Master's Thesis

End Effector Design and Control of Planetary Rover Robot Arm for Communication Relay Retrieval

Stefan Leutenegger

2008-10

Adviser: Vytas Sunspiral

Dr. Terrence Fong
Intelligent Robotics Group
Nasa Ames Research Center
Moffett Field, California, USA

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prof. Dr. Roland Siegwart
Autonomous Systems Lab
Swiss Federeal Institute of Technology (ETH)
Zurich, Switzerland

Abstract

K10, a prototype planetary rover automatically deploys communication relays during its exploration task, for the retrieval of which a 5-degree-of-freedom (DoF) robot Arm is considered to be mounted.

In order to establish a working setup capable of performing automatic pick-ups of those relay ”Bricks” at arbitrary positions and orientations, a suitable Gripper had to be developed first. Since the 5-DoF Arm lacks one orientation DoF, a rotation symmetric Gripper was designed that can always grasp one of the two flaps of the Brick.

The Arm was equipped with two control modes: it has the ability to precisely follow straight lines both in joint space as well as in Cartesian space. Despite this flexibility, motion planning was found to be indispensable: SBL, a state-of-the-art probabilistic motion planner was therefore implemented with major extensions and adaptions allowing to deal with the ability of the grasped Brick to align its orientation with gravity.

Zusammenfassung

K10 ist ein Prototyp eines Planeten-Rovers, der während seiner Erkundungsaufgaben vollautomatisch Kommunikationsrelays absetzt. Ein 5-freiheitsgradiger Roboterarm kann dafür eingesetzt werden, diese Relays im Nachhinein wieder einzusammeln.

Um eine funktionierende Testumgebung für automatisiertes Auflesen von Relays beliebiger Position und Orientierung zu schaffen, musste als Erstes ein Greifer konstruiert werden. Weil dem Arm ein Freiheitsgrad bezüglich Orientierung fehlt, hat der Greifer eine rotations-symmetrische Geometrie, die in jedem Fall erlaubt, die eine der beiden Klappen am Relay zu ergreifen.

Der Arm wurde mit zwei Regelungs-Modi ausgestattet: er kann sowohl geraden Linien im kartesischen Raum wie auch im Raum der Gelenkwinkel folgen. Trotz dieser Flexibilität hat sich herausgestellt, dass Motion Planning unabdingbar ist: darum ist SBL implementiert worden, ein dem Stand der Technik entsprechender probabilistischer Planner. Allerdings musste dieser signifikanten Anpassungen und Erweiterungen unterzogen werden, um die Tatsache zu bewältigen, dass das Relay nicht fix mit dem Greifer verbunden ist, sondern seine Orientierung der Schwerkraft anpassen kann.

Acknowledgments

There is a long list of people I want to thank for their support and help:

- First of all Prof. Roland Siegwart at ETH for letting me write my Master's Thesis here in California within IRG.
- Dr. Terry Fong, leader of IRG, for accepting me as an intern in his group and helping me with the associated paperwork.
- CMU Silicon Valley for their help with my visa and organizing talks.
- Vytas SunSpiral, my competent adviser, for his valuable inputs and guiding along with correcting the report at hand.
- All the IRG people who have always been willing to help me, to name a few Susan Lee for her advice in conjunction with mechanical design, Kyle Husmann for his help with the stereo camera, Mike Landis for lending me his high-end camera and converting video files, Camilla Ljungström for getting me started, and last but not least Daniel Chavez for helping me implementing the motion planning library.
- My family, who supported this undertaking from the very beginning.

Contents

List of Figures

List of Tables

1 Introduction

Future manned missions to the moon as part of the Constellation Project will be supported by various robotic equipment: IRG[1] is developing teleoperated and supervised ”utility” robot hardware and software as well as ground control structures and operational procedures. These systems are aimed at performing highly repetitive routine tasks in manned missions while not requiring continuous human robot interaction [2].

K10 is the most recent planetary rover used by IRG (described in Section 2.1). One of its capabilities is the automatic deployment of communication relays whenever keeping the wireless link to the base station requires it – which is a very powerful capability for planetary rovers of a certain degree of autonomy. There is not yet, however, a solution offered for the retrieval of those communication relays back to the base station. This capability is important for future manned missions with autonomous rover support, since the relays will not be available in big numbers and collecting them manually would be cumbersome. The Master's Thesis at hand is aimed at investigating the use of a 5-degree-of-freedom (DoF) manipulator arm mounted to K10 for the purpose of picking up the communication relays and carrying them.

The goal was set to establish a working K10 Arm test setup in the lab for autonomous relay pick-ups. In order to achieve this, the following work packages needed to be approached:

1. **Design of an end effector capable of grasping communication relays at arbitrary orientation**
   Hereby, the mechanical design needs to comply with several constraints, such as the lacking 6th DoF, or the weight limit. Conclusion of this first step necessitates some driver software development and integration of hardware and software with the Arm as well.

2. **Evaluation and Implementation of Control Algorithms**
   There are two modes of controlling the Arm that need to be implemented – which is joint space path following on the one hand and Cartesian space path following on the other hand.

[1] Intelligent Robotics Group, Nasa Ames Research Center, Moffett Field, California, USA

3. **Using stereo vision for relay position and attitude determination**
   Given the situation where the relay is in the field of view of the stereo camera as well as within the Arm workspace, it needs to be localized precisely: a tool is presented prompting the operator to select correlating points in both stereo images allowing relay localization.

4. **Motion planning**
   Even in an uncluttered environment, the Arm must avoid self collisions as well as collisions with the relay when moving towards it. Therefore a suitable motion planning algorithm needs to be implemented. Planning motion after grasping, i.e. with the relay device as part of the Arm proofed to be a particular challenge.

1.1 Previous Work

The Arm (including controller boards) was fully operational at the starting point of this project; it was mounted off the rover in the lab for testing. Some Arm control software from a previous project (all C++) was also available. Furthermore, the Master's Thesis of Camilla Ljungström [7], another intern student at IRG, is aimed at setting up operation of the Arm and adapting the software to the needs of K10 including joint space control algorithms. In terms of motion planning, the library provided by Jean-Claude Latombe's Motion Planning Group at Stanford University is available which was also used for motion planning of ATHLETE's FootFall[2] another project IRG is part of.

[2] All-Terrain-Hex-Limbed Extra Terrestrial Explorer, developed by the Jet Propulsion Laboratory (JPL)

2 Background

This Section provides an overview of the hardware related to the project. Furthermore, its broader context is being explained.

2.1 K10 Rover

K10 is a rover platform developed at the NASA Centers JPL and Ames aimed at testing several planetary surface science task. So far, the two newest K10s ”red” and ”black” have been used by IRG for testing topographic mapping, survey and site recon (science scouting), ground-penetrating radar, mapping wireless network coverage as well as to deploy Wi-Fi relay devices.

The rover is further equipped with a stereo camera used for navigation and mapping as well as with a GPS for testing on earth.

Deployment of communication relays (as presented in Section 2.3) is triggered fully automatically: they are being shot out of a container mounted at the rear side of K10 chargeable with up to six communication nodes. The opening of the spring loaded relay is initiated by K10 via Wi-Fi.

2.1.1 Technical Data

The overall height of the most recent K10 version (”red” and ”black”) sums up to 1.5 meters. The wheel axis distance measures 0.81 m and the wheel spacing is 0.78 m. As a consequence of the wheel diameter being just 0.30 m, K10 can only climb over rounded obstacles of up to 0.25-0.30 m height.

The overall mass amounts to around 50 kg, depending of course on the payload.

Measured maximum speeds on flat terrain are around 0.90 m/s. K10’s predecessor K9 which is based on JPL’s FIDO chassis can not exceed speeds of approximately 0.06 m/s.

The designated mounting point for the Arm lies centered on the front of K10 at a hight of 0.45 m.

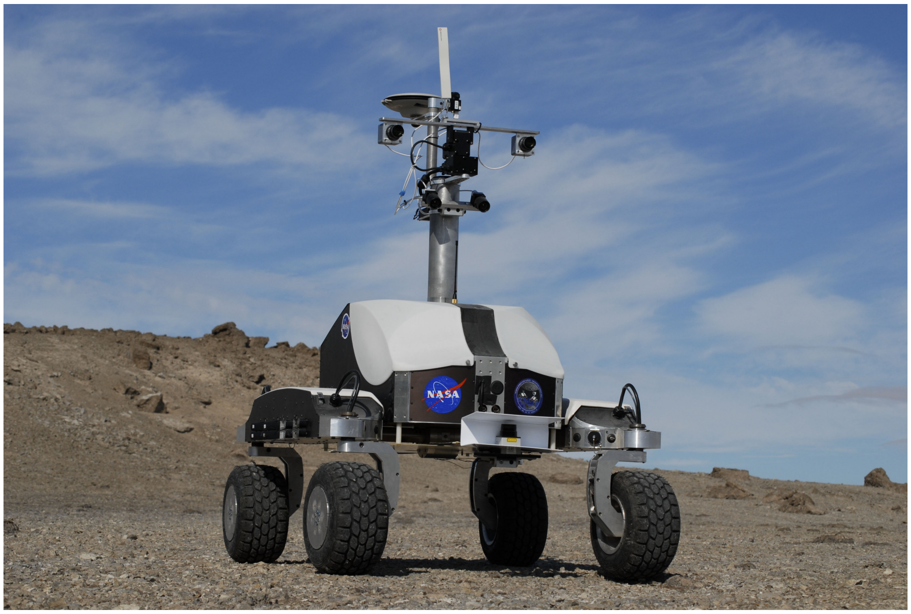


Figure 1: *K10 ("Black") Rover at Moses Lake Field Test 2008. Source: http://www.nasa.gov/centers/ames/multimedia/images/2008/moses_lake.html as on 09/13/2008*

2.2 5-DoF Arm

K10's predecessor, K9, was equipped with a 5-DoF Arm capable of performing geological analysis. The CHAMP (Camera Hand-lens MicroscoPe) microscopic camera was attached to the Arm as a tool. K9 would autonomously place the CHAMP against nearby rocks in order to acquire microscopic images of surface features to support physical characterization of rock geology.

The Arm features a chain of five revolute joints named after their human antetypes:

1. Waist

2. Shoulder

3. Elbow

4. Twist

5. Wrist

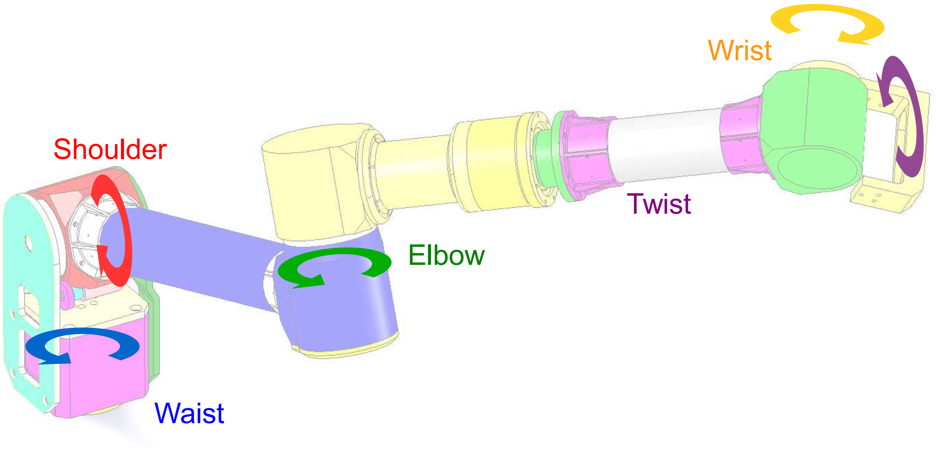


Figure 2: *CAD Arm model highlighting the kinematic chain of the Arm.*

The first three joints allow arbitrary positioning of the tool within the Arm workspace whereas the last two degrees of freedom allow orienting it. Note that only yaw and pitch are available; the third orientation degree of freedom is missing, i.e. there is no control over the roll.

The dimensions also resemble a human arm: both forearm and upper arm are each about 0.3 m long. Figure 2 depicts the Arm with its respective joints and coordinate frames. Sections 5.2 and 5.3 describe the forward as well as backward kinematics and provides a workspace analysis.

The five highly geared brush-type DC motors allow low power control of the joint angles. PIC-Servo Boards[3] assembled in a stack run a real-time control loop with the motors. The joint positions are sensed via encoders; the absolute angles are retrieved by reading potentiometers at initialization via a B&B data acquisition module[4].

2.3 Communication Relays

The Automatically Deployed Communication Relays (ADCR) by SPAWAR Systems provide extended communication range for Unmanned Ground Vehicles (UGV) beyond the operation within line of sight (LOS) to a control station. As the name is indicating, the devices are being deployed automatically whenever maintaining the link back to the control station requires to

[3] PIC-SERVO SC Motion Control Board; by Jeffrey Kerr LLC, Berkeley, CA 94708

[4] RS-232 Data Acquisition Module, Model 232SDA12; by B&B Electronics, Ottawa, IL 61350

do so.

The Relays operate using IP packets allowing communication over TCP/IP or UDP.

The deployer unit features six chambers being spring-loaded when charged with relays. The relays have the shape of a brick in that closed state, therefore they are also referred to as Comm Bricks. When a deploy action is initiated, the Brick is shot out of the deployer. A few seconds later, the Relay Brick will open its spring loaded flaps as well as unfold its spring loaded antenna. This opening also self-rights the Relay which brings the antenna in its vertical orientation.

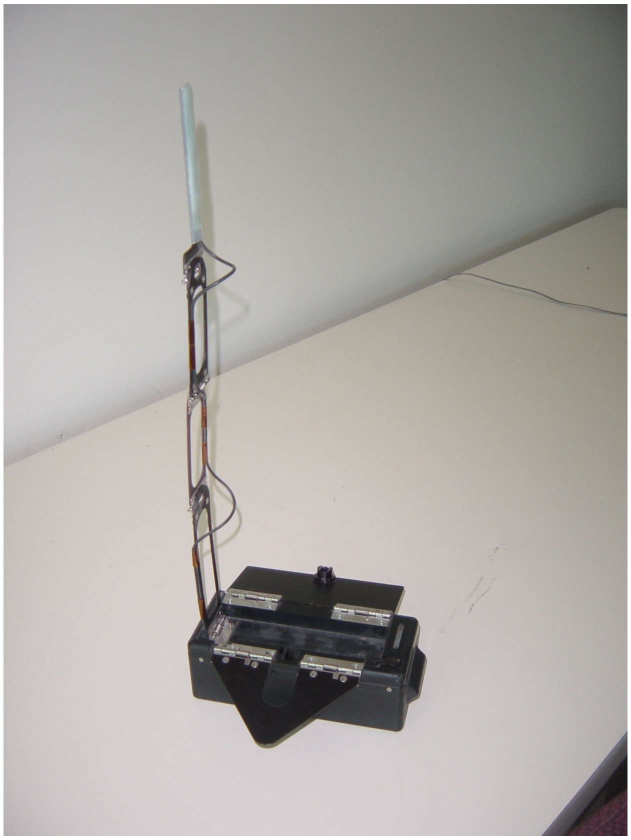


Figure 3: *Opened Communication Relay.*

A Brick is shown in Figure 3 in its opened state. Its overall height to the antenna tip measures 0.52 m, while having a mass of 0.49 kg.

3 State of the Art in Robotic Manipulators

In industrial manipulation robotics, 5-DoF arms are mainly used for welding, painting and cutting; these operations are invariant to the last orientation DoF, therefore it does not necessarily have to be articulated. Robots that need to grasp objects at arbitrary orientation are designed with at least 6 DoF – or more in order to achieve redundancy, making them more flexible in terms of path planning around obstacles as well as in terms of avoiding singularities in their workspaces.

3.1 End Effectors

The major challenge faced in end effector design is caused by the trade-off between their complexity (including number of DoF) and flexibility with respect to the range of objects that may be grasped [8]. Industrial robots tend to be equipped with highly specific but comparably simple tools (that may be exchangeable) such as 1-DoF grippers. Human hand resembling end effectors, however, are examples of the complex end of the end effector spectrum: their design complexness also imposes major challenges in terms of control. Figure 4 shows an example of an anthropomorphic hand [6].

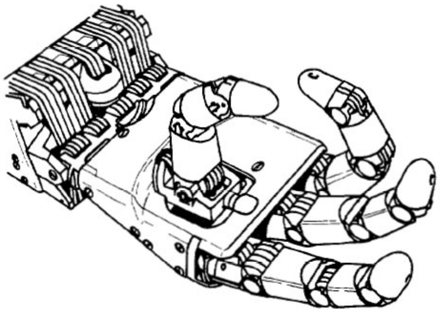


Figure 4: *The Utah/MIT dextrous hand. Copied from [6].*

In order to compensate for positioning imprecision and preventing damage to robot or object, force feedback may be desirable.

3.2 Motion Planning

Regarding an overview covering the state of the art in manipulator motion planning, the reader is referred to Section 7.2.

4 Mechanical Design of End Effector

The first task comprises designing, manufacturing and setting up operation of an end effector suitable for Communication Brick pick-up. As a consequence of the Arm being given only 5 DoF, there is an additional requirement for the end effector: it needs to perform a reliable grip and also be invariant to the last rotational DoF. Standard grippers as used in industrial automation generally do not fulfill this requirement and are likely to be of too big size. Therefore, a simple end effector principle was evaluated, designed, built and integrated with the Arm, including driver software development.

4.1 Kinematic Considerations

The kinematics of the 5-DoF robot Arm impose major design challenges limiting the collection of possible principles: as sketched in the previous Section 2.1.1, it will only be possible to specify the tool position as well as the direction it is pointing at; the last rotation (roll) can not be specified. Therefore, an end effector needs to be designed, which complies with this lack of the last DoF. This requirement is most easily fulfilled by end effectors which are rotation-symmetric with respect to an arbitrary axis not coincident with the Wrist axis.

A second consideration concerns the inverse kinematics: the K9 Arm arrangement did not allow a completely analytical solution. After having specified the tool pointing direction and position, a Nelder-Mead-minimization is searching for possible Wrist positions on a circle around the tool axis. Once the Wrist position is known, the different solutions for the first three joint angles (Waist, Shoulder, Elbow) can be found analytically. The remaining Twist and Wrist angle can thereafter be determined easily. The numerical part is only needed due to the tool frame origin offset from the Twist axis (i.e.search circle radius). Therefore, designing the end effector in such a way that this offset becomes zero would result in purely analytical inverse kinematics. This would be highly desirable, since especially for Cartesian space control the inverse kinematics will need to be solved frequently.

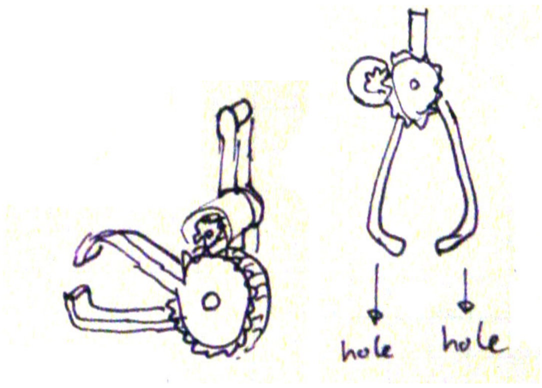
Note that this descriptive explanation does not contradict the fact stating it is possible to determine closed-form inverse kinematics if the last three rotation axes intersect in a 6-DoF manipulator with all joints revolute (i.e. decoupled position and orientation)[1]: In case the Arm had a sixth joint allowing tool roll as well, the desired tool frame orientation could be specified, thus the Wrist position is *a priori* known independent from any tool offset.

4.2 Principle Evaluation

The Comm Bricks (as presented in Section 2.3) open two thin flaps each featuring either a hole or a knob. Both may be used for form locking grasps. Note that the antenna was assumed to be too fragile to serve that purpose. In the following, an overview of different end effector principles considered for the evaluation is given.

4.2.1 Claws

A first class of end effectors comprises any claw-like tool making use of the hole in the smaller flap. Figure 5 shows two different designs.



(a) For grasping from the side.

(b) For grasping from the top.

Figure 5: *Two different claw types.*

Such claws are generally of a low complexity as well as low weight. A simple rotary motor could be used as actuator. Another advantage is that the brick attached may hang down in a lift or carry position without transferring additional forces to the motor as well as the structure.

It is important to note that only a claw of type (b) could grasp the Brick being arbitrarily oriented in space: for any roll around the tool axis, there will be a possible grasping location on an arc (min. 180°) around the hole (left-bottom-right). There will, however, even in this case be a certain complication in terms of inverse kinematics, since the resulting roll depends on the specified position or vice versa. Therefore, time consuming iterative parts would be needed similar as with the K9 CHAMP positioning.

A second drawback is shared by all principles involving entering the flap hole of diameter d_{hole} with (part of) the tool of diameter d_{tool}: for the lateral tool positioning error ϵ of the Arm, the following inequality must be fulfilled for successful grasping:

$$\epsilon \leq \frac{1}{2}\left(d_{hole} - d_{tool}\right) \tag{4.1}$$

In words, even for infinitely thin claws, a the tool positioning precision must not exceed $\frac{1}{2}d_{hole} = 12.7mm$. Note that the overall positioning precision is not only depending on calibration and modeling errors, but also on the vision based Brick localization error.

4.2.2 Axis Symmetrical Tools Locking in Hole

A tool of this class is meant to be inserted into the hole; as a second step a mechanism is locking the flap to the tool. There are different possible ways to perform that form-lock involving various actuators, Figures 6 and 7 depict two examples. The fact, that these tools are invariant to rotation along their axes is their major advantage: not only would they be able to grasp Bricks of arbitrary orientation, also the inverse kinematics offer an analytical solution due to symmetry.

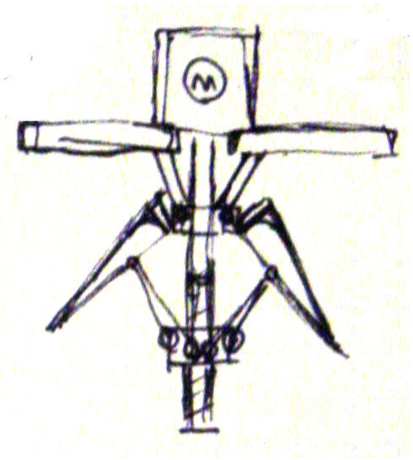


Figure 6: *Possible mechanism allowing the inserted tool to form-lock with the flap hole: it is using a combination of rotary actuator and worm shaft.*

Unfortunately, such tools require a high positioning precision. However, a very slim design is hardly possible due to a necessary high level of complexity with respect to the mechanics.

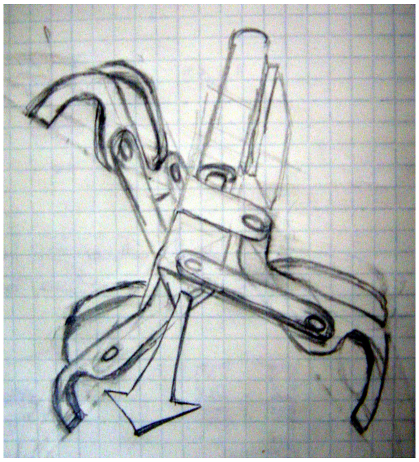


Figure 7: *More complex mechanism involving a linear actuator.*

4.2.3 Gripper Style

A gripper style end effector impresses with the opportunity for a very simple and therefore robust design. Furthermore, a Gripper may be designed rotation symmetric, such that the requirements for analytical inverse kinematics as well as for the ability of grasping Bricks at arbitrary orientation are met in an elegant way.

