12-2016

Design of a Flexible Control Platform and Miniature in vivo Robots for Laparo-Endoscopic Single-Site Surgeries

Lou P. Cubrich
University of Nebraska-Lincoln, lou.cubrich@gmail.com

DESIGN OF A FLEXIBLE CONTROL PLATFORM AND MINIATURE *IN VIVO* ROBOTS FOR LAPARO-ENDOSCOPIC SINGLE-SITE SURGERIES

by

Lou P. Cubrich

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Mechanical Engineering and Applied Mechanics

Under the Supervision of Professor Shane Farritor

Lincoln, Nebraska

December, 2016

DESIGN OF A FLEXIBLE CONTROL PLATFORM AND MINIATURE *IN VIVO* ROBOTS FOR LAPARO-ENDOSCOPIC SINGLE-SITE SURGERIES

Lou P. Cubrich, M.S.

University of Nebraska, 2016

Adviser: Shane Farritor

Minimally-invasive laparoscopic procedures have proven efficacy for a wide range of surgical procedures as well as benefits such as reducing scarring, infection, recovery time, and post-operative pain. While the procedures have many advantages, there are significant shortcomings such as limited instrument motion and reduced dexterity. In recent years, robotic surgical technology has overcome some of these limitations and has become an effective tool for many types of surgeries. These robotic platforms typically have an increased workspace, greater dexterity, improved ergonomics, and finer control than traditional laparoscopic methods. This thesis presents the designs of both a four degree-of-freedom (DOF) and 5-DOF miniature *in vivo* surgical robot as well as a software architecture for development and control of such robots. The proposed surgical platform consists of a two-armed robotic prototype, distributed motor control modules, custom robot control software, and remote surgeon console. A plug-in architecture in the control software provides the user a wide range of user input devices and control algorithms, including a numerical inverse kinematics solver, to allow intuitive control and rapid development of of future robot prototypes. A variety of experiments performed by a surgeon at the University of Nebraska Medical Center were used to evaluate the performance of the robotic platform.

ACKNOWLEDGMENTS

I would like to thank my adviser, Dr. Shane Farritor, for the opportunity to work on such an engaging project. I would also like to thank my colleges, especially Jay Carlson and Mark Reichenbach, for their support and significant contributions. Without their help, this work would have been much more difficult. Finally, I would like to thank my parents, Mitch and Nancy Cubrich. Without their constant encouragement and support, I would have never made it this far.

The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. This work was supported by the Office of the Assistant Secretary of Defense for Health Affairs under Award No. W81XWH-14-1-0058. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the Department of Defense. In conducting research using animals, the investigator(s) adheres to the laws of the United States and regulations of the Department of Agriculture. In the conduct of research utilizing recombinant DNA, the investigator adhered to NIH Guidelines for research involving recombinant DNA molecules. In the conduct of research involving hazardous organisms or toxins, the investigator adhered to the CDC-NIH Guide for Biosafety in Microbiological and Biomedical Laboratories.

Contents

List of Figures

List of Tables

Introduction

Traditional surgical procedures typically require large open incisions to provide the surgeon access to and visualization of the surgical site. In an effort to reduce recovery time, post-operative pain, and cosmetic effects, many of these procedures have been converted to minimally invasive surgeries (MIS). In the last 20 years, MIS has influenced the techniques used in nearly every specialty of surgical medicine. MIS procedures result in an 18% reduction in post-operative infection, as well as reducing blood loss, length of hospital stay, morbidity, and complication rates. Typical MIS procedures replace the large open incision with multiple small incisions which laparoscopic tools are inserted through. However, several downsides have emerged with laparoscopic procedures: there is a significant learning curve in the use of laparoscopic tools as the control is not intuitive, there is a reduction in visual feedback and dexterity, and the tools only work well for relatively simple procedures such as tissue removal and closure [30].

Laparoendoscopic single-site (LESS) surgery is a less common type of MIS which is performed entirely through a single incision, typically at the belly button. A laparoscope and several surgical instruments are inserted through a special gel diaphragm device at the incision to provide the surgeon access to the surgical site. Special bent laparoscopic instruments are usually required to accomplish the complex tasks required by the procedure. The abdominal cavity is filled with carbon dioxide gas to create a larger workspace for the surgeon. The gas is evacuated after the procedure is complete. While this type of MIS has even more benefits than traditional laparoscopic surgery, it is also requires more training because the laparoscopic instruments must be crossed at the incision site to improve triangulation.

In an attempt to reduce the limitations of MIS and LESS procedures, several laparoscopic surgical robot platforms have been introduced. Platforms such as the DaVinci Surgical System® from Intuitive Surgical are designed to manipulate laparoscopic tools as a natural extension of the surgeon's hands and eyes by mimicking the motions of the operator in a master-slave configuration. While these platforms are mature and greatly mitigate control and dexterity problems, they are still limited by their multi-incision design. These platforms are also generally very large and expensive, making them impractical for most smaller hospitals.

Completely insertable LESS *in vivo* robotic prototypes have been developed to address the limitations of currently available surgical robots. The surgical robotic devices were designed to be inserted through a single incision, allowing them to be rotated in the incision to provide access to all quadrants of the abdominal cavity. The devices have two independent arms with interchangeable tools and an integrated vision system. An example of the device within the abdominal cavity is shown in Figure 1.

This thesis presents the mechanical, electrical, and software design for this robotic platform. Two surgical robots are discussed, as well as the motor control system and user interface.

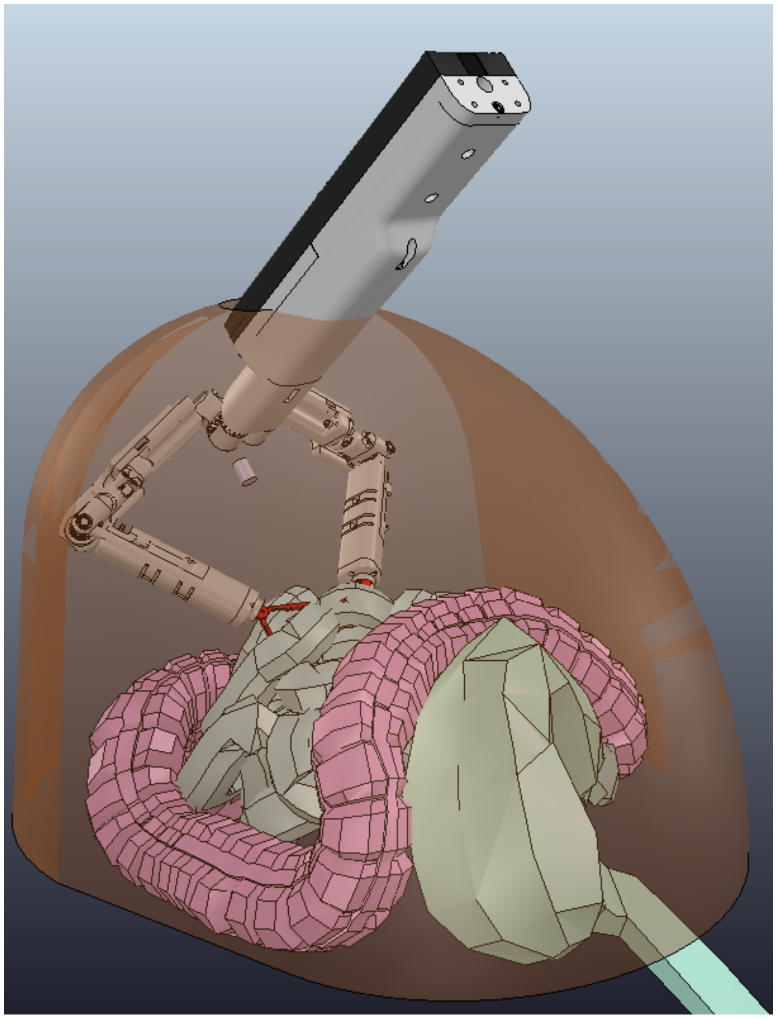


Figure 1: Miniature LESS robotic device inside an insufflated abdominal cavity.

Background

Open surgeries involve a large open incision that grants access to the surgical site and allows the surgeon to easily visualize and and manipulate the tissue and organs. While this method of surgery is typically the least difficult to perform, there are significant drawbacks due to the trauma caused by the surgery including increased recovery time and risk of infection. Many surgeries that were traditionally performed as open procedures are being converted to minimally invasive surgical (MIS) procedures [31]. MIS replaces the large open incision with one or more small incisions (0.5-1.5 cm) in which the surgeon inserts long, slender instruments and a camera into the patient's abdomen. Benefits of MIS include reduced trauma, postoperative pain, and recovery time. In fact, the length of post-operative hospital stay was reduced 28% for MIS abdominal surgeries including colectomy and cholecystectomy [1].

Minimally invasive surgery

Minimally Invasive Surgery

The shift from open to laparoscopic procedures began in the 1980s and initially resulted in significant morbidity and mortality due to lack of training, proper instrumentation, and standardization [36]. The single, large incision was replaced with 3-5 small incisions in which special ports, or trocars, are placed. The abdominal cavity is then insufflated with carbon dioxide to increase the volume of the abdominal cavity. Special laparoscopic tools are then inserted through the trocars to to grant the surgeon access to the internal organs. The trocars seal around the tools to maintain insufflation and tools can be removed and interchanged depending on the needs of the procedure.

While these procedures now produce superior outcomes over open surgeries, there are still many disadvantages. The laparoscopic technique suffers from restricted instrument motion due to the constraint of passing through the abdominal wall [24]. The surgeon must also learn to operate in a two-dimensional surgical field without depth perception. Maneuvering laparoscopic instruments results in increased muscle activity and often requires the surgeon to operate in non-ergonomic positions, increasing shoulder and spine discomfort compared to open procedures [2]. Despite the greater strain, laparoscopic surgery is still considered the standard of care for many simple surgical procedures.

Laparoendoscopic Single Site Surgery

Another method to increase the the benefits of MIS is laparo-endoscopic single-site (LESS) surgery. This form of MIS is similar to conventional laparoscopic procedures but, instead of using multiple incisions to access the surgical site, a single small incision (~2 cm) is used to pass multiple tools and camera into the abdominal cavity through a special gel diaphragm. The reduction of incisions provides improved cosmesis and minimizes the morbidity associated with multiple-incision procedures [10]. Although LESS improves patient outcomes, it requires special articulated tools that must be crossed at the incision, resulting in transposed instrument view (i.e, the instrument in the left hand operates on the right side of the monitor) and increased intracorporeal and extracorporeal instrument collisions [32].

Robotic MIS

Robotic Laparoscopic Surgery

With the advances in surgical medicine and robotic technology, many institutions saw the potential to combine robotics and MIS to reduce the shortcomings of traditional minimally invasive procedures. While the majority of these systems are not actually “robots”, they allow the surgeon to control surgical instruments through intuitive motions and eliminate the need to operate in non-ergonomic positions by using a master-slave style control scheme. Current robotic surgery results in lower blood loss, but also is associated with greater cost and longer surgery times [39].

Although it would seem to be a relatively new type of surgery, the first robot-assisted surgical procedure was actually performed in 1983 with the use of Anthrobot [30], a robot designed to assist in orthopedic procedures. The first robot approved by the Food and Drug Administration (FDA) for abdominal procedures was the Automated Endoscopic System for Optimal Positioning (AESOP) [29]. The platform consisted of a robotic arm that positioned a camera based on voice commands from the surgeon.

The most advanced commercially available robot for general surgery is currently the da Vinci Surgical System (Intuitive Surgical, Sunnyvale, CA). The system received approval from the FDA in 2000 and has since been the standard for robotic surgery [11]. The platform consists of externally actuated positioning arms, which control tools similar to laparoscopic instruments with up to 7 degrees of freedom, and a surgeon console. The robot has a stereoscopic vision system with up to 10x zoom

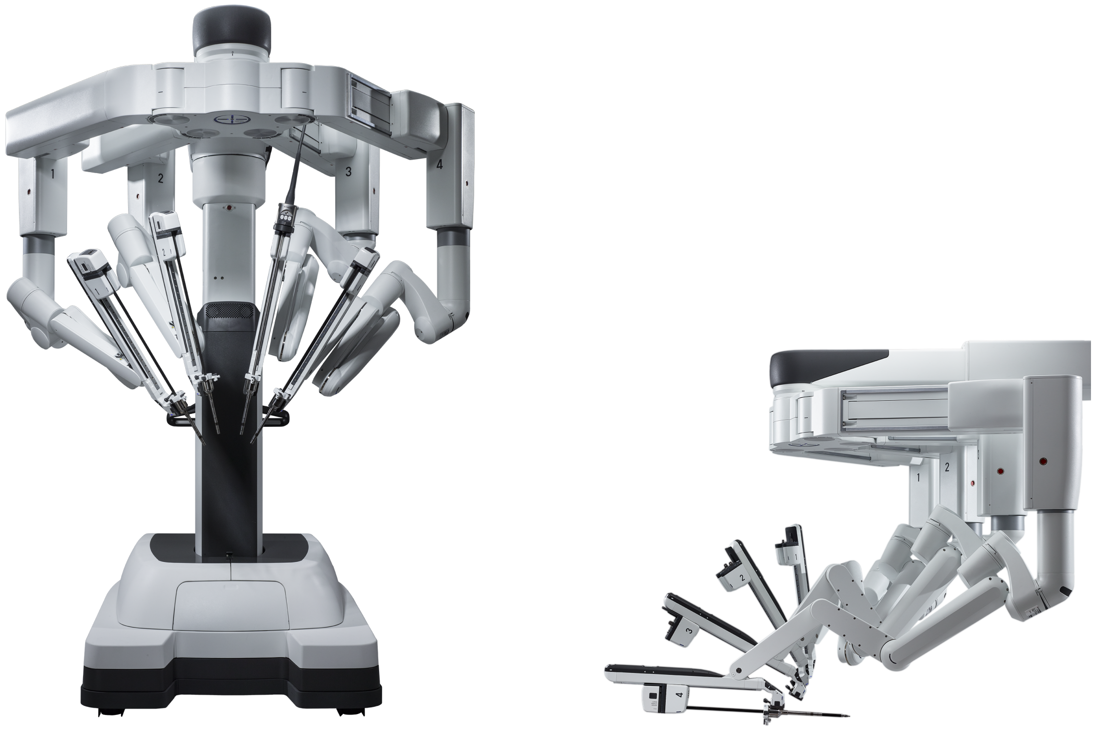


Figure 2: The daVinci® Surgical System, model Xi (©2014 Intuitive Surgical, Inc.)

and has the ability to filter out hand tremors and scale motions through an intuitive control interface [6]. While the platform greatly improves on conventional laparoscopic surgery, it also has some limitations including large size, high cost, crowding of the surgical site, and the need to be repositioned for complex surgeries [4]. The da Vinci system faces the problems that are inherent to multi-site surgeries, including a limited workspace and loss of haptic feedback.

Another notable robot MIS system under development is the Raven-II, which is a collaborative research project between multiple universities. The Raven-II has three 3-DOF arms that position interchangeable 4-DOF tools. The control system is built on an open-source platform developed by seven different universities [12]. While this platform offers the ability to develop custom control algorithms, the hardware is static

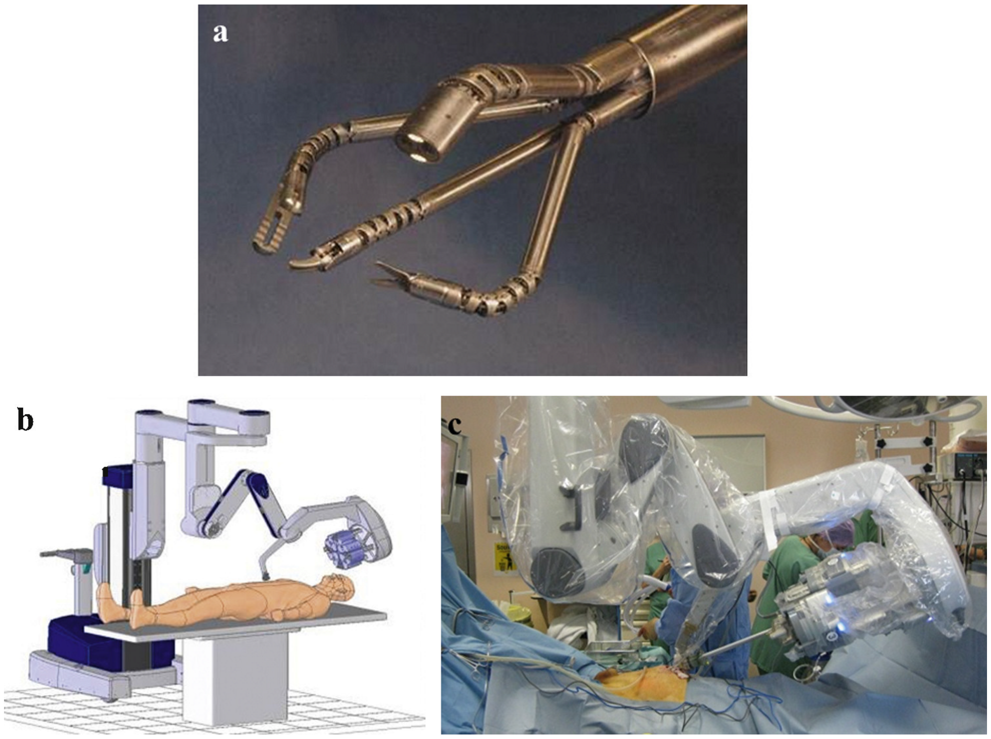


Figure 3: da Vinci SP Surgical System [14].

and has the downfalls of multi-incision MIS such as the need to reposition arms during complex procedures. Also, the dexterity of the tools is limited and there remains the potential for collisions of the tools outside the body.

Robotic Lapaendoscopic Single Site Surgery

Just as traditional laparoscopic procedures have begun converting to LESS, so have robotic assisted MIS started converting to robotic LESS, or R-LESS. The da Vinci platform has an experimental LESS platform which is undergoing clinical evaluation called the da Vinci SP® which is composed of three flexible arms and a stereoscopic camera that all go through a single port, as shown in Figure 3. While this platform is quite dexterous and has successfully performed surgeries through a single port [3], the system is extremely large and takes up a significant portion of the operating room.

Another R-LESS device that is being developed is a two-armed robot developed by the BioRobotics Institute at SSSA (Italy) called SPRINT. This device has two 6-DOF arms with end effectors and is controlled via a haptic interface. The arms are each 18 mm in diameter and are designed to be inserted through a 30-mm port [26, 23, 25].

A snake-like LESS robot is under development at Waseda University. It is positioned by a robotic arm and deploys tools out of the main tubular body. The system is actuated using a cable-driven system and has demonstrated cautery abilities. However, the system has problems with global positioning and triangulation and requires a custom interface of four Phantom Omni (SensAble Technologies, Wilmington, MA) haptic controllers to provide intuitive control [15, 16].

Development of various types of *in vivo* surgical devices in the Advanced Surgical Technologies Laboratory at the University of Nebraska-Lincoln has been occurring since the early 2000s. Platforms include two-wheeled robots, magnetically-coupled imaging devices, and rigidly mounted single-port devices [37, 38, 20, 17, 18, 21, 22, 27, 13]. The most recent work has focused on the development of two-armed miniature robots for R-LESS surgeries. These devices are designed to be inserted into an insufflated abdominal cavity through a single port to perform general abdominal procedures. These robots have successfully performed such operations as colectomies, cholecystectomies, and a hysterectomy. This thesis presents a new platform for the rapid development of such R-LESS devices and two robotic designs that utilize this platform. The most recent robot developed in the Advanced Surgical Technologies Laboratory was the Eric-Bot 2.0 (EB-2.0) [20]. This device has two independent 4-DOF arms and is made to be inserted through a single incicsion.

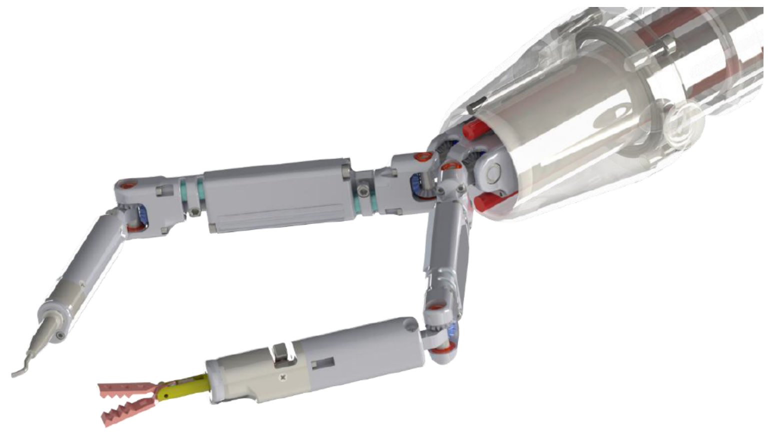


Figure 4: Eric-Bot 2.0 R-LESS device developed at UNL.

Design

Design Requirements

Among the factors that need to be considered in the development of *in vivo* R-LESS devices are force, velocity, dexterity, workspace, and size. The robot must be able to transmit enough force to perform surgical procedures and should be fast enough to give the operator a sense of control. The dexterity and workspace of the manipulators should be maximized while keeping the profile of the inserted device as small as possible.

Though it is difficult to quantify the forces and speeds required to perform a certain procedure, relevant data exists which characterizes the forces and speeds used during laparoscopic procedures.The BlueDRAGON device developed by the BioRobotics Lab at the University of Washington was used to measure the forces at the laparoscopic instrument handles for a variety of surgical procedures [19, 28]. The raw data from these studies showed a force of about 20 N in the direction of the tool axis and about 5 N perpendicular to the tool axis. Further analysis of the data yielded velocity data for surgical procedures. Angular velocities about the axes perpendicular to the instrument were 0.485 rad/sec and the velocity about the axis of the tool was 1.053 rad/sec. The velocity along the axis of the laparoscopic tool was 72 mm/sec. The linear velocities can be calculated using the reported tool length of 100-150 mm. The

Table 1: Force and velocity requirements for surgical procedures.

Force Direction	Value [N]	Velocity Direction	Value [mm/sec]
Fx	0.8	Vx	70
Fy	0.8	Vy	70
Fz	2.2	Vz	72
		$\omega_{grasper}$	1.053 rad/s

velocity requirements can be estimated from these data [20].

Force data were also collected in a study to determine the force needed to stretch the mesocolon for dissection [9]. Clamps were applied to the mesocolon in series with a spring scale and the surgeon applied tension at an angle of approximately 60 degrees from horizontal. The average pull force per clamp was 1.9±0.6 N, with a maximum of 3.1 N. Lehman et al. assumed an even distribution of forces between the remaining axes, yielding the forces shown in Table 1.

The two arms should have minimal cross-section area when in position for insertion. The shared workspace between the two arms should be maximized. The arms should be as dexterous as necessary without adding unnecessary complexity. The device should have a rigid mount outside the body with which to grossly position the arms within the abdominal cavity.

CubReich-Bot 1.0

The CubReich-Bot 1.0 (CRB-1.0) is a LESS surgical robot with two independent 4-DOF arms. Each arm is composed of a 2-DOF differential shoulder joint, upper arm link that houses a 1-DOF elbow joint, and a forearm link which houses a tool with a 1-DOF wrist. A flexible HD endoscope (Endoeye Flex, Olympus®) is inserted down a 5-mm port that runs down the length of the shoulder and is positioned directly below the shoulder joints. Each link houses the motors and control electronics needed for

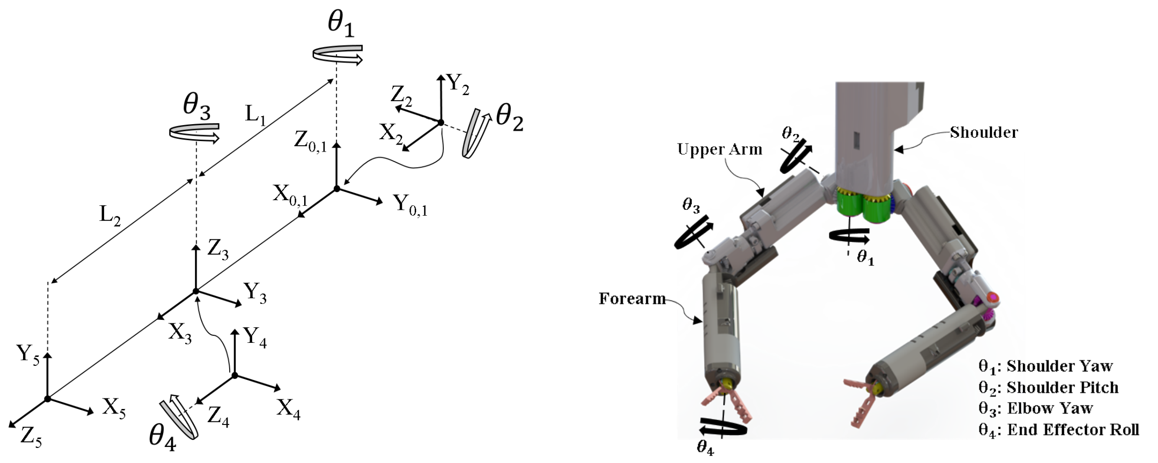


Figure 5: Kinematic frames assigned to CRB-1.0 and link naming convention.

its joints.

Kinematics

The kinematics for each of the robotic arms were analyzed using Denavit-Hartenberg (DH) parameters, which uses four parameters per joint to characterize manipulator kinematics. These parameters are d (offset along previous Z to common normal), θ (angle about previous Z), a (length of the common normal, X), and α (angle about common normal, X), taken in that order. Coordinate frames were attached to each joint with the Z axis of the frame along the axis of rotation and the X axis along the length of the link as shown in Figure 5. DH parameters were assigned based on the orientation of each of the frames and displayed in Table 2. It should be noted that the small offsets in the shoulder were disregarded to simplify the kinematic equations and allow a closed form solution.

