block molds. If you require a high-tear strength silicone, Mold Max® silicones are recommended. More information on Mold Max® silicones is available at www.smooth-on.com

OOMOO® 25 & 30 are suitable for a variety of art-related and industrial applications including making one and two-piece block molds for sculpture and prototype reproduction, casting plaster, resins and wax. OOMOO® silicones are also suitable for electrical potting and encapsulation applications.

TECHNICAL OVERVIEW

	A:B Mix Ratio by Volume	A:B Mix Ratio by Weight	Mixed Viscosity (ASTM D-2393)	Specific Gravity (g/cc) (ASTM D-1475)	Specific Volume (cu. in./lb.) (ASTM D-1475)	Pot Life (ASTM D-2471)	Cure Time	Color	Shore A Hardness (ASTM D-2240)	Tensile Strength (ASTM D-412)	Elongation at Break % (ASTM D-412)	Die C Tear Strength (ASTM D-624)
OOMOO® 25	1A:1B	100A:130B	4250 cps	1.34	20.6	15 min.	75 min.	Light Blue	25A	240 psi	250%	40 pli
OOMOO® 30	1A:1B	100A:130B	4250 cps	1.34	20.6	30 min.	6 hours	Lavender	30A	240 psi	250%	40 pli

Volume Resistance (ohm) (ASTM D-150-98): >1.0E+14

Volume Resistivity (ohm cm) (ASTM D-150-98): >7.363E+15

Dielectric Constant k' @ 100 Hz (ASTM D-150-98): 3.33

Dissipation Factor @ 100 Hz (ASTM D-150-98): 0.01

Dielectric Strength (V/mil) (ASTM D-147-97a): 357

Coefficient of Linear Expansion (um/m-°C) (ASTM E-831-06): 288

Thermal Conductivity (W/M*K) (ASTM E-1461): 0.37

Useful Temperature Range: -65°F to 400°F (-19°C to 205°C)

Shrinkage (in./in.) (ASTM D-2566): 0.0025

*All values measured after 7 days at 73°F/23°C

PROCESSING RECOMMENDATIONS

PREPARATION... Safety – Use in a properly ventilated area (“room size” ventilation). Wear safety glasses, long sleeves and rubber gloves to minimize contamination risk. Wear vinyl gloves only. Latex gloves will inhibit the cure of the rubber.

Store and use material at room temperature (73°F/23°C). Storing material at warmer temperatures will also reduce the usable shelf life of unused material. These products have a limited shelf life and should be used as soon as possible.

Cure Inhibition - Silicone rubber may be inhibited by certain contaminants in or on the pattern to be molded, resulting in tackiness at the pattern interface or a total lack of cure throughout the mold. If compatibility between the rubber and the surface is a concern, a small-scale test is recommended. Apply a small amount of rubber onto a non-critical area of the pattern. Inhibition has occurred if the rubber is gummy or uncured after the recommended cure time has passed. Materials found to cause cure inhibition include sulfur-based modeling clays and latex rubber. **To prevent inhibition apply a sealing agent . . . apply a “barrier coat” of clear acrylic lacquer sprayed onto the clay surface.**

Applying A Release Agent - Although not usually necessary, a release agent will make demolding easier when pouring into or over most surfaces. **Ease Release® 200** is a proven release agent for making molds with silicone rubber and for releasing new silicone from cured silicone. Mann Ease Release® products are available from Smooth-On or your Smooth-On distributor. **Because no two applications are quite the same, a small test application to determine suitability for your project is recommended if performance of this material is in question.**

Safety First!

The Material Safety Data Sheet (MSDS) for this or any Smooth-On product should be read prior to use and is available upon request from Smooth-On. All Smooth-On products are safe to use if directions are read and followed carefully.

Keep Out of Reach of Children

Be careful. Use only with adequate ventilation. Contact with skin and eyes may cause irritation. Flush eyes with water for 15 minutes and seek immediate medical attention. Remove from skin with waterless hand cleaner followed by soap and water.

Important: The information contained in this bulletin is considered accurate. However, no warranty is expressed or implied regarding the accuracy of the data, the results to be obtained from the use thereof, or that any such use will not infringe upon a patent. User shall determine the suitability of the product for the intended application and assume all risk and liability whatsoever in connection therewith.

MEASURING & MIXING...

Before you begin, pre-mix Part A thoroughly to re-disperse fillers that may have settled. After dispensing equal amounts of Parts A and B into mixing container, mix thoroughly for 3 minutes making sure that you scrape the sides and bottom of the mixing container several times. Mixture should have a uniform color with no color streaks. If you observe color streaks, continue mixing until they are eliminated.

POURING, CURING & PERFORMANCE ...

Pouring – For best results, pour your mixture in a single spot at the lowest point of the containment field. Let the rubber seek its level up and over the model. A uniform flow will help minimize entrapped air. The liquid rubber should level off at least 1/2" (1.3 cm) over the highest point of the model surface.

Curing – Allow to cure as prescribed (75 minutes for OOMOO® 25 and 6 hours for OOMOO® 30) at room temperature (73°F/23°C) before demolding. Post curing the mold an additional 4 hours at 150°F (65°C) will eliminate any residual moisture and alcohol which is a by product of the condensation process and may inhibit some resins. Allow mold to cool to room temperature before using. Do not cure rubber where temperature is less than 65°F/18°C.

Using The Mold – No release agent is necessary when casting wax or gypsum. Applying a release agent (Ease Release® 200) prior to casting polyurethane, polyester and epoxy resins is recommended to prevent sticking and mold degradation.

Mold Performance & Storage – The physical life of the mold depends on how you use it (materials cast, frequency, etc.). Casting abrasive materials such as concrete will quickly erode mold detail, while casting non-abrasive materials (wax) will not affect mold detail. Before storing, the mold should be cleaned with a soap solution and wiped fully dry. Two part (or more) molds should be assembled. Molds should be stored on a level surface in a cool, dry environment.



Call Us Anytime With Questions About Your Application.

Toll-free: (800) 762-0744 Fax: (610) 252-6200

The new www.smooth-on.com is loaded with information about mold making, casting and more.

Vita

Alex J. Burtness was born on March 16th, 1987 in Minneapolis, Minnesota. He later moved to Portland, Oregon and attended Sunset High School where he graduated in 2005. Following his graduation he accepted an appointment to the United State Naval Academy and was sworn in as a Midshipman on June 28th, 2006. At the Academy he majored in Systems Engineering and graduated with Honors and Distinction. In 2010 he was commissioned as an Ensign in the United States Navy and was accepted into the training pipeline to become an Explosive Ordnance Disposal Officer.

Following his commissioning, Alex was given orders to attend the Johns Hopkins University to finish his graduate studies in Mechanical Engineering while doing research for the Navy Explosive Ordnance Disposal Technology Division. In February 2011, Alex will report to the Naval Diving and Salvage Training Center in Panama City where he will begin training to receive his qualification as an Explosive Ordnance Disposal Officer.