However, it is not obvious how to accomplish positive fit: clearly, the hole in one of the flaps may not be of use to the Gripper. The knob, on the other hand, may be locked inside a *circular cavity* in the upper gripper blade, when the lover blade is shut. The only drawback with this design might be that its use is highly specific, only applicable to the Bricks. Other objects may possibly be grasped, however, without the form lock property.

4.2.4 Decision and Justification

Since the drawbacks of claws in terms of inverse kinematics are unacceptable, this kind of design was excluded first. Symmetric tools locking in the flap hole could be designed with a high effort, but compared to the gripper style, they do not offer any benefits. Therefore, the decision was made to design a rotation symmetric Gripper with a circular cavity form-locking the knob inside the upper blade.

4.3 Actuator Selection

Having selected the rotation symmetric Gripper design, the actuator is constrained to be linear. The selection criteria of the linear motor mainly consisted of cost, weight, and controllability. Most linear actuators are worm-geared electric motors. Solenoids were not considered as an alternative due to their three issues: comparably low exerted forces, highly varying force/velocity profiles, and very high power consumption in the ON-state.

While the total amount of linear actuators available is enormous, it is not easy to find miniature linear actuators. However, Firgelli[5] produces a series of uniquely light and low-cost linear actuators. The decision was made to purchase one of their L12 Miniature Actuators with the following options:

- Stroke 50 mm: this will allow reasonable clearance of the Gripper blades with respect to the Brick flap (and knob) also considering imprecise positioning.
- Gearing option 210: this high gearing comes with the advantage of the actuator being practically non-backdrivable as well as exerted forces reaching up to approximately 50 N. On the other hand, the maximum speed of just 5 mm/s may be considered a drawback. Compared to the Arm speed, however, that speed appears reasonable.
- Potentiometer position feedback without integrated controller board: this will leave the flexibility for customized control including force control. In contrast to simple limit switch feedback, this will allow driving the Gripper to arbitrary positions potentially even coordinated motion between Gripper and Arm. Unfortunately, there is no encoder position feedback option required for control via the PIC-Servo Boards in the same way as the Arm motors.

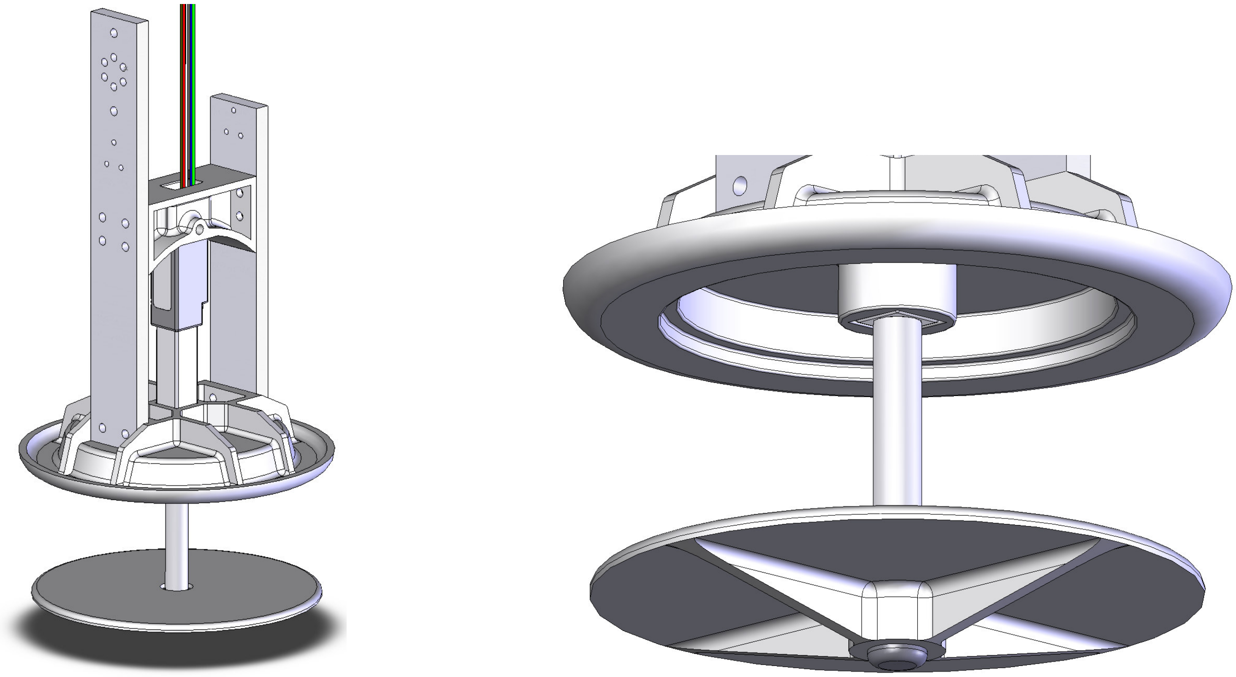
4.4 CAD Modeling

During the design process, it was decided to later manufacture most parts by rapid prototyping: thus high machining costs avoided and minimal limits are imposed on the geometry of the parts, leaving the freedom to optimize the structure.

[5] Firgelli Technologies Inc.,Victoria, BC, Canada

The drawbacks, however, are considerable as well: both the tensile strength as well as the rigidity of the respective plastics are limited compared to aluminum. For this reason, a support structure was built of two simple aluminum bars which are cheap to manufacture.

Figures 8 and 9 show the SolidWorks®CAD drawings of the final design: a motor support is attached to the two aluminum bars which absorbs the reaction forces from the gripping action. The motor is also sideways supported by the upper (fixed) Gripper blade being attached to the support bars. The lower blade is attached to the actuator shaft by a barrel nut allowing rotation around the actuator shaft axis.

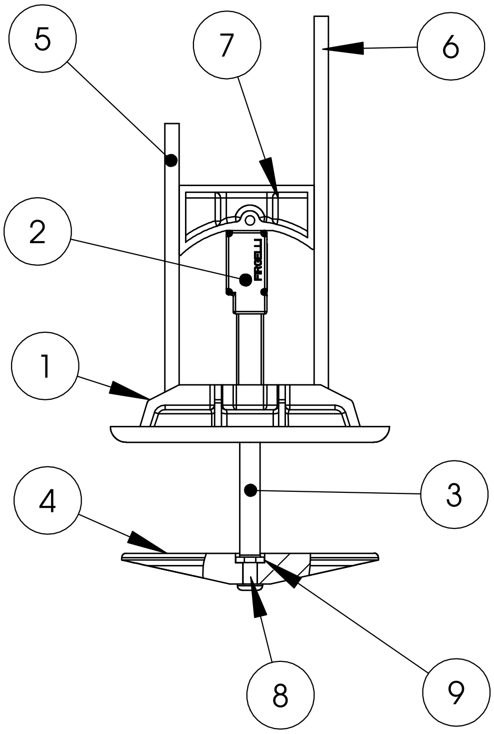


(a) Overall Gripper assembly.

(b) The upper blade features a central cavity form-locking the knob, when the lower blade is shut.

Figure 8: *3D CAD drawings of the Gripper assembly.*

The motor support was designed similar to the shape of an I-beam in order to enhance its bending stiffness. The upper Gripper blade was equipped with bending stiffeners on the outside. Also the lower blade features some stiffeners, but its rigidity was deliberately kept rather low in order to prevent stress concentration at the narrow attachment to the actuator shaft.



(1) Upper Gripper Blade
(2) Firgelli L12 Actuator
(3) Actuator Shaft
(4) Lower Gripper Blade
(5) Left Aluminum Bar
(6) Right Aluminum Bar
(7) Motor Support
(8) Barrel Nut
(9) Washer

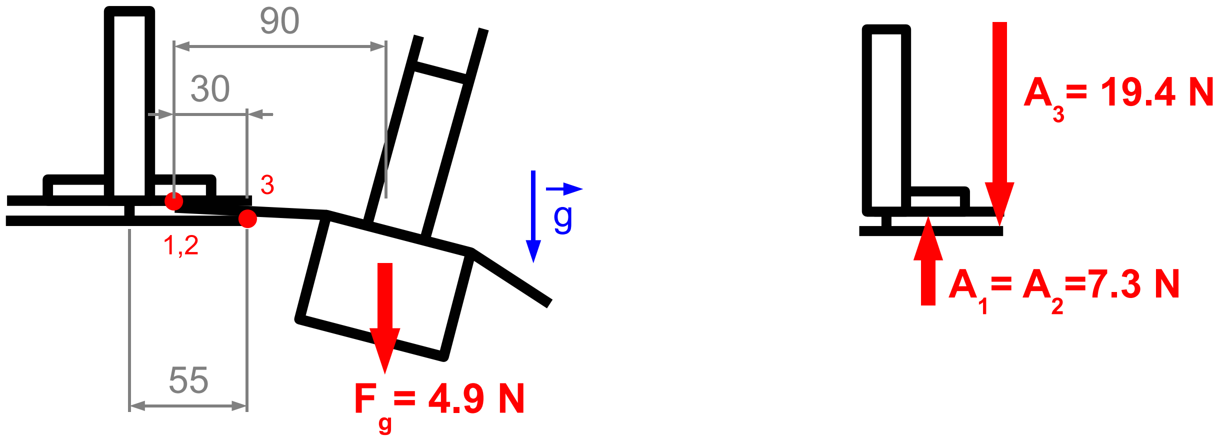
Figure 9: *2D Gripper assembly drawing.*

4.5 Force Estimation and Structural Analysis

Two worst case force scenarios are analyzed: the forces induced by holding the brick in horizontal extension, and the forces experienced during a motor stall caused by an incorrect grasp. In the first scenario (as shown in Figure 10), the assumption is made that three points on the horizontal Gripper blades support the Brick flap: the force to the upper blade is equally shared by the two contact points (1) and (2) while the lower blade supports the flap in one point (3). The numeric values of the respective forces induced to the Gripper are easy to obtain in this particular case.

The described situation also gives insight into the case where closing the Gripper involves lifting the Brick: the reaction force at point (3), $A_3 = 19.4$ N, is then equal to the force at the actuator. Since the specified maximum actuator force measures approximately 55 N, the motor is clearly strong enough to close the Gripper while lifting the Brick.

Another concern with respect to the actuator lies in bending torque at the point where the shaft leaves the housing. The manufacturer specifies a maximum side force at full actuator



(a) Gripper with Brick contacting at three points.

(b) Forces induced to the Gripper blades at the three contact points.

Figure 10: *First load scenario: the Brick weight F_g causes forces at three points A_1, A_2, and A_3.*

extension of 30 N. This corresponds to a torque $M_{s,max} = 1.5$ Nm. Notice that the applied torque measures not much less: $M_s \approx 1.1$ Nm. However, the maximum torque with the shaft not fully extended – which is the case here – will be higher.

In the second worst case scenario that is analyzed, the Gripper blades are assumed to accidentally grip an obstacle. This scenario is most likely to occur when the Gripper is imprecisely positioned and the knob jams. Since the stall force of the actuator measures at maximum 70 N, both upper as well as lower blade may be exposed to 70 N attacking at the respective blade edge in the worst case.

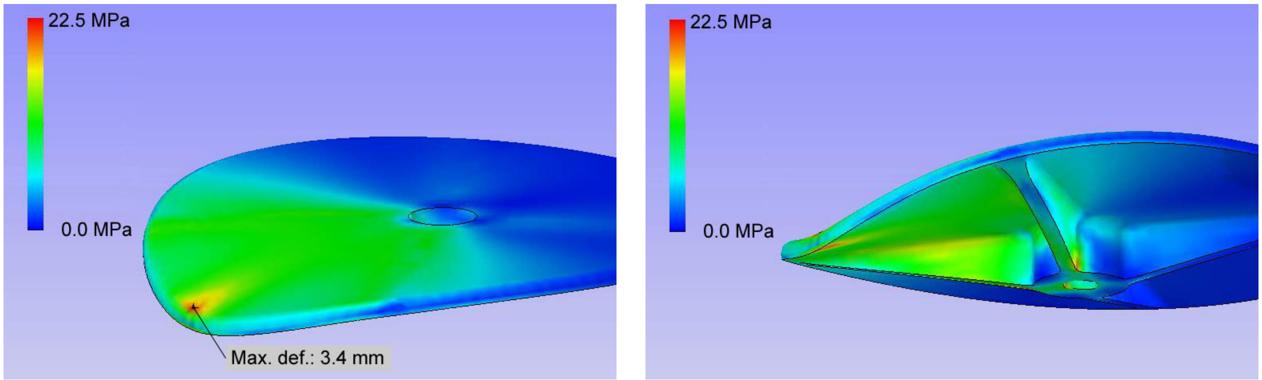
The resulting torque to the actuator shaft would be $M_s = 3.8$ Nm which is clearly too high at full extension with $M_{s,max} = 1.5$ Nm. Therefore it is essential to limit the actuator force while closing the Gripper to approximately 4s0 N just allowing lifting the Brick but protecting the actuator from damage. The following FEM analysis, however, is carried out assuming a maximum force of 70 N since it might be applied (not recommended) when the actuator is retracted.

4.5.1 FEM Analysis

In order to analyze the structure, COSMOS®XPress, an FEM tool built into SolidWorks®was used. Its functionality unfortunately is very limited in terms of specifying boundary conditions: forces may only be uniformly distributed over a surface and only rigid restraints at surfaces are allowed.

First, the aluminum bars are analyzed for the case the Gripper is pointing horizontally: the assumption is made that they are very rigid compared to the parts interconnecting them (motor mount and upper blade), therefore the interconnection is neglected which is a conservative assumption. Even in high bending load cases, such as when the Brick is suddenly adjusting its orientation to gravity, the deformations indeed stay low. Assuming a load of 20 N (i.e. load factor 4) equally partitioned to both bars, their bending deformation measures around 0.1 mm.

For all other parts, FullCure®VeroBlue material was used characterized by a yield strength of about 50 MPa and an elastic modulus of 2.7 GPa.



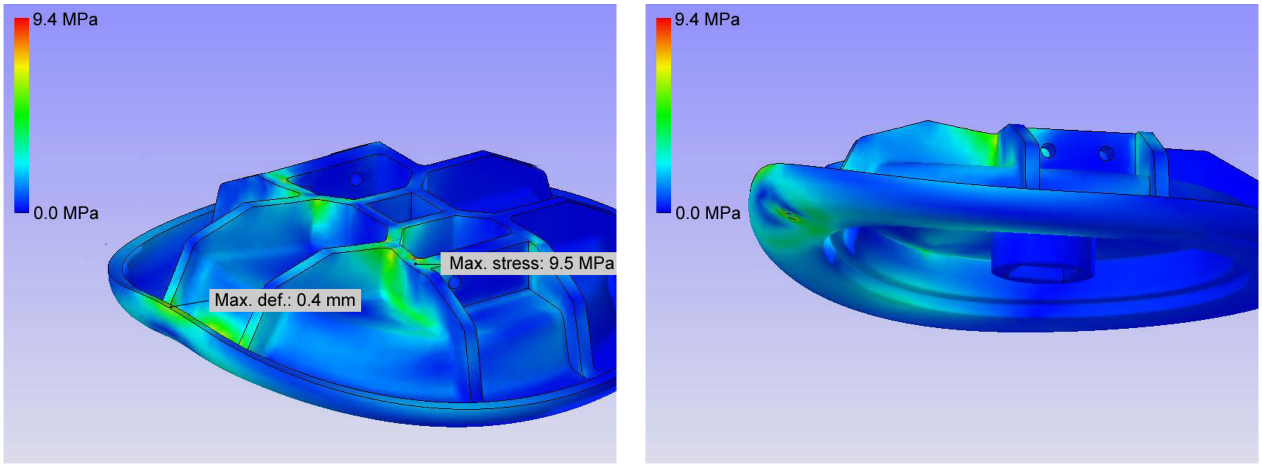
(a) Inner side (top).

(b) Outer side (bottom).

Figure 11: *Stress distribution in the upper Gripper blade with point force at the blade edge between the stiffeners. The fixation hole was constrained to be immovable.*

Figures 11 and 12 show the stress distribution and deformation for the lower and upper blade assuming 70 N applied at their edges. The maximum deformation of the lower blade is considerable: 3.4 mm displacement at 70 N is an extreme case, but also in normal operation yielding forces of up to 20 N, the deformation of the lower blade will be around 1 mm.

Finally, the motor support needs to be analyzed (see Figure 13): the maximum actuator



(a) Top view.

(b) Bottom view.

Figure 12: *Stress distribution in the upper Gripper blade with point force at the blade edge. The fixation to the bars was set immovable.*

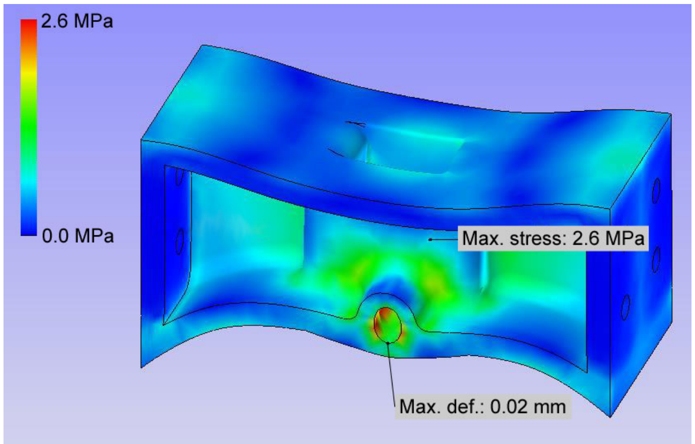


Figure 13: *Stress distribution in the motor support part: the force is applied to the lower half of the fixation hole. The aluminum bar mounting surfaces were set immovable.*

force of 70 N is applied to the fixation hole. It is important that the deformations remain small in order to guarantee precise control. In fact, the design shows a maximum displacement of only 0.02 mm.

4.6 Gripper Control

In order to operate the Gripper in a flexible way, it needs position control rather than just 'open' and 'close' commands. Furthermore, force feedback helps detect failures and allows control of applied forces. Both of these capabilities were implemented.

4.6.1 Control Architecture

The major difficulty in terms of writing a suitable Gripper driver is imposed by the fact that the actuator only features a potentiometer position sensor instead of an encoder. Therefore, it may not be controlled in a local control-loop by a PIC-Servo Board as the Arm joints.

The decision was made to extend the Gripper control-loop through the Linux box. The sample time t_s will therefore be comparably long as well as varying. On the other hand, control algorithms of arbitrary complexness are possible rather than PID control provided by the PIC-Servo Board. In particular, model based filtering of the position signal and force control may be implemented.

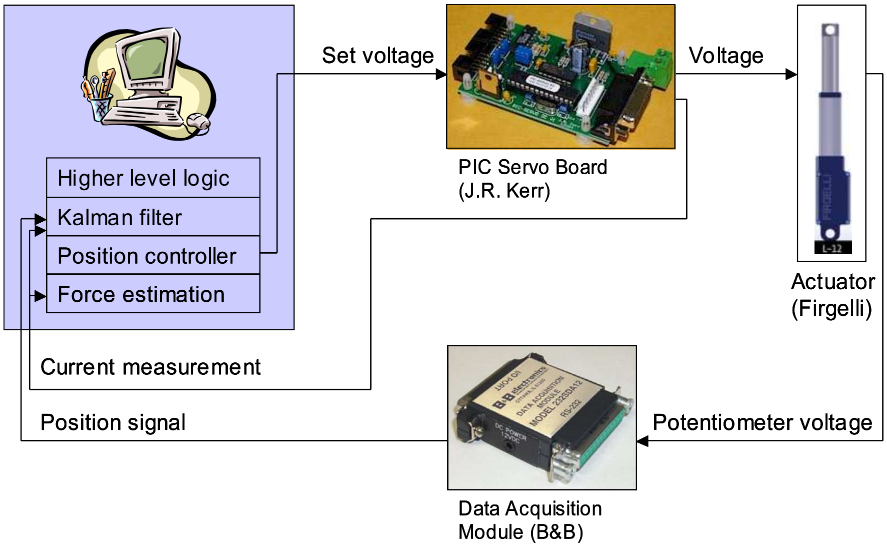


Figure 14: *Gripper control architecture.*

The overall control architecture of the Gripper is shown in Figure 14: in terms of position control, the PIC-Servo Board is used only to set the Gripper motor voltage. The position is sensed via the same B&B data acquisition module as the one reading the Arm potentiometers at initialization. Applied forces are sensed indirectly by measuring the current draw on the PIC-Servo Boards.

4.6.2 Position Control

Fast stall detection requires a low noise position signal. Unfortunately, the position measurement proved to be subjected to a significant noise level. By replacing the original wires to the potentiometer with shielded ones, the noise could be reduced. Nonetheless, the decision was made to implement a Kalman filter to smooth the signal without introducing delay: all low-pass or averaging filters that would be easy to implement introduce too much delay given the long sample time $t_s \approx 20$ ms.

The entry for a discrete variable sample time (Δt) model for the actuator position x is straightforward where the actuator speed u is seen as input to the system:

$$x_k = x_{k-1} + \Delta t_k \cdot u_k \tag{4.2}$$

Notice that the assumption is implicitly made that the speed u may be set directly, i.e. neglecting inertial effects. This is justified by the fact that the motor is highly geared. Determining the actuator speed u as a function of the terminal voltage V requires a (static) motor model. A simplified electric circuit is drawn in Figure 15 where the motor inductance is neglected and only the winding resistance R is considered. The induced (back electromagnetic force) voltage V_{emf} is seen as the input generating a corresponding motor speed in this quasi-static model (while the true physical causality is opposite).

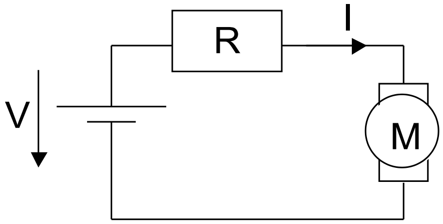


Figure 15: *Simplified Gripper motor electric circuit.*

$$V_{emf} = V - I \cdot R \tag{4.3}$$

The current I is unknown and depends on the force applied to the actuator. Since I is sensed, this measurement is used to adapt the model. Let k_v be the overall motor constant; the actuator speed u – which is the input to the dynamic system (4.2) – may now be expressed as:

$$u_k = \frac{1}{k_v} (V - I \cdot R) \tag{4.4}$$

Next, modeling uncertainty w as well as measurement noise v are added to the model:

$$x_k = x_{k-1} + \Delta t_k \cdot u_k + w_{k-1}, \ w_{k-1} \sim \mathcal{N}(0, q_{k-1}) \tag{4.5}$$

$$y_k = x_k + v_k, \ v_k \sim \mathcal{N}(0, r_k) \tag{4.6}$$

The overall model uncertainty was mainly determined by an error propagation of the uncertainty estimates on the different parameters. Notice that some of the model uncertainty is even caused by current measurement error. The measurement (y) error is estimated by measuring the variance of the signal.