Transformation matrices were derived for each frame with respect to the previous frame using the homogeneous transformation matrix and the DH parameters. The homogeneous transformation matrix for frame i with respect to the previous frame

Table 2: DH parameters for the right arm of the CRB-1.0 manipulator.

i	α_i	a_i	d_{i-1}	θ_{i-1}
1	0	0	0	0
2	$90°$	0	0	ϕ_1
3	$-90°$	$L_1 = 68.5$ mm	0	ϕ_2
4	$90°$	0	0	$\phi_3 + 90^{\text{o}}$
5	0	0	$L_2 = 96.4$ mm	ϕ_4

can be calculated from the definition of the DH parameters as

$$T_i^{i-1} = T_{z_{i-1}}(d_{i-1}) R_{z_{i-1}}(\theta_{i-1}) T_x(a_i) R_x(\alpha_i)$$

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_{i-1} & -\cos\alpha_i \sin\theta_{i-1} & \sin\alpha_i \sin\theta_{i-1} & a_i \cos\theta_{i-1} \\ \sin\theta_{i-1} & \cos\alpha_i \cos\theta_{i-1} & -\sin\alpha_i \cos\theta_{i-1} & a_i \sin\theta_{i-1} \\ 0 & \sin\alpha_i & \cos\alpha_i & d_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrices multiplied in order from first to last yield the transformation matrix of the end-effector frame {5} with respect to the base frame {0}. The full derivation can be found in Appendix A.

$$T_5^0 = \begin{bmatrix} -c_4(s_1s_3 - c_1c_2c_3) - c_1s_2s_4 & s_4(s_1s_3 - c_1c_2c_3) - c_1c_4s_2 & c_3s_1 + c_1c_2s_3 & L_1c_1c_2 + L_2\left(s_1c_3 - c_1c_2s_3\right) \\ c_4(c_1s_3 + c_2c_3s_1) - s_1s_2s_4 & -s_4(c_1s_3 + c_2c_3s_1) - c_4s_1s_2 & c_2s_1s_3 - c_1c_3 & L_1s_1c_2 - L_2\left(c_1c_3 + s_1c_2s_3\right) \\ c_2s_4 + c_4c_3s_2 & c_2c_4 - c_3s_2s_4 & s_2s_3 & L_1s_2 + L_2s_2s_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where c_i and s_i are the sine and cosine of θ_i, respectively. The forward kinematic equations can be pulled from the last column of this matrix and are

$$x = L_1c_1c_2 + L_2\left(s_1c_3 + c_1c_2s_3\right)$$

$$y = L_1s_1c_2 - L_2\left(c_1c_3 - s_1c_2s_3\right)$$

$$z = L_1s_2 + L_2s_2s_3$$

A closed form of the inverse kinematic equations was found by solving the forward kinematic equations for the joint angles in terms of x, y, and z. By disregarding solutions that yield an inverted shoulder or elbow joint, the following are the inverse kinematic equations:

$$\begin{aligned}
\theta_3 &= \pi - \arccos\left(\frac{L_1^2 + L_2^2 - L_{12}^2}{2L_1L_2}\right) \\
\theta_2 &= \pm\arctan\left(\frac{\sqrt{1-a^2}}{a}\right) \\
\theta_1 &= \begin{cases} \arctan\left(\frac{\sqrt{x^2+y^2-b^2}}{b}\right) & x > 0 \vee y > 0 \\ \arctan\left(\frac{\sqrt{x^2+y^2-b^2}}{b}\right) - 2\pi & x < 0 \wedge y < 0 \end{cases}
\end{aligned}$$

where

$$\begin{aligned}
L_{12} &= \sqrt{x^2 + y^2 + z^2} \\
a &= \frac{y}{L_1 + L_2c_3} \\
b &= L_1c_2 + L_1c_2c_3
\end{aligned}$$

The robot was designed to be inserted through the abdominal wall with both arms pointed straight downward in-line with the shoulder body. To prevent the manipulators from colliding with internal organs, the arms are bent to reduce their length along the incision axis, as shown in Figure 6.

Mechanical Design

The joint torques and angular velocities were calculated for each joint using the following formulas:

$$T_i = \eta^{n_i}\tau_i\eta_{m_i}\tau_{m_i}T_{m_i} \qquad \omega_i = \frac{\omega_{m_i}}{\tau_i\tau_{m_i}}$$

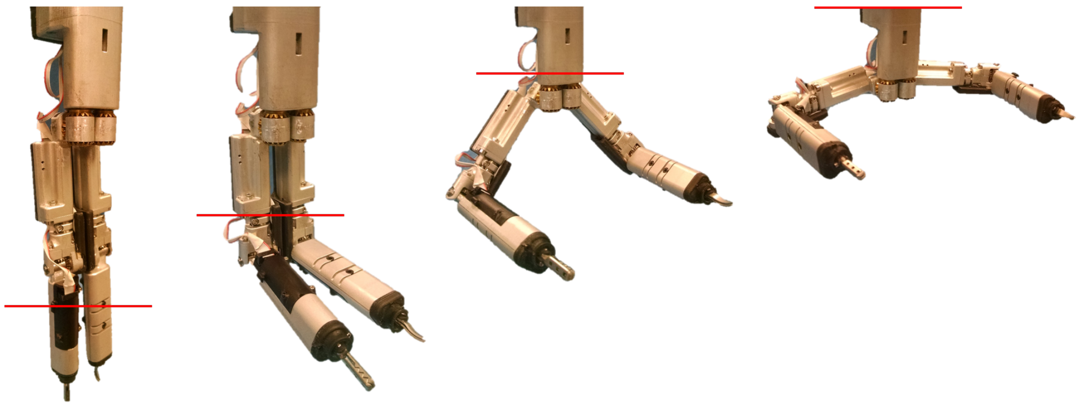


Figure 6: Robot insertion procedure, with the red line representing the abdominal wall.

where T_i and ω_i are the maximum torque and angular velocity at joint i, respectively, η is the efficiency of the gear mesh and is assumed to be 95% per gear mesh, n_i is the number of gear meshes in the drive train of joint i, τ_i is the gear reduction ratio in the drivetrain of joint i, η_{m_i} and τ_{m_i} are the efficiency and gear reduction ratio of the planetary gearbox coupled to the motor, and T_{m_i} and ω_{m_i} are the stall torque and no-load speed of the motor. The torques and angular velocities for each joint are tabulated in Table 3 at the end of this section. The motor and gearbox specifications can be found in the appendix.

A shoulder body was designed to have two independent shoulder joints, a small rigid profile for inserting into the abdominal cavity, and a small port through the body for a 5-mm endoscopic camera. The goal of the design was to reduce the size of the necessary incision and keep as much electronics out of the body as possible. A cross-section view of the inserted shoulder with a profile area of 8.75 cm^2 is shown in Figure 7.

The shoulder uses a concentric shaft design to transmit power from the four 12-mm motors housed in the shoulder to the two shoulder joints while maintaining a

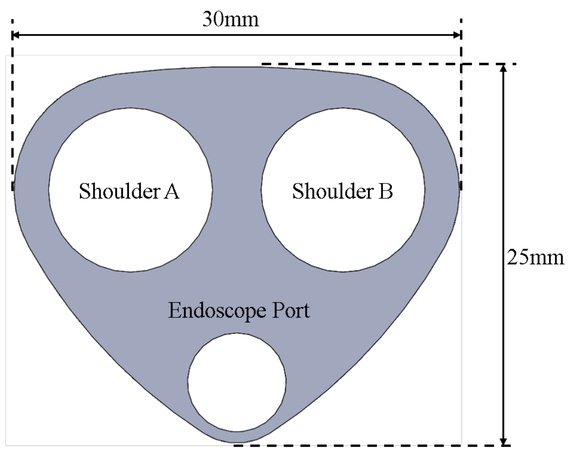


Figure 7: Cross-section view of the inserted portion of the shoulder with endoscope port.

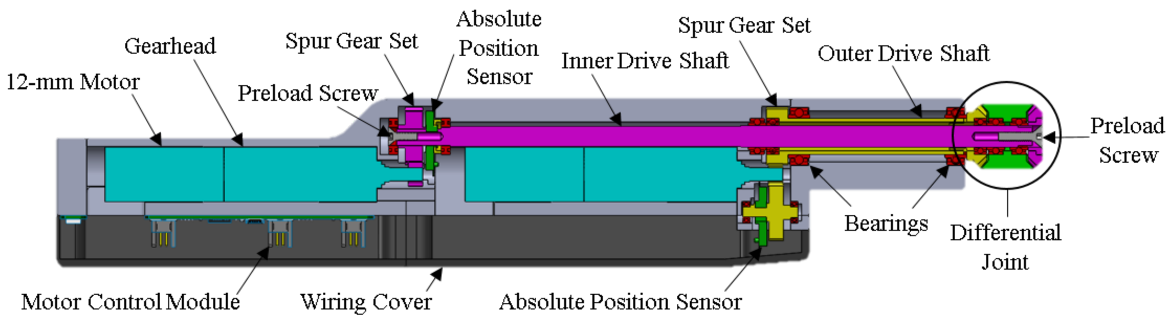


Figure 8: Four DOF shoulder drive train.

small profile. Absolute position sensors (potentiometers) were installed on each shaft to provide feedback to the motor controllers. The shafts were extended to allow for a length of reduced cross-sectional area to be inserted through the abdominal wall. The innermost shaft is tapped and a single screw provides the preload for the shoulder drive train. Each of the differential joints' drive shafts are driven by 12-mm Faulhaber® 1226 12V BLDC motors coupled to 256:1 planetary gearboxes. The motor-gearbox combinations drive their respective shafts through a 30:12 spur gear set. The shoulder yaw and pitch have a range of motion from -90 to 45 degrees.

A differential gear train for each shoulder, shown in Figure 9, allows the two joints

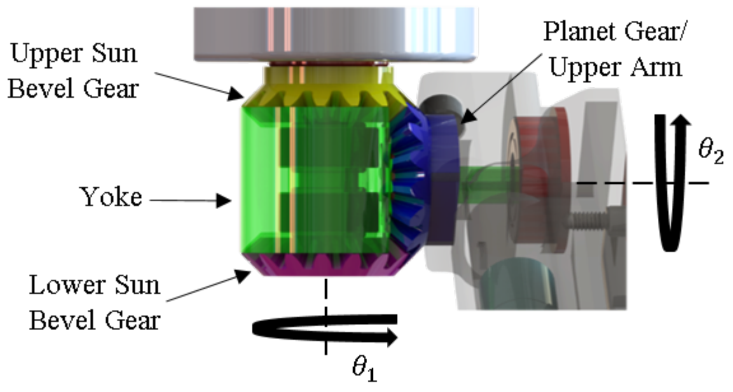


Figure 9: Differential shoulder joint.

to be compacted tightly together but also couples the two DOFs of the shoulder. Both the sun gears must rotate at the same rate and direction to produce pure θ_1 motion; to produce pure θ_2 motion, the sun gears must rotate at the same rate and in opposite directions. The upper and lower sun gear angles can be written in terms of the kinematic angles for use in the control scheme:

$$\theta_{upper} = \theta_1 + \theta_2 \qquad \theta_{lower} = \theta_1 - \theta_2$$

The planet gear of the shoulder differential is part of the upper arm link of the manipulator. The upper arm is mounted to the yoke of the differential joint and has a single DOF elbow joint. The upper arm houses a 6-mm BLDC motor and an on-board motor controller. The upper arm links were designed to sit in contact with each other when aligned straight downward. This creates a minimal profile and allows for a better seal around the robot during insertion into the abdominal cavity. The length of the upper arm, 68.5-mm, was determined by the length of the motor, gear train, and motor controller. A 6-mm Faulhaber® 0620 BLDC motor coupled to a 1024:1 planetary gearbox (60% efficient) transfers mechanical power from a spur gear set with a gear reduction of 16:10 to the joint through a bevel gear set. The

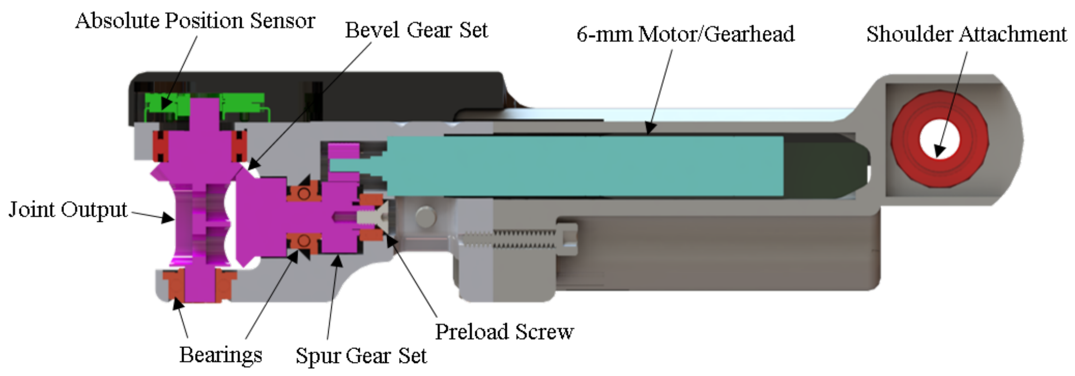


Figure 10: Upper arm drive train cross-section view.

elbow joint has a range of motion from -105 to 105 degrees, but is limited to 0 to 105 degrees by the inverse kinematics solution.

The robot forearm links connect to the elbow joint and house two motors, a motor controller, and a variety of custom tools that use a common interface. The two 6-mm Faulhaber 0620 12V BLDC motors are housed in the forearm and are constrained using motor clamping features. For the end effector roll drivetrain, a 12-tooth spur gear is press-fit onto the output shaft of the gearbox and is coupled to a 24-tooth spur gear about the circumference of the tool yoke. The tool actuation is driven by a 18:10 spur gear set, with the 10-tooth gear press-fit onto the output of the gearbox. A recessed feature at the end of the grasper link provides an area to mount a castration band and seal the robotic arm in a disposable plastic bag. The total length of the forearm link from elbow joint to grasper midpoint is 96.4 mm. A view exposing the drivetrain is shown in Figure 11.

A variety of surgical tools have been developed to use the forearm interface and be easily changed to enable improved functionality. These tools include graspers, surgical shears, mono-polar cautery hook, mono-polar cautery shears, and bi-polar cautery graspers. Each of these tools includes the drivetrain to mate to one or more

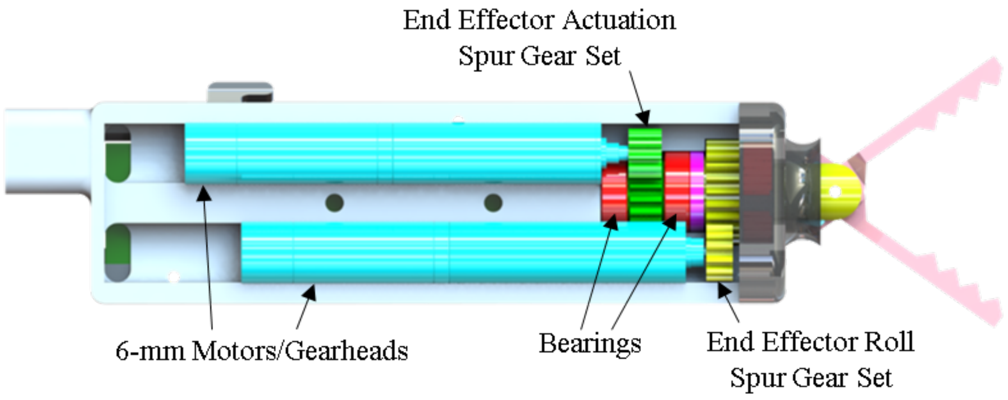

End Effector Actuation Spur Gear Set
6-mm Motors/Gearheads
Bearings
End Effector Roll Spur Gear Set

Figure 11: Forearm drive train cross-section view.

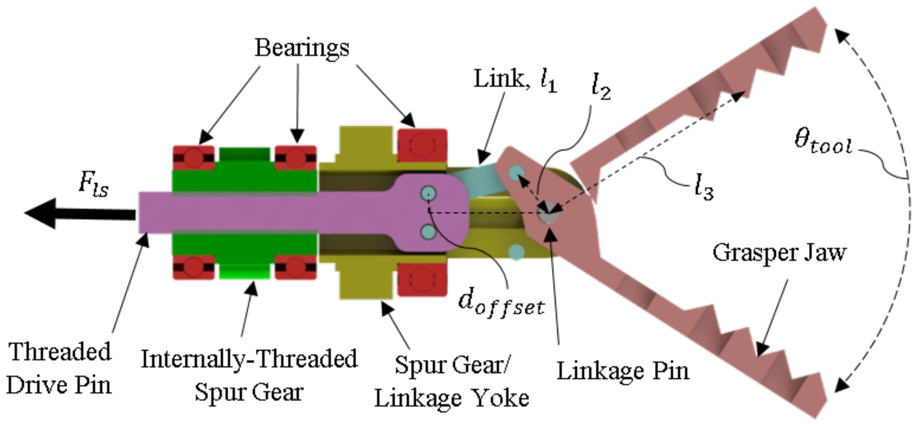


Figure 12: Cross-section view of an actuated tool.

of the forearm's motors and can be easy replaced by removing three screws. All tools have an end-effector roll DOF; the actuated tools (graspers, shears) use an internally threaded spur gear to drive a lead screw drive pin which, in turn, drives two links that are mated to the grasper or shear jaws, as shown in Figure 12.

The closing force for the actuated tools can be calculated by considering the force on the lead screw drive pin [5]:

$$F_{ls} = \frac{2T_m}{d_m}\left(\frac{\pi d_m - fl\sec\alpha}{l + \pi f d_m \sec\alpha}\right)$$

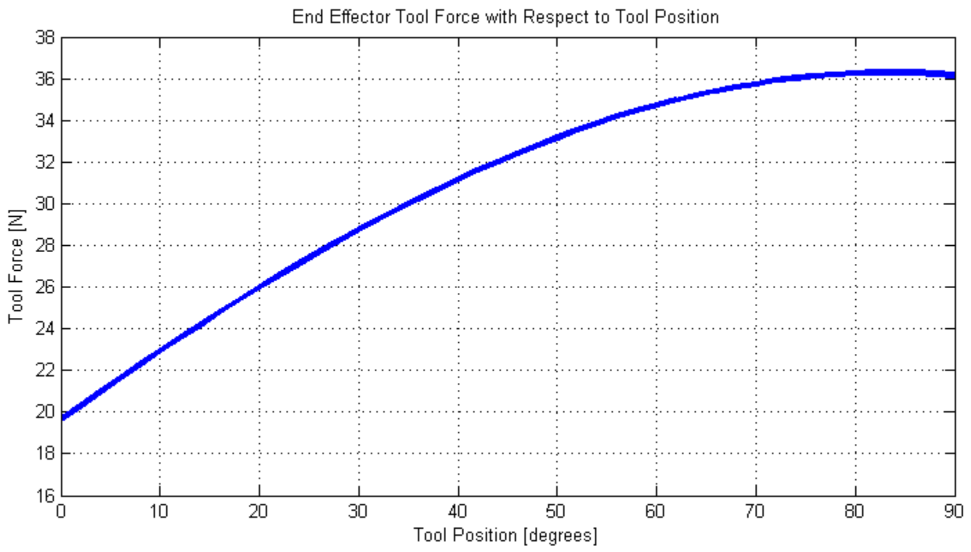


Figure 13: Tool force as a function of tool position.

where d_m is the mean of the major and minor diameters of the thread, f is the friction coefficient (steel-steel ~0.5), l is the lead of the thread, and α is the thread angle. Solving this equation for a #3-56 thread ($d_m = 2.23$ mm, $l = 0.45$ mm, $\alpha = 60$ř), the force of the lead screw can be estimated as 133.14 N. The tool drive links act as two-force members and can only transmit force along the length of the link. The lead screw force can be set to equal the link force in the direction of the drive pin motion. The grasper force with respect to tool position can calculated by applying the lead screw force to the lever arm perpendicular to the force at the linkage pin, as shown in Figure 13. By setting the sum of forces along the direction of the lead screw to zero for static equilibrium, the force equation in terms of the tool position is

$$F_{tool} = \frac{F_{ls}}{l_3}\left[l_2 \sin\theta_a + \tan\left(\arcsin\left(\frac{d_{offset} + l_2 \sin\theta_a}{l_1}\right)\right) l_2 \cos\theta_a\right]$$

where the variables are described in Figure 13 and $\theta_a = {}^{\theta_{tool}}/_{2} + 19^{\text{o}}$.

Table 3: CRB-1.0 joint torques and angular velocities.

i	n_i	τ_i	η_{m_i} [%]	τ_{m_i}	T_{m_i} [mNm]	ω_{m_i} [rpm]	T_i [mNm]	ω_i [rpm]
1	2	30:12 = 2.5	60	256	8.99	27,400	3115.57	42.81
2	2	30:12 = 2.5	60	256	8.99	27,400	3115.57	42.81
3	2	16:10 = 1.6	55	1024	0.551	37,300	448.11	22.77
4	1	24:12 = 2	55	1024	0.551	37,300	589.61	18.21
Tool	1	18:10 = 1.8	70	256	0.551	37,300	168.84	80.95

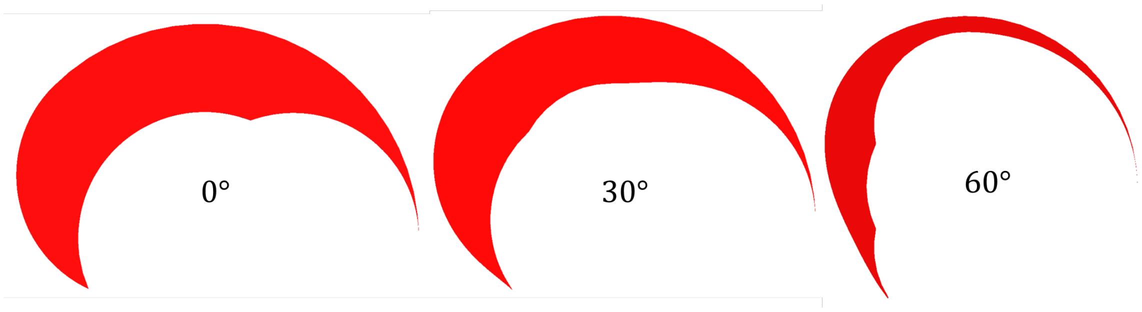


Figure 14: CRB-1.0 workspace slices with changing θ_2.

Workspace, Forces, & Velocity

The workspace of a robot can be defined as the volume that is reachable by the end effector of the manipulator. Because of the two-armed nature of this device, the workspace is the combined volume reachable by both arms. The volume of the workspace where both arms intersect is especially important because surgical tasks such as suturing, dissection, and tissue manipulation often require both arms to work together.

Due to the nature of the kinematic joint arrangement, the workspace is not uniform throughout the range of θ_2 and is especially narrow when θ_2 is 90 degrees and the arms are pointed straight downward. The workspace is the largest in the plane $\theta_2 = 0$. Slices of the right arm's workspace with increasing θ_2 are shown in Figure 14.

The volume of the workspace is therefore a very strange shape and was estimated by creating a CAD model. The reachable volume for each arm was found to be

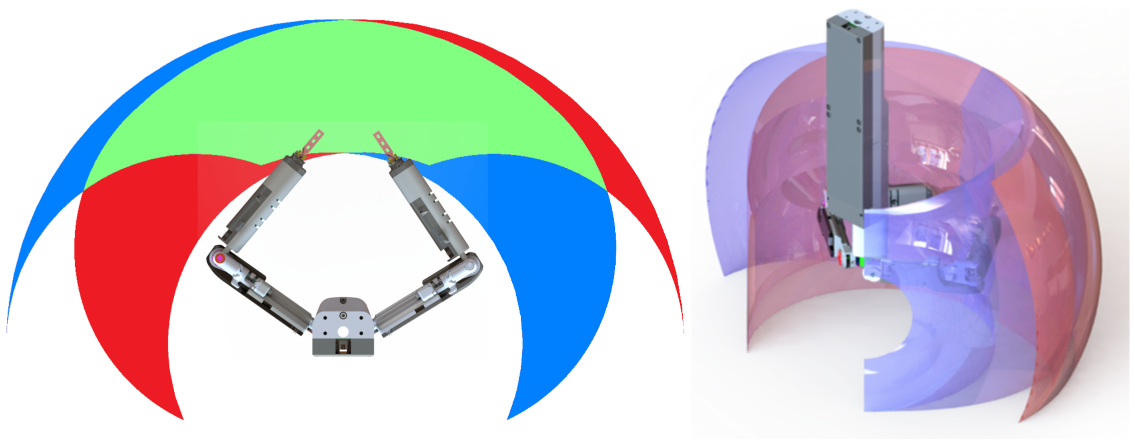


Figure 15: CRB-1.0 within bubbles of the right arm (red), left arm (blue), and shared (green) workspace.

3926.3 cm^3. The shared workspace volume between the two arms is 2215.8 cm^3 and the total workspace volume for both arms is 5636.8 cm^3. The shared workspace volume accounts for 39.3% of the total reachable volume of the robot. The robot inside its workspace is shown in Figure 15.

The forces, velocities, and dexterity at the end effector of the robotic arm were analyzed by considering the Jacobian of the transform matrix from the base frame to the end effector. The Jacobian is the first order partial derivative of the forward kinematics. The matrix can be used to derive both the static forces and velocities of the robotic manipulator. The Jacobian matrix with respect to frame $\{0\}$ can be written as

$$^0J(\theta) = \frac{\delta x}{\delta \theta}$$

where ^0J is the Jacobian with respect to the base frame, x is an vector containing the forward kinematic equations from the last column of the transformation matrix, and θ is the n x 1 array of joint angles. For the four-DOF robotic arm, the Jacobian matrix was derived as

$$^{0}J(\theta) = \begin{bmatrix} L_2(c_1c_3 - c_2s_1s_3) - L_1c_2s_1 & -L_1c_1s_2 - L_2c_1s_2s_3 & -L_2(s_1s_3 - c_1c_2c_3) \\ L_2(c_3s_1 + c_1c_2s_3) + L_1c_1c_2 & -L_1s_1s_2 - L_2s_1s_2s_3 & L_2(c_1s_3 + c_2c_3s_1) \\ 0 & L_1c_2 + L_2c_2s_3 & L_2c_3s_2 \end{bmatrix}$$

The Jacobian matrix can be used to determine the dexterity of a manipulator through a measure defined by Yoshikawa called the manipulability index [40, 33]. This measure describes the distance to singular configurations of the robotic manipulator and is defined as

$$w = \sqrt{det[J(\theta)J^T(\theta)]}.$$

The manipulability index was calculated across a cross-section of the 4-DOF robot's workspace of the right arm. The results were normalized to the maximum manipulability to produce a range with 1 being the highest manipulability and 0 being the lowest. The results are plotted in Figure 16. The figure shows a high manipulability value throughout the majority of the workspace, with the index dropping lower toward the edges of the workspace.