Since a positive correlation between position measurement noise and applied current had been identified, r_k was modeled as a heuristic function of I.

If the motor is stalled, the output voltage is reduced by the PIC-Servo Board to an unknown amount such that the current I is not exceeding a specified limit. This has the unfortunate consequence that the model becomes very inaccurate in that critical situation. Consequently, q_{k-1} must be increased drastically in that case.

Now, the Kalman prediction and update steps may be performed (see [3]).

Figure 16 shows a comparison between the measured position signal, an average filter and the Kalman filtered signal.

In terms of feedback control, a simple P-controller was chosen. In order to reject steady state error caused by friction, an I-component would theoretically be necessary, too. But this also requires implementing anti-reset-windup. Due to friction dead-band compensation, the steady state error proved to be very small compared to the required positioning precision, therefore the controller was left simple with proportional gain only.

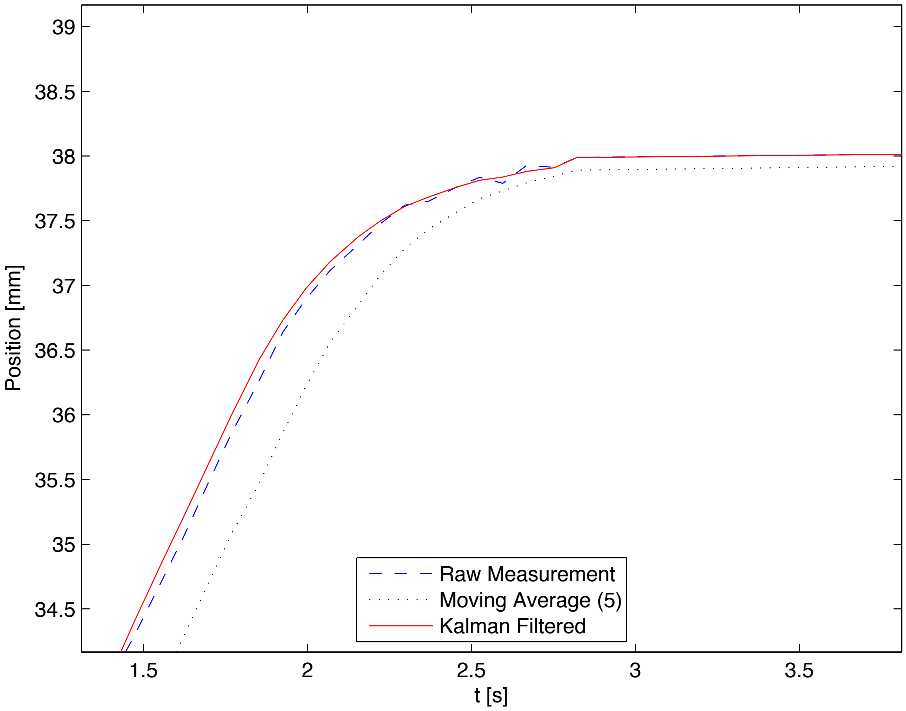


Figure 16: *Comparison between moving average and Kalman filtered Gripper position signal.*

4.6.3 Force Estimation and Control

The equality of electric power consumed and exerted with the Gripper yields the relationship between force and current:

$$F = k_v \cdot I \tag{4.7}$$

Notice that this force is of rather theoretical nature, since it also includes friction which is considerable given the high motor gear ratio (even involving a worm shaft). The friction force (assumed to be a constant) is however easy to determine by measuring the minimum current necessary to move the motor. The absolute value of the force at the actuator F_A may be expressed as:

$$F_A = |k_v \cdot I| - F_{fr}, \;\; F_{fr} > 0 \text{ N} \tag{4.8}$$

The overall control logic including force control is performed as follows: while the Gripper is in position control, the applied force is watched and being limited to a specified maximum. There are two maximum forces to be specified: one for 'underway' and a second one for the position target region. If a stalled stop is detected in the target region, the Gripper will still return *success* but with the information that the force was applied. In case of a stop underway, the Gripper will obviously return *failure*.

Notice that the resolution of the current measurement is only 8 bit and not very accurate. Furthermore, the settable force limit is only 7 bit resolution. Therefore the applied and measured forces will be neither precise nor accurate.

5 Arm Control

The following Section describes the Arm control as well as related issues: basically, all the necessary steps are covered of how to turn a sequence of waypoints into a trajectory and executing it with the Arm.

5.1 Control Architecture

Since precise path following is highly desired, the decision was made to use the PIC-Servo Boards in path point mode (PPM). The overall Arm control architecture is depicted in Figure 17. Running PPM requires generating a trajectory that is passed to the Boards which run an internal PID position control loop with the Arm servo motors. The trajectory consists of waypoints that will be visited chronologically with a constant time interval T (set to 33 ms in our case). Sections 5.5 and 5.6 cover this non-trivial step for the case of joint-space and Cartesian space trajectory generation, respectively.

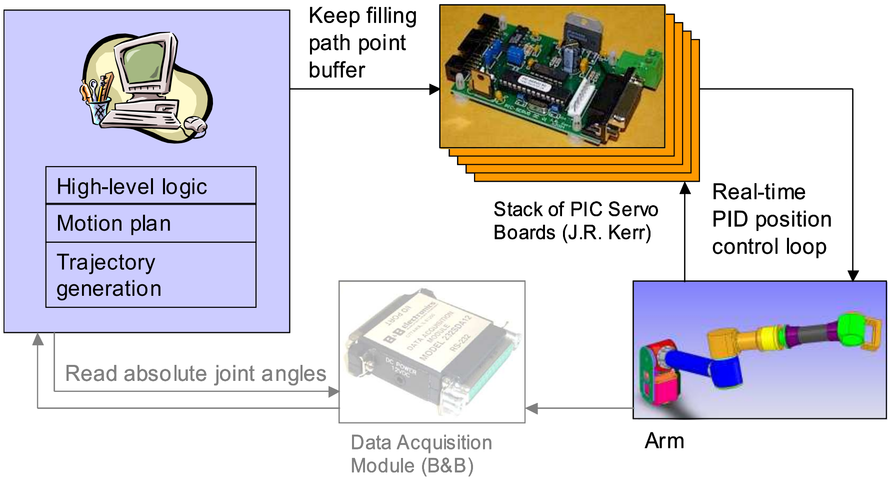

Keep filling path point buffer
High-level logic
Motion plan
Trajectory generation
Stack of PIC Servo Boards (J.R. Kerr)
Real-time PID position control loop
Read absolute joint angles
Data Acquisition Module (B&B)
Arm

Figure 17: *Arm control architecture.*

As the encoders only provide information on joint angle changes after initialization, the absolute joint positions need to be found: therefore, the joint position potentiometers are read, however, *only* at Arm initialization. The respective A/D conversion leads to a resolution of 0.27° which unfortunately causes a considerable amount of the Arm positioning imprecision.

5.2 Kinematics

In order to generate a trajectory in joint space, the kinematic chain needs to be analyzed. Figure 18 introduces the different coordinate frames needed as well as the joint angle and link length definitions. Tables 1 and 2 list the numeric values of the link lengths and joint angle limits, respectively.

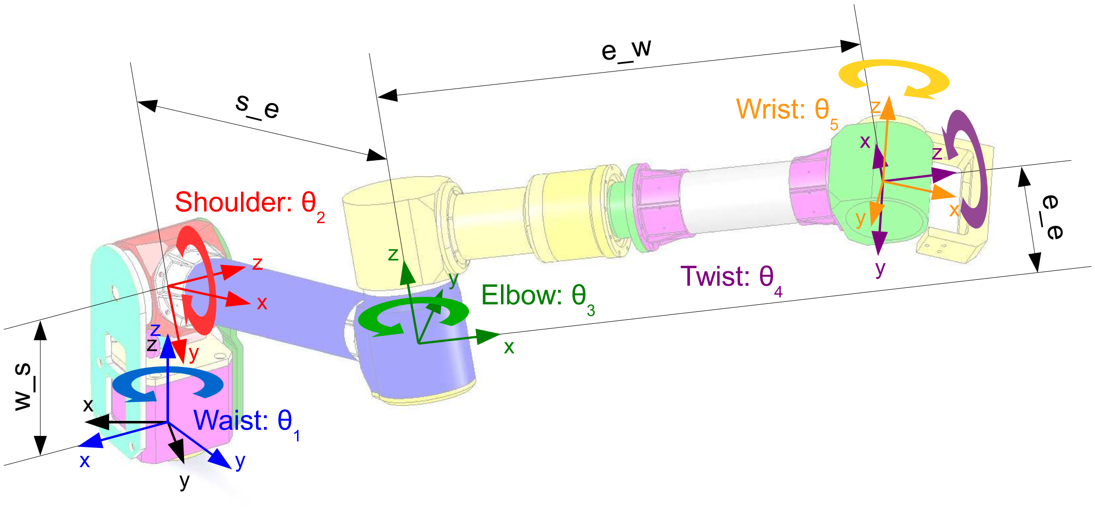


Figure 18: *Coordinate frame definition and respective transformations in the kinematic chain of the arm.*

Parameter	Description	Value
w_s	Waist-Shoulder	0.112 m
s_e	Shoulder-Elbow(bottom)	0.325 m
e_e	Elbow(bottom)-Elbow(top)	0.0692 m
e_w	Elbow(top)-Wrist	0.311 m

Table 1: *Link dimensions.*

5.2.1 Forward Kinematics

The homogeneous transformation matrix T_{05} transforming coordinates from the Wrist frame (5) to the Base frame (0) is derived in Appendix A.1.

Notice that the Twist frame as well as the Wrist frame definitions do not correspond to the Denavitt-Hartenberg convention. The deviation of the Twist frame yields the advantage

Joint name	Index	Lower limit	Upper limit
Waist	1	-3.14 rad	3.14 rad
Shoulder	2	-1.5 rad	0.4 rad
Elbow	3	-3.14 rad	3.14 rad
Twist	4	-3.14 rad	3.14 rad
Wrist	5	-2.0 rad	2.0 rad

Table 2: *Joint limits.*

that the last two transformations do not require translation anymore, so the decoupling of position and orientation is already being introduced at the level of intermediate coordinate frames. The Wrist frame alternate definition causes it to be aligned with the Base frame in zero configuration (all joint angles at 0 rad) which is a convenient property.

The Wrist position is obtained as:

$$\overrightarrow{OW} = \begin{pmatrix} T_{05}(1,4) \\ T_{05}(2,4) \\ T_{05}(3,4) \end{pmatrix} \tag{5.1}$$

In terms of orientation, the tool pointing vector $\vec{p}$ is considered which is the Wrist frame x-direction base vector:

$$\vec{p} = \begin{pmatrix} T_{05}(1,1) \\ T_{05}(2,1) \\ T_{05}(3,1) \end{pmatrix} \tag{5.2}$$

5.2.2 Inverse Kinematics

The inverse kinematics find the 5 joint angles given a desired Wrist location $\overrightarrow{OW}$ as well as pointing direction $\vec{p}$. The respective equations are derived in A.2 using a geometric approach. Given a desired position and pointing direction within the workspace as described in detail in the next Section 5.3, there are typically 4 solutions originating from the following two ambiguities:

- **Right and left Elbow**
  In order to reach a Wrist position, the Elbow may either be bent to the right or to the left.

- **Orientation ambiguity**
  There are two solutions for the Wrist and the Twist to achieve the desired orientation: it is (almost) every time possible to rotate the Twist ±180° and adjust the Wrist accordingly in order to obtain the exact same orientation.

The Elbow ambiguity is eliminated by restricting solutions to positive Elbow angles θ_3 (right Elbow). This restriction prevents dealing with the case of the Arm passing through a singularity from a right Elbow solution to a left Elbow solution or obversely.

Coping with the rotational ambiguity requires some more attention: in order to always choose the more reasonable solution, the start configuration is being passed to the inverse kinematics function, such that it can return the solution with the least change in joint angles (i.e.the closer solution).

5.3 Workspace Analysis

It is essential to have knowledge about the Arm workspace in order to determine at what positions and orientations Bricks may be picked. Furthermore, analysis of the workspace will allow comparing different Arm mounting alternatives. Finally, it will give a qualitative understanding of situations where Cartesian space trajectory generation is doomed to failure.

Naively, the assumption might be made that the reachable workspace has the geometry of a half sphere, since there is obviously control of the Arm radius with the Elbow as well as latitude and longitude control by Shoulder and waist, respectively. Unfortunately, however, with increasing Elbow bending the latitude control becomes limited. This leads to an unreachable sphere-like region on top of the Waist. Figure 19 illustrates the reachable workspace qualitatively.

5.3.1 Comparison of Different Mounting Alternatives

Two different mounting alternatives are being compared while also considering the Gripper orientation to be constrained perpendicular to the ground. This approximately corresponds to the pose at which Bricks are being grasped.

1. Arm mounted up-side down as on K9: the base frame z-axis points straight down.

2. Arm mounted sideways: this is the originally intended mounting for k10.

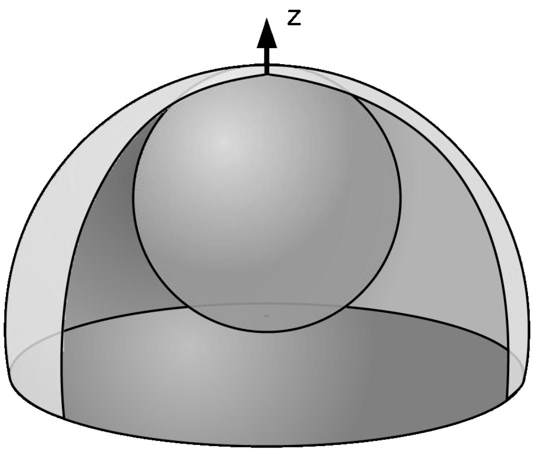


Figure 19: *Qualitative insight into the reachable workspace.*

Figure 20 compares the respective dexterous workspaces assuming a mounting approximately 0.4 m above the ground. Clearly, the sideways mounted Arm has a much more limited workspace, mostly caused by the central non-reachable space. Unfortunately, this non-reachable area would lie right in front of the rover (i.e.in the field of view of its stereo camera), therefore Bricks may not be grasped there.

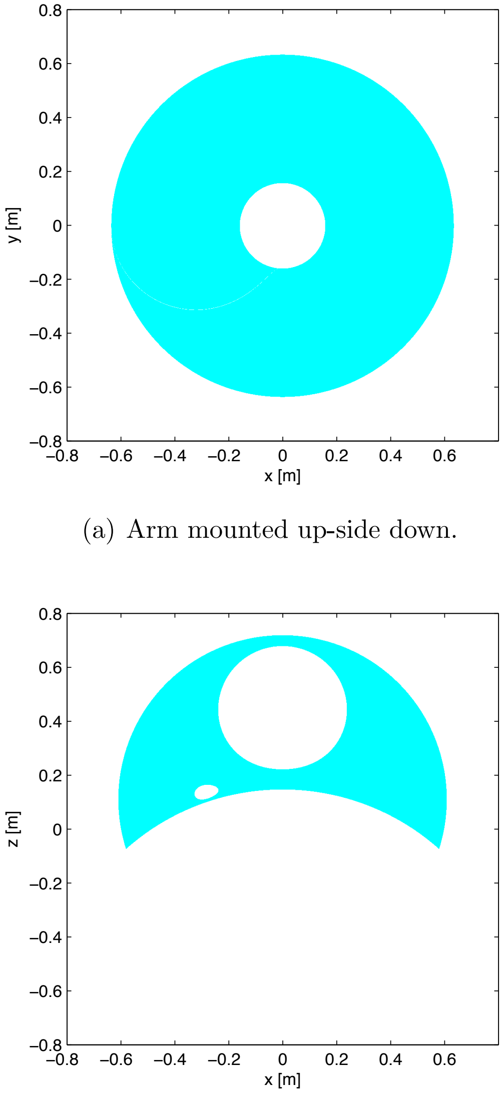
Another massive drawback of the sideways mounting alternative lies in the fact that the Waist motor needs to work against gravity, in particular with the Brick attached. The problem lies in the fact that the Waist motor was not designed for high torques, such that it would have to be equipped with a higher gear-ratio in order to prevent back-driving and stalling.

Consequently, the Arm will be mounted up-side down in the test setup and the same mounting is strongly recommended for a later integration of the Arm with K10.

5.4 Gray-Box Model of the Dynamics

Modeling the full dynamics of the Arm is strenuous and requires identification of many parameters such as moments of inertia. Fortunately, the motors are highly geared such that inertial effects may be neglected to a certain extent. Hence, the dynamics are governed by speed limits of the individual motors. Table 3 provides the numeric values.

Notice that the speed limits could potentially be optimized by making them adaptive to the current state of the Arm: the torques to each joint may be calculated using the Jacobian,



(a) Arm mounted up-side down.

(b) Arm mounted side-wise

Figure 20: *Comparison of cuts through the workspaces corresponding to the two Arm mounting alternatives. The coordinates reflect Wrist positions 0.2 m above ground with the Gripper pointing straight down.*

the weights of the links and the Brick as well as the direction of the gravity vector. This would, however, complicate the trajectory generation dramatically.

In order to account for start and stop acceleration being finite, limits are assumed that are

Joint name	Index	Speed Limit
Waist	1	0.2 rad/s
Shoulder	2	0.01 rad/s
Elbow	3	0.03 rad/s
Twist	4	0.03 rad/s
Wrist	5	0.1 rad/s

Table 3: *Speed limits.*

constant and therefore absolutely independent from the state of the Arm. These acceleration limits as shown in Table 4 do not reflect physical limits but are set low enough to not violate their true values in any state of the Arm even with Brick attached.

Joint name	Index	Acceleration Limit
Waist	1	0.2 rad/s^2
Shoulder	2	0.2 rad/s^2
Elbow	3	0.2 rad/s^2
Twist	4	0.2 rad/s^2
Wrist	5	0.2 rad/s^2

Table 4: *Acceleration limits.*

5.5 Joint Space Trajectory Generation

Joint space trajectories will be needed in conjunction with motion planning: given a sequence of waypoints (or milestones) as a set of the five joint angles, they need to be interconnected by linear segments in joint space such that neither the velocity limits nor the acceleration limits are violated.

Obviously, a sequence of strictly linear segments would require the Arm to stop at each waypoint, otherwise the acceleration would be infinite. Since this is not practical, some deviation to the linear segments is allowed near the waypoints. Notice that there exists no absolute time optimality for the generated trajectory, since the execution time depends on the specified allowed deviation at the waypoints.

The trajectory is computed as follows: first, the acceleration constraint is ignored and the waypoints are spaced in time such that the speed limits are just not violated for any of the five joints. Secondly, parabolic blends are inserted at the waypoints yielding exactly the maximum allowed acceleration for each of the joints. Notice that the generated trajectory is time optimal in the sense of there not existing a faster trajectory yielding the same path. For very short segments, inserting a blend at both ends might not be feasible; in that case, the linear segment needs adaption. Figure 21 illustrates these steps for one of the coordinated five joints.

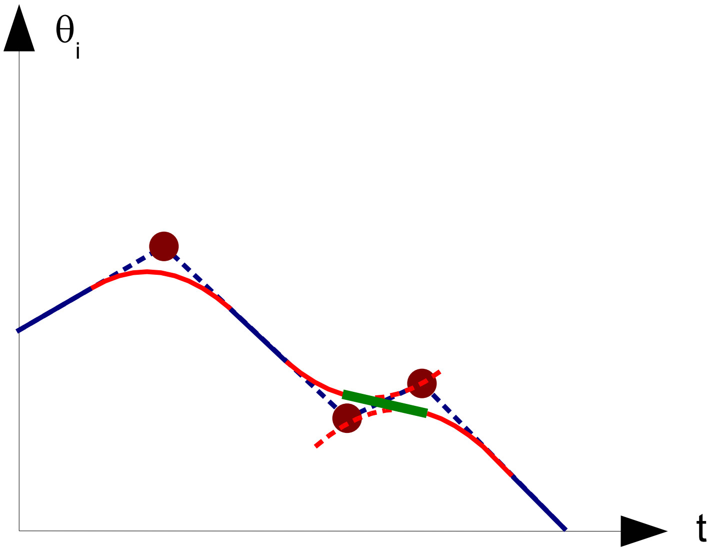


Figure 21: *Example illustration of the generation of the i^{th} joint trajectory: on the right, the linear segment needed to be adjusted (bold, green) such that it became a tangent to both blends. Notice that all blends are of the same curvature corresponding to the maximum acceleration of the particular joint; the linear segments, however, are not necessarily of maximum steepness, since one of the other four joints might hold the active speed limit constraint.*

5.6 Cartesian Space Trajectory Generation

In contrast to joint space trajectory generation as described in the previous Section 5.5, the goal is now to connect a sequence of waypoints by straight lines in Cartesian space. The waypoints are in this case given as a list of Wrist positions $\overrightarrow{OW}$ and Wrist x-axes pointing directions $\vec{p}$ Again, the goal is to generate a reasonably fast trajectory obeying speed and acceleration limits – which proved to be a non-trivial problem.

5.6.1 Underlying Theory

The trajectory $\theta\left(t\right)=\left[\theta_1\left(t\right),\theta_2\left(t\right),\theta_3\left(t\right),\theta_4\left(t\right),\theta_5\left(t\right)\right]$ ‘T will be obtained by numerical integration of the joint speeds $\dot{\theta}$ from the start configuration θ_0 with constant step-size T (i.e.the trajectory discretization time):

$$\theta\left(t\right)=\theta_0+\int_0^t\dot{\theta}\left(\tau\right)d\tau \tag{5.3}$$

This allows locally obeying the constraints in terms of speed as well as acceleration.

The relation between Cartesian velocity as well as rotation and joint velocities is given by the Jacobians J. Due to the separation of Wrist position and orientation, the Wrist velocity $\vec{v}_w$ is determined by the three first joint speeds only. Therefore we can write:

$$\vec{v}_w=J_{pos}(\theta_1,\theta_2,\theta_3)\cdot\begin{bmatrix}\dot{\theta}_1\\\dot{\theta}_2\\\dot{\theta}_3\end{bmatrix}\ ,\ J_{pos}\in\mathbb{R}^{3\times3} \tag{5.4}$$

For the i^{th} linear segment, the desired velocity $\vec{v}_{i,d}$ should drive the Wrist into direction to the next waypoint $\overrightarrow{OW}_i$:

$$\vec{v}_{i,d}=C_1\cdot\left(\overrightarrow{OW}_i-\overrightarrow{OW}_{i-1}\right) \tag{5.5}$$

Now equation 5.4 may be solved for $\left[\dot{\theta}_1,\dot{\theta}_2,\dot{\theta}_3\right]^T$ while choosing the unknown constant C_1 later such that both speed and acceleration constraints are not violated for any of the *five* joint speeds.

Next, the rotation part of the problem may be approached:

$$\vec{\omega}_w=J_{rot}(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5)\cdot\begin{bmatrix}\dot{\theta}_1\\\dot{\theta}_2\\\dot{\theta}_3\\\dot{\theta}_4\\\dot{\theta}_5\end{bmatrix}\ ,\ J_{rot}\in\mathbb{R}^{3\times5} \tag{5.6}$$

The desired Wrist rotation in the i^{th} segment is obtained as:

$$\vec{\omega}_{i,d} = C_2 \cdot (\vec{p}_{i-1} \times \vec{p}_i) + \dot{\epsilon} \cdot \vec{p} \tag{5.7}$$

Since there is no control for rotations around $\vec{p}$, an arbitrary rotation of magnitude $\dot{\epsilon}$ must is allowed: now, equation 5.6 may be solved for $\left[\dot{\theta}_4, \dot{\theta}_5, \dot{\epsilon}\right]^T$. The constant C_2 may, however, not be chosen, since rotation of $\vec{p}$ must be coordinated with the Wrist velocity $\vec{v}_w$; the infinitesimally traveled distance ds and the traveled angle $d\alpha$ need to be in the right relation to each-other:

$$d\alpha \cdot \Delta s = ds \cdot \Delta\alpha \Leftrightarrow C_2 \cdot |\vec{p}_{i-1} \times \vec{p}_i| = C_1 \cdot \angle(\vec{p}_i, \vec{p}_{i-1}) \tag{5.8}$$

5.6.2 Edge Smoothing

Again, assembling a path as a sequence of exact linear segments would result in required full stops at each waypoint. Therefore, the edges need to be smoothed, i.e. the path is preprocessed. In several iterations, the edges of the position path are chamfered such that they approximate a circular arc (see Figure 22): every waypoint m_{k-1} is replaced by two waypoints at the k^{th} iteration such that the new waypoints lie *both* at a distance C from m_{k-1} on the original legs of length $l_{left,k-1}$ and $l_{right,k-1}$. The following choice of C will force the circular

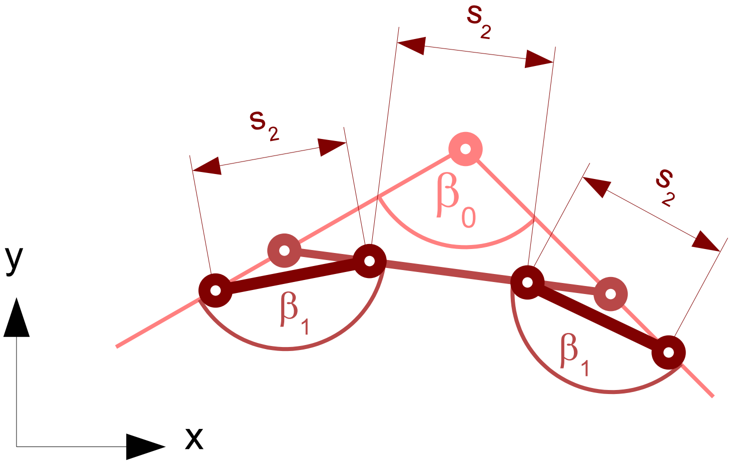


Figure 22: *Example of the iterative chamfering of the Cartesian space path applying two iterations $k = 1$ and $k = 2$.*

arc approximation, i.e. the chamfer lengths at one waypoint become all equal to a certain value s_k:

$$C_k = \min \left\{ C_{max}, \min \left\{ l_{left,k-1}, l_{right,k-1} \right\} \cdot \frac{1}{2\left(1 + \sin \frac{\beta_{k-1}}{2}\right)} \right\} \tag{5.9}$$

At the same time as the position path is smoothed, the pointing vector $\vec{p}$ path is smoothed as well: for each of the new inserted or deleted waypoint m, the corresponding $\vec{p}$ is found and inserted or deleted, respectively.

Notice that the described iteration will not yield equal arc radius' at each corner; but the maximum deviation of the smoothed path to each of the original waypoints is bounded to C_{max}.

5.6.3 Speed Adjustments and Numeric Integration

Now, the major challenge is to turn the smoothed path into its time-optimal trajectory by numeric integration.

Complying with the speed limits of the individual joints is not difficult to accomplish by scaling the calculated vector of desired joint speeds $\dot{\theta} \in \mathbb{R}^5$ (calculation explained in Section 5.6.1) at each step of the integration, such that the velocity constraints are just met.

Complying with the acceleration constraint, however, is more difficult to achieve, because $\dot{\theta}$ may need to be adjusted in order to not violate acceleration constraints in *future* integration steps (looking ahead required). The acceleration limit condition at the r^{th} integration step of step-size T is formulated as:

$$\frac{\left|\dot{\theta}_{j,r} - \dot{\theta}_{j,r-1}\right|}{T} \leq \ddot{\theta}_j, \; j = 1, ..., 5 \tag{5.10}$$

This keeps the maximum deviation small enough. The following procedure is applied in order to find out about the maximum speeds allowed at each of the i waypoints (of the smoothed path):

1. Determine the maximum speed possible (only considering the speed limits) immediately before ($\dot{\theta_-}$) and after ($\dot{\theta_+}$) the waypoint.

2. The velocity change in case of full speed at this waypoint is $\Delta\dot{\theta} = \dot{\theta_+} - \dot{\theta_-}$. Compare

this to the maximum allowed velocity change vector *of the same direction* according to (5.10). The outcome is a *relative maximum speed with respect to maximum speed* assigned to each waypoint.

The algorithm finally executing the numeric integration has to look as follows for the r^{th} step:

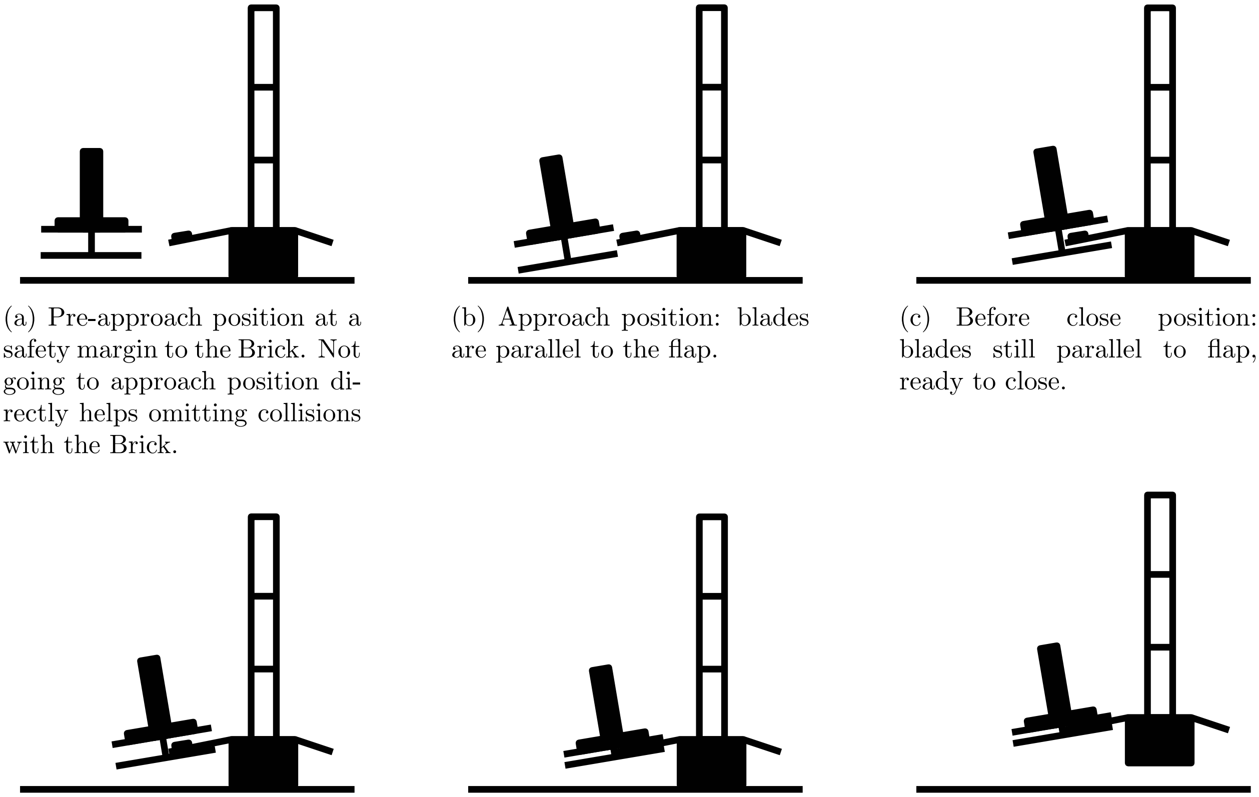
1. Calculate the desired joint speed direction $\dot{\theta}_d(\vec{v}_d, \vec{\omega}_d, \theta_{r-1}) \in \mathbb{R}^5$
2. Check the relative speeds on a hypothetic full acceleration slow-down along the path until full stop: are there any violations to any of the assigned maximum relative speeds at any of the waypoints passed during that acceleration?
    - If yes, set $\left|\left|\dot{\theta}_r\right|\right|$ as small as possible (complying with acceleration and speed limits) into the calculated desired direction (maximum deceleration).
    - If no, set $\left|\left|\dot{\theta}_r\right|\right|$ as high as possible (complying with acceleration and speed limits) into the calculated desired direction (full speed or maximum acceleration).
3. Find the closest ideal position, calculate its corresponding orientation, and apply the inverse kinematics; this yields the ideal joint angles θ_d. Reject the error by adding a proportional control output $k_p \cdot (\theta_d - \theta_{r-1})$ to $\dot{\theta}_r$.
4. Perform the integration: $\theta_r = \theta_{r-1} + T \cdot \dot{\theta}_r$.
5. Append θ_r to the trajectory.

5.7 Pick-Up without Motion Planning

Setting up a pick-up procedure without the capability of planning the motion requires predefining the entire paths. Notice that collision detection is not performed at this stage, since it constitutes a major challenge faced and solved in conjunction with the motion planning implementation as addressed in Section 7.

1. The approach to the Brick may be performed either as a joint space trajectory or as a Cartesian space trajectory with an inserted via-point to avoid the central non-reachable space.

2. The grasping procedure as depicted in Figure 23 is being executed as a Cartesian space trajectory.



(a) Pre-approach position at a safety margin to the Brick. Not going to approach position directly helps omitting collisions with the Brick.

(b) Approach position: blades are parallel to the flap.

(c) Before close position: blades still parallel to flap, ready to close.

(d) The Gripper is partially closed in order to avoid collision with the lower blade and the ground later.

(e) Closed Gripper position: simultaneously to closing the Gripper, the Wrist was moved downward.

(f) The Brick is lifted to a safe distance from the ground.

Figure 23: *Grasping procedure of the Gripper/Arm.*

3. The Gripper pointing direction $\vec{p}$ is turned into the horizontal plane letting the Brick adjust itself to gravity and hang down in the Gripper. This motion is characterized by a Cartesian space path in order to avoid collisions between the Brick and the ground in this delicate phase.

4. Finally, the Arm is moved back to its carry position, again, either by Cartesian trajectory with via-point or directly by following a joint-space trajectory.

6 Determining Relay Position and Attitude using Stereo Vision

Equipping the test setup with a stereo camera offers several important capabilities: on the one hand, it may be used to identify key-points of Bricks in both left and right image followed by triangulation of the points and determining the position as well as the orientation of the Bricks.

Furthermore, it will allow full terrain reconstruction using IRG's *Stereo Pipeline* software: the generated environment mesh would then serve as one of the inputs to the motion planner. Theoretically, the Brick could also be identified in the stereo-reconstructed terrain. Due to the brick featuring a uniform black surface without any texture, this kind of Brick localization was supposed to be of low precision. Also, comparing the reconstructed terrain to a 3D Brick model is a non-trivial and computationally expensive step. Therefore this approach will not be further discussed.

6.1 Camera Calibration and Model

The MATLAB® Camera Calibration Toolbox allows an efficient calibration of the camera pair: first, the intrinsic parameters of the individual cameras are found. As a second step, the stereo calibration is performed which allows refining both camera models as well as finding the extrinsic parameters (at least relative to one of the cameras) within one step.

The camera models are stored in the form of Tsai's camera model [11] [12].

Finding the absolute position and orientation of the stereo pair with respect to the Arm base frame had to be delayed until after the implementation of triangulation: once points may be identified in 3D space with respect to the left camera frame, this last calibration step may be performed.

6.2 Integration of Vision Workbench

The found Tsai camera calibration is used to initialize Vision Workbench (VW)[6] pinhole camera models: these camera model objects allow both *forward projection* of points into the image plane as well as finding the *line of sight* of points in the image.

6.3 User Interaction

The test setup and workflow require human interaction of the following form: the user needs to select four points painted on the Brick flaps in both images, such that they may be triangulated. In the future, finding these key-points can be automated using state-of-the-art vision techniques.

The required human interaction necessitated a graphical user interface (GUI). Originally, it was given two capabilities:

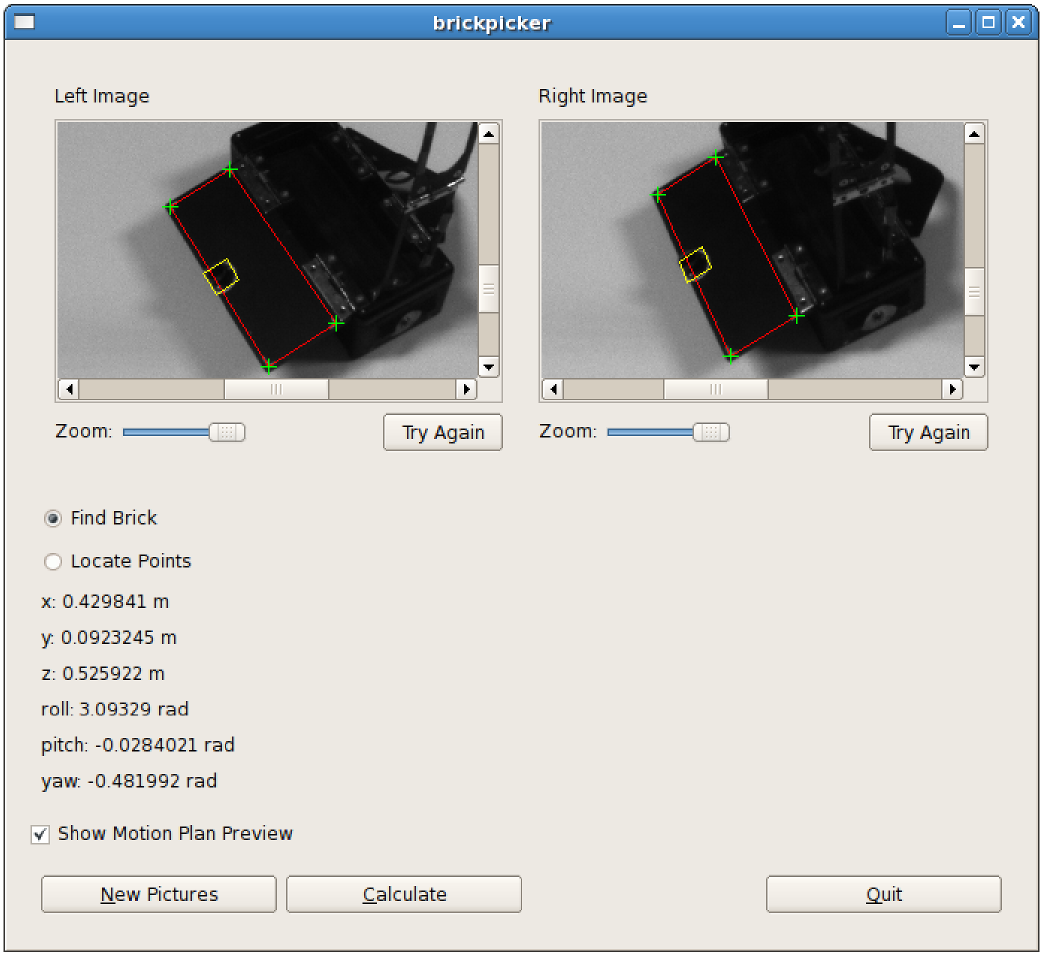
- Localizing arbitrary points for testing as well as calibration of the absolute position and orientation of the stereo camera frame.
- Localizing the Brick: the positions of four points on the Brick flap are found allowing to estimate the flap plane and therefore also the Brick orientation and position.

The GUI as programmed using Trolltec® QT® 4 is shown in Figure 24. The user needs to select four points in the same order in both images, allowing the triangulation of those points and finding the Brick position and orientation. As a verification for the user, the knob of the Brick flap is forward-projected into both images. Additionally, the position as well as roll, pitch and yaw of the Brick are displayed numerically.

6.4 Triangulation and Brick Localization

Assuming the image coordinates of a set of points in 3D space are given for both the left and the right image, the coordinates of this point in space may be found, i.e.*triangulated* [4].

[6] VW is a general purpose image processing and computer vision library developed by the Autonomous Systems and Robotics (ASR) Area in the Intelligent Systems Division at the NASA Ames Research Center. It has been publicly released under the terms of the NASA Open Source Software Agreement.
Web: http://ti.arc.nasa.gov/projects/visionworkbench/



● Find Brick

○ Locate Points

x: 0.429841 m

y: 0.0923245 m

z: 0.525922 m

roll: 3.09329 rad

pitch: -0.0284021 rad

yaw: -0.481992 rad

☑ Show Motion Plan Preview

Figure 24: *Brick pick GUI after the user had selected the points and the Brick was localized.*

Since the image plane coordinates will always be affected by errors, the lines of sight generally will neither intersect with each-other nor with the true 3D location of the respective point. Therefore, the *mid-point method* was used to estimate the point location: hereby, the minimal Euclidean distance between the two lines of sight is found and the point is assumed in the middle of that distance.

After the points (position vectors) $\overrightarrow{OP_1}$, $\overrightarrow{OP_2}$, $\overrightarrow{OP_3}$, and $\overrightarrow{OP_4}$ are found, the Brick may be localized. The points are to be selected in mathematically positive sense when looking on the flap and start with the point closest to the antenna. Since there is redundant information (4 instead of minimally 3 points), there was some averaging applied which increases the robustness. First, the center c of the flap plane is found:

$$c = \frac{\overrightarrow{OP_1} + \overrightarrow{OP_2} + \overrightarrow{OP_3} + \overrightarrow{OP_4}}{4} \tag{6.1}$$

The base vectors of the Brick flap frame (index f) are determined in the following way:

$$e_{x,f} = \frac{\left(\overrightarrow{OP_2} - \overrightarrow{OP_1}\right) + \left(\overrightarrow{OP_3} - \overrightarrow{OP_4}\right)}{\left|\left(\overrightarrow{OP_2} - \overrightarrow{OP_1}\right) + \left(\overrightarrow{OP_3} - \overrightarrow{OP_4}\right)\right|} \tag{6.2}$$

$$e_{z,f} = \frac{e_{x,f} \times \left(\left(\overrightarrow{OP_3} - \overrightarrow{OP_2}\right) + \left(\overrightarrow{OP_1} - \overrightarrow{OP_4}\right)\right)}{\left|e_{x,f} \times \left(\left(\overrightarrow{OP_3} - \overrightarrow{OP_2}\right) + \left(\overrightarrow{OP_1} - \overrightarrow{OP_4}\right)\right)\right|} \tag{6.3}$$

$$e_{y,f} = e_{z,f} \times e_{x,f} \tag{6.4}$$

Notice that the flap x-direction is found via the usually longer edges of the selected quadrangle providing more accuracy than calculating the y-direction.

Once the flap coordinate system is found, the knowledge of the Brick geometry enables determining the Brick frame origin and base vectors.

Note that the Brick orientation and position is represented by its homogeneous transformation matrix with respect to the Arm Base frame only. Therefore, mathematical singularities of a roll-pitch-yaw-representation are avoided.

7 Implementing Motion Planning

The term *Motion Planning* is used here in its classical sense of generating a path in configuration space connecting a goal configuration to a start configuration without causing any collisions in-between. For several reasons, equipping the Arm with motion planning proved to be indispensable:

- The environment is modeled as a simple L-shape consisting of ground plane and planer wall to avoid collisions with the rover: consequently, the Wrist may be guided with precise paths which are known to be collision free – but the rest of the Arm could still collide with the environment or with itself. Straight line paths both in Cartesian space as well as in joint space are in fact likely to cause collisions.

- Therefore, some degree of collision checking would have had to be implemented in any case which is an intricate step: adding a motion planner to the collision checker appeared to be comparably facile while adding an enormous amount of value.

- The Brick itself, it's antenna in particular, forms a considerable obstacle both for the first part of the pick-up process as well as when attached to the Gripper in the second phase. Without equipping the Arm with any kind of intelligence, the positions and orientations of Bricks that may be picked up would be extremely limited.

- As laid out in Section 5.7, a large part of the pick-up motion would need to be defined in Cartesian space in order to ensure the right behavior of the grasped Brick. But any pre-planned Cartesian space path increases the risk of leaving the workspace and further reduces the amount of allowed Brick positions and orientations that may be grasped.

Implementing a motion planner therefore appeared to be of high value allowing to increase the overall system robustness.

7.1 Terms and Definitions

- Configuration q:
  A set of joint angles θ with $\theta_i \in [\theta_{i,min}, \theta_{i,max}] \in \mathbb{R}, \theta_{i,min} > -\pi, \theta_{i,max} < \pi, i = 1, ..., 5.$

- Configuration space or C-space $\mathcal{C}$:
  The space spanned by configurations: $\mathcal{C} = [\theta_{i,min}, \theta_{i,max}]^5 \subset \mathbb{R}^5, i = 1, ..., 5$. Notice that all θ_i are bounded with $\pm\pi$ in our case. Therefore, the introduction of a distance measure on the C-space will be straightforward.

- $\mathcal{C}_{FREE} \subseteq \mathcal{C}$:
  Subset of $\mathcal{C}$ that is *feasible*, i.e.there are no self collisions and no environment collisions

- $\mathcal{C}_{OBS} = \mathcal{C} \backslash \mathcal{C}_{FREE}$

- Edge:
  Straight line in the C-space connecting two configurations.

- Visibility of an edge:
  An edge is called *visible* if all its configurations are feasible.

7.2 State of the Art in Probabilistic Motion Planning

As soon as planning problems are to be solved in more than three degrees of freedom, straight forward (complete or exact) planning that is guaranteed to find the shortest path in configuration space is computationally too expensive [1], since that requires constructing and then searching the entire free C-space.

Current research is focused on an alternative named *Sampling-Based Algorithms*: only random samples of configurations are checked for their feasibility and later tested for collision free connections – which proves to be computationally cheaper. Unfortunately, they do not guarantee finding the shortest path, but at least, they return a feasible path with increasing time, if one exists. This property is called *Probabilistic Completeness*.

Probabilistic roadmaps (PRM) have proven to solve high-dimensional path planning problems effectively [1]. A PRM is built of sampled feasible configurations called *milestones* that are interconnected by local paths. While searching the roadmap for a feasible path, the planner spends most of its time checking for collisions. Therefore, it is crucial to implement a fast collision checker and to find algorithms that limit unnecessary checks.

An important distinction of probabilistic roadmaps is whether they are constructed for *single query* or *multiple query*. In frequently changing environments as is the case with Brick retrieval, it is reasonable to focus on single query.