The equation for the no-load end effector velocity, assuming no gravitational effects and a massless arm, can be derived from the definition of the Jacobian with a minimal amount of manipulation [7].

$$J(\theta) = \frac{\partial x}{\partial \theta} = \frac{\partial x}{\partial t}\frac{\partial t}{\partial \theta} \longrightarrow \frac{\partial x}{\partial t} = J(\theta)\frac{\partial \theta}{\partial t}$$

or

$$\dot{x} = J\dot{\theta}$$

where $\dot{x}$ is the vector of linear and angular velocities and $\dot{\theta}$ is the array of joint an-

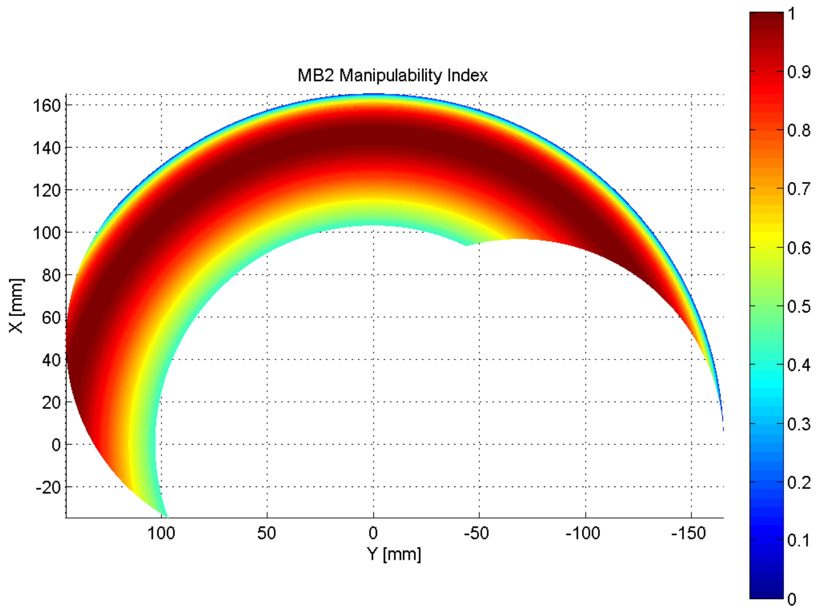


Figure 16: Manipulability index across the right arm workspace.

gular velocities. The theoretical maximum velocity of the end effector was calculated through a cross-section of the workspace by using the maximum angular velocities at each joint from Table 3. The maximum velocity of the end effector in the direction of each principal axis is displayed in Figure 17. While the numbers are only an estimation, they provide insight into the capabilities of the manipulator throughout the workspace. The mean velocities for the X, Y, and Z-directions are 489.4 mm/sec, 631.5 mm/sec, and 534.6 mm/sec, respectively. The minimum velocity capability for the x and y-directions is zero, but this only occurs when the arm is completely extended in the X or Y-direction, respectively. The minimum velocity capability in the Z-direction is 195.6 mm/sec and occurs when the elbow is turned to its limit. The maximum velocity for the X and Y-directions is 969.4 mm/sec and 739.6 mm/sec in the Z-direction.

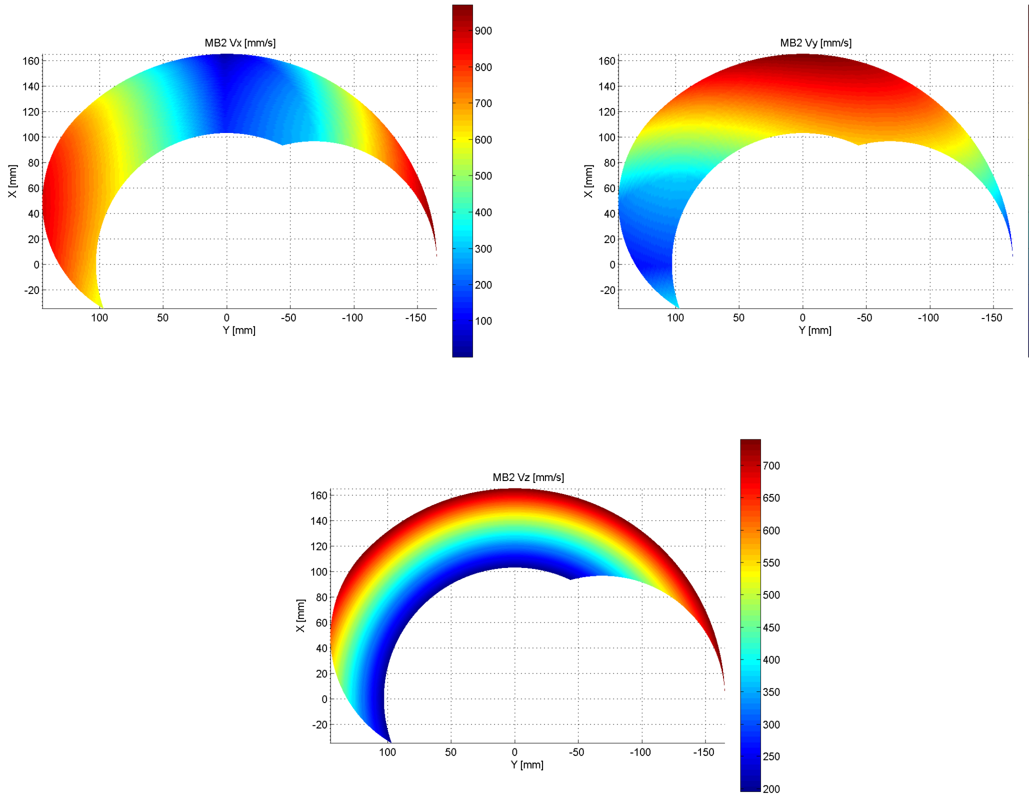


Figure 17: Maximum velocity of the end effector in the X, Y, and Z directions for the right arm.

The equation that relates joint torques to end effector force through the principle of virtual work [7] is

$$\tau =^{0} J^{T} F$$

where τ is the array of joint torques, ${}^{0}J^{T}$is the transpose of the Jacobian with respect to the base frame, and F is the 6x1 force/torque array. The forces at the end effector were numerically solved for across a cross-section of the workspace using the maximum joint torques from Table 3 and are shown in Figure 18. The forces were bounded with an upper limit of 30 N. The mean force in the X-, Y-, and Z-directions is 11.14 N, 9.65 N, and 24.81 N, respectively. The minimum forces are 4.6 N for the X and

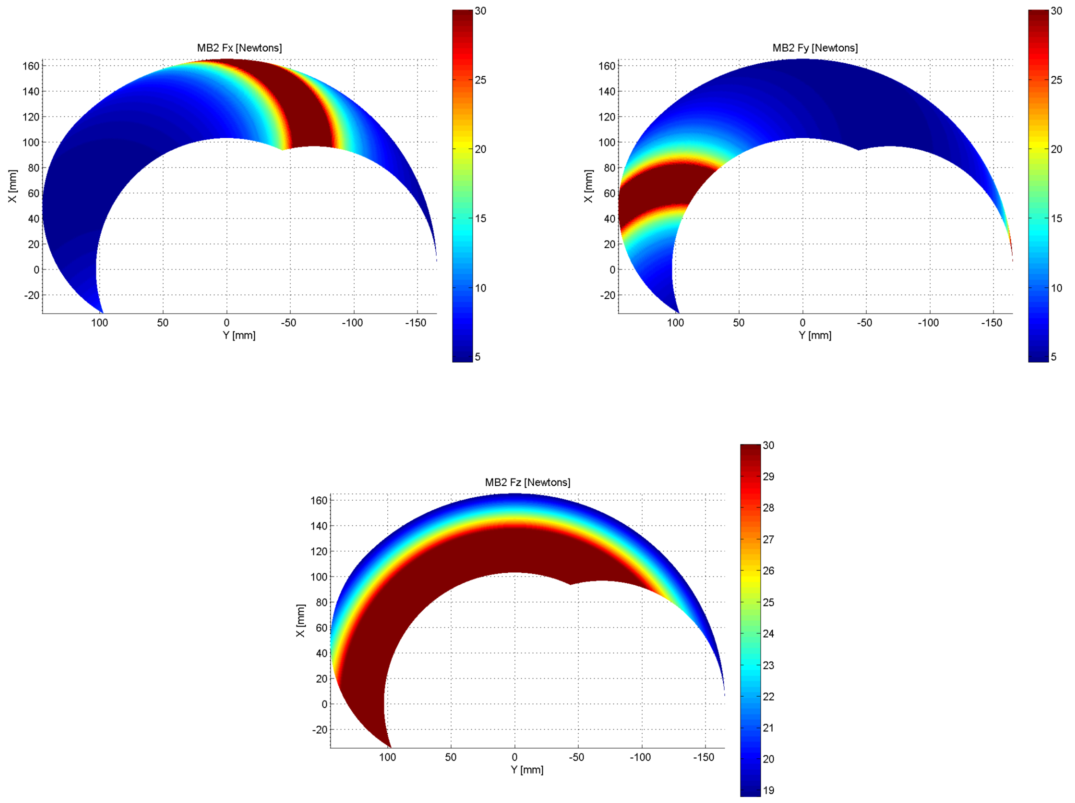


Figure 18: Maximum end effector force in the X, Y, and Z directions through a cross-section of the workspace.

Y-directions and 18.8 N for the Z-direction. The maximum force is 30 N for each direction.

Lou-Bot 1.0

A second version of miniature surgical robot was designed to take advantage of lessons learned in the CRB-1.0 design. The primary differences between Lou-Bot and CRB-1.0 lie in the design of the shoulder. The concentric shaft design was extended to allow for deeper insertion of the robot and to keep motors and control electronics outside of the body. An additional motor and driveshaft were added to each shoulder

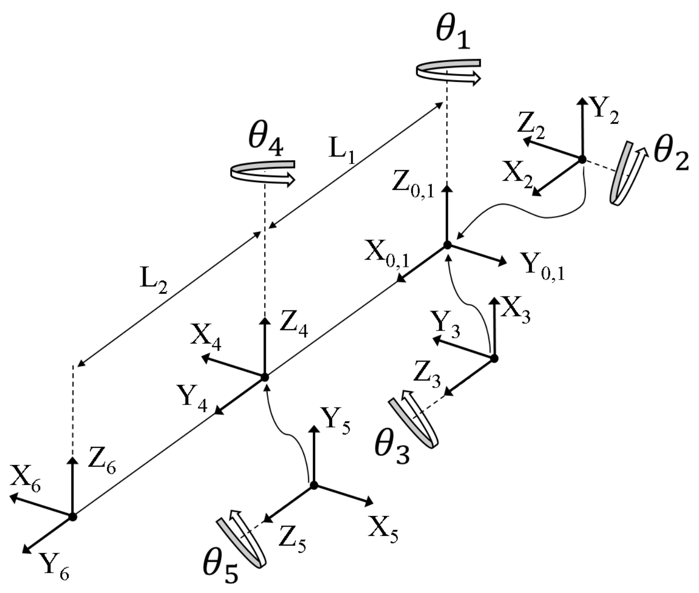


Figure 19: Kinematic frames assigned to the right arm of Lou-Bot 1.0.

drive assembly to create two independent 3-DOF shoulder joints in the same profile as the previously described shoulder. A custom miniature camera system is used in place of the endoscope, reducing the shoulder profile further by eliminating the need for a 5-mm port down the length of the shoulder. The inserted profile of the LB-1.0 shoulder has a cross-section area of 4.13 cm^2, which is 47% of the size of the CRB-1.0 shoulder.

Kinematics

Similarly to the previously described robot, kinematic frames were assigned to each joint of the arm and the DH parameters were extracted. The frame assignment is shown in Figure 19 with the corresponding DH parameters in Table 4. The sixth frame represents the end effector position and orientation.

The transformation matrix was derived as previously defined using the homoge-

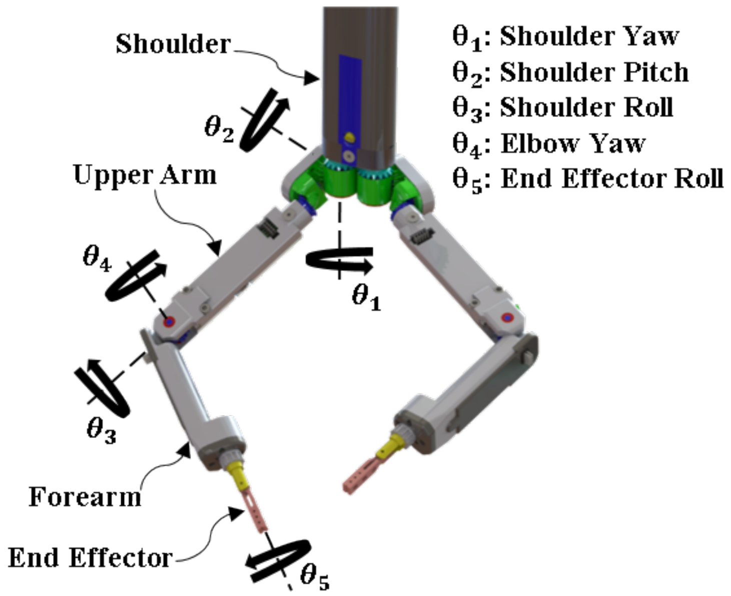


Figure 20: Lou-Bot 1.0 with rotation axes defined.

Table 4: DH parameters for the right arm of Lou-Bot 1.0.

i	α_i	a_i	d_{i-1}	θ_{i-1}
1	0	0	0	0
2	90°	0	0	ϕ_1
3	90°	0	0	$\phi_2 + 90°$
4	90°	0	$L_1 = 87.6$ mm	$\phi_3 + 90°$
5	90°	0	0	$\phi_4 + 180°$
6	90°	0	$L_2 = 86.6$ mm	$\phi_5 + 180°$

neous transformation matrix derived earlier.

$$T_6^0 = \begin{bmatrix} R_{00} & R_{01} & R_{02} & x \\ R_{10} & R_{11} & R_{12} & y \\ R_{20} & R_{21} & R_{22} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$R_{00} = c_5(c_4(s_1s_3 + c_1c_2c_3) + c_1s_4s_2) + s_5(c_3s_1 - c_1c_2s_3) \qquad R_{01} = c_1c_4s_2 - s_4(s_1s_3 + c_1c_2c_3)$$
$$R_{02} = s_5(c_4(s_1s_3 + c_1c_2c_3) + c_1s_4s_2) - c_5(c_3s_1 - c_1c_2s_3) \qquad R_{10} = -c_5(c_4(c_1s_3 - c_2c_3s_1) - s_1s_4s_2) - s_5(c_1c_3 + c_2s_1s_3)$$
$$R_{11} = s_4(c_1s_3 - c_2c_3s_1) + c_4s_1s_2 \qquad R_{12} = c_5(c_1c_3 + c_2s_1s_3) - s_5(c_4(c_1s_3 - c_2c_3s_1) - s_1s_4s_2)$$
$$R_{20} = -c_5(c_2s_4 - c_4c_3s_2) - s_5s_2s_3 \qquad R_{21} = -c_4c_2 - c_3s_4s_2$$
$$R_{22} = c_5s_2s_3 - s_5(c_2s_4 - c_4c_3s_2)$$

and

$$x = L_1c_1s_2 - L_2(s_4(s_1s_3 + c_1c_2c_3) - c_1c_4s_2)$$
$$y = L_2(s_4(c_1s_3 - c_2c_3s_1) + c_4s_1s_2) + L_1s_1s_2$$
$$z = -L_2(c_4c_2 + c_3s_4s_2) - L_1c_2$$

As can be seen by comparing the transformation matrices of the two robot manipulators, adding the extra degree of freedom significantly increases the complexity of the forward kinematics. Also, it becomes necessary to define both the position and orientation of the end effector to solve for the joint angles because there are multiple orientations possible at each reachable position in the workspace. Due to this added complexity, a closed form solution for the inverse kinematics was not calculated. Instead, a numerical inverse kinematics solver was implemented based on the cyclic coordinate descent method described by Wang (see section on inverse kinematics solver) [35].

Mechanical Design

The shoulder body houses six 12-mm Faulhaber® 1226 BLDC motors coupled to 256:1 planetary gearboxes, three for each shoulder joint. The motors are mounted radially about the driveshafts using face mount screws on each gearbox. A 14-tooth spur gear is press-fit onto the output of each gearbox and is then coupled to a 30-tooth spur gear. The 30-tooth spur gears are machined from stock pinion wire, allowing

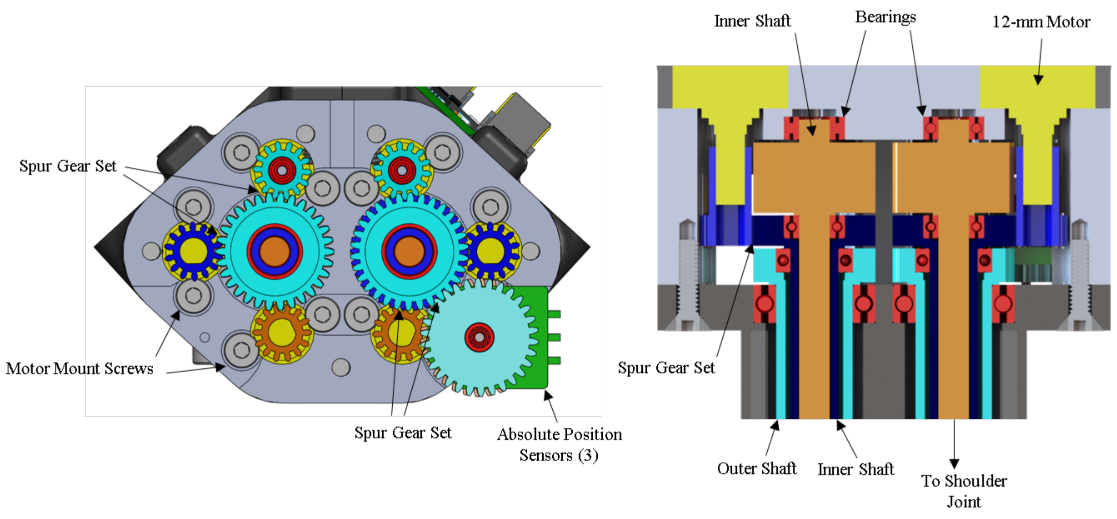


Figure 21: LB-1.0 drivetrain cross-section view.

the driveshafts to be turned down from a long piece of the pinion wire and making the gears and driveshaft a single part. Each of the spur gear shafts is also coupled to a thin spur gear mounted on an absolute position sensor. The outer and middle driveshafts are hollow, enabling a compact nested concentric driveshaft design. The shafts are four inches long to enable the shoulder motors and electronics to remain outside the body when the arms are inserted. The outer and middle shafts have a castled feature at their ends and the innermost shaft has a flatted feature to mate with the differential shoulder joint gears. The torques and angular velocities for each joint are tabulated at the end of this section.

The 3-DOF shoulder joint is similar to the 2-DOF shoulder joint but has added complexity to produce a spherical joint. The same differential joint as CRB-1.0 is used to produce the shoulder pitch and yaw joints with the outer shaft driving the upper sun bevel gear and the innermost shaft driving the lower sun bevel gear, both of which mate to a planet bevel gear machined into the shoulder output body. In order to create a third DOF which intersects the axes of the first two shoulder DOFs and

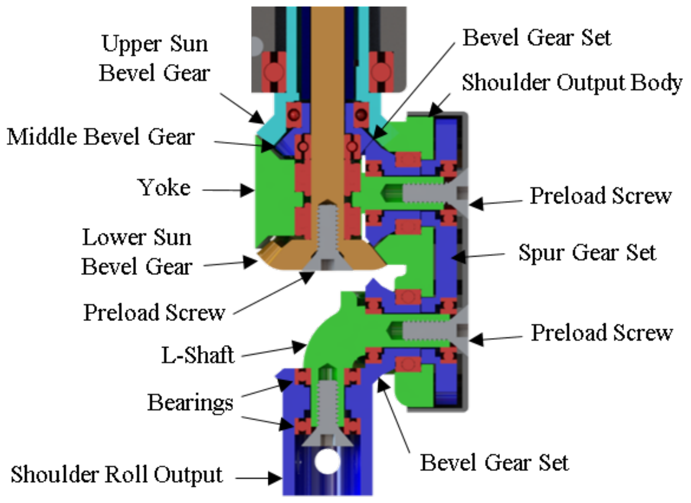


Figure 22: LB1.0 shoulder joint cross-section view.

therefore creates a spherical joint, a relatively complicated drivetrain is needed. The middle driveshaft mates to a smaller bevel gear which is nested inside the upper bevel gear. This smaller bevel gear drives an identical gear housed in the shoulder output body which is geometrically mated to a spur gear, both of which are mounted on the differential yoke output shaft. The spur gear drives an identical spur gear which is then geometrically mated to another bevel gear. An L-shaft is inserted through both the spur and bevel gear and is geometrically constrained to the shoulder output body. Finally, the bevel gear drives another bevel gear which is mounted to the other end of the L-shaft to produce the shoulder roll DOF. The upper arm is attached to this final bevel gear. This 3-DOF shoulder joint is not truly spherical as the shoulder yaw and roll become aligned when the arm is pointed straight downward. Similar to the 2-DOF differential joint, there is coupling between the three shoulder DOFs. The equations for the three bevel gear angles in terms of the kinematic angles are

$$\theta_{upper} = \theta_1 + \theta_2 \qquad \theta_{lower} = \theta_1 - \theta_2 \qquad \theta_{middle} = \theta_3 - \theta_1 - \theta_2$$

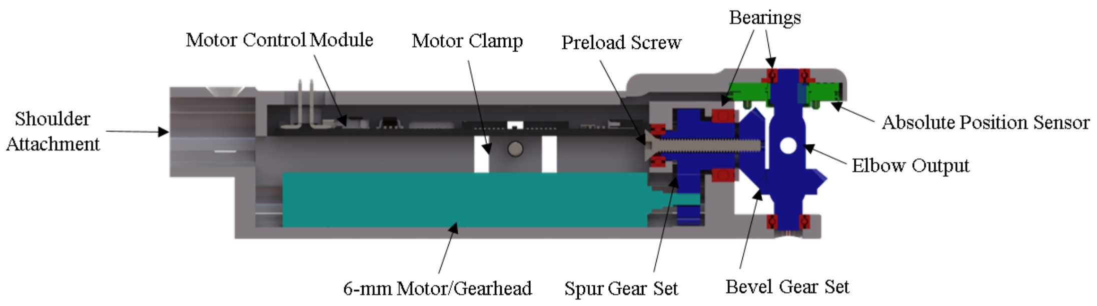


Figure 23: LB-1.0 upper arm link cross-section view.

The upper arm link of LB-1.0 is very similar to the upper arm of CRB-2.0. A 6-mm Faulhaber 0620 12V BLDC motor coupled to a 1024:1 gearbox is constrained inside the link through a motor clamping feature. A 12-tooth spur gear is press-fit onto the output of the gearbox and is coupled to an 18-tooth spur gear. The 18-tooth spur gear is geometrically mated to a bevel gear through a castle feature which drives the output shaft of the elbow joint. The main difference between with the LB-1.0 elbow design is in how the upper arm forearm link is attached to the elbow joint. The elbow joint range of motion was increased by removing the ability to move to negative positions. This greatly increased the workspace of the manipulator by allowing the elbow to move to a position from 0-150 degrees. The length of the upper arm link, 87.6-mm, is longer than MB-2.0 due to the added DOF in the shoulder joint.

The LB-1.0 forearm link overlaps the elbow joint which reduces its overall length. The forearm links connect to the elbow joint and house two motors, a motor controller, and a variety of custom tools that use a common interface. The two 6-mm Faulhaber 0620 12V BLDC motors are housed in the forearm and are constrained using motor clamping features. For the end effector roll drivetrain, a 12-tooth spur gear is press-fit onto the output shaft of the gearbox and is coupled to a 24-tooth spur gear about the circumference of the tool yoke. The tool actuation is driven by a 24:12 spur gear set, with the 12-tooth gear press-fit onto the output of the gearbox. The total length of

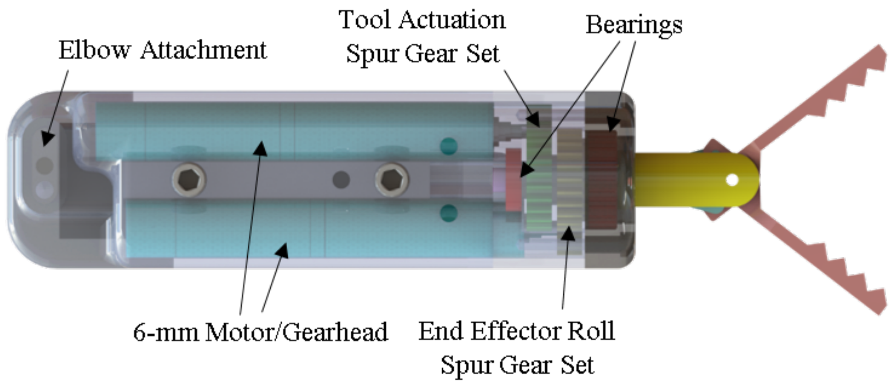


Figure 24: LB-1.0 forearm link with labeled components.

Table 5: LB-1.0 theoretical joint torques and angular velocities.

i	n_i	τ_i	η_{m_i} [%]	τ_{m_i}	T_{m_i} [mNm]	ω_{m_i} [rpm]	T_i [mNm]	ω_i [rpm]
1	2	30:14 = 2.14	60	256	8.99	27,400	2670.49	49.95
2	2	30:14 = 2.14	60	256	8.99	27,400	2670.49	49.95
3	4	30:14 = 2.14	60	256	8.99	27,400	2410.12	49.95
4	2	18:12 = 1.5	55	1024	0.551	37,300	420.10	24.28
5	1	24:12 = 2	55	1024	0.551	37,300	589.61	18.21

the forearm link from elbow joint axis to grasper midpoint is 86.6-mm. The forearm with components labeled is shown in Figure 24.

Workspace, Forces, & Velocity

The workspace of LB-1.0 was greatly increased over the CR-1.0 workspace through the additional DOF and the increased range of the ϑ_4 elbow joint. The volume of the LB-1.0 workspace was modeled using CAD software. The workspace volume for each arm is 9467.7 cm^3, with 7658.9 cm^3 of shared workspace between the two arms. This results in a total of 11276.5 cm^{3}of workspace for the robot. The shared workspace is 67.9% of the total robot workspace. The robot with its reachable workspace is shown in Figure 25.

The velocity, force, and dexterity at the end effector were analyzed using the same

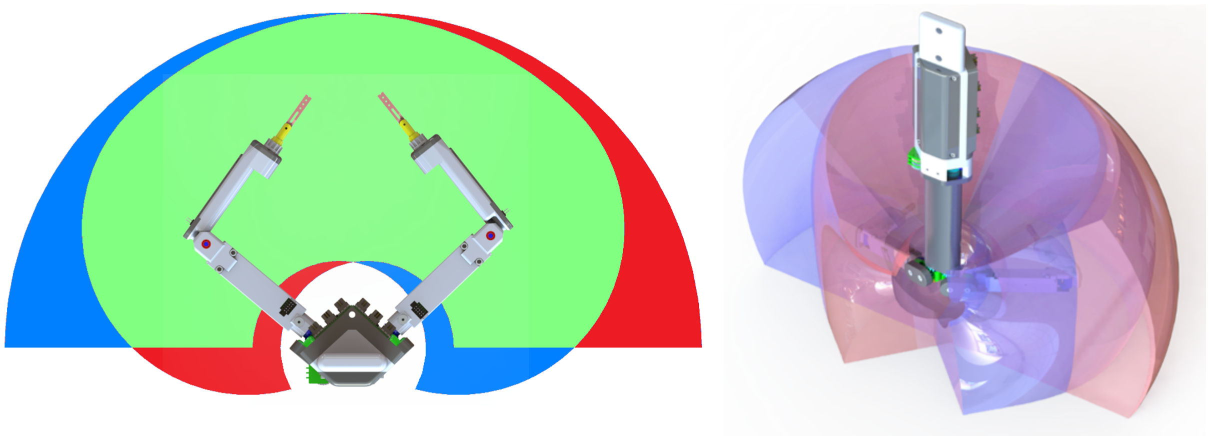


Figure 25: LB-1.0 within bubbles of the right arm (red), left arm (blue), and shared (green) workspace.

methods as for CRB-1.0. The Jacobian matrix with respect to the base frame was calculated using the MATLAB jacobian() function as

$$ {}^{0}J(\theta) = \begin{bmatrix} J_{X1} & J_{X2} & J_{X3} & J_{X4} & J_{X5} \\ J_{Y1} & J_{Y2} & J_{Y3} & J_{Y4} & J_{Y5} \\ J_{Z1} & J_{Z2} & J_{Z3} & J_{Z4} & J_{Z5} \end{bmatrix} $$

where

$$J_{X1} = -L_2(s_4(c_1s_3 - c_2c_3s_1) + c_4s_1s_2) - L_1s_1s_2$$

$$J_{Y1} = L_1c_1s_2 - L_2(s_4(s_1s_3 + c_1c_2c_3) - c_1c_4s_2)$$

$$J_{X2} = L_2(c_1c_4c_2 + c_1c_3s_4s_2) + L_1c_1c_2$$

$$J_{Y2} = L_2(c_4c_2s_1 + c_3s_1s_4s_2) + L_1c_2s_1$$

$$J_{Z2} = L_2(c_4s_2 - c_2c_3s_4) + L_1s_2$$

$$J_{X3} = -L_2s_4(c_3s_1 - c_1c_2s_3)$$

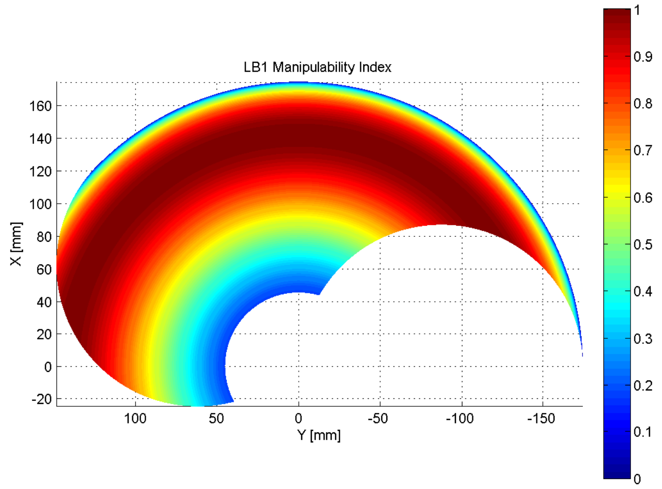


Figure 26: LB-1.0 manipulabilty index across cross-section of workspace.

$$J_{Y3} = L_2 s_4 (c_1 c_3 + c_2 s_1 s_3)$$

$$J_{Z3} = L_2 s_4 s_2 s_3$$

$$J_{X4} = -L_2 (c_4 (s_1 s_3 + c_1 c_2 c_3) + c_1 s_4 s_2)$$

$$J_{Y4} = L_2 (c_4 (c_1 s_3 - c_2 c_3 s_1) - s_1 s_4 s_2)$$

$$J_{Z4} = L_2 (c_2 s_4 - c_4 c_3 s_2)$$

$$J_{Z1} = 0 \quad J_{X5} = 0 \quad J_{Y5} = 0 \quad J_{Z5} = 0$$

The manipulabilty index for LB-1.0 was calculated using the formula described previously. The values were normalized to the maximum value resulting in a high values near 1 and low values near zero. The values across a cross-section of the right arm's workspace are shown in Figure 26.

The theoretical maximum velocities in the direction of each of the three principal axes were calculated assuming a massless arm using the the maximum angular velocities for each joint in Table 5. The results for a cross-section of the workspace are plotted in Figure 27. The mean velocities for the X-, Y-, and Z-directions are

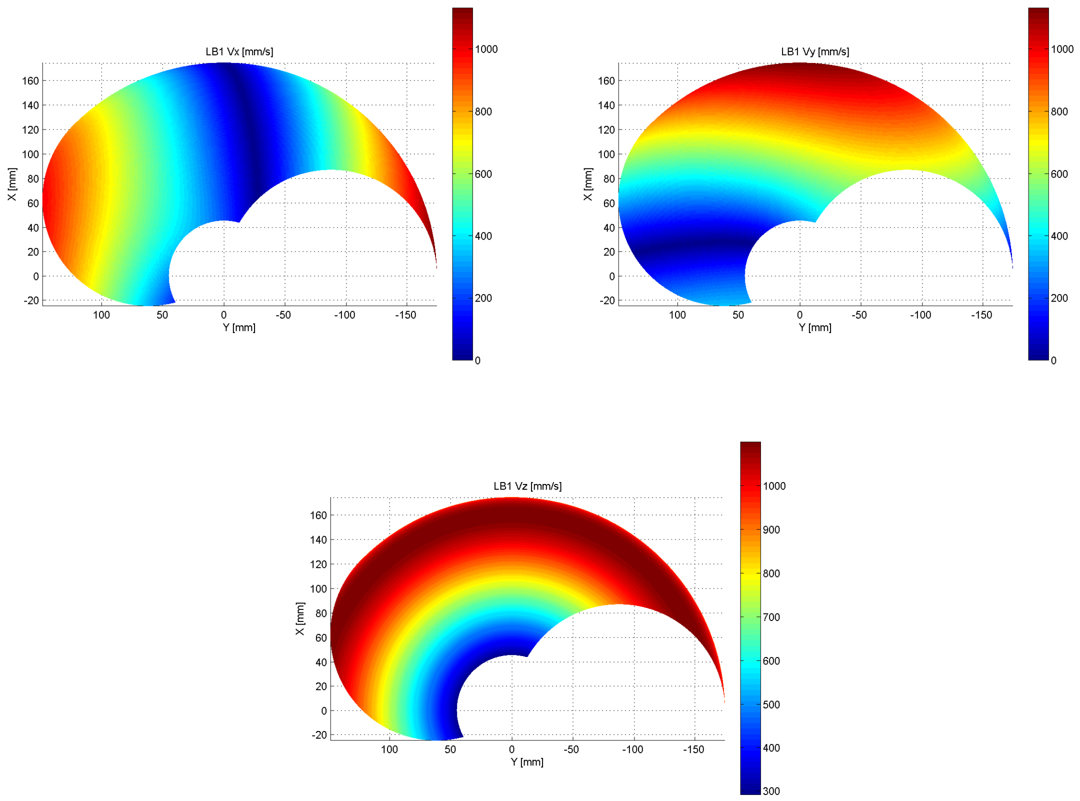


Figure 27: LB-1.0 theoretical end effector velocities in the direction of the three principal axes.

486.8 mm/sec, 601.9 mm/sec, and 865.6 mm/sec, respectively. The minimum velocity capability for the X and Y-directions is zero, but this only occurs when the arm is completely extended in the X or Y-direction, respectively. The minimum velocity capability in the Z-direction is 292.8 mm/sec and occurs when the elbow is turned to its limit. The maximum velocity for the X and Y-directions is 1131.2 mm/sec and 1098.6 mm/sec in the Z-direction.

The theoretical end effector forces were calculated based on the maximum joint torques in Table 5. The forces in the directions of the three principal axes are shown in Figure 28. The forces were bounded with an upper limit of 30 N. The mean force

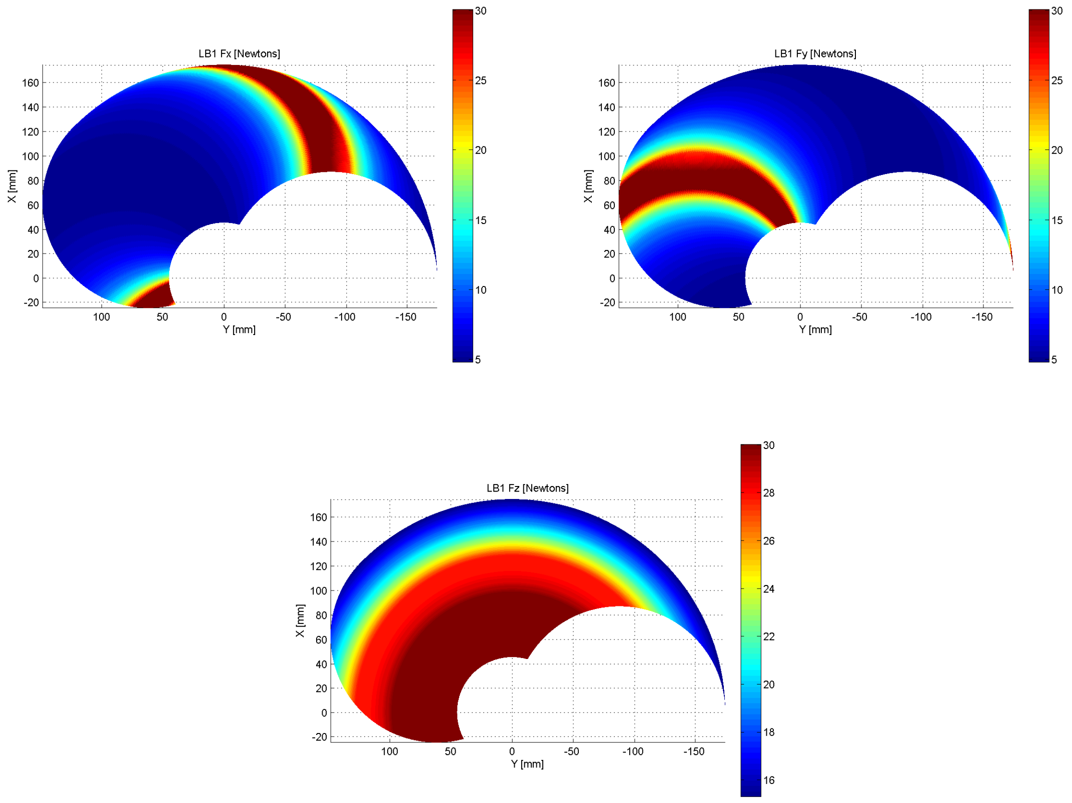


Figure 28: LB-1.0 theoretical end effector forces in the direction of the three principal axes.

in the X-, Y-, and Z-directions is 10.53 N, 10.71 N, and 23.39 N, respectively. The minimum forces are 4.8 N for the X and Y-directions and 15.3 N for the Z-direction. The maximum force is 30 N for each direction.

Motor Control Modules

Each joint of the previously described robots has a local motor control module that is responsible for controlling the motor in that joint. Each of these modules share the same bus for power and data . This simplifies cable management, as there are only four wires that run the length of each arm. The modules are composed of a 75

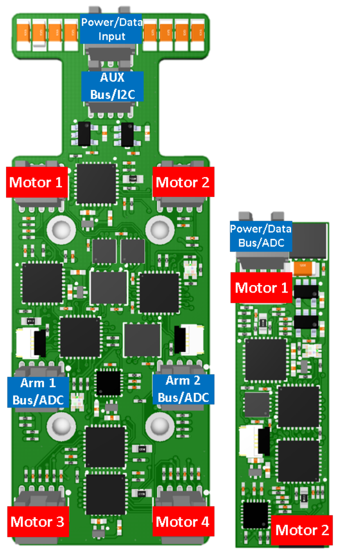


Figure 29: Shoulder and arm versions of the motor control modules.

MHz Cortex-M0+ microcontroller, two brushless DC bridge drivers, a 9-DOF inertial measurement unit, high-speed non-volatile FRAM memory, and a RS-485 transceiver connected to the data bus [8]. The schematic for the electrical design in shown in the appendix.

The modules interface with the bus using micro-pitch 2x5 crimp-on headers and 0.25” pitch cable, enabling very easy cable assembly. Since the bus uses only four wires, the unused wires on the 10-pin ribbon cable can either be removed or used for local auxiliary functionality; the unused wires can be used as analog or digital I/O, or as an I^2C bus, with the local motor control module acting as a bus master.

Two different forms of the PCB were created; one version was designed to drive four motors and be housed on the shoulder body and the other was designed to drive two motors and be installed on the arms. The two versions are shown in Figure 29 and an example of the motor control module layout is shown in Figure 30.

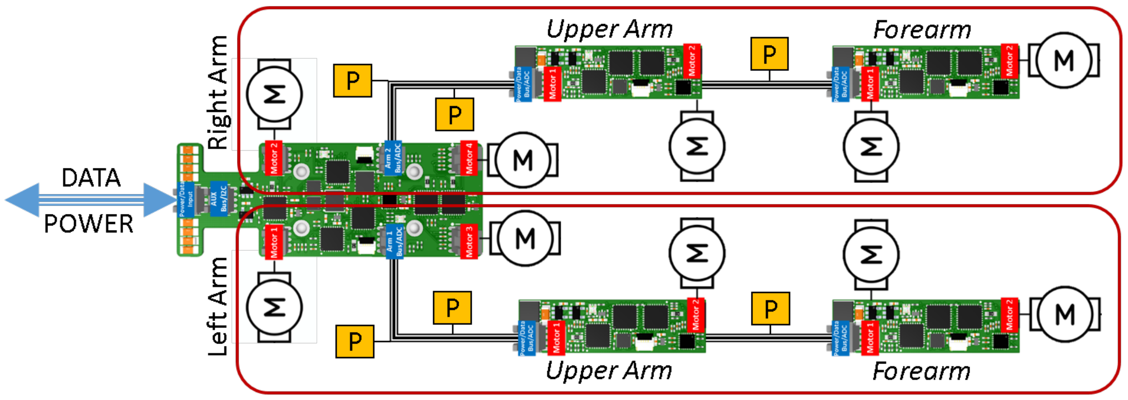


Figure 30: Motor control module layout for the 4-DOF robot, where (M) are motors and [P] are absolute position sensors.

Position Control

The brushless DC motors for each joint are commuted using three Hall effect sensors built into the actuators. This Hall sensor feedback is also used for relative position control of each joint through a simple proportional control scheme. The BLDC bridge drivers are controlled through an 8-bit pulse width modulated (PWM) pin on the MCU which sets the power sent to the motor as a value from 0-255.

The angular resolution for each joint is sufficient due to the high gear ratios coupled to each motor (256:1 or higher). Internal counters on the MCU are updated on interrupt events driven by changes in the Hall sensor signals. The direction of the motor can be determined by looking at two of the Hall signals when the interrupt is triggered, as shown in Figure 31. If the both of the signals are in the same state, the motor is rotating clockwise and the counter is increased; if the signals are opposite, the motor is rotating counter-clockwise and the counter is decreased. Initially the counters were updated only on the rising edge of the Hall signal, but this was found to lose position when close to the angular setpoint and changing directions quickly. Modifying the interrupt to trigger on both the rising and falling edges of the Hall

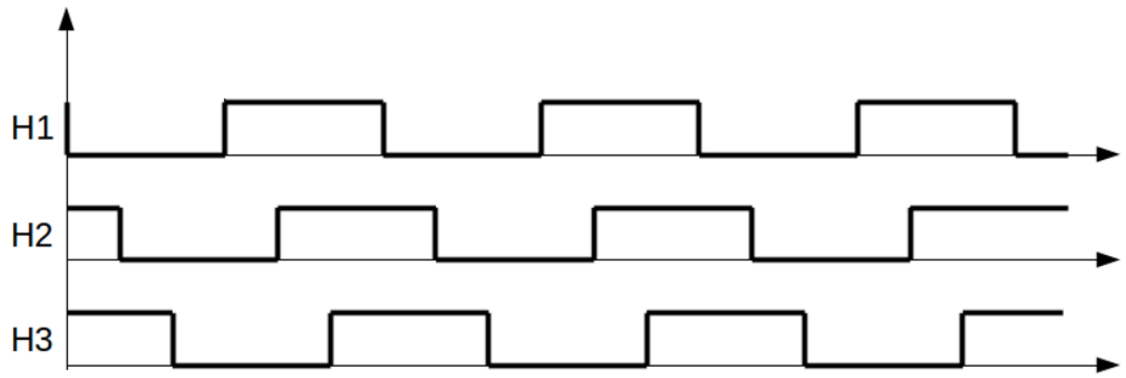


Figure 31: Hall sensor feedback from rotating brushless DC motor.

signal solved this problem.

Absolute position feedback is provided by potentiometers installed at the outputs of each joint. The potentiometers are wired as voltage dividers with the outputs wired to the analog inputs at the local motor module bus. These sensors could be used in the position control loop, but the small potentiometers used in the robot design have a non-linear position-to-output ratio which would require additional computation in the control loop. The sensors are instead used to provide absolute positioning during robot initialization.

Current-Torque Control

The current consumption of each of the BLDC bridge drivers is fed into the MCU for measurement. This data signal is fairly noisy and not very useful without conditioning. A moving average of the current measurement is used to reduce the noise in the signal and produce usable data. Instead of keeping a large array of past current values, computing the sum, and dividing it by the total number of samples, a more computationally efficient method was used. The moving average was calculated as

$$MA = (MA_p + I - MA_p/N)/N$$

where MA is the moving average, MA_p is the previous moving average, I is the current measurement, and N is the number of samples in the moving average. This technique has the advantage of not needing to store measurement values. If N is a power of two the division can be performed as a bit-shift, which is computationally efficient.

The conditioned data is used to perform current limiting for each motor. This is accomplished by setting a current threshold for each motor. The position control scheme is overridden when the current measurement is above the set threshold and the BLDC bridge driver PWM control pin is incremented down until the current falls below the limit. An example of the current limiting is shown in Figure 32. The current limiting method effectively limits the torque of the motor because the motor current is proportional to the output torque. This is useful to protect the mechanical components of each joint's drivetrain from overloading.

Non-Volatile Memory

The motor control modules use the non-volatile FRAM (Ferroelectric Random-Access Memory) to retain motor control information while powered off. The motor control mode, proportional gain, setpoint, current position, gear ratio, speed limit, current limit, and home value for the absolute position sensor are stored for each joint. This is very helpful in the event of power loss during operation because the robot can simply be re-powered and continue operation without any initialization.

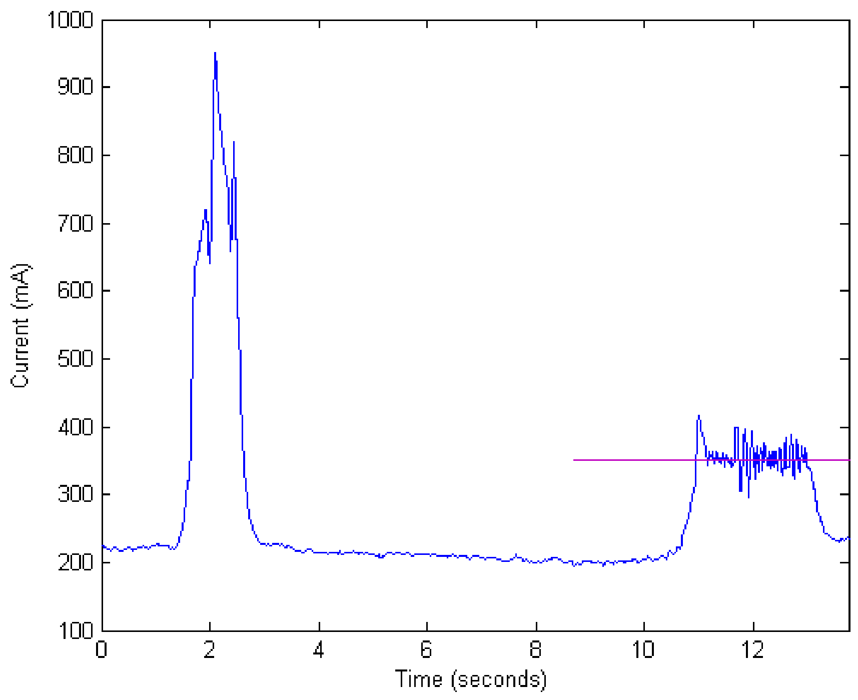
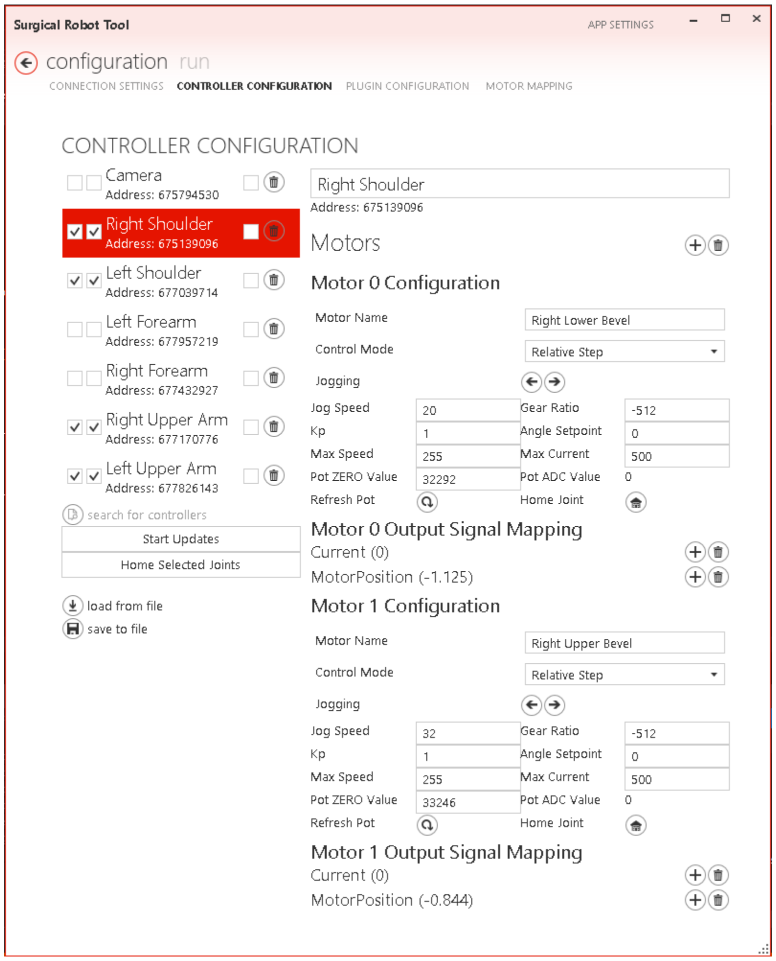


Figure 32: Current versus time for a non-limited and limited motor load.

Custom Software Stack

Instead of using off-the-shelf control/automation software (such as the National Instruments LabVIEW package), a custom software stack built on the .NET framework was developed to provide a flexible control structure to accommodate a wide variety of input and output devices. The program was designed to provide the core robot services and move all other functionality to an extensible plug-in infrastructure. The software architecture is composed of a communications layer (which abstracts the hardware communication transport between the computer and robot), a robot layer (which abstracts a specific set of motors, control modules, and robot-specific parameters), motor command and feedback layer (which defines joint-specific gear ratios, current limits, setpoints, jogging capabilities, and position feedback), and a plug-in



Surgical Robot Tool

configuration run

CONNECTION SETTINGS · **CONTROLLER CONFIGURATION** · PLUGIN CONFIGURATION · MOTOR MAPPING

CONTROLLER CONFIGURATION

☐ ☐	Camera	Address: 675794530
✓ ✓	Right Shoulder	Address: 675139096
✓ ✓	Left Shoulder	Address: 677039714
☐ ☐	Left Forearm	Address: 677957219
☐ ☐	Right Forearm	Address: 677432927
✓ ✓	Right Upper Arm	Address: 677170776
✓ ✓	Left Upper Arm	Address: 677826143

Right Shoulder
Address: 675139096

Motors

Motor 0 Configuration

Motor Name	Right Lower Bevel		
Control Mode	Relative Step		
Jog Speed	20	Gear Ratio	-512
Kp	1	Angle Setpoint	0
Max Speed	255	Max Current	500
Pot ZERO Value	32292	Pot ADC Value	0

Motor 0 Output Signal Mapping
Current (0)
MotorPosition (-1.125)

Motor 1 Configuration

Motor Name	Right Upper Bevel		
Control Mode	Relative Step		
Jog Speed	32	Gear Ratio	-512
Kp	1	Angle Setpoint	0
Max Speed	255	Max Current	500
Pot ZERO Value	33246	Pot ADC Value	0

Motor 1 Output Signal Mapping
Current (0)
MotorPosition (-0.844)

Figure 33: Robot control software user interface.

architecture (where all other run-time function is programmed) [8]. A screenshot of the software's controller configuration user interface is shown in Figure 33.