The following list gives a short overview of different probabilistic motion planners as they evolved during the last 30 years [1]:

- *Basic PRM*

  Initially, node sampling was done at a uniform distribution in C-space. This first version is obviously probabilistically complete and could solve a wide variety of higher degree planning problems. Unfortunately, this simple strategy is not very fast, when used for single query planning.

- *EST: Expansive-Spaces Trees*

  EST was aimed at being an efficient single query planner that can deal with kinodynamic constraints. It proved to be probabilistically complete. EST constructs a tree rooted at the start configuration. Random samples are taken in the neighborhood of an existing milestone m in the roadmap. In contrary to basic PRM, the new samples are only added to the tree as a child of m, if the respective edge is visible. The milestone to which a new child is attempted to be added is chosen with a carefully defined probability distribution in order to prevent oversampling of certain regions. Once the tree has grown close to the goal, connecting it to the tree is attempted.

- *SBL*

  SBL is a variation of EST where ”S” stands for single query, the ”B” for bi-directional and ”L” for lazy evaluation. There are two main differences to EST: first, there are two trees grown both from goal and start configuration. This generally leads to faster convergence. The second important difference lies in the lazy evaluation: visibility checks of the edges are not performed at addition of new milestones but delayed until checking the overall path. Therefore, the number of unnecessary edge checks is decreased.

- *RRT: Rapidly Exploring Random Trees*

  This planner was again initially developed for single query planning problems under kinodynamic constraints. It also grows two trees, but the expansion step is slightly different: rather than sampling a configuration in the neighborhood of a milestone, RRT samples a totally random configuration and then looks for the closest milestone. It will then move that sample configuration on a straight line closer to that milestone such that the distance metric becomes *step_size*. The choice of this parameter is critical and may also be adaptive with respect to the closeness to obstacles.

7.2.1 The SBL Planner in Detail

Basically, the SBL planner grows two trees T_{init} and T_{goal}, rooted at the initial configuration q_{init} and the goal configuration q_{goal}, respectively. Figure 25 depicts a two-dimensional example of two trees.

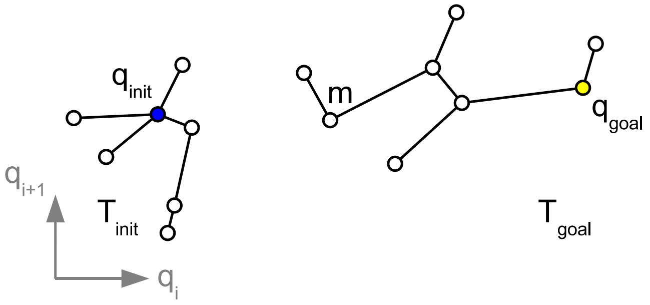


Figure 25: *SBL Tree*

The overall algorithm looks as follows [9]:

Algorithm 1: SBL PLANNER

```
Install $q_{\text{init}}$ and $q_{\text{goal}}$ as the roots of $T_{\text{init}}$ and $T_{\text{goal}}$, respectively;
for $i = 1$ to $s$ do
    EXPAND-TREE;
    $\tau \leftarrow$ CONNECT-TREES;
    if $\tau \neq nil$ then
        return $\tau$;
    end
end
return failure
```

Note that the algorithm may return failure, which does not imply there is no feasible path, but the algorithm has not found one within s iterations.

The tree expansion step adds a milestone as a child to an existing milestone in the graph:

Algorithm 2: EXPAND-TREE

```
Pick $T$ to be either $T_{\text{init}}$ or $T_{\text{goal}}$ with probability 0.5;
repeat
    Pick a milestone $m$ from $T$ at random using probability distribution $\pi(m)$;
    for $i = 1$ to max_iter do
        Pick a random $q$ in the neighborhood of $m$: $||q - m||_\infty < \rho/i$;
        if $q$ collision-free then
            install it in $T$ as child of $m$;
        end
    end
until new milestone configuration q generated ;
```

The probability distribution $\pi(m)$ is chosen to be inversely proportional to the current density of milestones which will prevent oversampling of certain regions. The neighborhood radius starts at an initial ρ and is then gradually decreased with each unsuccessful try which increases the probability of picking a feasible configuration. This strategy lets the algorithm take big steps in free space and adapt to smaller steps in cluttered C-space. An EXPAND-TREE step is illustrated in Figure 26.

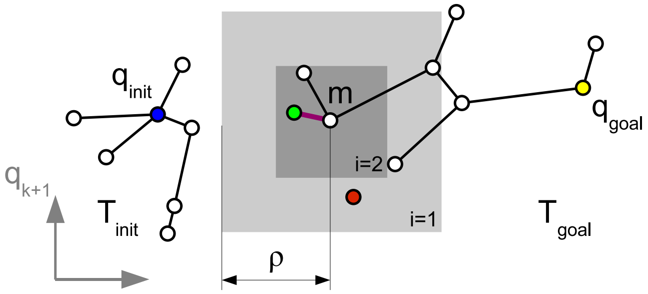


Figure 26: *EXPAND-TREE step at milestone m: in this example, the first sampled configuration was infeasible (thus marked red). Consequently, the neighborhood is shrinked for the second attempt (i = 2). Here, a feasible configuration is sampled (green) and EXPAND-TREES installs it as a child of m.*

In each main loop iteration, the attempt is made to connect the trees (see figure 27) by a bridge w and find a possible path:

Algorithm 3: CONNECT-TREE

```
$n \leftarrow$ most recently added milestone ;
$n' \leftarrow$ closest milestone to m in the tree not containing $m$;
**if** $||n - n'||_\infty < \rho$ **then**
    Connect $n$ and $n'$ by a bridge $w$;
    **return** *TEST-PATH*($\tau$);
    **return** *nil*;
**end**
```

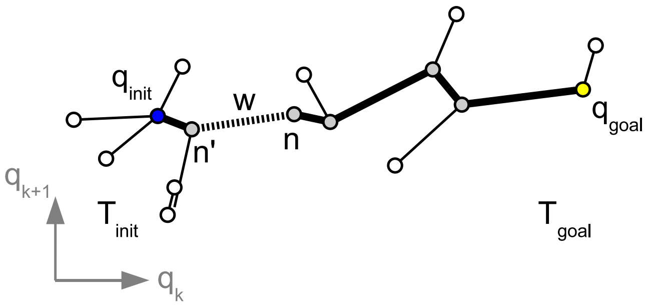


Figure 27: *Connect-Tree Step: since the most recently added milestone n is close to n′ in the other tree, the bridge w is created.*

The TEST-PATH algorithm finally checks all the edges of the path. If a collision is detected in any of the edges, the subtree before or after the non-feasible edge needs to be reconnected to T_{init} or T_{goal}, respectively.

In order to make the path checking as efficient as possible, the TEST-SEGMENT algorithm requires some attention: it makes sense to check the edges by bisection, i.e.always checking the middle point: the probability that the middle point is infeasible is the highest, since it is the furthest away from feasible configurations. The tested segments are labeled with a value indicating to what extent they have been tested, i.e.the length of the non-tested subsegments. As soon as the distance between tested configurations on an edge becomes smaller than ϵ, the edge is labeled *safe*.

The TEST-PATH algorithm now maintains a priority queue U with the edges to test, starting with the one of highest distance between tested points: the aim is that the non-feasible part of the path is detected as early as possible. Notice that an overall edge check is only done once but it may span several TEST-SEGMENT and even TEST-PATH calls.

Algorithm 4: TEST-PATH(τ)

```
while $U$ is not empty do
    u ← extract($U$);
    if $TEST\text{-}SEGMENT(u)==collision$ then
        if $u$ child of $T_{init}$ then
            remove $u$ from $T_{\text{init}}$;
            connect subtree after $u$ to $q_{\text{goal}}$;
        end
        if $u$ child of $T_{goal}$ then
            remove $u$ from $T_{\text{goal}}$;
            connect subtree before $u$ to $q_{\text{init}}$;
        end
        return $nil$;
    end
    if $u$ not marked safe then
        re-insert $u$ into $U$;
    end
    return $\tau$;
end
```

Figure 28 shows the re-connected example trees after one of the edges proofed to be infeasible.

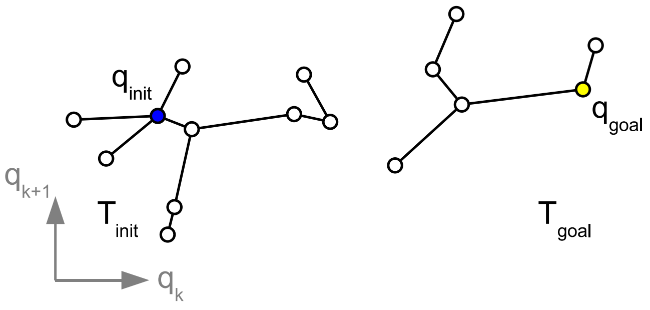


Figure 28: *In the example, the TEST-PATH failed since TEST-SEGMENT found one edge to be non-visible. Therefore, that edge is removed and the trees are reconnected accordingly.*

7.3 Problem Statement

The goal is implementing a modified SBL planner that is capable of returning feasible paths for the K10 Arm both with an empty Gripper as well as with the Brick attached.

The first case is a straightforward use of the existing SBL algorithm already implemented in the motion planning library.

The second case, however complicates the situation, because the brick may or may not adjust its orientation around the Gripper axis to gravity. In fact, the adjustment is a needed property allowing bringing Arm and Brick into a *carry pose* that relieves Gripper and Arm from unnecessary torques. Allowing this mostly uncontrolled roll motion around the Gripper axis may be seen as a necessary compensation for the lacking degree of freedom. Figure 29 illustrates the swing-down adjustment of the Brick as well as the carry pose. There are, however, considerable consequences of this uncontrolled Brick adjustment: the collision checking step is no longer just a function of the Arm configuration; the angles at which the Brick may be attached to the Gripper need to be modeled with a memory of the previous possible Brick angles which makes it a dynamic system. The problem is not only restricted to modeling these dynamics in a realistic way, but there are also necessary adaptations to the SBL algorithm and post-processing, mostly caused by the fact that collision checks need to be performed in a chronological way.

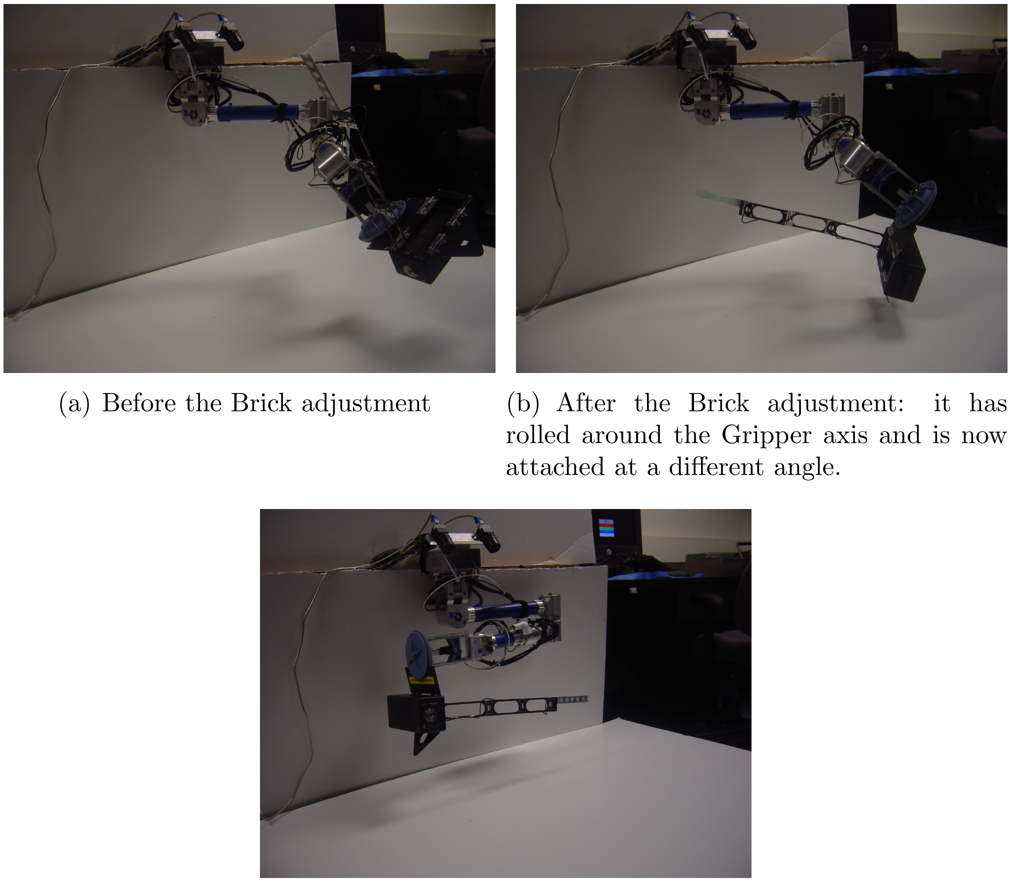
7.4 Geometric Model for Collision Checking

Both for basic collision checking without the Brick attached as well as for the more challenging situation of the Brick being part of the Arm, a model of the environment and the Arm geometry including kinematic relations is needed.

7.4.1 Robot Model

The CAD model of the Arm, the Gripper and the Brick had to be dramatically simplified in order to ensure fast collision checking. The resulting geometries are stored as triangulated meshes consisting of approximately 100 to 500 triangles per body. Figure 30 shows a comparison between the original CAD model and the simplified version.

The Brick may either be part of the robot model or not, dynamically loadable and removable at runtime. Figure 31 shows the Brick model. Since its antenna is flexible, consisting of



(a) Before the Brick adjustment

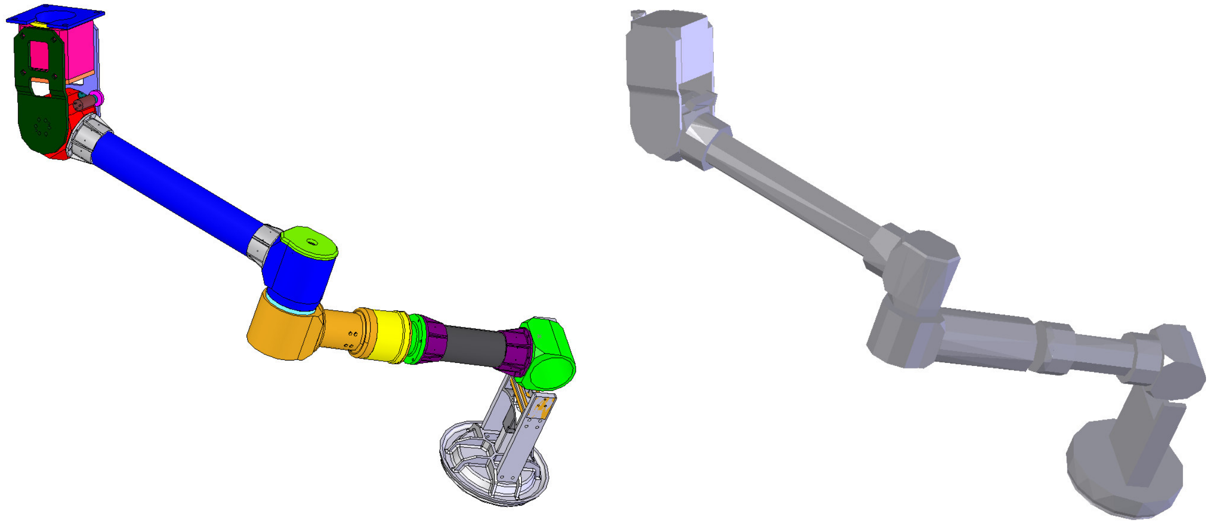
(b) After the Brick adjustment: it has rolled around the Gripper axis and is now attached at a different angle.

(c) Finally Arm and Brick reach the carry pose.

Figure 29: *Brick pick-up and carry pose.*

four links interconnected with springs, the position uncertainty grows with the height of the antenna. Therefore it was modeled as cylindrical sector such that the antenna will always stay within this bounding box.

The robot model needs 2 extra DoF that are not part of the C-space: a prismatic joint to model the Gripper and a revolute joint to model the Brick's grasped position in the Gripper. For the prismatic joint, its extension e is always known and can be set externally. On the other hand, the Brick location in the gripper, tracked as the angle φ, is not known *a priori.*



(a) Original CAD model.

(b) Simplified and triangulated model used for collision checking.

Figure 30: *CAD Model Simplification*

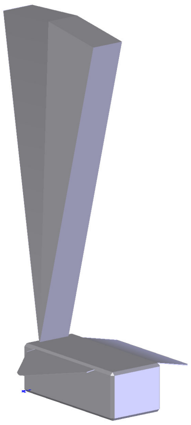


Figure 31: *Simplified Brick CAD model with antenna modeled as cylindrical sector due to uncertainty.*

7.4.2 Environment Model

In a future version of this project, K10 will provide the stereo-reconstructed terrain as environment mesh for picking up Bricks. The Brick will be part of that terrain, therefore it needs to be removed carefully without leaving any obstacle artifacts.

The position and attitude of the Brick has already been determined by stereo vision at this stage. Therefore, the terrain mesh is overlaid with the Brick model at its estimated position and orientation. Now, all triangles overlapping with the Brick or being closer than a certain margin are flattened into the plane the brick is standing on.

For the first phase, where the Arm is approaching the Brick, the Brick is part of the environment, therefore its model is merged with the environment mesh.

As there was no time left to tackle the implementation of stereo reconstruction yet, a planar environment was assumed in the form of a simple L-shape representing ground and rover. The Brick removal process as described above, however, is already implemented.

7.5 Advanced Collision Checking with Attached Brick

This subsection presents a solution to the problem of the angle φ the Brick is attached to the Gripper not being known *a priori* for any configuration q. The presented models also require significant adaptations to the SBL planner as well as to the post-processing of found paths.

7.5.1 Three Contact Point Friction Model

The first question to be answered is: given a configuration q and the Brick angle φ, will the brick adjust its orientation (φ) to gravity? Because of the extremely nonlinear transition from static to dynamic friction as well as model uncertainties, this question cannot be answered with in a binary sense of ”yes” or ”no”; it may, however, be given an answer from the ternary set ”sure not” (0), ”sure yes” (1) or ”maybe” (2). We are therefore looking for a function ADJUST:

$$\mathrm{ADJUST}(q, \varphi) = \begin{cases} 0 & \text{meaning ”sure not”;} \\ 1 & \text{meaning ”sure yes”;} \\ 2 & \text{meaning ”maybe”.} \end{cases} \tag{7.1}$$

In order to find this function, a friction model is needed. The following modeling assumptions are made hereby:

- Any dynamic effects introduced by Arm motion are neglected.

- Both Brick and Gripper are rigid bodies. There are neither deflections nor forces from deformations taken into account.
- The only degree of freedom of the Brick with respect to the Gripper is a rotation φ around the Gripper axis.
- The Brick is supported by three points of contact at all times; no reaction torques originate from these points.

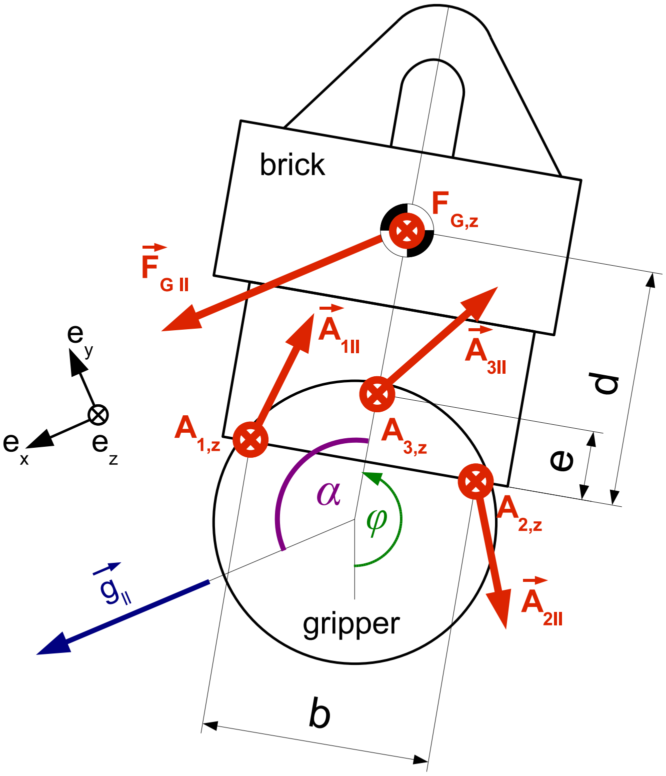


Figure 32: *Sketch with definitions needed for the friction model: top view from Wrist onto Gripper plane.*

Figure 32 shows a top view from on the Gripper plane and defines respective forces. All vectors are decomposed into their z-component normal to the Gripper plane and a component in the plane. The angle $\alpha(q, \varphi)$ between the gravity vector projected into the Gripper plane $\vec{g}_{||}$ and the direction to the Brick center of gravity (CoG) is the first relevant measure. Notice that α is defined as $\alpha \in [0, \pi]$. The second deciding angle is the tilt angle of the Gripper plane $\gamma(q)$. Knowing the tilt angle γ, the weight components are given as:

$$F_{Gz} = m_{Brick} \cdot g \cdot \cos\gamma \tag{7.2}$$

$$F_{G||} = m_{Brick} \cdot g \cdot \sin\gamma \tag{7.3}$$

Note that the Brick CoG where the weight attacks lies at an offset c below the Gripper plane.

The attempt is made to determine, whether a static equilibrium is possible or not, given all the parameters including the static friction coefficient μ between Gripper blade and Brick flap.

There are three forces each with three components that we need to solve for. Unfortunately, the requirement for static equilibrium only provides six equations (force and torque equilibria), therefore we are dealing with a *statically undefined* problem. This particular choice of force component decomposition, however, lets us solve a statically defined subproblem: the normal component of each of the three reaction forces A_{iz} at the contact points may be determined.

$$\begin{array}{l} M_x = 0: \\ M_y = 0: \\ R_z = 0: \end{array} \begin{bmatrix} -\frac{b}{2}\cos\alpha & \frac{b}{2}\cos\alpha & e\sin\alpha \\ -\frac{b}{2}\sin\alpha & \frac{b}{2}\sin\alpha & -e\cos\alpha \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} A_{1z} \\ A_{2z} \\ A_{3z} \end{bmatrix} = \begin{bmatrix} -F_{Gz} d \sin\alpha \\ F_{Gz} d \cos\alpha - F_{G||} c \\ -F_{Gz} \end{bmatrix} \tag{7.4}$$

Since equation (7.4) is linear in A_z the solution is directly obtained as:

$$\cdot \begin{bmatrix} A_{1z} \\ A_{2z} \\ A_{3z} \end{bmatrix} = \begin{bmatrix} \left(\frac{d}{e} - 1\right) F_{Gz} - \frac{c}{e}\cos\alpha F_{G||} + 2\frac{c}{b}\sin\alpha F_{G||} \\ \left(\frac{d}{e} - 1\right) F_{Gz} - \frac{c}{e}\cos\alpha F_{G||} - 2\frac{c}{b}\sin\alpha F_{G||} \\ -\frac{d}{e} F_{Gz} + \frac{c}{e}\cos\alpha F_{G||} \end{bmatrix} \tag{7.5}$$

Knowing A_{iz}, the requirement for static friction may be formulated:

$$\left|\left|\vec{A}_{i||}\right|\right| \leq \mu \cdot |A_{iz}| \tag{7.6}$$

In order to be able to answer the question stated in the beginning in the sense of equation (7.1), (7.6) is evaluated for two different coefficients μ constituting a lower bound (μ_l) and an upper bound (μ_u) of the static friction (not to be confused with dynamic and static friction coefficients). This is how both model uncertainties as well as the probabilistic behavior of the transition from static friction to dynamic friction are dealt with. Thus the function ADJUST (7.1) can be formulated as:

$$\text{ADJUST}(q,\varphi) = \begin{cases} 0 & \text{if (7.6) holds with } \mu_l; \\ 1 & \text{if (7.6) holds with } \mu_u; \\ 2 & \text{else, i.e.if (7.6) holds with } \mu_u \text{ but not with } \mu_l. \end{cases} \tag{7.7}$$

The force distribution among the components in the Gripper plane ($\vec{A}_{i||}$), i.e.the friction forces, is not entirely defined. Therefore, the following assumption is made: if there is *any* force distribution complying with the requirement for static friction (7.6) as well as with the remaining force and torque equilibria, then the respective static equilibrium is assumed to be held.

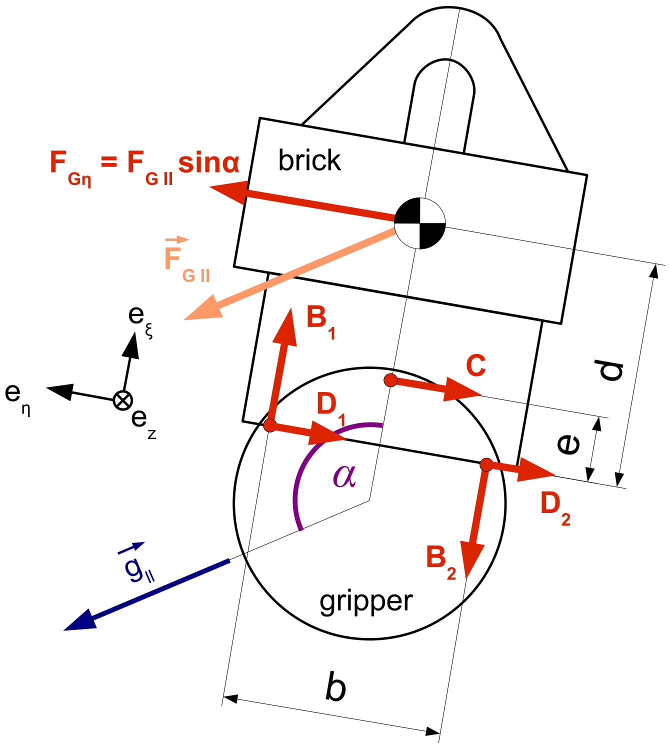


Figure 33: *Forces in the Gripper plane*

Consider Figure 33 introducing the decomposition of the forces in the Gripper plane into components along η- and ξ-axes which are aligned with the Brick frame. It is reasonable to assume that any weight in ξ-direction will be opposed by either the knob supported in the upper Gripper blade ($\alpha < \pi/2$) or the flap touching the Gripper motor plunger ($\alpha > \pi/2$). Therefore, the ξ-component of $F_{G||}$ is not considered.

The equations for static equilibrium are now:

$$\begin{aligned} R_\xi: &\quad B_1 = B_2 = B \\ R_\eta: &\quad F_{G\eta} - D_1 - D_2 - C = 0 \\ M_z: &\quad 2 \cdot B \cdot \tfrac{b}{2} - F_{G\eta} \cdot d = 0 \end{aligned} \tag{7.8}$$

Now, the condition as in (7.6) may be expressed as a set of three conditions to be fulfilled:

$$\begin{aligned} &(1) \quad C^2 \leq (\mu A_{3z})^2 \\ &(2) \quad B^2 \leq (\mu \cdot \min\{A_{1z}, A_{2z}\})^2 \\ &(3) \quad (F_{G\eta} - C)^2 \leq (\mu A_{1z})^2 + (\mu A_{1z})^2 - 2B^2 \end{aligned} \tag{7.9}$$

The first inequality (1) makes sure the limit for C is respected. The second condition (2) ensures B alone is not violating the maximum force allowed for static friction at either contact point 1 or 2, whichever has a smaller limit; inequality (3) finally checks there is the possibility of distributing $F_{G\eta} - C$ among point 1 and 2 without exceeding any of the respective limits.

As stated above, the static equilibrium is assumed to be feasible if there exists *any* C such that the above inequalities are fulfilled (notice that there is only one degree of freedom).

Figure 34 shows a generated map of ADJUST(γ, α) for $\mu_l = 0.2$ and $\mu_u = 0.4$.

7.5.2 Possible Brick Angles Modeled as a Finite State Machine

Knowing what might happen to the Brick given its angle φ along with a momentary configuration q lets us keep track of all the possible angles φ that a collision checker must check with. Notice that this propagation of possible Brick angles is only possible, if the chronological sequence of configurations q is known. Therefore, the model being presented in the following is only of use for checking complete paths and *not* for sampling random configurations to whose parent configurations are unknown.

Given is path to be checked as a finite set of n subsequent configurations $q_0, ..., q_n$. Be ψ a list of length m: it is the discretization of the angle interval $[-\pi, \pi]$ into m cells. The r^{th} entry corresponds to an angle φ, carrying the information on whether the Brick could be attached at that angle or not. The relationship between the list index r and the physical

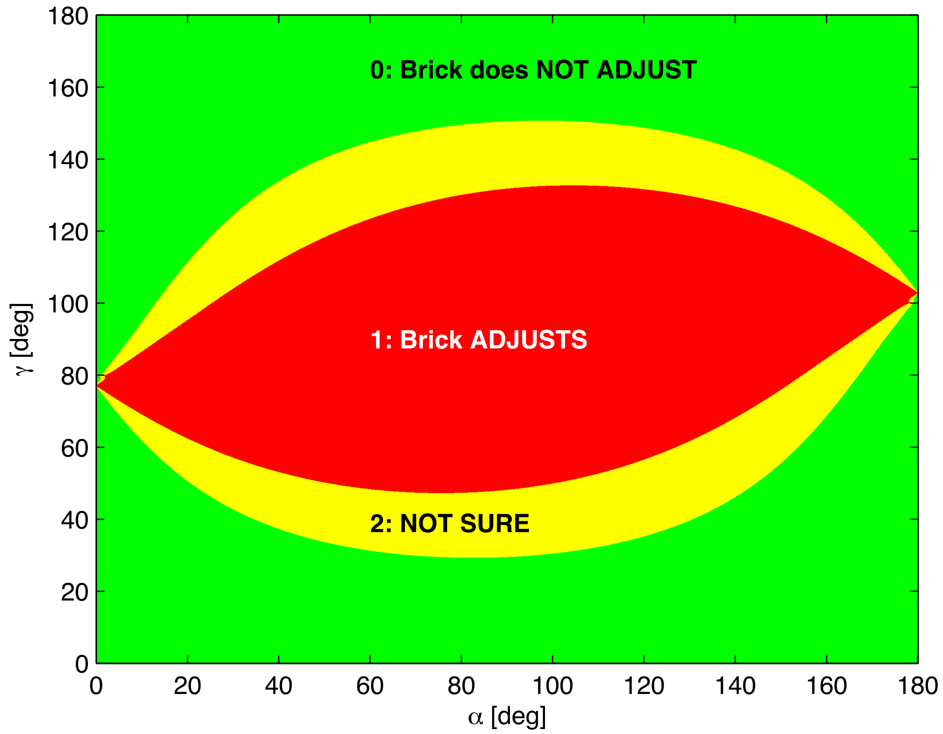


Figure 34: *Map of the 'ADJUST' function depending on tilt angle γ and brick angle to gravity α. $\mu_l = 0.2$ and $\mu_u = 0.4$.*

angle φ_r is given as:

$$\varphi_r = \frac{r-1}{m} - 0.5 \cdot 2\pi, r = 1, ..., m \tag{7.10}$$

The list ψ is also referred to as *checklist*, since it contains the information on what Brick angles φ the collision checker will have to test. Each of the m cells in ψ holds a ternary *state* (not to be confused with the coincidently also ternary output of function ADJUST):

$$\psi_r = \begin{cases} 0 & \text{if the Brick is for sure not attached at the corresponding angle } \varphi; \\ 1 & \text{if the Brick may me attached motionlessly at the corresponding angle } \varphi; \\ 2 & \text{if the Brick might be moving through the correspontíng angle } \varphi. \end{cases} \tag{7.11}$$

The propagation of ψ through the different subsequent configurations q_k has the characteristics of a *Finite State Machine* which is a *Deterministic Automaton* meeting the following prerequisites [5]:

- Ψ is the set of automaton *states* with $\psi \in \Psi$
- The Arm configuration $q \in \mathcal{C}$ is an *input letter*
- There is a deterministic *transition function* δ with:

$$\delta : \Psi \times \mathcal{C} \rightarrow \Psi \tag{7.12}$$

- ψ_0 is the *start state* of the automaton
- F is the set of *accepted letters*, in this case $F = \mathcal{C}$

The initial state of the automaton may easily be determined, since the angle φ_0 at which the brick will initially be attached to the Gripper can be calculated. Therefore, all cells of ψ_0 are set to zero except those whose corresponding angles are close to ϕ (some uncertainty about φ_0 is considered).

In the following, a transition function is presented defining $\psi_k = \delta(\psi_{k-1}, q_k)$.

1. For each cell $\psi_{k-1,r} \in \psi_{k-1}$ that is non-zero (i.e.the Brick could be attached at the corresponding angle), the following steps are performed:
   (a) find the angle φ_r corresponding to cell $\psi_{k,r}$. Call the function ADJUST(q_k, φ_r) and remember the outcome.
   (b) If the outcome of ADJUST is non-zero, we need to deal with the fact that the Brick may adjust its orientation from this point. Therefore, all cells in a temporary checklist ψ_{temp} that correspond to Brick angles between φ_r and the angle corresponding to the projected gravity vector $\vec{g}_{||}$ change state from 0 to 2.

2. Transfer the states of ψ_{k-1} to ψ_k according to the following table:

		Outcome of ADJUST(q_k, ϕ_r)		
		0	1	2
State of $\psi_{k-1,r}$	(0)	(0)	(0)	(0)
	1	1	2	1
	2	1	0	1

   There is one exception to the above rules: if the angle the Brick adjusts is smaller than π/m in absolute value, then the respective cell is set to 1 in any case: this ensures that the Brick cannot get ’lost’ in the checklist ψ_k.

3. Finally, ψ_k and ψ_{temp} are merged: each cell that is 0 in ψ_k but 2 in ψ_{temp} is changed to 2 in ψ_k.

The collision checker now needs to check with each of the Brick angles φ that have a non-zero corresponding cell in the checklists.

Figures 35 and 36 show an example sequence of Arm configurations q and their corresponding propagation of ψ.

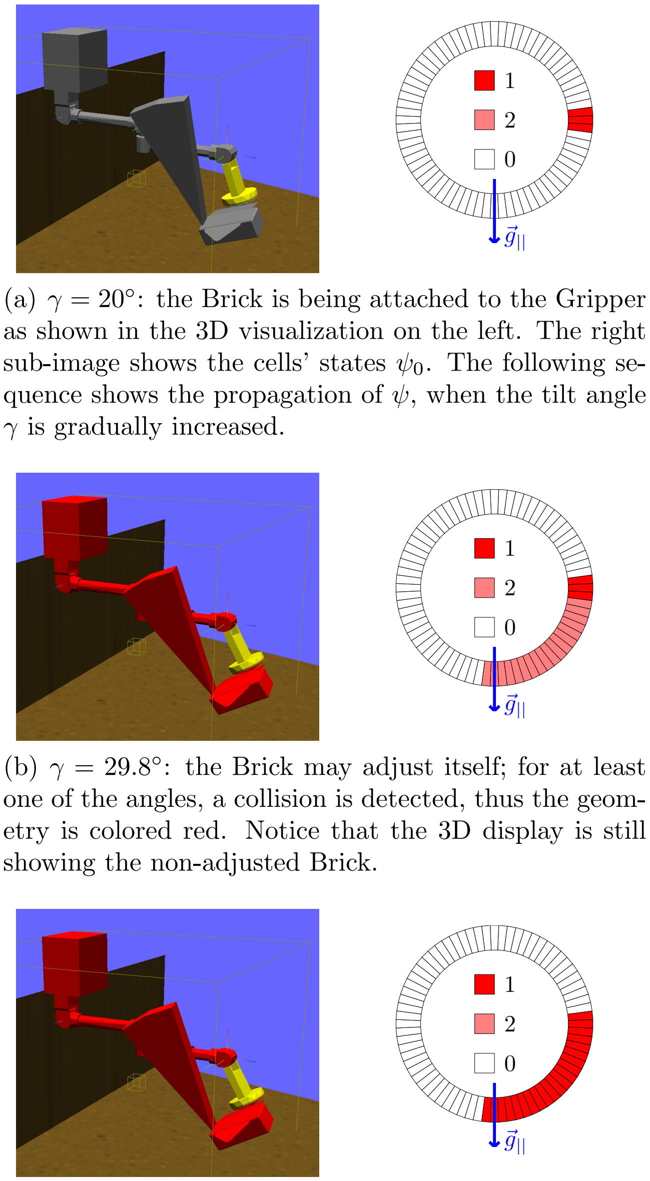
7.6 Necessary Adaptations to the SBL Planner

Collision checking is performed in the sampling step as well as in the path checking step of the SBL planner. Therefore, these two steps need to undergo adaptations.

7.6.1 Collision Checking for Random Configurations

In the phase SBL is sampling random configurations, i.e.before a possible plan is known, the chronological sequence of configurations before the sampled one is completely unknown. Therefore, the possible Brick angles φ of that Arm configuration is also unknown to a certain extent. Still, a collision check needs to be performed for each sample.

A very conservative approach would be to check for collisions with all possible angles φ. Unfortunately, the antenna of the brick is so long that this will always cause a collision with the Elbow at certain angles φ; therefore that conservative approach would yield a collision in any case.

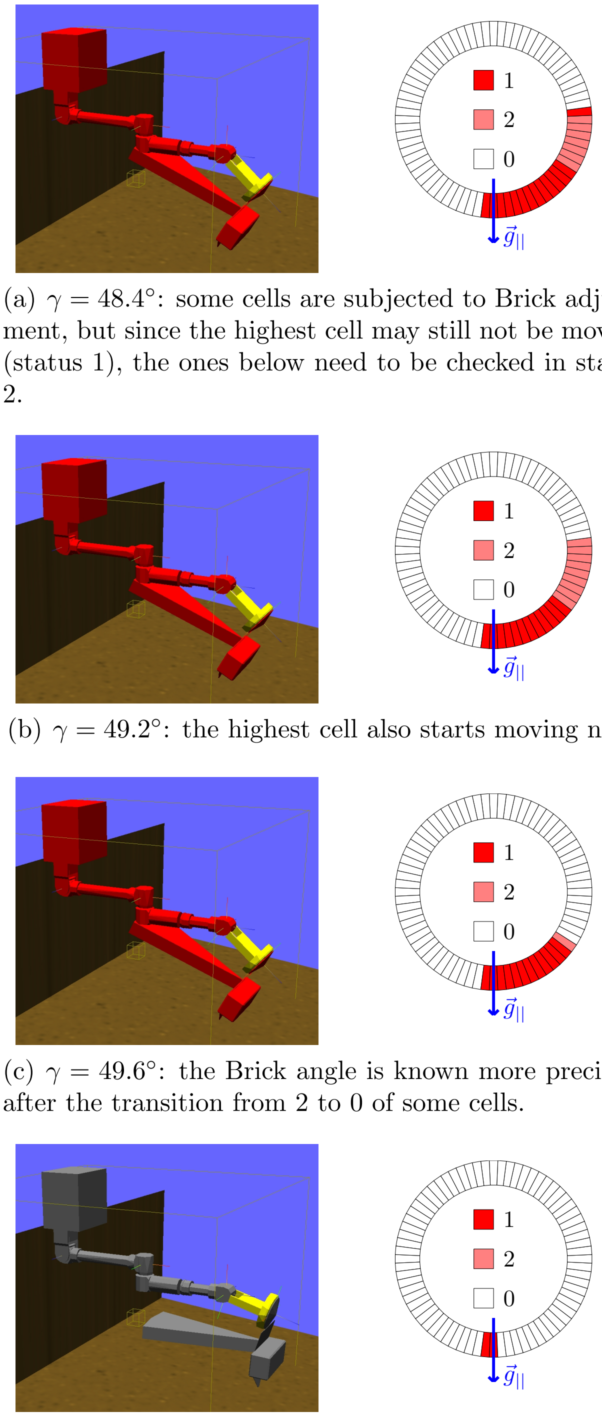


(a) $\gamma = 20°$: the Brick is being attached to the Gripper as shown in the 3D visualization on the left. The right sub-image shows the cells' states ψ_0. The following sequence shows the propagation of ψ, when the tilt angle γ is gradually increased.

(b) $\gamma = 29.8°$: the Brick may adjust itself; for at least one of the angles, a collision is detected, thus the geometry is colored red. Notice that the 3D display is still showing the non-adjusted Brick.

(c) $\gamma = 30.2°$: The Brick could have stopped along its path of adjusting down, wherever a still state is possible, the cell values changed from 2 to 1.

Figure 35: *Illustration of automaton state propagation part I.*



(a) $\gamma = 48.4°$: some cells are subjected to Brick adjustment, but since the highest cell may still not be moving (status 1), the ones below need to be checked in status 2.

(b) $\gamma = 49.2°$: the highest cell also starts moving now.

(c) $\gamma = 49.6°$: the Brick angle is known more precisely after the transition from 2 to 0 of some cells.

(d) $\gamma = 73.6°$: the Brick angle has gradually become known more precisely as the Gripper plane was tilted.

Figure 36: *Illustration of automaton state propagation part II.*

It is important to mention that there is no requirement for collisions to be detected at this stage already, since the complete path will be checked anyways. For a fast convergence of the search, however, it is desirable to achieve as much correlation between this first feasibility check and the feasibility check for the same configuration when checking the path. The angle φ_0 hereby plays an important role: it represents the angle at which the Brick was grasped.

The following procedure was applied in order to check for collision of a sample:

1. Assume the Brick is attached at an angle to $\vec{g}_{||}$ $\alpha = \pi/3$ and retrieve the corresponding output from the ADJUST function.

2. If the adjustment is not sure, check for collisions with some angles inbetween φ_0 and the angle of the gravity-adjusted Brick. Else, adjust the Brick angle φ – or not – according to the output of ADJUST and perform *one* collision check.

All of the assumptions are justified by the fact that they approximate reality for sampled configurations on a path close to the initial configuration q_0.

7.6.2 Path Checking Adaptations

In order to be able to use the advanced collision checking mode as developed in Section 7.5, the CHECK-PATH algorithm of SBL needs to be adapted such that the configurations are checked in chronological order allowing the propagation of the *checklist* ψ. As an unfortunate consequence of this requirement, all of the optimized edge checking needs to be switched off:

1. All edges must be checked chronologically, i.e.begin at the start tree root; priorization of edge checking is excluded.

2. Each edge must be checked at finest resolution only: the check of the interpolated points between the start and end configuration of the edge need to be checked chronologically as well. Checking edges by bisection is impossible.

Notice that remembering edges labeled *safe* may be done the same way as in the original SBL planner: the enabling fact lies in the topology of the graph. It is a property of trees that the path from the root to any node is always unique within the start tree. Notice that the goal tree can not have any checked edges: as soon as an edge proves to be invisible, CHECK-PATH moves all previous edges to the start tree. Therefore, edge remembering can be performed in the following way:

- All checked (i.e.visible) edges are remembered. Each edge also stores the checklist ψ as it looked like when the edge checking reached the edge end configuration.

- When CHECK-PATH is starting its check at the root of the start tree it does not check the edges as long as they are labeled *safe*; instead, the checklist ψ of the respective edge is retrieved.

- As soon as an edge is not marked *safe*, all following edges will be checked while continuously propagating the checklist ψ (and storing it to the respective edge).

Figure 37 shows an example of the two SBL trees as presented to the modified CHECK-PATH.

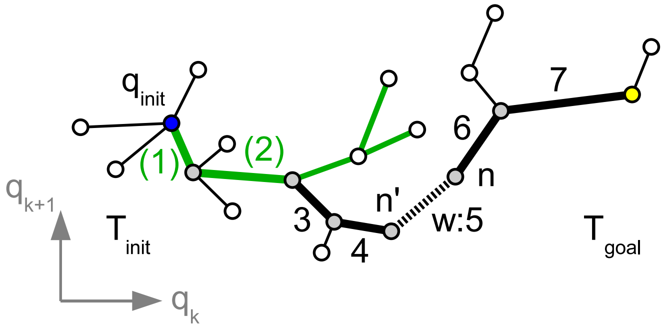


Figure 37: *Illustration of connected example trees with their respective path to be checked chronologically: some previous unsuccessful path checks have left some branches of the start tree labeled* safe *(marked green). Therefore, edge (1) and (2) are not checked; the edge checking begins with edge 3 after retrieving the stored checklist ψ from edge (2).*

Note that the lazy evaluation of SBL is a necessary precondition for the needed chronological collision checks: were the edges to new milestones to be checked when they are generated, then the goal tree would be impossible to build since there is no information on the configuration history to that milestone available.

7.7 Necessary Adaptations to the Path Simplification

Once a feasible motion plan is found that path generally is unnaturally non-smooth and far from optimal. Therefore, some post-processing is desirable. SunSpiral, Chavez *et al.*[10] suggest constructing a graph with all visible interconnections between the milestones of the found path. Subsequently, the Dijkstra Shortest Path is run with that graph and in most cases a simpler feasible path is returned.

Since edges cannot be checked independently when constructing a graph in the case at hand, the idea of lazy evaluation is applied also to this path simplification step: the graph is therefore simply constructed with *all* interconnections – except the one connecting goal and start directly, since that trivial solution is already known to be infeasible. Next, Dijkstra is run and the returned path is checked in a chronological way from start to goal configuration. If an edge happens to be invisible, it is removed from the graph and the Dijkstra search is run again. At some point, a feasible path will be found.