Communication Layer

The communication layer provides an architecture that abstracts computer-robot communication, allowing support to be built for serial ,USB, Bluetooth, or any other arbitrary communication interface. The current robot hardware only allows for serial communication, abstracted using a USB interface on the computer side. This abstraction also allows for a “virtual robot” simulation interface to be built to be used as a software dummy for development of control algorithms without the need for hardware to be present [8].

Robot Layer

The robot layer handles control and data services to discover, control, configure, and read motor control modules. Each robot configuration has a collection of zero or more controllers, which themselves can have zero, one, or two motors. Controllers are auto-discovered and identified by unique controller ID numbers. Each controller has a “friendly name” property that can be used to describe information such as the location of the motor control module. Motors can be added to the controller once it is instantiated. Each motor has a “name” property which can be used to describe the joint, as well as controls for jogging the motor, keying in setpoints manually, and setting the joint properties (gear ratio, current limits, maximum joint speed, and proportional gain for the position control loop). Each motor also has a button that can be used to home the joint based on the absolute position senor. Each motor also has two outputs which can routed arbitrarily: motor current and position [8].

Plug-in Layer

The robot control software utilizes an extensible plug-in infrastructure that allows individual software modules to publish and subscribe to data. Each plug-in is composed of a configuration pane, any number of named inputs, and any number of named outputs. To increase performance, the plug-ins are implemented as a derived class with no dynamic typing. This restricts all inputs and outputs to real-valued numbers represented by double-precision floating point numbers. Because each plug-in is implemented as a class, the programming language includes built-in capabilities to discover plug-ins that are part of the compiled assembly and instantiate the same plug-in one or more times [8].

Once a plugin is instantiated, its input list (containing zero or more items) is added to the global input registry. When a plugin is removed, its inputs are also removed from the list. Plug-ins can also have zero or more outputs. The base plugin GUI provides an interface to direct the outputs to zero or more inputs in the global input registry. Data is "pushed" from plug-in to plug-in, starting at the user input and ending with commands to the robot.

Besides the core robot control functionality, all other functionality during run-time is written into plug-ins. The plug-ins handle the basic functions necessary to control the robot, such as inverse kinematics and user input interfaces, as well as others that provide higher level functionality. Some provide haptic workspace functionality, the ability to record and play back robot actions, or scale the workspace for finer control of robot motion. A section of the most relevant plug-ins will be discussed next.

Geomagic Touch

The Geomagic Touch plug-in interfaces to the popular off-the-shelf haptic controller with the robot control software. The Geomagic Touch® is a cable-driven, motorized haptic device that provides position and orientation feedback from the controller as well as three-DOF of force feedback into the user's hand. As such, the plug-in provides bi-directional communication between the controller and master computer allowing the controller to output raw position and orientation with respect to the controller's base frame as well as receive force input from any number of plug-ins.

Several plug-ins were developed to condition the raw position data from the con-

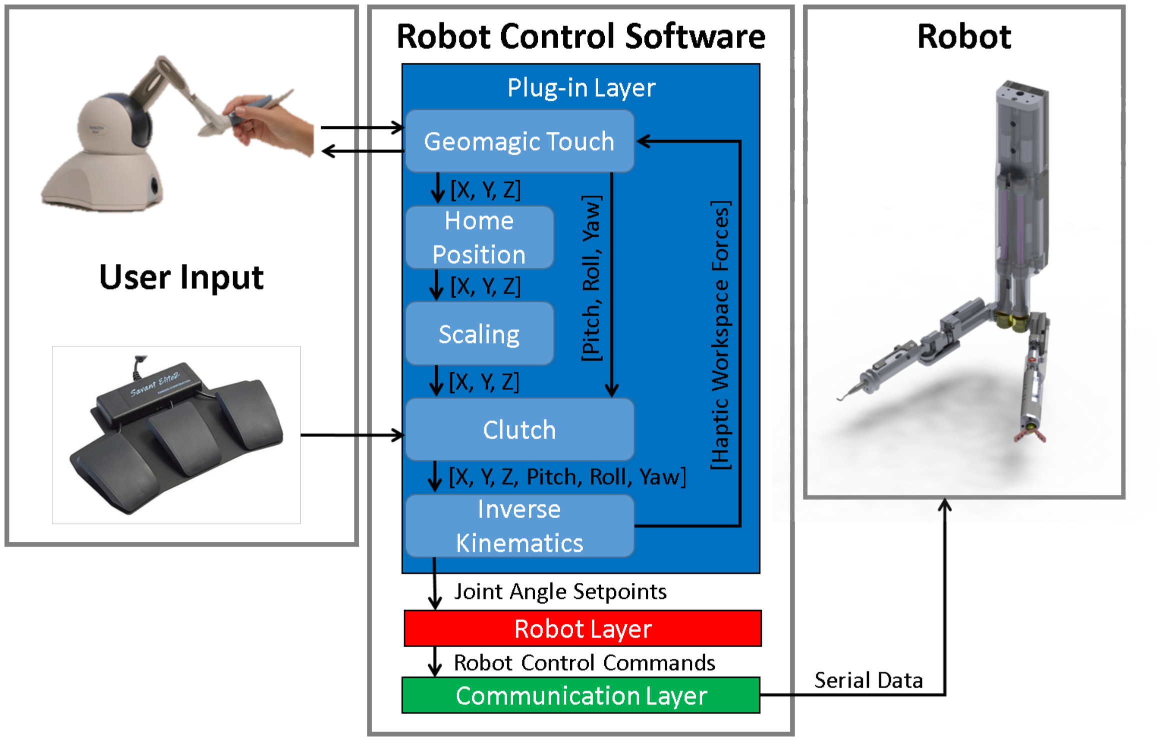


Figure 34: Flow chart of a typical plug-in architecture for the surgical robot.

troller. Among these plug-ins are are homing, scaling, and clutching functions. The homing plug-in was developed to reduce the complexity of the inverse kinematics by allowing the user to invert the raw position data and set a new base frame for the device, allowing the use of the same kinematic model for both arms. The scaling plug-in was developed to allow scaling of the input position by a user-defined ratio. This function allows the user to have very fine control of the robot position, if desired. When the user input is scaled down, the controller may run into the physical limits of the device before the physical limits of the robot arm. To address this issue, a clutching plug-in was developed. This function allows the user to press and hold an input button and move the controllers back into their useful volume without changing the position of the robot.

Hardware Interface

Plug-ins have been developed to interface additional hardware with the control software. Any microcontroller with serial communications can be combined with digital or analog input hardware to create custom controllers that easily integrate with the robot control architecture. Currently, the only input hardware used are a set of foot-pedals to complement the haptic controllers.

Haptic Workspace

The haptic feedback capabilities of the controllers allow the implementation of force barriers to keep the controller inside the robot's reachable workspace. Viscous forces are also applied to the controllers to try to reduce hand tremors in the operator and damp oscillations if the controllers are dropped. Additionally, a plug-in function to haptically lock the controllers and pause the position output to the robot has been developed.

In an effort to make the haptic workspace as generic as possible, haptic forces are calculated using the forward kinematics of the robot and joint limits. If the controller enters a spot that the robot cannot reach, the forward kinematics are calculated for the actual robot position based on the limited joint angles and subtracted from the position of the controller. The resulting displacement vector is multiplied by a spring constant to yield a force vector pushing the controller to the actual robot position, shown in the following equation:

$$\vec{F} = k_s(P_d - P_h)$$

where k_s is the spring constant, P_d is the desired position, and P_h is the actual position from the forward kinematics.

Numerical Inverse Kinematics Solver

While deriving a closed form inverse kinematic solution for a simple manipulator is fairly simple, the difficulty increases quickly as the number of joints is increased. Some manipulator configurations do not have a closed form solution at all. To address this problem and allow rapid development of different manipulator configurations, a numerical inverse kinematics solver algorithm was implemented based on the work of Li-Chung Tommy Wang [35]. This algorithm uses a cyclic coordinate descent (CCD) method to iteratively solve the inverse kinematics problem with only the DH parameters of the manipulator.

The algorithm starts with an initial guess for the joint angles and applies forward recursion formulas to determine the location and orientation of each of the kinematic frames with respect to the base frame [34]. The forward recursion formulas are

$$x_{i+1} = x_i cos\theta_i + y_i sin\theta_i$$
$$z_{i+1} = z_i cos\alpha_i + (x_{i+1} \times z_i) sin\alpha_i$$
$$y_{i+1} = z_{i+1} \times x_{i+1}$$
$$p_i^* = d_i z_i + a_i x_{i+1}$$
$${}^0p_{i+1} ={}^0 p_i - p_i^*$$

for $i = 0$ to $n - 1$ and where x_i, y_i, z_i are the the unit vectors for the orientation of the frame $[i]$, p_i^* is the position of frame $[i]$ with respect to the previous frame, and ${}^0p_i$ is the position of frame $[i]$ with respect to the base frame. These equations show that the position and orientation of each kinematic frame can be calculated if the base frame location/orientation and DH parameters are known. The position and orientation of the base frame are static and are defined as

$$x_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad y_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad {}^0p_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Once the position of each of the frames is defined, the current end effector position (P_h) is defined as the position of the last kinematic frame. The relative positions of each of the kinematic frames with respect to the end effector position is calculated and is defined as P_{ih} for each joint i. The error between the the desired position (P_d) and current position is

$$\begin{aligned} \Delta P(\vec{\theta}) &= (P_d - P_h) \cdot (P_d - P_h) \\ \Delta O(\vec{\theta}) &= \sum_{j=1}^{3} \sigma_j (R_{dj} \cdot R_{hj}(\vec{\theta}) - 1)^2 \\ E &= \Delta P(\vec{\theta}) + \Delta O(\vec{\theta}) \end{aligned}$$

where $\vec{\theta}$ is the vector containing the joint variable values, $\Delta P(\vec{\theta})$ is the position error, $\Delta O(\vec{\theta})$ is the orientation error, and E is the total error of using the current joint angles. In some practical applications, the orientation of the end effector may not need to be specified. In these cases,

$$\sigma_j = \begin{cases} 1 & \text{if the jth direction needs specifiying} \\ 0 & \text{otherwise.} \end{cases}$$

where $j = 1, 2, 3$ correspond to the X, Y, and Z components of the end effector frame, respectively. In the case of the four-DOF robot arm, only wrist roll orientation is possible, which corresponds to the z-component of the frame, so $\sigma_1$and σ_2 would be set equal to zero.

The objective function to minimize is the equation for the error (E). The CCD method is a heuristic direct search method with each cycle consisting of n steps. At the ith step of each cycle (i varying from 1 to n) only the ith joint angle variable is changed to minimize the objective function. The position and orientation of the end effector is updated only after each completed cycle through the joints. The cyclic process is continued until the value of the error function reaches a predetermined small tolerance.

In the case of a rotational joint, as is the case with all of the surgical robot joints, P_{ih} can be considered as a vector from the ith joint to the end effector. The expression for the P_{ih} vector rotated about the z_i axis by angle ϕ can be written as

$$P'_{ih}(\phi) = R(z_i, \phi)P_{ih}$$

where $R(z_i, \phi)$ is the 3×3 rotation matrix about z_i. Because the other joints are not allowed to move, the position error for the joint then becomes

$$\Delta p(\phi) = (P_{id} - P'_{ih}(\phi)) \cdot (P_{id} - P'_{ih}(\phi)).$$

Substituting the equation for P'_{ih} into the error equation and noting that the equation for R is orthogonal yields

$$\Delta p(\phi) = P_{id} \cdot P_{id} + P_{ih} \cdot P_{ih} - 2P_{id} \cdot (R(z_i, \phi)P_{ih}).$$

Minimizing the position error equation becomes the same as maximizing the negative part of the expression because the values of P_{id} and P_{ih} are constants for the cycle, yielding the following expression to be maximized:

$$g_1(\phi) = P_{id} \cdot (R(z_i, \phi) P_{ih}).$$

Using the same logic for the orientation error, the expression for the end effector orientation vectors are rotated about z_i by the angle $d\theta$ is

$$r'_{hj}(\phi) = R(z_i, \phi) R_{hj} \qquad \text{for } j = 1 \text{ to } 3$$

and the orientation error becomes

$$\Delta o(\phi) = \sum_{j=1}^{3} (R_{dj} \cdot r'_{hj}(\phi) - 1)^2$$

Since both R_{dj} and r'_{hj} are both unit vectors, they can be related to the direction angle $\psi_j(\phi)$ between them with the following expression:

$$R_{dj} \cdot r'_{hj}(\phi) = cos\psi_j(\phi).$$

The orientation error can then be written

$$\Delta o(\phi) = \sum_{j=1}^{3} (cos\psi_j(\phi) - 1)^2.$$

Due to the fact that $cos\psi_j$ is bounded between +1 and -1, minimizing the orientation error is the same as maximizing

$$g_2(\phi) = \sum_{j=1}^{3} \sigma_j cos\psi_j(\phi)$$

where σ_j is defined as before. Combining the two expressions to be maximized yields the new objective function for the joint

$$g(\phi) = w_p g_1(\phi) + w_o g_2(\phi)$$

where w_p and w_o are arbitrary weighting factors for the position and orientation, respectively.

The problem now becomes finding the value of ϕ such that the objective function is maximized. This can be accomplished analytically by utilizing the vector form of Rodrigues' equation and substituting the expression into the new objective function. The objective function becomes

$$g(\phi) = k_1(1 - cos\phi) + k_2 cos\phi + k_3 sin\phi$$

where k_1, k_2, and k_3 are constant coefficients defined as

$$\begin{aligned} k_1 &= w_p(P_{id} \cdot z_i)(P_{ih} \cdot z_i) + w_o \sum_{j=1}^{3} \sigma_j (R_{dj} \cdot z_i)(R_{hj} \cdot z_i) \\ k_2 &= w_p(P_{id} \cdot P_{ih}) + w_o \sum_{j=1}^{3} \sigma_j (R_{dj} \cdot R_{hj}) \\ k_3 &= z_i \cdot \left[w_p(P_{ih} \times P_{id}) + w_o \sum_{j=1}^{3} \sigma_j (R_{hj} \times R_{dj}) \right]. \end{aligned}$$

The objective function is maximized when

$$\frac{dg(\phi)}{d\phi} = (k_1 - k_2)\sin\phi + k_3 \cos\phi = 0$$

and

$$\frac{d^2 g(\phi)}{d\phi^2} = (k_1 - k_2)\cos\phi - k_3 \sin\phi < 0.$$

A value for ϕ can be found easily by solving the first derivative equation and checking the result with the second derivative condition

$$\phi = \arctan\left(\frac{k_3}{k_2 - k_1}\right) \qquad \phi > \arctan\left(\frac{k_1 - k_2}{k_3}\right).$$

The value of ϕ should bounded at the upper and lower mechanical limits of the joint for solutions outside of the joint's range. The implemented code is shown in the appendix.

V-REP Simulation Interface

A plugin has been created in our robot control application to interface with the Virtual Robot Experimental Platform (V-REP) (Coppeliar Robotics) software and allow direct control of the virtual robot using our kinematic models. The V-REP software has many features including the ability to place vision sensors on the robot, view the simulation through the vision sensor's perspective, as well as physics modeling and primitive collision detection. The plugin is very simple to use and the operation of the virtual robot is the same as operating the robot hardware. The plug-in uses the open-source V-REP Remote API framework to connect to the V-REP software as a client and starts a communication thread over a network IP. The thread is used to request all joint handles for the currently open model. An input is added to the input signal registry for each joint, at which point any other plug-in can output joint angles to the robot simulation using the same input/output mapping used to communicate with the actual robot as shown in Figure 35.

The V-REP platform is specifically built for the modeling of robotic systems. It includes tools to quickly generate kinematic models from DH parameters and vise-versa which, coupled with the numerical inverse kinematics solver, enables rapid testing of new manipulator configurations without the need to build any hardware. The simu-

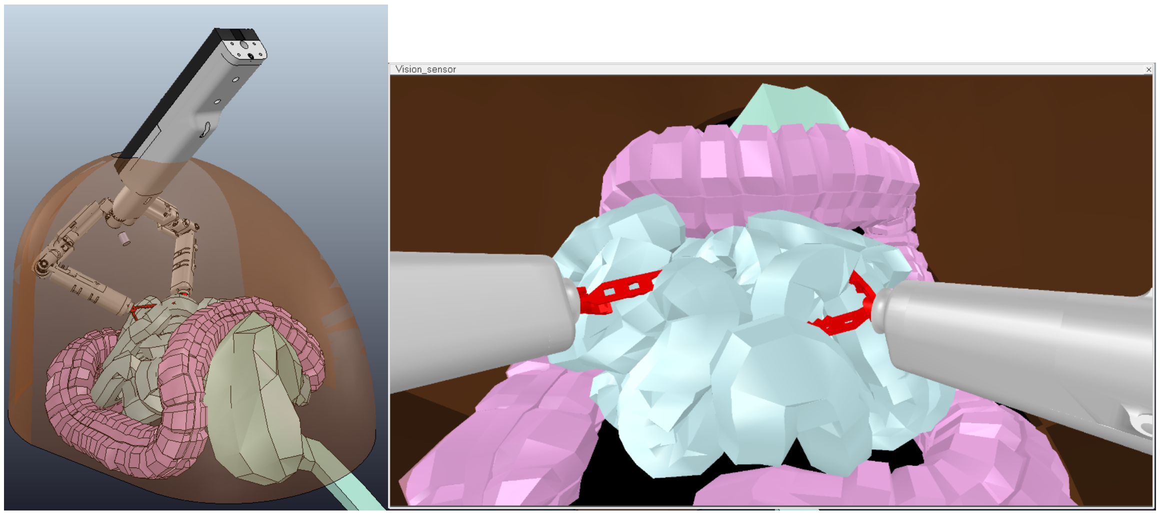


Figure 35: V-REP simulation of the four-DOF robot inside the insuflated abdominal cavity with simulated robot view.

lated vision sensors also allow the testing of proposed vision systems also shown in Figure 35.

Surgeon Console

A mobile surgeon console was constructed from an Ergotron WorkFit-C sit-stand computer cart. The keyboard platform was removed and replaced with a custom controller mount. Brackets were made to attach a rack-mounted computer to the stand. A power strip which powers all of the needed user control devices is attached to the side of the cart, making only a single outlet need to power the entire user interface. The monitor and controllers can be adjusted independently to the operator's preference. A HD-SDI/HDMI video recorder card (Blackmagic DeckLink BDLKMIN-REC) was added to the computer to enable interfacing with the HD endoscope video feed. The console is a convenient mobile platform, and additional user input devices can easily be added through the extra USB ports and the software plug-in interface.

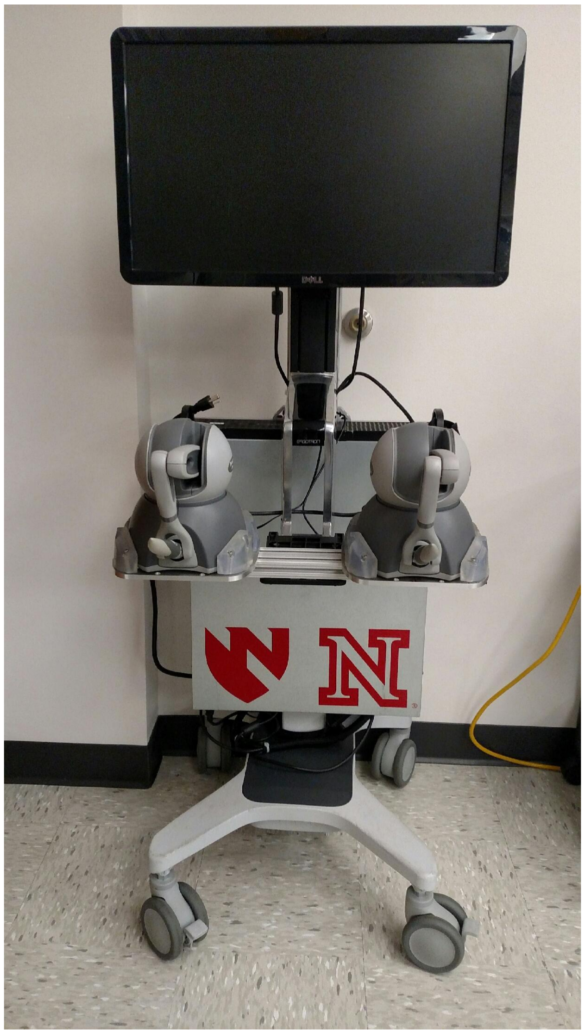


Figure 36: Remote surgeon user interface for the robot control platform.

Experimental Results

CRB-1.0 has been extensively tested in both benchtop and live animal studies in a porcine model. The live animal studies were performed when the platform was not quite mature, and hardware failures occurred in all tests. The hardware failures all occured during the insertion process of the robot through a gel diaphragm into the abdominal cavity. These procedures had to be converted to an open procedure, shown in Figure 37. The robot demonstrated sufficent strength to manipulate organs and also demonstrated mono-polar cautery cabability. The surgeon felt that the robot responded accurately to the given commands and provided smooth and intuitive control of the robotic manipulators.

In order to prevent further hardware failures during the insertion process, the arms

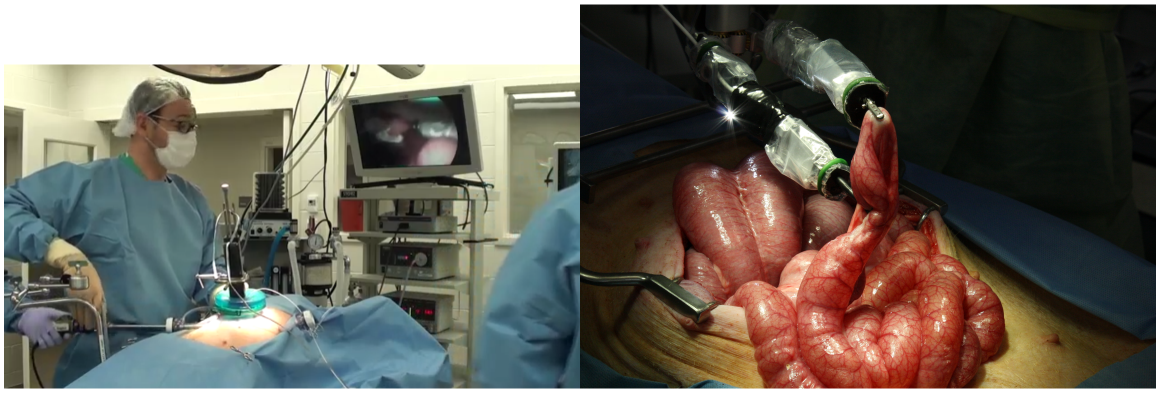


Figure 37: Live animal tests with robot inserted through gel diaphragm (left), and open procedure (right).

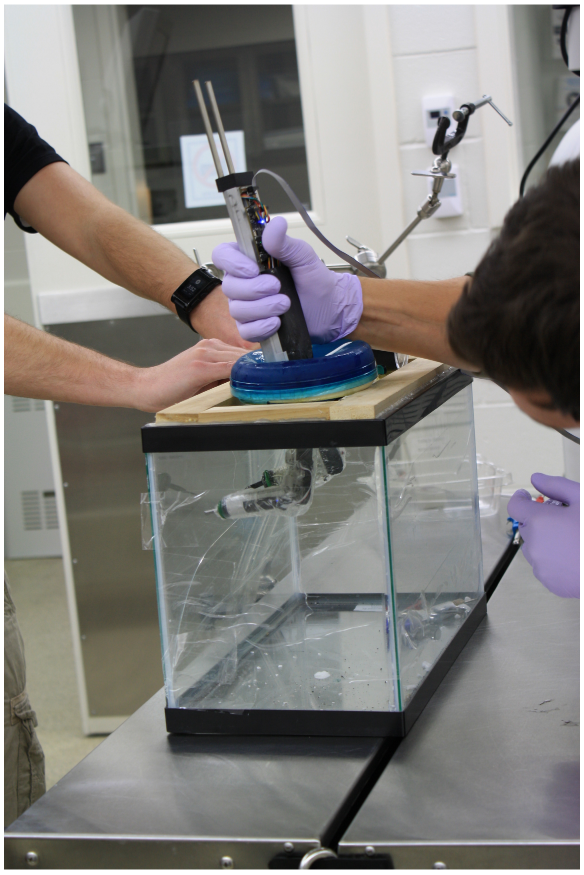


Figure 38: Insertion procedure test using a pressurized chamber.

were modified and the insertion procedure was tested extensively using a pressurized chamber in place of the abdominal cavity, shown in Figure 38. CRB-1.0 was able to successfully perform the insertion procedure in the benchtop test bed 15 consecutive times without any hardware or electrical failures.