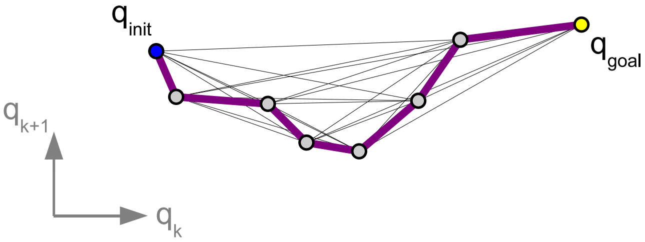


Figure 38: *Constructed graph as input to the Dijkstra shortest path search.*

Figure 38 shows the example path with the constructed graph for Dijkstra's Algorithm. Note that the number of edges e is $\mathcal{O}(m^2)$ with the number m of milestones the path consists of (including start and goal i.e.$m > 2$):

$$e = \frac{m \cdot (m-1)}{2} - 1 \tag{7.13}$$

Of more interest is the maximum number of iterations n_d. At each unsuccessful iteration, there is an edge being deleted. Therefore n_d is found as:

$$n_d = e - (m-1) = \frac{m \cdot (m-3)}{2} \tag{7.14}$$

This makes the simplification step fast compared to the SBL motion planning step.

Notice that the found path is not guaranteed to be the shortest feasible one of the constructed graph: edges might get deleted that would be feasible in another choice of route being shorter than the one the algorithm finally finds. This unfortunate fact may not be easily circumvented, since the only alternative would be to brute-force check all possible paths in the order of increasing cost. This would yield a maximum number of necessary iterations n_b of $\mathcal{O}(2^m)$ – which is not practicable for larger numbers of milestones:

$$n_b = (m-2) \cdot 2^{m-3} \tag{7.15}$$

It is worth mentioning, however, that even without this guarantee, the algorithm will still simplify the path somehow. In some cases, there are even attempts of the algorithm to delete an edge of the originally found path (feasible in its entirety) just because a different way was chosen to that edge. The deletion of those edges needs to be prevented by removing the previous edge instead.

The selection of a suitable cost for each of the edges connecting milestone m_i and m_j deserves some attention: the choice is depending on what is attempted to be minimized. Choosing the cost function to be $||m_i - m_j||_2$ minimizes the overall motion and may thus be seen as energy minimizing. Attempting to minimize the time spent for the motion would require selecting a weighted ∞-norm, where the weights are the inverse of the respective joint speed maxima. This choice has the drawback of not penalizing unnecessary motion of the non-deciding joints. Hence, a compromise between energy and time minimizing has been chosen by defining the cost function as a weighted 2-norm.

In order to further smooth the path, each of the edges is bisected and the Dijkstra-Simplification is run again. This is repeated several times until the improvement becomes lower than a specified threshold. One iteration of this bisection is shown in Figure 39; the final result of the path is displayed in Figure 40.

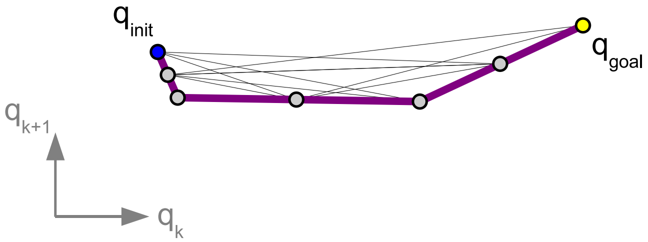


Figure 39: *Bisected path for re-running Dijkstra-Simplification.*

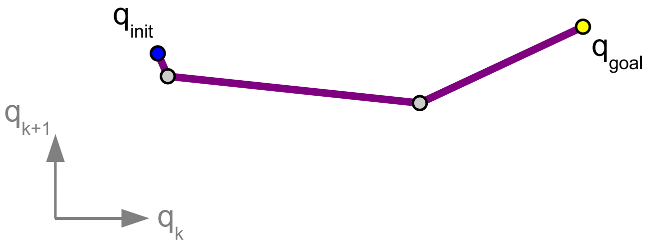


Figure 40: *Final example path: bisection stopped after one iteration.*

7.8 Overall Pick-up Planning

The overall pick-up now looks as follows:

1. Check reachability of the Brick.
2. Plan and execute a motion to the Brick with the Brick being part of the environment
3. Grasp the Brick with a motion predefined in Cartesian space (i.e.no planning): generate trajectory, check it for collision and execute. The corresponding procedure is presented in Subsection 5.7, Figure 23.
4. Plan a motion to the carry pose: hereby, the Brick is attached to the Gripper and motion planning is performed with the adapted SBL planner.

8 Results and Discussion

At the end of the project, a lab test bench was established capable of robust Brick pick-ups. Minimal user interaction is required for the Brick detection; supervision and debugging tools are available during operation. Appendix B provides all the information needed on the hardware and software usage. Figure 41 presents an example of a complete successful cycle.

8.1 Gripper Operation

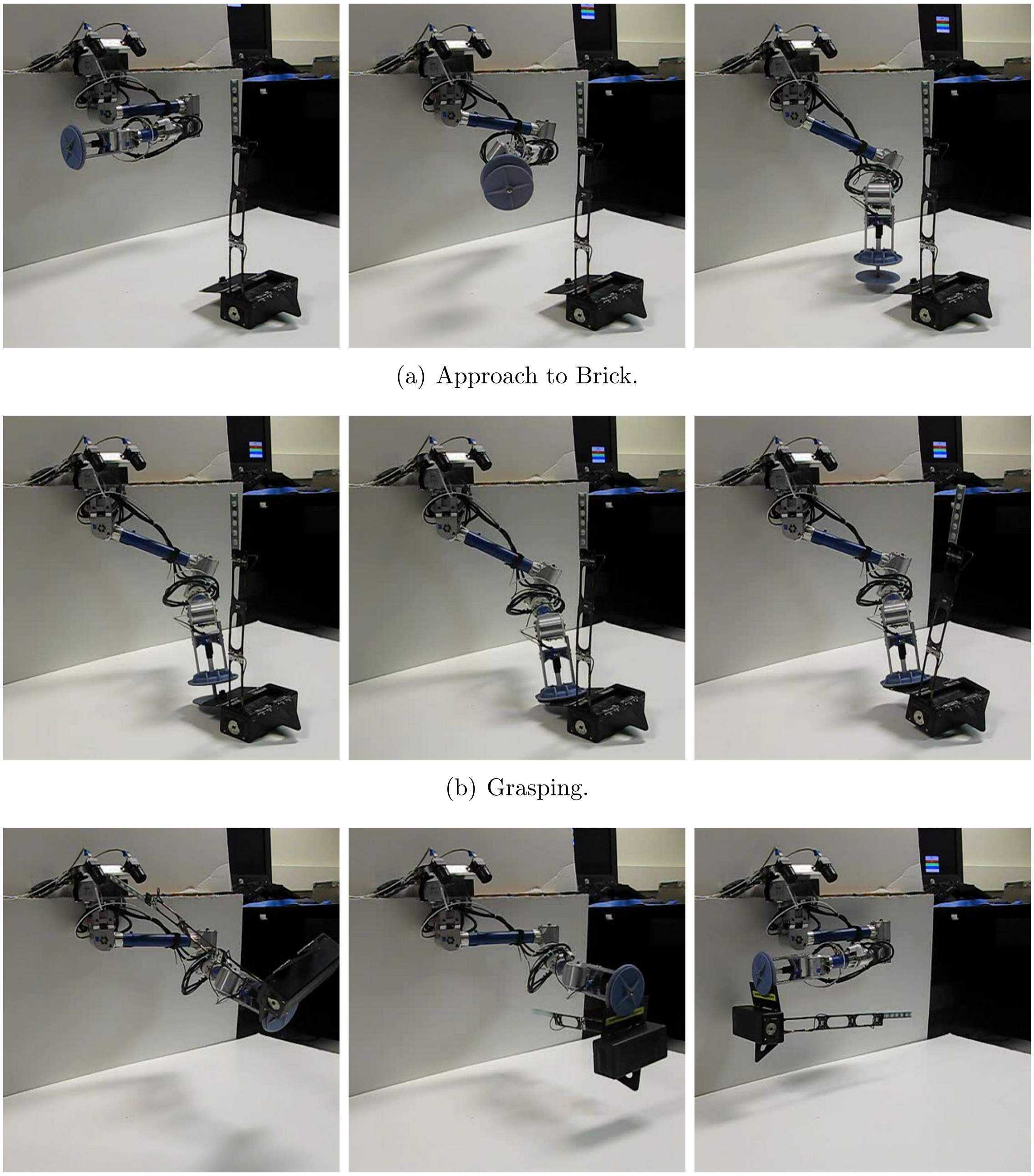
The Gripper showed robust operation due to the filtered position signal and its capability of detecting forces and reacting accordingly.

Despite its low weight, the Gripper structure proved to be very rigid except for the lower blade: the respective deformation leads to the fact that the Brick is not exactly attached as in the geometric model. This imprecision may cause problems in conjunction with collision checking and motion planning. This may, however, be recovered by either modeling the flexion or – much easier – by adding some buffer to the Brick model.

8.2 Pick-Up with and without Motion Planning

Using the Arm and the Gripper without motion planning is doomed to frequent failure: when following a Cartesian space path only, collisions between Brick/Wrist and ground may at least be successfully avoided. However, the respective long trajectories were very often not generated due to singularities and workspace leaving.

When using a joint space trajectory for the approach and the retrieval, the operation had to be aborted frequently due to immanent crashes. Notice that the whole Gripper tilting and Brick adjustment phase absolutely requires Cartesian space path following: that way, the Brick is guaranteed not to crash into the ground or, more likely, not to hit the Arm with its antenna by swinging down the wrong side. It proved to be this very part of the Cartesian space trajectory that very often ran into singularities, especially approaching Twist or Wrist joint limits during the 90° Gripper orientation change.



(a) Approach to Brick.

(b) Grasping.

(c) Taking the Brick to carry position.

Figure 41: *Full pick-up cycle with the final test bench version.*

8.3 Experiences with the Motion Planner

As expected, the implemented motion planner finally removed all the difficulties experienced when trying to pick up without motion planning and enhanced robustness and autonomy by orders of magnitude. The required changes and additions to the collision checker and planner proved to be far more extensive than expected, but finally the motion planner showed satisfying results.

The only drawback lies in additional computation time for planning of motion with the Brick attached: pick-ups from certain positions and orientations proved to require non-obvious paths which may take the planner up to a minute or two. Equally often, however, the latencies are in the order of a few seconds or less.

The reason for the rather high computation times are diverse. Primarily, the configuration space ends up being highly constrained when the Brick is attached. This is compounded by the fact that the possible Brick angles spans a large interval under certain circumstances. Secondly, the friction model computation and numerous collision checks with different Brick angles increase the average latency of the collision checker considerably. A further source for the slowdown certainly lies in the fact that all the smartness of the path checking in terms of edge priorization and bisection had to be turned off. Personally, I see a major cause for slow convergence in the non-causality of the following sense: when the edges of a path are checked and one of them is found to be infeasible, this result may have been caused by the choice of path *earlier on*: in other words, the infeasible segment that is removed from the search tree may not be the cause for the non-feasibility of the path; by choosing a different path beforehand, that edge could still be feasible. Also, sampled milestones (i.e. not in collision) may be located arbitrarily close to each-other but their interconnection might still be infeasible.

We can conclude here that the SBL, while efficient in standard cases and convenient because we had experience with it, might not have been the optimal planner to use in this case. A single directional planner without lazy evaluation such as EST would allow growing a tree of *feasible* branches from the start configuration in a natural, chronological manner. Notice that the constraint experienced here – i.e.the chronological evolution of the possible Brick angles – is stronger than common kino-dynamic ones which do not exclude growing a goal tree, since evolution of motion backwards in time is not debarred in that case.

8.4 On the Use of a 5-DoF Manipulator Arm for Object Grasping

The thesis at hand proved that a 5-DoF Manipulator Arm may be used to grasp certain objects at arbitrary position and orientation. The difficulties imposed by the lacking sixth DoF, however, are far wider than originally expected: first, an appropriate Gripper design involving rotation symmetry was supposed to easily overcome the constricted maneuverability. At the same time, the inverse kinematics can be solved analytically with this choice. Notice that adding the lacking roll DoF to the Wrist would allow a closed-form inverse kinematics solution for *arbitrary tool designs*.

The pure grasping problem was therefore solved mechanically. This came, however, with the price of having to allow a mostly uncontrolled gravity adjustment around the roll axis. The consequences thereof are extensive in terms of collision checking which in turn also affect the way motion planning may be performed.

Basically, all major difficulties experienced during the course of the project been caused by the lacking DoF. The essential lesson learned with this respect is probably that manipulators originally aimed to look at things must be adapted carefully to the new needs of grasping: having full control of both position and orientation is a necessary prerequisite when trying to avoid those issues. Also, this will allow designing a much more generic and flexible end effector capable of grasping a wide variety of objects apart from SPAWAR Communication Relays. Having a redundant Arm with at least 7 DoF is recommended, since that will allow avoiding singularities and expanding the workspace towards its inside.

9 Summary and Contributions

The major challenge during the first phase of the project was to design a Gripper capable of dealing with the lacking orientation DoF of the Arm. A rotation symmetric geometry was chosen making it invariant to that missing roll freedom and at the same time yielding analytical inverse kinematics. The Gripper grasps one of the two Brick flaps and form-locks a knob located on the top side of that flap inside a circular cavity in the upper Gripper blade. In terms of Gripper control, besides precise positioning also force feedback and limitation were necessary to be implemented, yielding robust operation.

As a second work package, two Arm control modes were implemented: on the one hand it may follow straight lines in Cartesian space and on the other hand in joint space by executing a pre-computed near-time-optimal trajectory.

In order to automate the Brick localization process, a stereo camera was mounted to the test setup. At this stage, however, minimal user interaction is necessary: four key-points need to be clicked in both images such that triangulation of the points and finally Brick position and orientation determination can be performed subsequently.

Towards the end of the project it became clear that motion planning needed to be added. Therefore, SBL, a state-of-the-art probabilistic planner was chosen for implementation. The related difficulties proved to be far wider than expected for the case where the Brick is grasped. Because the Brick is not attached rigidly to the Gripper and adjusts its orientation to gravity, the collision checker needed to undergo drastic changes. It is not the Arm configuration alone that determines whether a collision occurs. It is the Brick orientation that is in turn depending on the Arm configuration history which needs to be considered when checking for collision. Consequently, a sophisticated friction model had to be established allowing to predict the different possible angles at which the Brick may be attached to the Gripper. Since this implies that configurations may only be checked for collision in a chronological way, major adaptions to the SBL sampling, path checking and the path post-processing had to be performed.

The resulting setup works reliably and robustly with the only drawback of slightly high latencies in the motion planning step for the case with the Brick being grasped. It has been proven that a 5-DoF Arm may be used for grasping objects at arbitrary positions and orientations. The related difficulties, however, exceeded the expected amount dramatically. The conclusion to be drawn is hence obvious: 6 or even more DoF should be available for object grasping.

9.1 Future Work

The apparent next step comprises integrating the Arm and Gripper with the K10 rover. Besides the hardware and software integration as well as calibration and parameter identification, the following steps need to be approached:

- First of all, the test software will have to be set up as a client-server application, such that the K10 is not blocked due to running the user interface.
- Secondly, the Brick localization will need to be fully automated. On one hand, the Brick needs to be coarsely localized and the rover will need to be driven close to the Brick such that it is both in the field of view of the stereo camera as well as in the pickable workspace of the Arm. On the other hand, fiducial identification in the stereo images will be needed in order to locate the points that are now being selected by a user.
- Also, the stereo-reconstructed terrain must be used as the environment collision mesh instead of the idealized test bench L-shape.
- Finally, a drop-Brick procedure will have to be determined.

Looking at long term future work, the recommendation is made to equip the Arm with the lacking roll DoF in conjunction with designing a more generic Gripper not having to deal with lacking maneuverability.

A Kinematic Model

A.1 Forward Kinematics

The homogeneous transformation matrices T_{ij} transforming coordinates from frame $j = i+1$ to frame i are obtained in straight-forward manner as:

$$T_{01} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

$$T_{12} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & w\_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}$$

$$T_{23} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & s\_e \\ 0 & 0 & -1 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$T_{34} = \begin{bmatrix} 0 & 0 & 1 & e\_w \\ -\cos\theta_4 & \sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & e\_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$T_{45} = \begin{bmatrix} -\cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_5 & -\cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

The overall transformation from the Wrist frame (5) into the Base frame (0) is:

$$T_{05} = T_{01} \cdot T_{12} \cdot T_{23} \cdot T_{34} \cdot T_{45} \tag{A.6}$$

The Wrist position in the Base Frame is obtained as:

$$\begin{bmatrix} {}_0\overrightarrow{OW} \\ 1 \end{bmatrix} = \begin{bmatrix} {}_0\vec{w} \\ 1 \end{bmatrix} = T_{05} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} T_{05}(1,4) \\ T_{05}(2,4) \\ T_{05}(3,4) \end{pmatrix} \\ 1 \end{bmatrix} \tag{A.7}$$

And the tool pointing vector $\vec{p}$ that corresponds to the Wrist frame x-axis (i.e.${}_5\vec{p} = (1,0,0)^T$) becomes:

$$\begin{bmatrix} {}_0\vec{p} \\ 0 \end{bmatrix} = T_{05} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} T_{05}(1,1) \\ T_{05}(2,1) \\ T_{05}(3,1) \end{pmatrix} \\ 0 \end{bmatrix} \tag{A.8}$$

A.2 Analytical Inverse Kinematics

Figure A.2 shows a schematic Arm and all the subsidiary lines, distances and angles needed for the derivation of the analytical inverse kinematics.

The inverse kinematics function IK (.) takes the desired position ${}_0\vec{w} ={}_0 \overrightarrow{OW} = (x,y,z)^T$, pointing vector ${}_0\vec{p} = (p_x, p_y, p_z)^T$ as well as the start configuration $\theta_s = [\theta_{s,1}, \theta_{s,2}, \theta_{s,3}, \theta_{s,4}, \theta_{s,5}]^T$ as an input in order to obtain the joint angles (or configuration) θ:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \mathrm{IK}\left(\vec{w}, \vec{p}, \theta_s\right) \tag{A.9}$$

In the following, one of many ways of obtaining that function is being presented. First, the Wrist position problem is solved, since it is only determined by θ_1, θ_2 and θ_3. For a start, the Arm radius is needed:

$$r = \sqrt{x^2 + y^2 + (z - w\_s)^2} \tag{A.10}$$

Next, the important plane (S,E,F) is considered: it is the plane spanned by upper Arm and

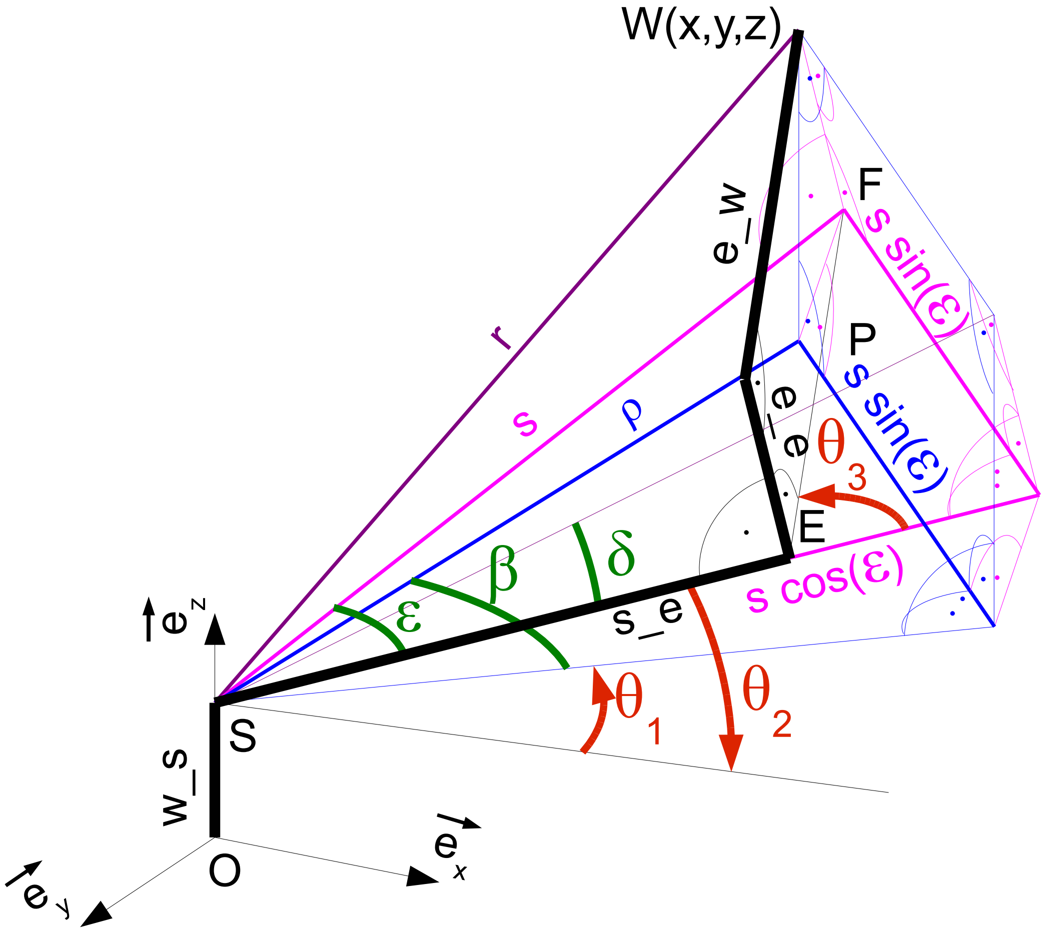


Figure 42: *Subsidiary sketch for th derivation of the inverse kinematics.*

forearm projected down along e_e. Side s corresponds to r projected into that plane.

$$s = \sqrt{r^2 - e\_e^2} \tag{A.11}$$

Applying the Cosine Law directly yields $|\theta_3|$.

$$|\theta_3| = \pi - \text{acos}\left(\frac{s^2 - e\_w^2 - s\_e^2}{-2 \cdot s\_e \cdot e\_w}\right) \tag{A.12}$$

The sign may be chosen positive for the 'right Elbow' solution or negative for 'left Elbow'.