Both mono-polar and bipolar cautery were tested in benchtop studies with animal tissue, as shown in Figure 39. Significant shielding was required on the mono-polar cautery power wire to prevent electromagnetic noise from resetting the motor control modules. It was also learned that the end effectors needed to be completely isolated from any ground due the nature of mono-polar cautery; the pad that is placed underneath the tissue to be cauterized provides the high voltage and the tool acts as the

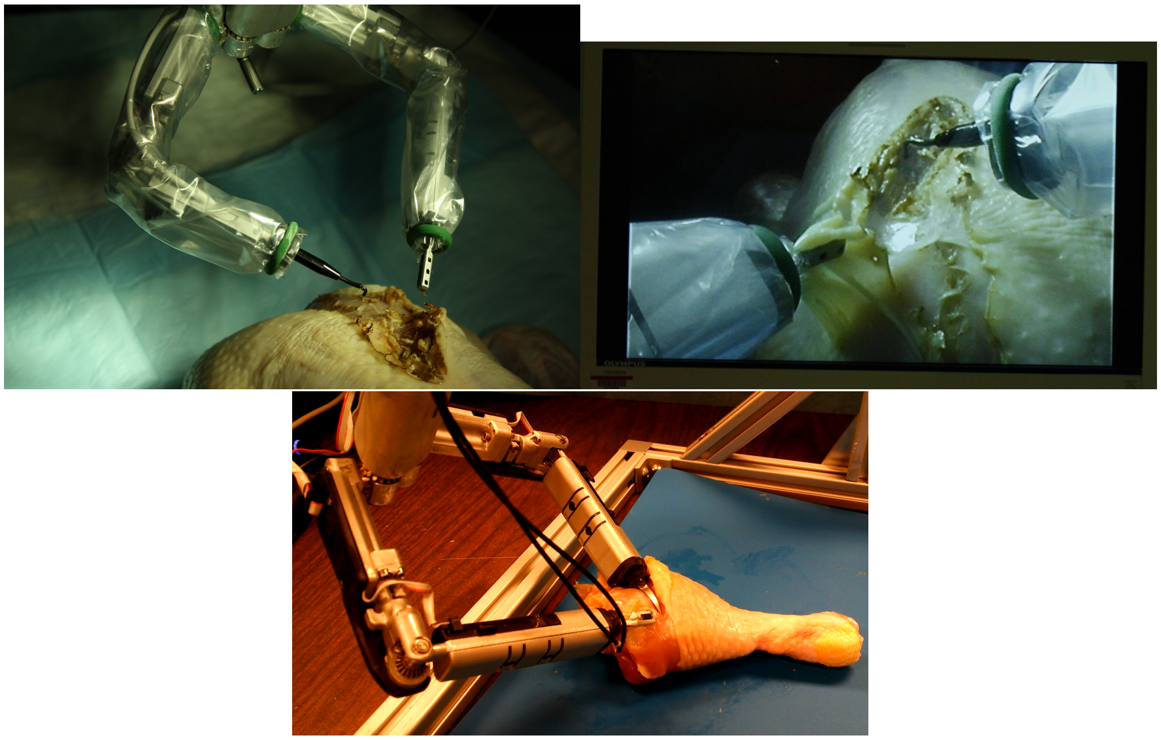


Figure 39: Mono-polar cautery (top) and bi-polar cautery (bottom) benchtop tests with animal tissue.

ground. The RS-485 serial communication protocol was found to provide sufficient protection from the electromagnetic noise generated by the cautery. The bi-polar cautery uses a much lower power than the mono-polar cautery and did not cause any problems with the control electronics.

The LB-1.0 device has not been tested as extensively as the CRB-1.0. This is due to the later development of the robot and the significant time that was required to develop the numerical inverse kinematics algorithm for its control. Successful control of a single arm has been demonstrated, but further benchtop studies are needed to more fully evaluate the devices capabilities.

Conclusions

This thesis presents several advancements in the field of single-incision robotic surgery. Two miniature surgical robots using the same distributed motor control modules have been developed to run on a flexible software stack purposely built to facilitate rapid development. The theoretical analyses of the devices were presented and both meet the proposed requirements to perform surgical tasks.

Both devices met the design requirments set at the beginning of the design chapter throughout almost all of their workspaces. The only areas where the target velocities were not met are on the very edges of the workspace when the manipulators start hitting singularities in the kinematics. The force requirement of 2.2 N was well exceded, with minimum forces of 4.6 N in CRB-1.0 and 4.8 N in LB-1.0. These minimum forces were much greater than the previous device, EB-2.0, which had minimum force values in the range of 0.8 N in some areas of the workspace. The average velocities for both designed devices was greater than the average velocity of EB-2.0.

The CRB-1.0 device has a total workspace volume of 5636.8 cm^3 and and shared workspace volume of 2215.8 cm^3. The limited range of the elbow joint made this workspace smaller than the EB-2.0 workspace, which had a total volume of 7431.2 cm^3 and shared volume of 3838.2 cm^3 [20]. CRB-1.0 was evaluated through multiple benchtop and *in vivo* animal experiments where it demonstrated the dexterity needed to perform simple laparoscopic procedures. While several shortcomings were found in the dexterity and workspace of the robot, the tests proved the effectiveness of the control system, electro-cautery tools, and insertion protocol. LB1.0 was developed to enhance the dexterity and workspace deficiencies of the CRB-1.0 device.

The LB-1.0 device greatly improves on the workspace of CRB-1.0. LB-1.0 has a 200% larger total workspace (11276.5 cm^3) than the CRB-1.0, and a 345% larger shared workspace (7658.9 cm^3). The LB-1.0 workspace surpasses the capabilities of the EB-2.0. The LB-1.0 also has an inserted profile which is 47% of the size of the CRB-1.0 and an integrated camera system. Further benchtop studies will be performed with this device to more completely evaluate its capabilities.

The overall system is compact and low-power, with all robot communications through a single USB port. The control software package can be run on any computer with a Windows operating system. The motor control modules provide joint position and torque control, with additional motor controllers easily added to a system by simply plugging them into the power/data bus.

Bibliography

[1] Samir Agarwal, Mikhail Gincherman, Elisa Birnbaum, James W. Fleshman, and Matthew Mutch. Comparison of long-term follow up of laparoscopic versus open colectomy for transverse colon cancer. *Proceedings (Baylor University. Medical Center)*, 28(3):296–299, July 2015.

[2] A. Alarcon and R. Berguer. A comparison of operating room crowding between open and laparoscopic operations. *Surgical Endoscopy*, 10(9):916–919, September 1996.

[3] Riccardo Autorino, Jihad H. Kaouk, Jens-Uwe Stolzenburg, Inderbir S. Gill, Alex Mottrie, Ash Tewari, and Jeffrey A. Cadeddu. Current status and future directions of robotic single-site surgery: a systematic review. *European Urology*, 63(2):266–280, February 2013.

[4] G. H. Ballantyne. Robotic surgery, telerobotic surgery, telepresence, and telementoring. Review of early clinical results. *Surgical Endoscopy*, 16(10):1389–1402, October 2002.

[5] Richard Budynas and Keith Nisbett. *Shigley's Mechanical Engineering Design.* McGraw-Hill Science/Engineering/Math, 9 edition edition, March 2010.

[6] F. Corcione, C. Esposito, D. Cuccurullo, A. Settembre, N. Miranda, F. Amato, F. Pirozzi, and P. Caiazzo. Advantages and limits of robot-assisted laparoscopic surgery: preliminary experience. *Surgical Endoscopy And Other Interventional Techniques*, 19(1):117–119, January 2005.

[7] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson/Prentice Hall, 2005. Google-Books-ID: MqMeAQAAIAAJ.

[8] Lou Cubrich, Mark A. Reichenbach, Jay D. Carlson, Andrew Pracht, Benjamin Terry, Dmitri Oleynikov, and Shane Farritor. A Four-DOF Laparo-Endoscopic Single Site Platform for Rapidly-Developing Next-Generation Surgical Robotics. *Journal of Medical Robotics Research*, page 1650006, July 2016.

[9] H. de Visser, E. a. M. Heijnsdijk, J. L. Herder, and P. V. Pistecky. Forces and displacements in colon surgery. *Surgical Endoscopy*, 16(10):1426–1430, October 2002.

[10] Amanda Nickles Fader and Pedro F. Escobar. Laparoendoscopic single-site surgery (LESS) in gynecologic oncology: technique and initial report. *Gynecologic Oncology*, 114(2):157–161, August 2009.

[11] Gary S. Guthart and J. K. Salisbury Jr. The IntuitiveTM Telesurgery System: Overview and Application. In *ResearchGate*, volume 1, pages 618–621 vol.1, February 2000.

[12] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White. Raven-II: An Open Platform for Surgical Robotics Research. *IEEE Transactions on Biomedical Engineering*, 60(4):954–959, April 2013.

[13] Jeff A. Hawks. Improved mobile wireless in vivo surgical robots: Modular design, experimental results, and analysis. *ResearchGate*, December 2010.

[14] Jihad H. Kaouk, Georges-Pascal Haber, Riccardo Autorino, Sebastien Crouzet, Adil Ouzzane, Vincent Flamand, and Arnauld Villers. A novel robotic system for single-port urologic surgery: first clinical investigation. *European Urology*, 66(6):1033–1043, December 2014.

[15] Yo Kobayashi, Yuta Sekiguchi, Takehiko Noguchi, Yu Takahashi, Quanquan Liu, Susumu Oguri, Kazutaka Toyoda, Munenori Uemura, Satoshi Ieiri, Morimasa Tomikawa, Takeshi Ohdaira, Makoto Hashizume, and Masaktsu G. Fujie. Development of a robotic system with six-degrees-of-freedom robotic tool manipulators for single-port surgery: Robotic manipulators for single-port surgery. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 11(2):235–246, June 2015.

[16] Yo Kobayashi, Yu Tomono, Yuta Sekiguchi, Hiroki Watanabe, Kazutaka Toyoda, Kozo Konishi, Morimasa Tomikawa, Satoshi Ieiri, Kazuo Tanoue, Makoto Hashizume, and Masaktsu G. Fujie. A surgical robot with vision field control for single port endoscopic surgery. *The international journal of medical robotics + computer assisted surgery: MRCAS*, 6(4):454–464, December 2010.

[17] Amy C. Lehman, Nathan A. Wood, Shane Farritor, Matthew R. Goede, and Dmitry Oleynikov. Dexterous miniature robot for advanced minimally invasive surgery. *Surgical Endoscopy*, 25(1):119–123, January 2011.

[18] Amy Catherine Lehman. Miniature in vivo robots for minimally invasive surgery. *ETD collection for University of Nebraska - Lincoln*, pages 1–160, January 2012.

[19] Mitchell J. H. Lum, Jacob Rosen, Mika N. Sinanan, and Blake Hannaford. Optimization of a spherical mechanism for a minimally invasive surgical robot: theoretical and experimental approaches. *IEEE transactions on bio-medical engineering*, 53(7):1440–1445, July 2006.

[20] Eric Markvicka. Design and Development of a Miniature *In Vivo* Surgical Robot with Distributed Motor Control for Laparoendoscopic Single-Site Surgery. *Mechanical (and Materials) Engineering – Dissertations, Theses, and Student Research*, August 2014.

[21] Ryan McCormick. SIX DEGREE OF FREEDOM MINIATURE *IN VIVO* ROBOT FOR LAPAROENDOSCOPIC SINGLE-SITE SURGERY. *Mechanical (and Materials) Engineering – Dissertations, Theses, and Student Research*, August 2011.

[22] Jack Mondry. Design and Development of a Four Degree of Freedom *In Vivo* Surgical Robot for Laparoendoscopic Single-Site Surgery. *Embargoed Master's Theses*, August 2012.

[23] M. Niccolini, G. Petroni, A. Menciassi, and P. Dario. Real-time control architecture of a novel Single-Port lapaRoscopy bimaNual roboT (SPRINT). In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3395–3400, May 2012.

[24] In Ja Park, Gyu-Seog Choi, Kyoung-Hoon Lim, Byung-Mo Kang, and Soo-Han Jun. Multidimensional analysis of the learning curve for laparoscopic colorectal surgery: lessons from 1,000 cases of laparoscopic colorectal surgery. *Surgical Endoscopy*, 23(4):839–846, April 2009.

[25] Gianluigi Petroni, Marta Niccolini, Arianna Menciassi, Paolo Dario, and Alfred Cuschieri. A novel intracorporeal assembling robotic system for single-port laparoscopic surgery. *Surgical Endoscopy*, 27(2):665–670, February 2013.

[26] Claudio Quaglia, Gianluigi Petroni, Marta Niccolini, Sebastiano Caccavaro, Paolo Dario, and Arianna Menciassi. Design of a Compact Robotic Manipulator for Single-Port Laparoscopy. *Journal of Mechanical Design*, 136(10):105001–105001, July 2014.

[27] Mark Reichenbach. GROSS POSITIONING ARM FOR *IN VIVO* ROBOTIC SURGERY. *Embargoed Master's Theses*, August 2016.

[28] Jacob Rosen, Jeffrey D. Brown, Marco Barreca, Lily Chang, Blake Hannaford, and Mika Sinanan. The Blue DRAGON–a system for monitoring the kinematics and the dynamics of endoscopic tools in minimally invasive surgery for objective laparoscopic skill assessment. *Studies in Health Technology and Informatics*, 85:412–418, 2002.

[29] Richard M. Satava. Surgical robotics: the early chronicles: a personal historical perspective. *Surgical Laparoscopy, Endoscopy & Percutaneous Techniques*, 12(1):6–16, February 2002.

[30] Giuseppe Spinoglio, Alessandra Marano, Fabio Priora, Fabio Melandro, and Giampaolo Formisano. History of Robotic Surgery. In Giuseppe Spinoglio, editor, *Robotic Surgery*, Updates in Surgery, pages 1–12. Springer Milan, 2015. DOI: 10.1007/978-88-470-5714-2_1.

[31] Claudia A. Steiner, Eric B. Bass, Mark A. Talamini, Henry A. Pitt, and Earl P. Steinberg. Surgical Rates and Operative Mortality for Open and Laparoscopic

Cholecystectomy in Maryland. *New England Journal of Medicine*, 330(6):403–408, February 1994.

[32] Julio Teixeira, Kevin McGill, Nina Koshy, James McGinty, and George Todd. Laparoscopic single-site surgery for placement of adjustable gastric band–a series of 22 cases. *Surgery for Obesity and Related Diseases: Official Journal of the American Society for Bariatric Surgery*, 6(1):41–45, February 2010.

[33] Nikolaus Vahrenkamp, Tamim Asfour, Giorgio Metta, Giulio Sandini, and RÃŒdiger Dillmann. Manipulability Analysis. In *ResearchGate*, November 2012.

[34] L. Wang and B. Ravani. Recursive computations of kinematic and dynamic equations for mechanical manipulators. *IEEE Journal on Robotics and Automation*, 1(3):124–131, September 1985.

[35] L.-C. T. Wang and Chih Cheng Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *ResearchGate*, 7(4):489–499, September 1991.

[36] D. C. Wherry, C. G. Rob, M. R. Marohn, and N. M. Rich. An external audit of laparoscopic cholecystectomy performed in medical treatment facilities of the department of Defense. *Annals of Surgery*, 220(5):626–634, November 1994.

[37] T. D. Wortman, R. L. McCormick, E. J. Markvicka, T. P. Frederick, S. M. Farritor, and D. Oleynikov. Multi-Functional Surgical Robot for Laparo-Endoscopic Single-Site Colectomies. pages 653–658, January 2011.

[38] Tyler D. Wortman, Kyle W. Strabala, Amy C. Lehman, Shane M. Farritor, and Dmitry Oleynikov. Laparoendoscopic single-site surgery using a multi-functional

miniature in vivo robot. *The international journal of medical robotics + computer assisted surgery: MRCAS*, 7(1):17–21, March 2011.

[39] Yongzhi Yang, Feng Wang, Peng Zhang, Chenzhang Shi, Yang Zou, Huanlong Qin, and Yanlei Ma. Robot-assisted versus conventional laparoscopic surgery for colorectal disease, focusing on rectal cancer: a meta-analysis. *Annals of Surgical Oncology*, 19(12):3727–3736, November 2012.

[40] Tsuneo Yoshikawa. Manipulability of Robotic Mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, June 1985.

Appendix

Kinematic Analysis

The homogeneous transformation matrix as derived from the definition of the Denavit-Hartenberg parameters is

$$T_i^{i-1} = T_{z_{i-1}}(d_{i-1}) R_{z_{i-1}}(\theta_{i-1}) T_x(a_i) R_x(\alpha_i)$$

where

$$T_{z_{i-1}}(d_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_{z_{i-1}}(\theta_{i-1}) = \begin{bmatrix} \cos\theta_{i-1} & -\sin\theta_{i-1} & 0 & 0 \\ \sin\theta_{i-1} & \cos\theta_{i-1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_x(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Multiplying these matrices in order yields the homogeneous transformation matrix through linear algebra:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_{i-1} & -\cos\alpha_i \sin\theta_{i-1} & \sin\alpha_i \sin\theta_{i-1} & a_i \cos\theta_{i-1} \\ \sin\theta_{i-1} & \cos\alpha_i \cos\theta_{i-1} & -\sin\alpha_i \cos\theta_{i-1} & a_i \sin\theta_{i-1} \\ 0 & \sin\alpha_i & \cos\alpha_i & d_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CubReich-Bot 1.0

The transformation matrices for each kinematic from with respect to the previous frame are found by plugging in the DH parameters:

i	α_i	a_i	d_{i-1}	θ_{i-1}
1	0	0	0	0
2	90°	0	0	ϕ_1
3	−90°	$L_1 = 68.5$ mm	0	ϕ_2
4	90°	0	0	$\phi_3 + 90^{\text{o}}$
5	0	0	$L_2 = 96.4$ mm	ϕ_4

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_2^1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_3^2 = \begin{bmatrix} c_2 & 0 & -s_2 & L_1 c_2 \\ s_2 & 0 & c_2 & L_1 s_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_4^3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_5^4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these in order yields the transformation matrix of the end effector with respect to the base frame:

$$T_5^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4$$

$$T_5^0 = \begin{bmatrix} -c_4(s_1 s_3 - c_1 c_2 c_3) - c_1 s_2 s_4 & s_4(s_1 s_3 - c_1 c_2 c_3) - c_1 c_4 s_2 & c_3 s_1 + c_1 c_2 s_3 & L_1 c_1 c_2 + L_2\left(s_1 c_3 - c_1 c_2 s_3\right) \\ c_4(c_1 s_3 + c_2 c_3 s_1) - s_1 s_2 s_4 & -s_4(c_1 s_3 + c_2 c_3 s_1) - c_4 s_1 s_2 & c_2 s_1 s_3 - c_1 c_3 & L_1 s_1 c_2 - L_2\left(c_1 c_3 + s_1 c_2 s_3\right) \\ c_2 s_4 + c_4 c_3 s_2 & c_2 c_4 - c_3 s_2 s_4 & s_2 s_3 & L_1 s_2 + L_2 s_2 s_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Lou-Bot 1.0

The transformation matrices for each kinematic from with respect to the previous frame are found by plugging in the DH parameters:

i	α_i	a_i	d_{i-1}	θ_{i-1}
1	0	0	0	0
2	$90°$	0	0	θ_1
3	$90°$	0	0	$\theta_2 + 90°$
4	$90°$	0	$L_1 = 87.6$ mm	$\theta_3 + 90°$
5	$90°$	0	0	$\theta_4 + 180°$
6	$90°$	0	$L_2 = 86.6$ mm	$\theta_5 + 180°$

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_2^1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_3^2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_4^3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_5^4 = \begin{bmatrix} -c_4 & 0 & -s_4 & 0 \\ -s_4 & 0 & c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_6^5 = \begin{bmatrix} -c_5 & 0 & -s_5 & 0 \\ -s_5 & 0 & c_5 & 0 \\ 0 & 1 & 0 & L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these in order yields the transformation matrix of the end effector with respect to the base frame:

$$T_6^0 = \begin{bmatrix} R_{00} & R_{01} & R_{02} & x \\ R_{10} & R_{11} & R_{12} & y \\ R_{20} & R_{21} & R_{22} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$R_{00} = c_5(c_4(s_1s_3 + c_1c_2c_3) + c_1s_4s_2) + s_5(c_3s_1 - c_1c_2s_3) \qquad R_{01} = c_1c_4s_2 - s_4(s_1s_3 + c_1c_2c_3)$$

$$R_{02} = s_5(c_4(s_1s_3 + c_1c_2c_3) + c_1s_4s_2) - c_5(c_3s_1 - c_1c_2s_3) \qquad R_{10} = -c_5(c_4(c_1s_3 - c_2c_3s_1) - s_1s_4s_2) - s_5(c_1c_3 + c_2s_1s_3)$$

$$R_{11} = s_4(c_1s_3 - c_2c_3s_1) + c_4s_1s_2 \qquad R_{12} = c_5(c_1c_3 + c_2s_1s_3) - s_5(c_4(c_1s_3 - c_2c_3s_1) - s_1s_4s_2)$$

$$R_{20} = -c_5(c_2s_4 - c_4c_3s_2) - s_5s_2s_3 \qquad R_{21} = -c_4c_2 - c_3s_4s_2$$

$$R_{22} = c_5s_2s_3 - s_5(c_2s_4 - c_4c_3s_2)$$

and

$$x = L_1c_1s_2 - L_2(s_4(s_1s_3 + c_1c_2c_3) - c_1c_4s_2)$$

$$y = L_2(s_4(c_1s_3 - c_2c_3s_1) + c_4s_1s_2) + L_1s_1s_2$$

$$z = -L_2(c_4c_2 + c_3s_4s_2) - L_1c_2$$

Supplemental Code

CubReich-Bot 1.0 Capabilities MATLAB Function

```
function [X,Y,w,Vx,Vy,Vz,Fx,Fy,Fz] = MEATBOTcapabilities(savePics)
% define step size in degrees
step = 1;
% define length of upper arm (L1) and forearm (L2) links
L1 = 68.58;
L2 = 96.393;
% set theta2 to zero to keep arm in X-Y plane
t2 = 0;
% preallocate memory for data arrays
xSize = (135/step)+1;
ySize = (105/step)+1;
X = zeros(xSize,ySize);
Y = zeros(xSize,ySize);
w = zeros(xSize,ySize);
Vx = zeros(xSize,ySize);
Vy = zeros(xSize,ySize);
Vz = zeros(xSize,ySize);
Fx = zeros(xSize,ySize);
Fy = zeros(xSize,ySize);
Fz = zeros(xSize,ySize);
% set maximum velocities for each joint in rad/s
jointVel = zeros(3);
jointVel(1) = 4.4831;
jointVel(2) = 4.4831;
jointVel(3) = 2.3845;
% set maximum joint torques for each joint in mN-m
jointTorque = zeros(3);
jointTorque(1) = 3115.57;
jointTorque(2) = 3115.57;
jointTorque(3) = 448.11;
% set maximum starting force and force step for the numerical force solver
forceMax = 30;
forceStep = 0.1;

m = 0;
for t1 = -90:step:45
    m = m + 1;
    n = 0;
    for t3 = 0:step:105
        n = n + 1;

        % calculate forward kinematics
        X(m,n) = L2*(cosd(90 + t3)*sind(t1) + cosd(t1)*cosd(t2)*sind(90 + t3)) + L1*cosd(t1)*cosd(t2);
        Y(m,n) = L1*cosd(t2)*sind(t1) - L2*(cosd(t1)*cosd(90 + t3) - cosd(t2)*sind(t1)*sind(90 + t3));
        % calcuate Jacobian matrix
        J = [ L2*(cosd(t1)*cosd(90 + t3) - cosd(t2)*sind(t1)*sind(90 + t3)) - L1*cosd(t2)*sind(t1), - L1*cosd(t1)*sind(t2) - L2*cosd(t1)*sind(t2)*sind(90 + t3), -L2*(sind(t1)*sind(90 + t3) - cosd(t1)*cosd(t2)*cosd(90 + t3)); L2*(cosd(90 + t3)*sind(t1) + cosd(t1)*cosd(t2)*sind(90 + t3)) + L1*cosd(t1)*cosd(t2), - L1*sind(t1)*sind(t2) - L2*sind(t1)*sind(t2)*sind(90 + t3),  L2*(cosd(t1)*sind(90 + t3) + cosd(t2)*cosd(90 + t3)*sind(t1)); 0, L1*cosd(t2) + L2*cosd(t2)*sind(90 + t3), L2*cosd(90 + t3)*sind(t2)];
        % calcuate Jacobian transpose matrix
        Jt = [ L2*(cosd(t1)*cosd(90 + t3) - cosd(t2)*sind(t1)*sind(90 + t3)) - L1*cosd(t2)*sind(t1), L2*(cosd(90 + t3)*sind(t1) + cosd(t1)*cosd(t2)*sind(90 + t3)) + L1*cosd(t1)*cosd(t2), 0; - L1*cosd(t1)*sind(t2) - L2*cosd(t1)*sind(t2)*sind(90 + t3), - L1*sind(t1)*sind(t2) - L2*sind(t1)*sind(t2)*sind(90 + t3), L1*cosd(t2) + L2*cosd(t2)*sind(90 + t3); -L2*(sind(t1)*sind(90 + t3) - cosd(t1)*cosd(t2)*cosd(90 + t3)), L2*(cosd(t1)*sind(90 + t3) + cosd(t2)*cosd(90 + t3)*sind(t1)), L2*cosd(90 + t3)*sind(t2)];
        % calculate manipulability index
        w(m,n) = sqrt(abs(det(J*Jt)));
        % calculate velocity
        for i=-1:1
            for j=-1:1
                for k=-1:1
                    V = J*[i*jointVel(1);j*jointVel(2);k*jointVel(3)];
                    if abs(V(1)) > Vx(m,n)
                        Vx(m,n) = abs(V(1));
                    end
                    if abs(V(2)) > Vy(m,n)
                        Vy(m,n) = abs(V(2));
                    end
                    if abs(V(3)) > Vz(m,n)
                        Vz(m,n) = abs(V(3));
                    end
                end
            end
        end
        % calculate force in x direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[testForce;0;0];
            for i = 1:3
               if abs(T(i)) > jointTorque(i)
                   strong = false;
               end
            end
            if strong
                weak = false;
                Fx(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end
        % calculate force in y direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[0;testForce;0];
            for i = 1:3
```

```
               if abs(T(i)) > jointTorque(i)
                   strong = false;
               end
            end
            if strong
                weak = false;
                Fy(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end
        % calculate force in z direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[0;0;testForce];
            for i = 1:3
               if abs(T(i)) > jointTorque(i)
                   strong = false;
               end
            end
            if strong
                weak = false;
                Fz(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end

    end
end
% normalize manipulability index to maximum
w = w/max(w(:));
% plot figure for manipulability index
figure(1)
surface(X,Y,zeros(size(X)),w,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Manipulability␣Index')
if savePics
    saveas(gcf,'MB2-manipulability.tif')
end
% plot figure for Vx
figure(2)
surface(X,Y,zeros(size(X)),Vx,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Vx␣[mm/s]')
if savePics
    saveas(gcf,'MB2-Vx.tif')
end
% plot figure for Vy
figure(3)
```

```
surface(X,Y,zeros(size(X)),Vy,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Vy␣[mm/s]')
if savePics
    saveas(gcf,'MB2-Vy.tif')
end
% plot figure for Vz
figure(4)
surface(X,Y,zeros(size(X)),Vz,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Vz␣[mm/s]')
if savePics
    saveas(gcf,'MB2-Vz.tif')
end
% plot figure for Fx
figure(5)
surface(X,Y,zeros(size(X)),Fx,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Fx␣[Newtons]')
if savePics
    saveas(gcf,'MB2-Fx.tif')
end
% plot figure for Fy
figure(6)
surface(X,Y,zeros(size(X)),Fy,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('MB2␣Fy␣[Newtons]')
if savePics
    saveas(gcf,'MB2-Fy.tif')
end
% plot figure for Fz
figure(7)
surface(X,Y,zeros(size(X)),Fz,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
```

```
title('MB2␣Fz␣[Newtons]')
if savePics
    saveas(gcf,'MB2-Fz.tif')
end
```

Lou-Bot 1.0 Capabilities MATLAB Function

```
function [X,Y,w,Vx,Vy,Vz,Fx,Fy,Fz] = LOUBOTcapabilities(savePics)
% define step size in degrees
step = 1;
% define length of upper arm (L1) and forearm (L2) links
L1 = 87.57;
L2 = 86.59;
% set theta2 and theta3 to zero to keep arm in X-Y plane
t2 = 0;
t3 = 0;
% preallocate memory for data arrays
xSize = (135/step)+1;
ySize = (150/step)+1;
X = zeros(xSize,ySize);
Y = zeros(xSize,ySize);
w = zeros(xSize,ySize);
Vx = zeros(xSize,ySize);
Vy = zeros(xSize,ySize);
Vz = zeros(xSize,ySize);
Fx = zeros(xSize,ySize);
Fy = zeros(xSize,ySize);
Fz = zeros(xSize,ySize);
% set maximum velocities for each joint in rad/s
jointVel = zeros(4);
jointVel(1) = 5.2308;
jointVel(2) = 5.2308;
jointVel(3) = 5.2308;
jointVel(4) = 2.5426;
% set maximum joint torques for each joint in mN-m
jointTorque = zeros(4);
jointTorque(1) = 2670.49;
jointTorque(2) = 2670.49;
jointTorque(3) = 2410.12;
jointTorque(4) = 420.1;
% set maximum starting force and force step for the numerical force solver
forceMax = 30;
forceStep = 0.1;

m = 0;
for t1 = -90:step:45
    m = m + 1;
    n = 0;
    for t4 = 0:step:150
        n = n + 1;

        % calculate forward kinematics
        X(m,n) = L1*cosd(t1)*sind(90 + t2) - L2*(sind(t4)*(sind(t1)*sind(90 + t3) +
    cosd(t1)*cosd(90 + t2)*cosd(90 + t3)) - cosd(t1)*cosd(t4)*sind(90 + t2));
        Y(m,n) = L2*(sind(t4)*(cosd(t1)*sind(90 + t3) - cosd(90 + t2)*cosd(90 + t3)*
    sind(t1)) + cosd(t4)*sind(t1)*sind(90 + t2)) + L1*sind(t1)*sind(90 + t2);
        % calcuate Jacobian matrix
        J = [ - L2*(sind(t4)*(cosd(t1)*sind(90 + t3) - cosd(90 + t2)*cosd(90 + t3)*
    sind(t1)) + cosd(t4)*sind(t1)*sind(90 + t2)) - L1*sind(t1)*sind(90 + t2), L2*(
    cosd(t1)*cosd(t4)*cosd(90 + t2) + cosd(t1)*cosd(90 + t3)*sind(t4)*sind(90 + t2))
    + L1*cosd(t1)*cosd(90 + t2), -L2*sind(t4)*(cosd(90 + t3)*sind(t1) - cosd(t1)*cosd
    (90 + t2)*sind(90 + t3)), -L2*(cosd(t4)*(sind(t1)*sind(90 + t3) + cosd(t1)*cosd
    (90 + t2)*cosd(90 + t3)) + cosd(t1)*sind(t4)*sind(90 + t2)); L1*cosd(t1)*sind(90
    + t2) - L2*(sind(t4)*(sind(t1)*sind(90 + t3) + cosd(t1)*cosd(90 + t2)*cosd(90 +
    t3)) - cosd(t1)*cosd(t4)*sind(90 + t2)), L2*(cosd(t4)*cosd(90 + t2)*sind(t1) +
    cosd(90 + t3)*sind(t1)*sind(t4)*sind(90 + t2)) + L1*cosd(90 + t2)*sind(t1),  L2*
    sind(t4)*(cosd(t1)*cosd(90 + t3) + cosd(90 + t2)*sind(t1)*sind(90 + t3)),  L2*(
    cosd(t4)*(cosd(t1)*sind(90 + t3) - cosd(90 + t2)*cosd(90 + t3)*sind(t1)) - sind(
    t1)*sind(t4)*sind(90 + t2)); 0, L2*(cosd(t4)*sind(90 + t2) - cosd(90 + t2)*cosd
    (90 + t3)*sind(t4)) + L1*sind(90 + t2), L2*sind(t4)*sind(90 + t2)*sind(90 + t3),
    L2*(cosd(90 + t2)*sind(t4) - cosd(t4)*cosd(90 + t3)*sind(90 + t2))];
        % calcuate Jacobian transpose matrix
        Jt = [ - L2*(sind(t4)*(cosd(t1)*sind(90 + t3) - cosd(90 + t2)*cosd(90 + t3)*
    sind(t1)) + cosd(t4)*sind(t1)*sind(90 + t2)) - L1*sind(t1)*sind(90 + t2), L1*cosd
    (t1)*sind(90 + t2) - L2*(sind(t4)*(sind(t1)*sind(90 + t3) + cosd(t1)*cosd(90 + t2
    )*cosd(90 + t3)) - cosd(t1)*cosd(t4)*sind(90 + t2)), 0; L2*(cosd(t1)*cosd(t4)*
    cosd(90 + t2) + cosd(t1)*cosd(90 + t3)*sind(t4)*sind(90 + t2)) + L1*cosd(t1)*cosd
    (90 + t2), L2*(cosd(t4)*cosd(90 + t2)*sind(t1) + cosd(90 + t3)*sind(t1)*sind(t4)*
    sind(90 + t2)) + L1*cosd(90 + t2)*sind(t1), L2*(cosd(t4)*sind(90 + t2) - cosd(90
    + t2)*cosd(90 + t3)*sind(t4)) + L1*sind(90 + t2); -L2*sind(t4)*(cosd(90 + t3)*
    sind(t1) - cosd(t1)*cosd(90 + t2)*sind(90 + t3)), L2*sind(t4)*(cosd(t1)*cosd(90 +
     t3) + cosd(90 + t2)*sind(t1)*sind(90 + t3)), L2*sind(t4)*sind(90 + t2)*sind(90 +
     t3); -L2*(cosd(t4)*(sind(t1)*sind(90 + t3) + cosd(t1)*cosd(90 + t2)*cosd(90 + t3
    )) + cosd(t1)*sind(t4)*sind(90 + t2)), L2*(cosd(t4)*(cosd(t1)*sind(90 + t3) -
    cosd(90 + t2)*cosd(90 + t3)*sind(t1)) - sind(t1)*sind(t4)*sind(90 + t2)), L2*(
    cosd(90 + t2)*sind(t4) - cosd(t4)*cosd(90 + t3)*sind(90 + t2))];
        % calculate manipulability index
        w(m,n) = sqrt(abs(det(J*Jt)));
        % calculate maximum velocity
        for i=-1:1
            for j=-1:1
                for k=-1:1
                    for l=-1:1
                        V = J*[i*jointVel(1);j*jointVel(2);k*jointVel(3);l*jointVel
    (4)];
                        if abs(V(1)) > Vx(m,n)
                            Vx(m,n) = abs(V(1));
                        end
                        if abs(V(2)) > Vy(m,n)
                            Vy(m,n) = abs(V(2));
                        end
                        if abs(V(3)) > Vz(m,n)
                            Vz(m,n) = abs(V(3));
                        end
                    end
                end
            end
        end
        V = J*[jointVel(1);jointVel(2);jointVel(3);jointVel(4)];
        Vx(m,n) = abs(V(1));
        Vy(m,n) = abs(V(2));
        Vz(m,n) = abs(V(3));
        % calculate force in x direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[testForce;0;0];
            for i = 1:4
               if abs(T(i)) > jointTorque(i)
                   strong = false;
```

```
                end
            end
            if strong
                weak = false;
                Fx(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end
        % calculate force in y direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[0;testForce;0];
            for i = 1:4
                if abs(T(i)) > jointTorque(i)
                    strong = false;
                end
            end
            if strong
                weak = false;
                Fy(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end
        % calculate force in z direction
        testForce = forceMax;
        weak = true;
        while weak
            strong = true;
            T = Jt*[0;0;testForce];
            for i = 1:4
                if abs(T(i)) > jointTorque(i)
                    strong = false;
                end
            end
            if strong
                weak = false;
                Fz(m,n) = testForce;
            else
                testForce = testForce - forceStep;
            end
        end

    end
end
% normalize manipulability index to maximum
w = w/max(w(:));
% plot figure for manipulability index
figure(1)
surface(X,Y,zeros(size(X)),w,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Manipulability␣Index')
if savePics
    saveas(gcf,'LB1-manipulability.png')
end
```

```
% plot figure for Vx
figure(2)
surface(X,Y,zeros(size(X)),Vx,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Vx␣[mm/s]')
if savePics
    saveas(gcf,'LB1-Vx.png')
end
% plot figure for Vy
figure(3)
surface(X,Y,zeros(size(X)),Vy,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Vy␣[mm/s]')
if savePics
    saveas(gcf,'LB1-Vy.png')
end
% plot figure for Vz
figure(4)
surface(X,Y,zeros(size(X)),Vz,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Vz␣[mm/s]')
if savePics
    saveas(gcf,'LB1-Vz.png')
end
% plot figure for Fx
figure(5)
surface(X,Y,zeros(size(X)),Fx,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Fx␣[Newtons]')
if savePics
    saveas(gcf,'LB1-Fx.png')
end
% plot figure for Fy
figure(6)
surface(X,Y,zeros(size(X)),Fy,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
```

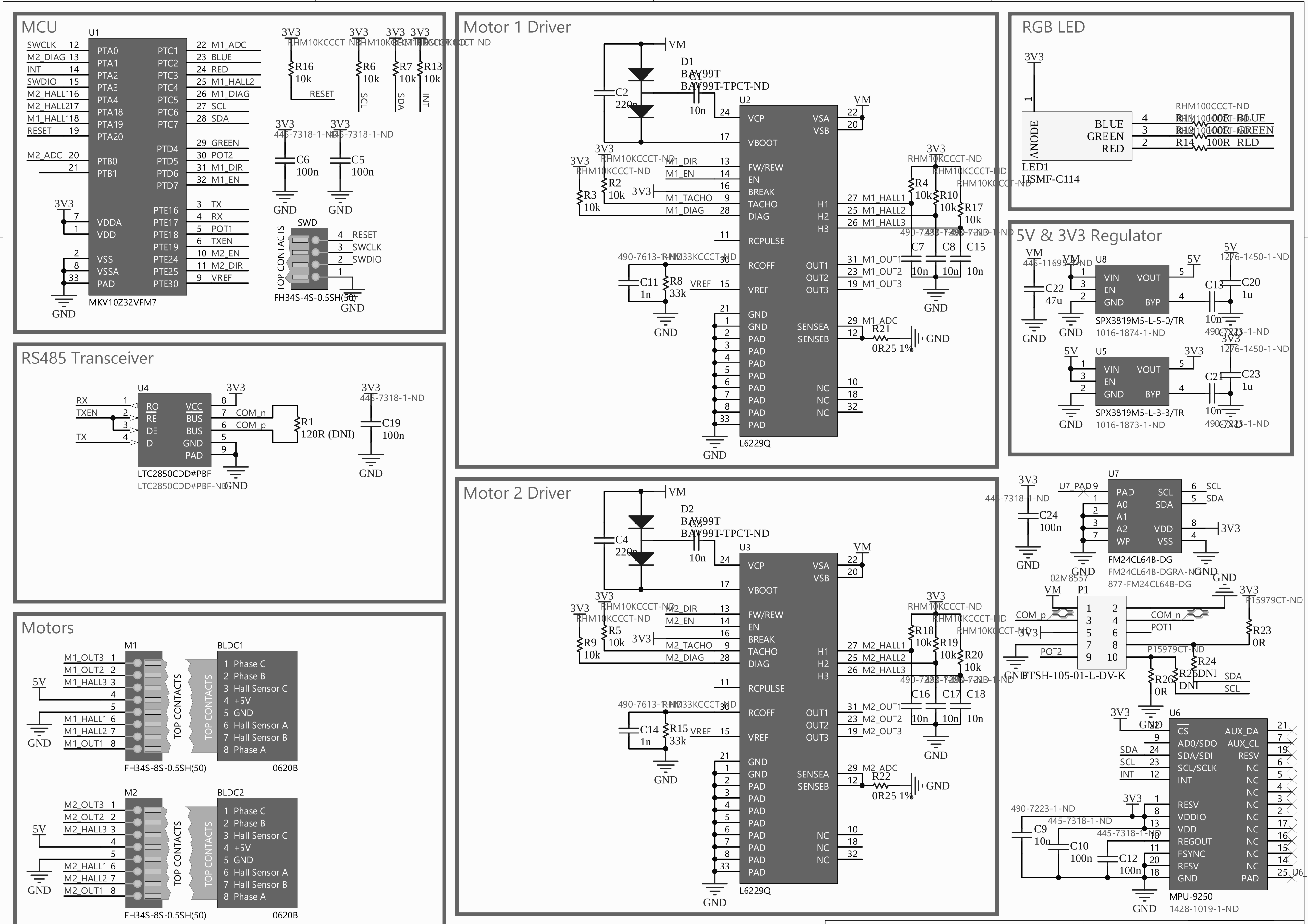
```
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Fy␣[Newtons]')
if savePics
    saveas(gcf,'LB1-Fy.png')
end
% plot figure for Fz
figure(7)
surface(X,Y,zeros(size(X)),Fz,'LineStyle','none');
view(-90,90)
colorbar;
axis equal tight
grid on
xlabel('X␣[mm]')
ylabel('Y␣[mm]')
zlabel('Z␣[mm]')
title('LB1␣Fz␣[Newtons]')
if savePics
    saveas(gcf,'LB1-Fz.png')
end
```

Inverse Kinematics Solver

```
 using System;
using System.Diagnostics;
using System.Windows;
using System.Windows.Media.Media3D;

namespace Kinematics
{
    public class IKSolver : Kinematic
    {
        private double[] radAngle;      // array of joint angles in radians
        private double[] thetaOffset;   // array of theta offsets from DH parameters
        private Vector3D[,] frame;      // array of joint frame vectors
        private Vector3D Pd;            // desired position vector
        private Vector3D Ph;            // position of end effector
        private Vector3D[] Rd;          // desired orientation of end effector
        private Vector3D[] Rh;          // orientation of end effector
        private Vector3D[] Pih;         // array of relative position of end effector
     with respect to each frame
        private double Eo;              // orientation error
        private double Ec;              // current error
        private double Ep;              // previous error
        private bool Initialized = false;
        private double maxForce = 4;

        const int IK_MAX_TRIES = 15000;

        /// <summary>
        /// index = 0         1          2          3
        /// alpha(i-1)      a(i-i)     d(i)       theta(i)
        /// </summary>
        public double[,] DHparameters { get; set; }

        /// <summary>
        /// This returns the number of links in the manipulator
        /// </summary>
        public int N { get; set; }

        /// <summary>
        /// This returns the weights (0 or 1) of each end effector orientations
        /// </summary>
        public bool[] Sigma { get; set; }

        /// <summary>
        /// This returns the minimum and maximum angles for each joint in degrees
        /// </summary>
        public Point[] MinMax { get; set; }

        /// <summary>
        /// This returns the joint coupling of the manipulator
        /// </summary>
        public CouplingType Coupling { get; set; }

        /// <summary>
        /// This returns whether or not to output workspace forces
        /// </summary>
        public bool OutputWorkspace { get; set; }

        /// <summary>
        /// This returns whether or not to invert workspace forces
        /// </summary>
        public bool[] InvertForces { get; set; }

        /// <summary>
        /// This returns the names of the angle outputs
        /// </summary>
        public string[] OutputStrings { get; set; }

        /// <summary>
        /// Stop criteria for CCD
        /// </summary>
        public double IK_POS_THRESH { get; set; }

        /// <summary>
        /// Criteria to begin BGFS optimizer
        /// </summary>
        public double BETA { get; set; }

    protected override double[] getJointAngles(Point3D Position, Point3D Orientation)
        {
            if (!Initialized)
            {
                radAngle = new double[N + 1];
                radAngle.Initialize();

                thetaOffset = new double[N + 1];
                thetaOffset[0] = 0;
                for (int i = 1; i <= N; i++)
                {
                    thetaOffset[i] = DHparameters[i - 1, 3] * Math.PI / 180;
                }

                Initialized = true;
            }
            // create desired position vector
            Pd = new Vector3D(Position.X, Position.Y, Position.Z);
            // create desired orientation vector from roll, pitch, yaw
            Rd = new Vector3D[3];
            // convert to radians
            Orientation.X = Orientation.X * Math.PI / 180;
            Orientation.Y = Orientation.Y * Math.PI / 180;
            Orientation.Z = Orientation.Z * Math.PI / 180;
            // convert roll/pitch/yaw to rotation matrix

            Rd[0].X = Math.Cos(Orientation.Y) * Math.Cos(Orientation.Z);
            Rd[0].Y = Math.Sin(Orientation.Z) * Math.Cos(Orientation.Y);
            Rd[0].Z = -Math.Sin(Orientation.Y);
            Rd[1].X = Math.Cos(Orientation.Z) * Math.Sin(Orientation.Y) * Math.Sin(
    Orientation.X) - Math.Sin(Orientation.Z) * Math.Cos(Orientation.X);
            Rd[1].Y = Math.Sin(Orientation.X) * Math.Sin(Orientation.Y) * Math.Sin(
    Orientation.Z) + Math.Cos(Orientation.X) * Math.Cos(Orientation.Z);
            Rd[1].Z = Math.Cos(Orientation.Y) * Math.Sin(Orientation.X);
            Rd[2].X = Math.Cos(Orientation.X) * Math.Sin(Orientation.Y) * Math.Cos(
    Orientation.Z) + Math.Sin(Orientation.X) * Math.Sin(Orientation.Z);
            Rd[2].Y = Math.Sin(Orientation.Z) * Math.Sin(Orientation.Y) * Math.Cos(
    Orientation.X) - Math.Cos(Orientation.Z) * Math.Sin(Orientation.X);
            Rd[2].Z = Math.Cos(Orientation.Y) * Math.Cos(Orientation.X);

            Rh = new Vector3D[3];
            // declare 3D array for each joint frame axis (xi, yi, zi, Pi)
            frame = new Vector3D[N + 1, 4];
            frame.Initialize();
            // set base frame
            frame[0, 0].X = 1;
            frame[0, 1].Y = 1;
            frame[0, 2].Z = 1;

            Vector3D[] Pstar = new Vector3D[N];
            Pstar.Initialize();

            int link = N;
            int tries = 0;
            bool solved = false;
            // begin Cyclic Coordinate Descent loop
            do
            {
                // initialize frame positions
                for (int i = 0; i < N + 1; i++)
                {
                    frame[i, 3].X = 0;
                    frame[i, 3].Y = 0;
                    frame[i, 3].Z = 0;
                }
                // forward recurrsion formulas for frame position and orientation
                for (int i = 1; i <= N; i++)
                {
                    // x(i) orientation vector
                    frame[i, 0] = Vector3D.Add((Vector3D.Multiply((Math.Cos(radAngle[
    i - 1] + thetaOffset[i])), frame[(i - 1), 0])), (Vector3D.Multiply((Math.Sin(
    radAngle[i - 1] + thetaOffset[i])), frame[(i - 1), 1])));
                    // z(i) orientation vector
                    frame[i, 2] = Vector3D.Add((Vector3D.Multiply(Math.Cos(
    DHparameters[i - 1, 0] * Math.PI / 180), frame[(i - 1), 2])), Vector3D.Multiply(
    Math.Sin(DHparameters[(i - 1), 0] * Math.PI / 180), Vector3D.CrossProduct(frame[i
    , 0], frame[(i - 1), 2])));
                    // y(i) orientation vector
                    frame[i, 1] = Vector3D.CrossProduct(frame[i, 2], frame[i, 0]);
                }
                // P* --> relative positions of next frame wrt present frame
                for (int i = 0; i < N; i++)
                {
                    Pstar[i] = Vector3D.Add(Vector3D.Multiply(DHparameters[i, 2],
    frame[i, 2]), Vector3D.Multiply(DHparameters[i, 1], frame[i + 1, 0]));
                }
```

```
                //P(i) --> frame positions wrt base frame
                for (int i = 1; i < N + 1; i++)
                {
                    frame[i, 3] = frame[i - 1, 3] + Pstar[i - 1];
                }
                // set position of end effector
                Ph = frame[N, 3];

                // compute relative positions
                Pih = new Vector3D[N + 1];
                for (int i = N - 1; i >= 0; i--)
                {
                    Pih[i] = Vector3D.Subtract(Ph, frame[i, 3]);
                }
                // set end effector orientation
                Rh[0] = frame[N, 0];
                Rh[1] = frame[N, 1];
                Rh[2] = frame[N, 2];
                // calculate orientation error
                Eo = 0;
                for (int i = 0; i < 3; i++)
                {
                    if (Sigma[i])
                    {
                        Eo += Math.Pow((Vector3D.DotProduct(Rd[i], Rh[i]) - 1), 2);
                    }
                }
                // calculate current position error
                Ec = Eo + Vector3D.DotProduct(Vector3D.Subtract(Pd, Ph), Vector3D.Subtract(Pd, Ph));

                if (solved)
                    break;

                if ((Ec > IK_POS_THRESH) && (Ec < BETA) && (Ec > Math.Pow(Ep, 2))) // begin BFGS optimization
                {
                    double epsg = 0.0000000001;
                    double epsf = 0;
                    double epsx = 0;
                    int maxits = 0;              // maximum number of iterations, for unlimited = 0
                    double stpmax = 0;
                    double[] scale = new double[N];
                    double[] optiAngle = new double[N];
                    for (int i = 0; i < N; i++)
                    {
                        scale[i] = 2;
                        optiAngle[i] = radAngle[i + 1];
                    }
                    alglib.minlbfgsstate state;
                    alglib.minlbfgsreport rep;

                    alglib.minlbfgscreate(4, optiAngle, out state);          // create optimizer with current joint angles for initial values
                    alglib.minlbfgssetcond(state, epsg, epsf, epsx, maxits); // set optimizer options
                    alglib.minlbfgssetstpmax(state, stpmax);
                    alglib.minlbfgsoptimize(state, function1_grad, null, null); // optimize
                    alglib.minlbfgsresults(state, out optiAngle, out rep); // get results
                    for (int i = 0; i < N; i++)
                    {
                        radAngle[i + 1] = optiAngle[i];
                        // adjust angle based on joint limits
                        if (radAngle[i + 1] < (MinMax[i].X * Math.PI / 180))
                            radAngle[i + 1] = MinMax[i].X * Math.PI / 180;
                        else if (radAngle[i + 1] > (MinMax[i].Y * Math.PI / 180))
                            radAngle[i + 1] = MinMax[i].Y * Math.PI / 180;
                    }
                    solved = true;
                }
                else if (Ec > IK_POS_THRESH) // begin Cyclic Coordinate Descent loop
                {

                    // create target effector position vector
                    Vector3D Pid = Vector3D.Subtract(Pd, frame[link, 3]);
                    double wp = 1;      // position weight
                    double wo = 1;      // orientation weight
                    // calculate values for adjustment angle
                    double k1 = 0;
                    for (int i = 0; i < 3; i++)
                    {
                        if (Sigma[i])
                            k1 += wo * Vector3D.DotProduct(Rd[i], frame[link, 2]) *
        Vector3D.DotProduct(Rh[i], frame[link, 2]);
                    }
                    k1 += wp * Vector3D.DotProduct(Pid, frame[link, 2]) * Vector3D.
        DotProduct(Pih[link], frame[link, 2]);

                    double k2 = 0;
                    for (int i = 0; i < 3; i++)
                    {
                        if (Sigma[i])
                            k2 += wo * Vector3D.DotProduct(Rd[i], Rh[i]);
                    }
                    k2 += wp * Vector3D.DotProduct(Pid, Pih[link]);

                    double k3 = 0;
                    Vector3D ko3 = new Vector3D();
                    for (int i = 0; i < 3; i++)
                    {
                        if (Sigma[i])
                            ko3 = Vector3D.Add(ko3, wo * Vector3D.CrossProduct(Rh[i],
         Rd[i]));
                    }
                    k3 = Vector3D.DotProduct(frame[link, 2], Vector3D.Add(wp *
        Vector3D.CrossProduct(Pih[link], Pid), ko3));
                    double turnAngle;
                    // minimize position and orientation error
                    if ((k1 - k2) != 0)
                        turnAngle = Math.Atan(-k3 / (k1 - k2));
                    else
                        turnAngle = 0;
                    radAngle[link] += turnAngle;
                    // adjust angle based on joint limits
                    if (radAngle[link] < (MinMax[link - 1].X * Math.PI / 180))
                        radAngle[link] = MinMax[link - 1].X * Math.PI / 180;
                    else if (radAngle[link] > (MinMax[link - 1].Y * Math.PI / 180))
                        radAngle[link] = MinMax[link - 1].Y * Math.PI / 180;
                    if (double.IsNaN(radAngle[link]))
                        radAngle[link] = 0;

                    // backward recurssion through joints for CCD
                    if (link-- < 2) link = N;
                }
                // set previous error value for next loop
                Ep = Ec;
            }
            while (tries++ < IK_MAX_TRIES && Ec > IK_POS_THRESH);

            if (solved)
                Debug.WriteLine("BFGS");
            Debug.WriteLine("Error: "+Convert.ToString(Ec));
            Debug.WriteLine("Iterations: " + Convert.ToString(tries));

            double[] angles;
            // check if we are outputting workspace forces
            if (OutputWorkspace)
            {
                double forceGain = 0.5;
                angles = new double[N + 2];
                // calculate workspace forces if our position error is greater than the threshold
                if (Ec > IK_POS_THRESH)
                {
                    Vector3D forces = Vector3D.Multiply(forceGain, Vector3D.Subtract(Pd, Ph));
                    // invert forces if desired
                    angles[N - 1] = InvertForces[0] ? -forces.X : forces.X;
                    angles[N] = InvertForces[1] ? -forces.Y : forces.Y;
                    angles[N + 1] = InvertForces[2] ? -forces.Z : forces.Z;
                    for (int i = N - 1; i < N + 2; i++)
                    {
                        if (angles[i] > maxForce) angles[i] = maxForce;
                        else if (angles[i] < -maxForce) angles[i] = -maxForce;
                    }
                }
                else
                {
                    // no workspace force if we can reach desired point
                    angles[N - 1] = 0;
                    angles[N] = 0;
                    angles[N + 1] = 0;
                }
            }
            else
                angles = new double[N];
            // change output angles based on joint coupling
            switch (Coupling)
            {
                case CouplingType.None:
                    //convert angles to degrees
                    for (int i = 0; i < N; i++)
                    {
                        angles[i] = radAngle[i + 1] * 180 / Math.PI;
                    }
                    break;
                case CouplingType.ShoulderTwoDOF:
                    angles[0] = (radAngle[1] + radAngle[2]) * 180 / Math.PI;
                    angles[1] = (radAngle[1] - radAngle[2]) * 180 / Math.PI;
                    for (int i = 2; i < N - 1; i++)
                    {
                        angles[i] = radAngle[i + 1] * 180 / Math.PI;
                    }
                    break;
            }
            return angles;
        }

        public void function1_grad(double[] q, ref double func, double[] grad, object obj)
        {
            Vector3D[] Pstar = new Vector3D[N];
            Pstar.Initialize();

            // initialize frame positions
            for (int i = 0; i < N + 1; i++)
            {
                frame[i, 3].X = 0;
                frame[i, 3].Y = 0;
                frame[i, 3].Z = 0;
            }
            // forward recurrsion formulas for frame position and orientation
            for (int i = 1; i <= N; i++)
            {
                // x(i) orientation vector
                frame[i, 0] = Vector3D.Add((Vector3D.Multiply((Math.Cos(radAngle[i - 1] + thetaOffset[i])), frame[(i - 1), 0])), (Vector3D.Multiply((Math.Sin(radAngle[i - 1] + thetaOffset[i])), frame[(i - 1), 1])));
                // z(i) orientation vector
                frame[i, 2] = Vector3D.Add((Vector3D.Multiply(Math.Cos(DHparameters[i - 1, 0] * Math.PI / 180), frame[(i - 1), 2])), Vector3D.Multiply(Math.Sin(DHparameters[(i - 1), 0] * Math.PI / 180), Vector3D.CrossProduct(frame[i, 0], frame[(i - 1), 2])));
                // y(i) orientation vector
                frame[i, 1] = Vector3D.CrossProduct(frame[i, 2], frame[i, 0]);
            }
            // P* --> relative positions of next frame wrt present frame
            for (int i = 0; i < N; i++)
            {
                Pstar[i] = Vector3D.Add(Vector3D.Multiply(DHparameters[i, 2], frame[i, 2]), Vector3D.Multiply(DHparameters[i, 1], frame[i + 1, 0]));
            }
            //P(i) --> frame positions wrt base frame
            for (int i = 1; i < N + 1; i++)
            {
                frame[i, 3] = frame[i - 1, 3] + Pstar[i - 1];
            }
            // set position of end effector
            Ph = frame[N, 3];

            // compute relative positions
            Pih = new Vector3D[N + 1];
            for (int i = N - 1; i >= 0; i--)
            {
                Pih[i] = Vector3D.Subtract(Ph, frame[i, 3]);
            }
            // set end effector orientation
            Rh[0] = frame[N, 0];
            Rh[1] = frame[N, 1];
            Rh[2] = frame[N, 2];
            // calculate orientation error
            Eo = 0;
            for (int i = 0; i < 3; i++)
            {
                if (Sigma[i])
                {
                    Eo += Math.Pow((Vector3D.DotProduct(Rd[i], Rh[i]) - 1), 2);
                }
            }
            // function to be minimized
            func = Eo + Vector3D.DotProduct(Vector3D.Subtract(Pd, Ph), Vector3D.Subtract(Pd, Ph));
            // declare gradient vector elements for each joint
            for (int i = 0; i < N-1; i++)
            {
                Vector3D grad0 = new Vector3D();
                for (int j = 0; j < 3; j++ )
                {
                    if (Sigma[j])
                        grad0 = Vector3D.Add(grad0, (Vector3D.DotProduct(Rd[j], Rh[j]) - 1) * Vector3D.CrossProduct(Rh[j], Rd[j]));
                }
                grad[i] = Vector3D.DotProduct(Vector3D.Multiply(2, frame[i, 2]), Vector3D.Add((Vector3D.CrossProduct(Vector3D.Subtract(Pd, Ph), Pih[i])), grad0));
            }
        }
```

Motor Control Module Schematics



KV10 BLDC Control

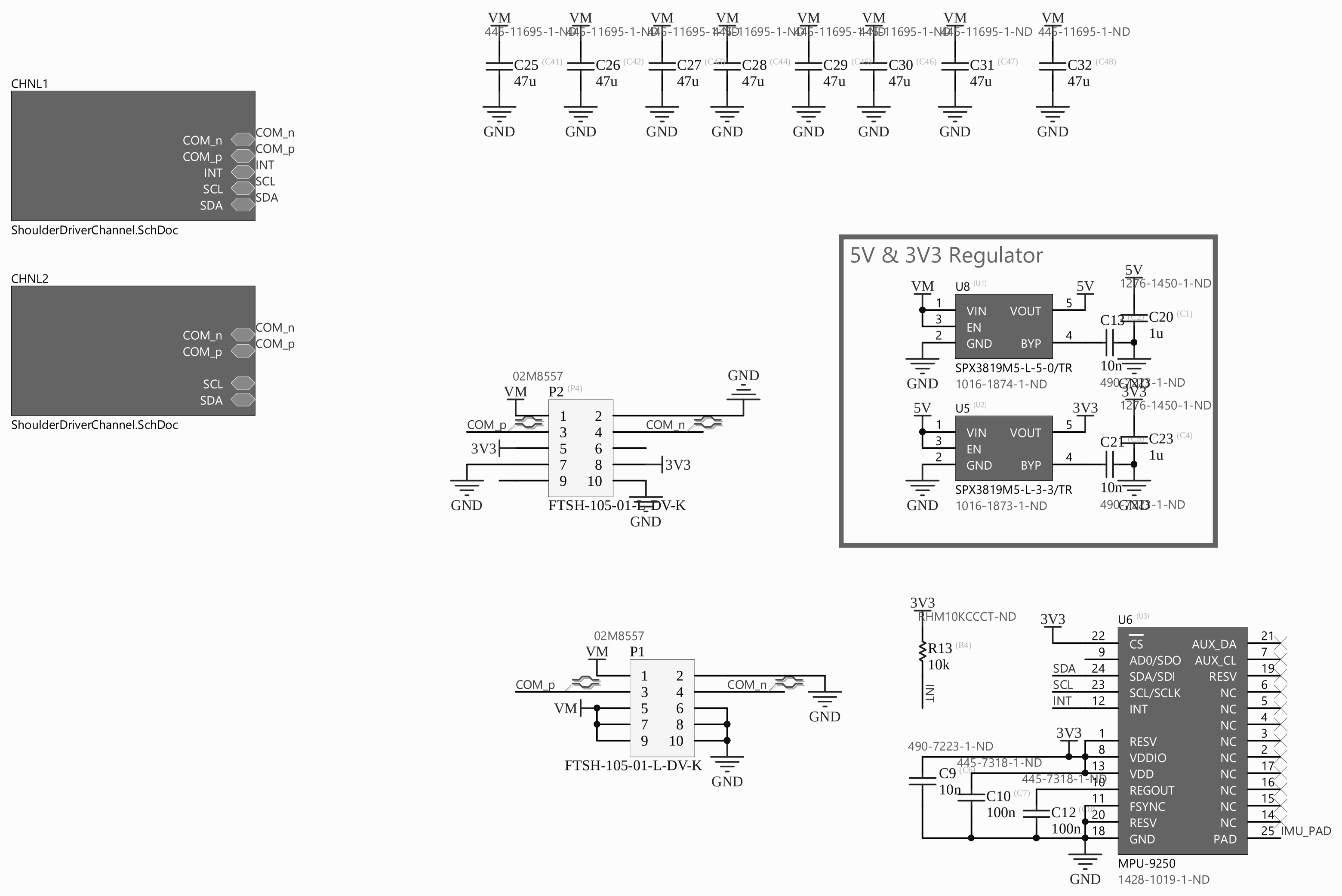
Revision B1 | Project: KV10BLDCControl.PrjPcb

11/21/2016 4:06:08 PM | Sheet 1 of 1

C:\Users\lcubrich\Documents\GitHub\arm-driver\KV10BLDCControl.SchDoc

Jay Carlson
209N SEC
P.O. Box 880511
Lincoln, NE 68588-0511

B



Title		
Size Letter	Number	Revision
Date: 11/21/2016		Sheet of
File: C:\Users\..\ShoulderDriver.SchDoc		Drawn By:

Motor Datasheets

FAULHABER

Brushless DC-Servomotors

2 Pole Technology

0,36 mNm

1,7 W

Series 0620 ... B

Values at 22°C and nominal voltage		0620 K	006 B	012 B	
1	Nominal voltage	U_N	6	12	V
2	Terminal resistance, phase-phase	R	8,8	60,2	Ω
3	Efficiency, max.	$\eta_{max.}$	51	50	%
4	No-load speed	n_0	48 600	37 300	min^{-1}
5	No-load current, typ. (with shaft ø 1 mm)	I_0	0,056	0,018	A
6	Stall torque	M_H	0,732	0,551	mNm
7	Friction torque, static	C_0	0,011	0,011	mNm
8	Friction torque, dynamic	C_v	$1{,}02 \cdot 10^{-6}$	$1{,}02 \cdot 10^{-6}$	mNm/min^{-1}
9	Speed constant	k_n	8 761	3 386	min^{-1}/V
10	Back-EMF constant	k_E	0,114	0,295	mV/min^{-1}
11	Torque constant	k_M	1,09	2,82	mNm/A
12	Current constant	k_I	0,917	0,355	A/mNm
13	Slope of n-M curve	$\Delta n/\Delta M$	70 730	72 289	min^{-1}/mNm
14	Terminal inductance, phase-phase	L	28	192	µH
15	Mechanical time constant	τ_m	7	7,2	ms
16	Rotor inertia	J	0,0095	0,0095	gcm^2
17	Angular acceleration	$\alpha_{max.}$	771	580	$\cdot 10^3 rad/s^2$

18	Thermal resistance	R_{th1} / R_{th2}	13,2 / 84,3	K/W
19	Thermal time constant	τ_{w1} / τ_{w2}	1,1 / 89	s
20	Operating temperature range:			
	– motor		-20 ... +100	°C
	– winding, max. permissible		+125	°C
21	Shaft bearings		ball bearings, preloaded	
22	Shaft load max.:			
	– with shaft diameter		1	mm
	– radial at 10 000 min^{-1} (4 mm from mounting flange)		2	N
	– axial at 10 000 min^{-1} (push only)		0,6	N
	– axial at standstill (push only)		10	N
23	Shaft play:			
	– radial	≤	0,012	mm
	– axial	=	0	mm
24	Housing material		aluminium, black anodized	
25	Mass		2,5	g
26	Direction of rotation		electronically reversible	
27	Speed up to	$n_{max.}$	100 000	min^{-1}
28	Number of pole pairs		1	
29	Hall sensors		digital	
30	Magnet material		NdFeB	

Rated values for continuous operation			006 B	012 B	
31	Rated torque	M_N	0,28	0,3	mNm
32	Rated current (thermal limit)	I_N	0,311	0,122	A
33	Rated speed	n_N	21 820	7 290	min^{-1}

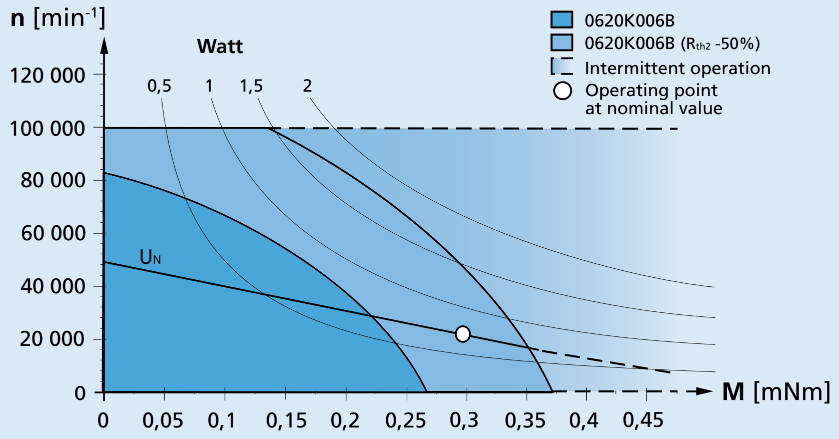
Note: Rated values are calculated with nominal voltage and at a 22°C ambient temperature. The R_{th2} value has been reduced by 25%.

Note:

The diagram indicates the recommended speed in relation to the available torque at the output shaft for a given ambient temperature of 22°C.

The diagram shows the motor in a completely insulated as well as thermally coupled condition (R_{th2} 50% reduced).

The nominal voltage (U_N) curve shows the operating point at nominal voltage in the insulated and thermally coupled condition. Any points of operation above the curve at nominal voltage will require a higher operating voltage. Any points below the nominal voltage curve will require less voltage.



Recommended operation areas (example: nominal voltage 6V)

For notes on technical data and lifetime performance refer to "Technical Information".
Edition 2016

Planetary Gearheads

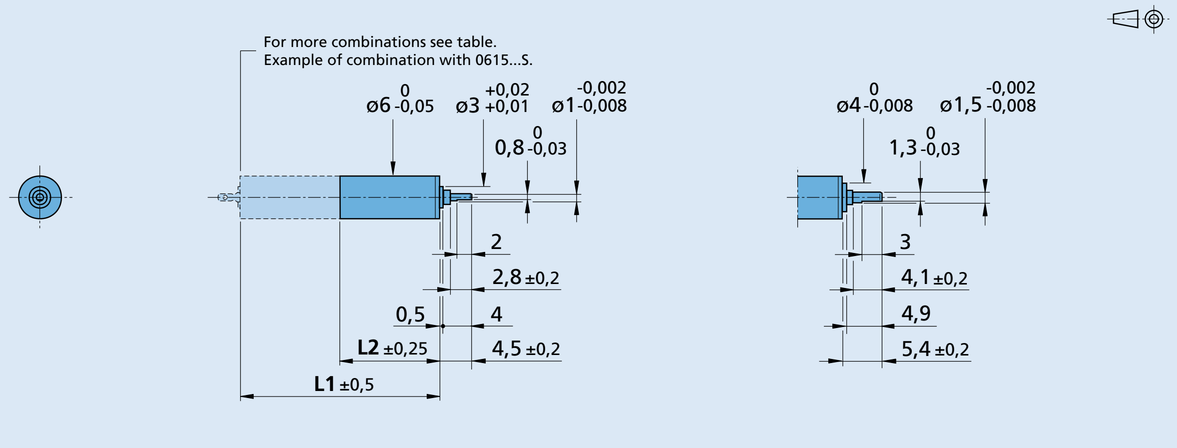
25 mNm

For combination with
DC-Micromotors
Brushless DC-Motors
Stepper Motors

Series 06/1

	06/1	06/1K
Housing material	steel	steel
Geartrain material	steel	steel
Recommended max. input speed for:		
– continuous operation	8 000 min^{-1}	8 000 min^{-1}
Backlash, at no-load	≤ 3 °	≤ 3 °
Bearings on output shaft	sintered bearings	ball bearings
Shaft load, max.:		
– radial (3,5 mm from mounting face)	≤ 0,5 N	≤ 5 N
– axial	≤ 0,5 N	≤ 3 N
Shaft press fit force, max.	≤ 3,5 N	≤ 5 N
Shaft play		
– radial (3,5 mm from mounting face)	≤ 0,06 mm	≤ 0,06 mm
– axial	≤ 0,1 mm	≤ 0,05 mm
Operating temperature range	- 30 ... + 100 °C	- 30 ... + 100 °C

Specifications							
Number of gear stages		1	2	3	4	5	6
Continuous torque	mNm	25	25	25	25	25	25
Intermittent torque	mNm	35	35	35	35	35	35
Mass without motor, ca.	g	2	2,8	3,4	4	4,4	5
Efficiency, max.	%	90	80	70	60	55	48
Direction of rotation, drive to output		=	=	=	=	=	=
Reduction ratio (exact)		4:1	16:1	64:1	256:1	1 024:1	4 096:1
L2 [mm] = length without motor		9,2	11,9	14,6	17,3	20,0	22,7
L1 [mm] = length with motor	0615C...S	24,2	26,9	29,6	32,3	35,0	37,7
	0515C...B	23,8	26,5	29,2	31,9	34,6	37,3
	0620C...B	29,2	31,9	34,6	37,3	40,0	42,7
	FDM0620...-35	18,7	21,4	24,1	26,8	29,5	32,2



06/1

06/1 K (L1, L2 = +1)

For notes on technical data and lifetime performance refer to "Technical Information".
Edition 2016

Brushless DC-Servomotors

2 Pole Technology

2,6 mNm

9,9 W

Series 1226 ... B

Values at 22°C and nominal voltage		1226 S		006 B	012 B	
1	Nominal voltage	U_N		6	12	V
2	Terminal resistance, phase-phase	R		2,2	5,45	Ω
3	Efficiency, max.	$\eta_{max.}$		71	72	%
4	No-load speed	n_0		21 000	27 400	min^{-1}
5	No-load current, typ. (with shaft ø 1,2 mm)	I_0		0,07	0,054	A
6	Stall torque	M_H		7,24	8,99	mNm
7	Friction torque, static	C_0		0,073	0,073	mNm
8	Friction torque, dynamic	C_v		$5{,}3 \cdot 10^{-6}$	$5{,}3 \cdot 10^{-6}$	mNm/min^{-1}
9	Speed constant	k_n		3 563	2 318	min^{-1}/V
10	Back-EMF constant	k_E		0,281	0,431	mV/min^{-1}
11	Torque constant	k_M		2,68	4,12	mNm/A
12	Current constant	k_I		0,373	0,243	A/mNm
13	Slope of n-M curve	$\Delta n/\Delta M$		2 925	3 066	min^{-1}/mNm
14	Terminal inductance, phase-phase	L		36	85	µH
15	Mechanical time constant	τ_m		4,4	4,7	ms
16	Rotor inertia	J		0,15	0,15	gcm^2
17	Angular acceleration	$\alpha_{max.}$		499	621	$\cdot 10^3 rad/s^2$
18	Thermal resistance	R_{th1} / R_{th2}	7,3 / 36,6			K/W
19	Thermal time constant	τ_{w1} / τ_{w2}	3,2 / 207			s
20	Operating temperature range:					
	– motor		-20 ... +100			°C
	– winding, max. permissible		+125			°C
21	Shaft bearings		ball bearings, preloaded			
22	Shaft load max.:					
	– with shaft diameter		1,2			mm
	– radial at 10 000 min^{-1} (4 mm from mounting flange)		5			N
	– axial at 10 000 min^{-1} (push only)		2,5			N
	– axial at standstill (push only)		11			N
23	Shaft play:					
	– radial	≤	0,012			mm
	– axial	=	0			mm
24	Housing material		aluminium, black anodized			
25	Mass		13			g
26	Direction of rotation		electronically reversible			
27	Speed up to	$n_{max.}$	79 000			min^{-1}
28	Number of pole pairs		1			
29	Hall sensors		digital			
30	Magnet material		NdFeB			

Rated values for continuous operation						
31	Rated torque	M_N		2,13	1,97	mNm
32	Rated current (thermal limit)	I_N		0,932	0,573	A
33	Rated speed	n_N		12 480	19 670	min^{-1}

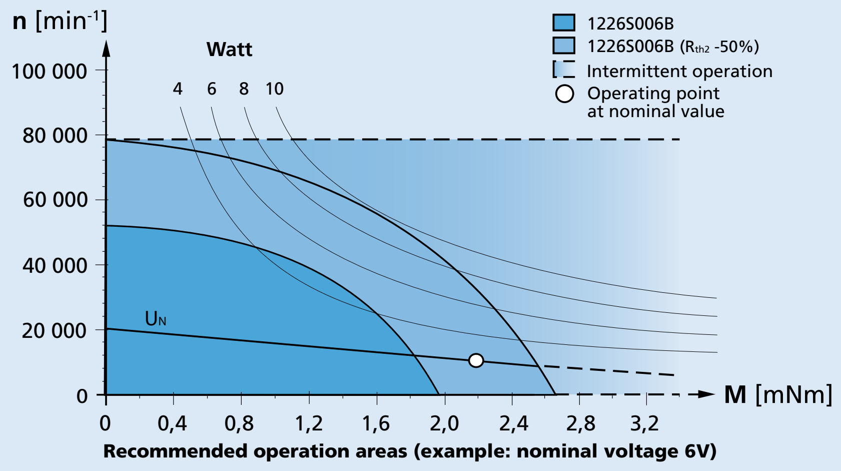
Note: Rated values are calculated with nominal voltage and at a 22°C ambient temperature. The R_{th2} value has been reduced by 25%.

Note:

The diagram indicates the recommended speed in relation to the available torque at the output shaft for a given ambient temperature of 22°C.

The diagram shows the motor in a completely insulated as well as thermally coupled condition (R_{th2} 50% reduced).

The nominal voltage (U_N) curve shows the operating point at nominal voltage in the insulated and thermally coupled condition. Any points of operation above the curve at nominal voltage will require a higher operating voltage. Any points below the nominal voltage curve will require less voltage.



Recommended operation areas (example: nominal voltage 6V)

For notes on technical data and lifetime performance refer to "Technical Information".
Edition 2016

FAULHABER

Planetary Gearheads

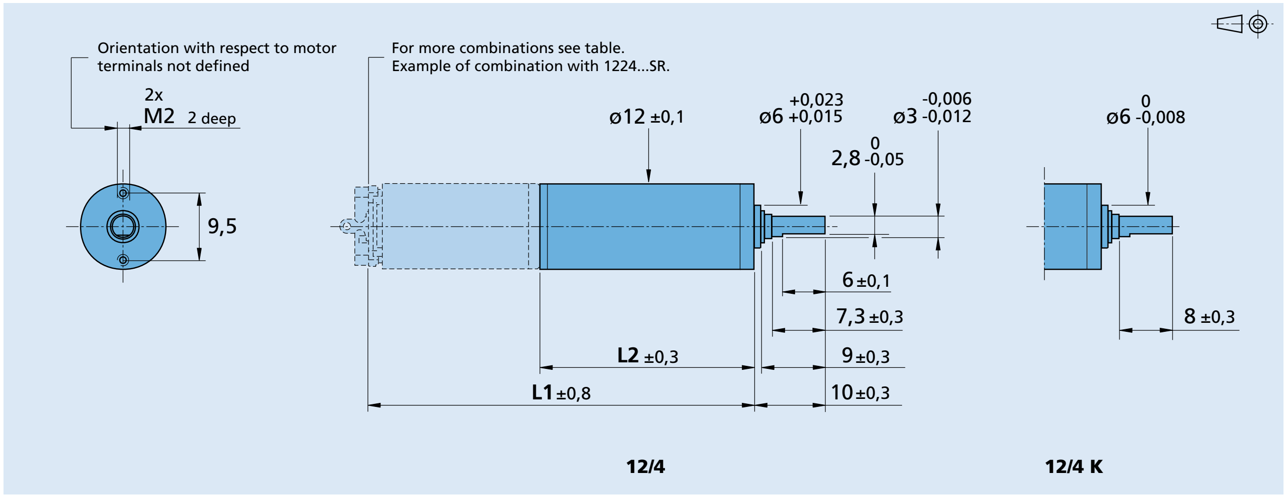
0,3 Nm

For combination with
DC-Micromotors
Brushless DC-Motors
Stepper Motors

Series 12/4

	12/4	12/4K
Housing material	metal	metal
Geartrain material	metal	metal
Recommended max. input speed for:		
– continuous operation	5 000 min^{-1}	5 000 min^{-1}
Backlash, at no-load	≤ 3 °	≤ 3 °
Bearings on output shaft	sintered bearings	ball bearings, preloaded
Shaft load, max.:		
– radial (6 mm from mounting face)	≤ 4 N	≤ 20 N
– axial	≤ 3 N	≤ 5 N
Shaft press fit force, max.	≤ 15 N	≤ 5 N
Shaft play		
– radial (6 mm from mounting face)	≤ 0,05 mm	≤ 0,04 mm
– axial	≤ 0,1 mm	= 0 mm
Operating temperature range	- 30 ... + 100 °C	- 30 ... + 100 °C

Specifications						
Number of gear stages		**1**	**2**	**3**	**4**	**5**
Continuous torque	mNm	300	300	300	300	300
Intermittent torque	mNm	450	450	450	450	450
Mass without motor, ca.	g	12	15	18	21	24
Efficiency, max.	%	90	80	70	60	55
Direction of rotation, drive to output		=	=	=	=	=
Reduction ratio (exact)		**4:1**	**16:1**	**64:1**	**256:1**	**1 024:1**
L2 [mm] = length without motor		15,1	19,7	24,3	28,9	33,5
L1 [mm] = length with motor	1024A...S	38,8	43,4	48,0	52,6	57,2
	1224A...SR	39,3	43,9	48,5	53,1	57,7
	1028A...B	43,2	47,8	52,4	57,0	61,6
	1218A...B	33,1	37,7	42,3	46,9	51,5
	1226A...B	41,1	45,7	50,3	54,9	59,5
	ADM1220S...-59	32,5	37,1	41,7	46,3	50,9



For notes on technical data and lifetime performance refer to "Technical Information".
Edition 2016