Therefore the following variable is introduced:

$$rightelbow = \begin{cases} 1 & \text{if right Elbow desired;} \\ -1 & \text{if left Elbow desired.} \end{cases} \tag{A.13}$$

Now θ_3 is defined as:

$$\theta_3 = elbowright \cdot |\theta_3| \tag{A.14}$$

Applying the Cosine Law for a second time yields the following needed auxiliary angle

$$\epsilon = \text{acos}\left(\frac{e\_w^2 - s\_e^2 - s^2}{-2 \cdot s \cdot s\_e}\right) \tag{A.15}$$

Knowing ϵ, δ may be evaluated:

$$\delta = \text{atan2}\left(e\_e, s \cdot \cos\epsilon\right) \tag{A.16}$$

Next, the plane through S and P parallel to the xy-plane is considered. ρ is the projection of r into the plane:

$$\rho = \sqrt{x^2 + y^2} \tag{A.17}$$

Therefore:

$$\beta = \text{asin}\left(\frac{s \cdot \sin\epsilon}{\rho}\right) \tag{A.18}$$

Now, the two remaining angles can be evaluated:

$$\theta_2 = -\left(\text{asin}\left(\frac{z - w\_s}{\sqrt{r^2 - s^2 \cdot \sin^2\epsilon}}\right) - \delta\right) \tag{A.19}$$

$$\theta_1 = \text{atan2}\left(y, x\right) - elbowright \cdot \beta \tag{A.20}$$

There remain the last two angles θ_4 and θ_5 that adjust the pointing vector $\vec{p}$ to be solved for. Let R_{ij} be the upper left 3×3 rotation part of T_{ij}. $\vec{p}$ represented in the Elbow frame (3) becomes:

$$_3\vec{p} = R_{30} \cdot {}_0\vec{p} \tag{A.21}$$

with

$$R_{30} = (R_{01} \cdot R_{12} \cdot R_{23})^T \tag{A.22}$$

Now, the following equation must hold:

$$R_{34} \cdot R_{45} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = {}_3\vec{p} \tag{A.23}$$

which corresponds to:

$$\begin{pmatrix} \cos\theta_5 \\ \cos\theta_4 \cdot \sin\theta_5 \\ \sin\theta_4 \cdot \sin\theta_5 \end{pmatrix} = {}_3\vec{p} = \begin{pmatrix} {}_3p_1 \\ {}_3p_2 \\ {}_3p_3 \end{pmatrix} \tag{A.24}$$

First, θ_5 is obtained as

$$\theta_5 = \pm\text{acos}\left({}_3p_1\right) \tag{A.25}$$

Finally, θ_4 is determined as

$$\theta_4 = \pm\text{acos}\left(\frac{{}_3p_2}{\sin\theta_5)}\right) \tag{A.26}$$

Fulfilling the last equation A.27 will determine the sign of θ_4. Note that it will never create a discrepancy since $\vec{p}$ is of norm 1.

$$\sin\theta_4 \cdot \sin\theta_5 = {}_3p3 \tag{A.27}$$

The sign of θ_5 is chosen such that $||\theta - \theta_s||_2$ is minimized (minimal motion).

In the case of an orientation singularity with $\sin\theta_5 = 0$ (stretched Wrist), equation A.24 will allow an arbitrary choice for θ_4. In that case, θ_4 is set to $\theta_{s,4}$ in order to prevent unnecessary motion.

B Guide to the Current Software and Hardware Usage

B.1 Building the Software

Notice that you will need to run the software on RedHat 5.

B.1.1 Motion Planning Library

1. Check out the Motion Planning Library from: https://babelfish.arc.nasa.gov/svn/athletefootfall/branches/athletefootfall_RedHat5/motionPlanning/

2. Look at `./K10ArmConfig/SOURCEME.cshrc` and make sure you have the same environment variables set accordingly – including the `LD_LIBRARY_PATH`.

3. Edit the top-level `makefile.config`:

   - Set the `PLANNER_ROOT` variable.
   - The motion planning library requires `LAPACK`. Specify the respective path.

4. Change directory to `${PLANNER_ROOT}/Library/KrisLibrary/include` and edit the `makefile.config` the same way.

5. Still in the `KrisLibrary` directory build all the sub-libraries by `make all`. Then, build the `KrisLibrary` with `make KrisLibrary`.

6. Do a `cd ../..` to `${PLANNER_ROOT}/Library/`. Build the remaining libraries:

   - Configure `freeglut` with `./configure --prefix=./glut` and build it with `make; make install`.
   - Build `glui`, `PQP` and `Robotics` only typing `make`; `glpk` is not needed.

7. Change directory to `${PLANNER_ROOT}` and build the Motion Planning Library: `make ArmPlannerLib`. For testing you may also want to do a `make Test` and `make Display`. `Test` lets you play around with the Arm simulation including collision checker; `Display` tests the motion planner for a highly constrained situation with the Brick attached to the Arm.

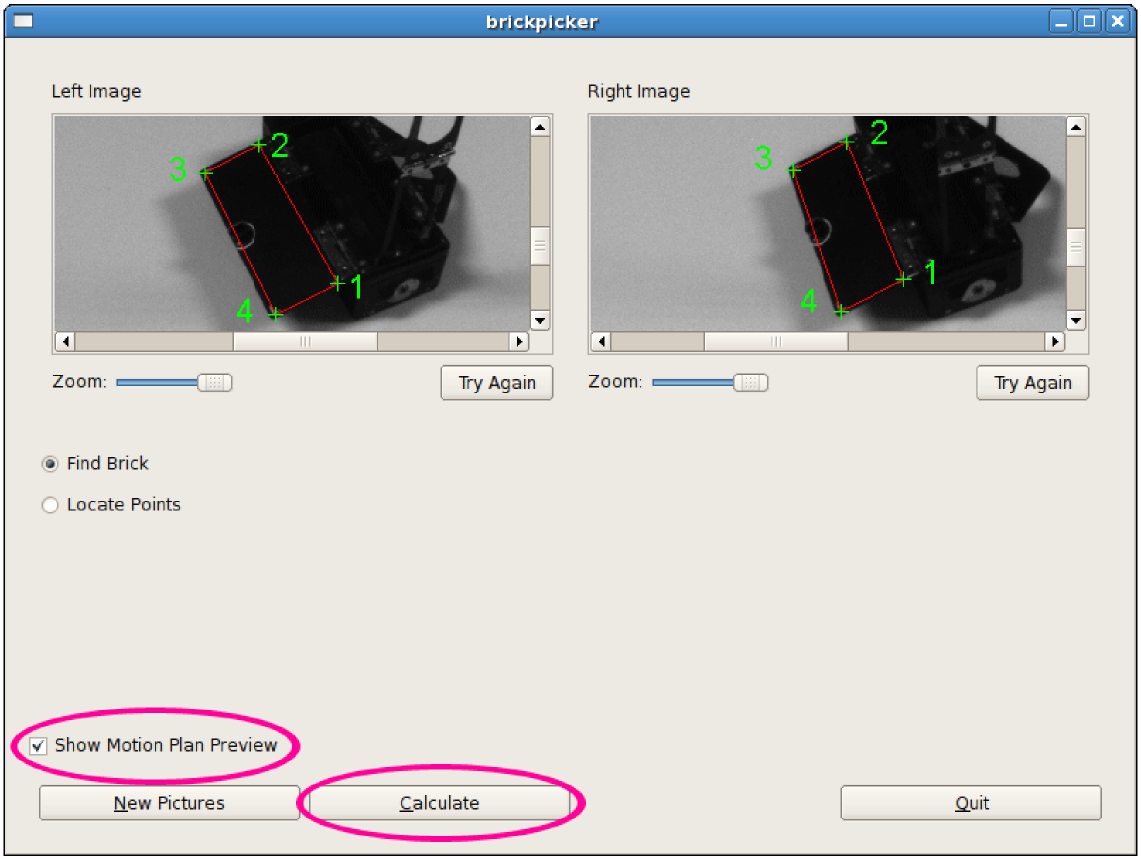
B.1.2 Rover Software

1. If not installed, check out `roversw` from:
   https://babelfish.arc.nasa.gov/svn/roversw/trunk/
2. Open `Makefile.top` and add `k9` to `SUBDIRS`.
3. Open `configure.sh` and uncomment the section on the k9 module.
4. Run `./configure.sh`. This will also build and install the whole `roversw` project.
5. The executable `brickpicker` is located in `./build/k9/src/k9arm/`. In `./k9/src/k9arm/` you will find all the source code; if you want to also build the test programs, edit the `CMakeLists.txt` located here accordingly.

B.2 Pick-Up Workflow

For help with the hardware setup see the wiring diagram in B.3. In the following, the standard workflow to use the test environment is described:

1. Make sure the stereo camera is plugged in and the Arm power supply is switched on.
2. The waist potentiometer cogwheel sometimes gets misaligned: make sure the two red dots match and realign carefully if necessary.
3. Run the `brickpicker` application. If this is the first time you run it after switching on the power supply, quit `brickpicker` and run it again: for some reason, the Wrist motor would not work the first time after power up.
4. Use zoom and scrollbars to see the flap of the Brick in both images at maximum resolution. You may also take 'New Pictures', if the position of the Brick has changed.
5. Click the marked white corners on the flap in both images: it is important that you start with the point closest to the antenna and go around in mathematically positive sense (CCW). Make sure the points are selected precisely, since this is critical for the localization. Use the 'Try Again' button if you are not satisfied with the clicked points.



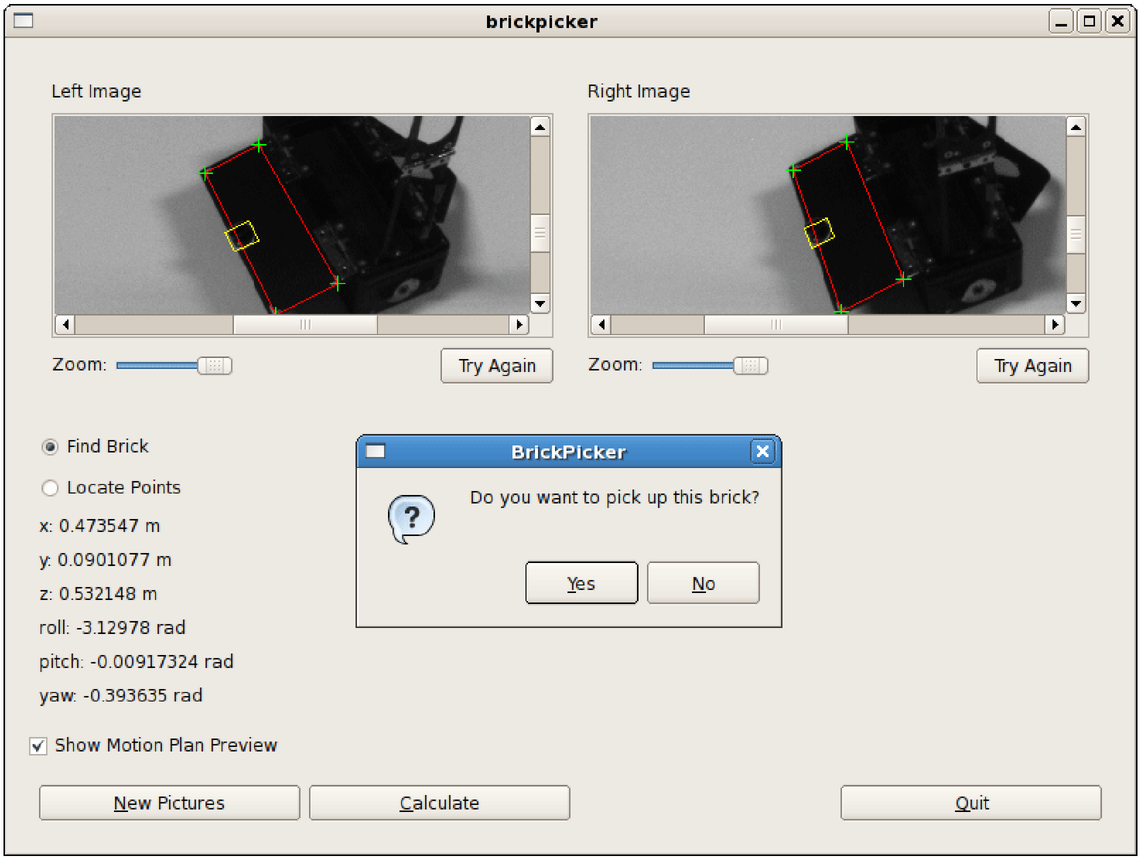
● Find Brick

○ Locate Points

☑ Show Motion Plan Preview

6. Select the 'Show Motion Plan Preview' checkbox if you want to see the result of the planner which is highly recommended for testing.

7. Click the 'Calculate' button: the Brick is localized and the result can be verified both graphically in the form of a forward projection of the knob location (and orientation) into the two pictures as well as by checking the displayed coordinates and roll/pitch/yaw-parameters.



● Find Brick

○ Locate Points

x: 0.473547 m

y: 0.0901077 m

z: 0.532148 m

roll: -3.12978 rad

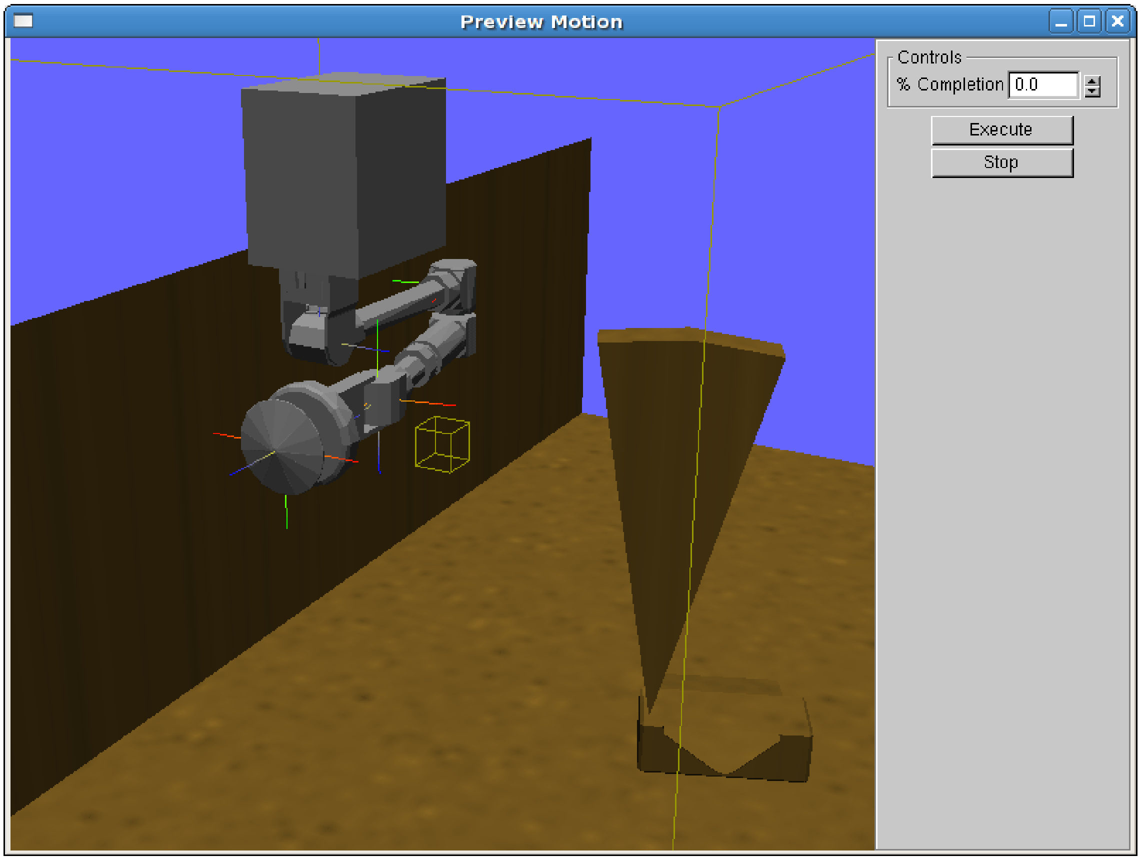
pitch: -0.00917324 rad

yaw: -0.393635 rad

☑ Show Motion Plan Preview

Do you want to pick up this brick?

8. The planned motion to the approach position is viewable after usually little latency in a new window:



% Completion 0.0

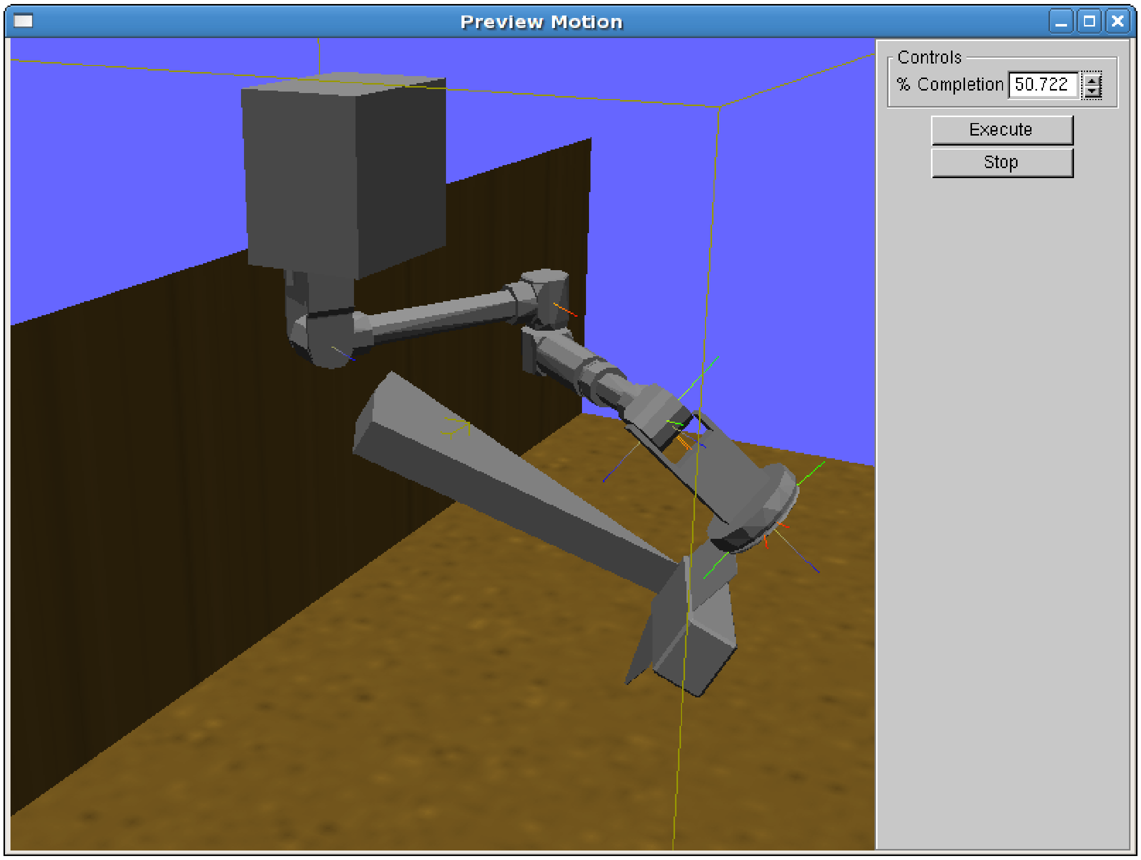
- Use the `Mouse_Button + Mouse` to change the camera view angle.
- Use `Mouse_Button + Shift + Mouse_Up/Mouse_Down` to zoom out/in.
- Use `Mouse_Button + Ctrl + Mouse` to pan.
- Adjust the degree of completion using the `up/down` arrows of the spinner in conjunction with `Mouse_Button + Mouse_Up/Mouse_Down`

Press 'Execute' or 'Stop'; it is very important *never* to close the window, else you will not be able to preview anymore.

9. The Arm moves to pre-approach position.
10. Another motion preview is shown for the little motion to the effective approach position very close to the Brick; notice that the Brick was therefore removed from the environment.

11. The Brick is grasped and lifted a little. Make sure the grasping is performed successfully; there is currently no automatic reaction if not.

12. The planning of a motion with the Brick to the carry position may take around two minutes in certain cases, so be patient. At success, the preview is shown. Make sure you preview the motion chronologically, you should not go back. There might be very short periods of detected collisions that you may ignore originating from a different checking resolution in the preview than in the motion planner.



% Completion 50.722

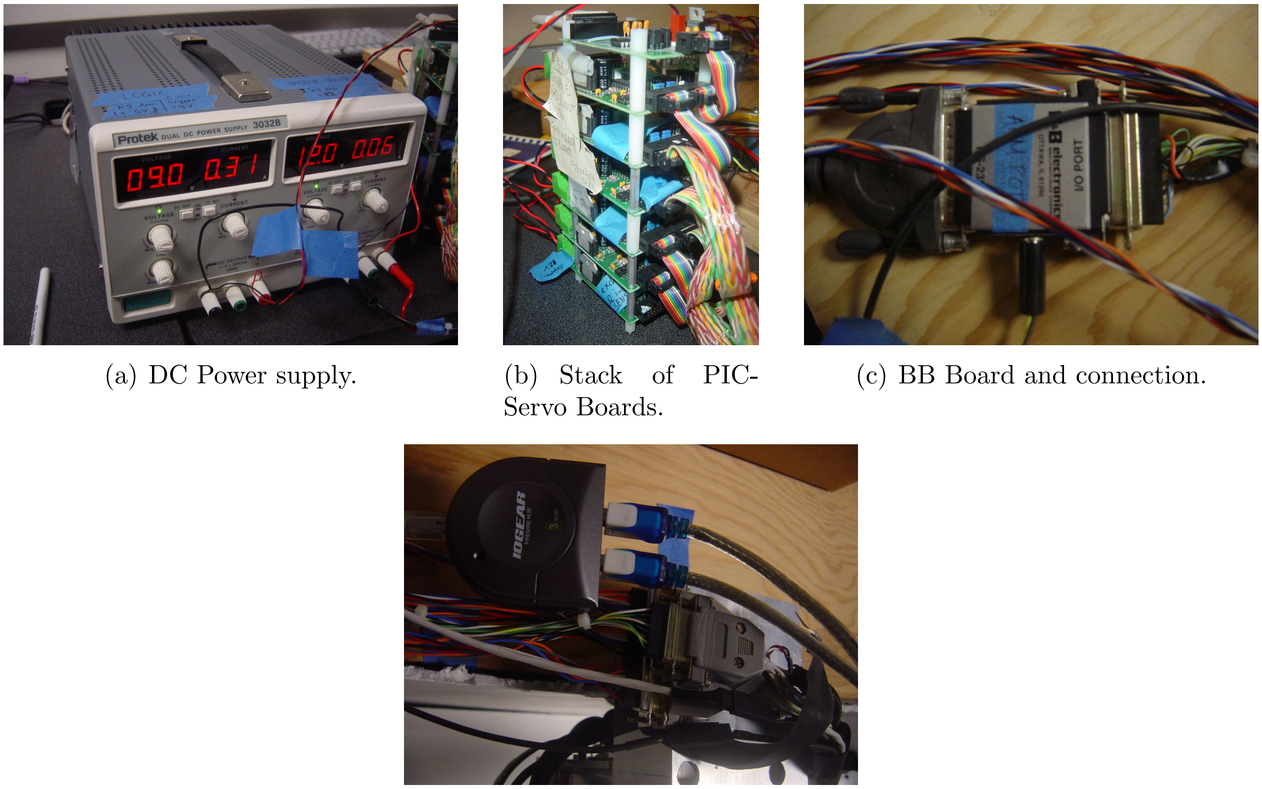
13. Watch the Arm take the Brick to carry position.

14. To conclude the cycle, you are prompted to hold the Brick: the Gripper will open (and close immediately thereafter). Finally, new pictures are taken and you may start a new pick-up cycle.

If something fails fatally, you will have the ability of switching off the power and aborting the program execution with `Ctrl + C`. Run the `test_k10Arm` program to recover from the respective situation by selecting the appropriate steps: usually '`open Gripper`' ('`o`') followed by '`home`' ('`h`').

B.3 Wiring

You will need a DC power supply with 9V output for the PIC-Servo logic supply as well as 12V for powering Arm motors, the Gripper, and the BB Board. See the respective documentation of PIC-Servo Boards and BB Board. The wiring between boards, Arm servo motors, Gripper motor and the potentiometers was set up as shown on the next page. Figure 43 shows the important components and connections of the used hardware setup.



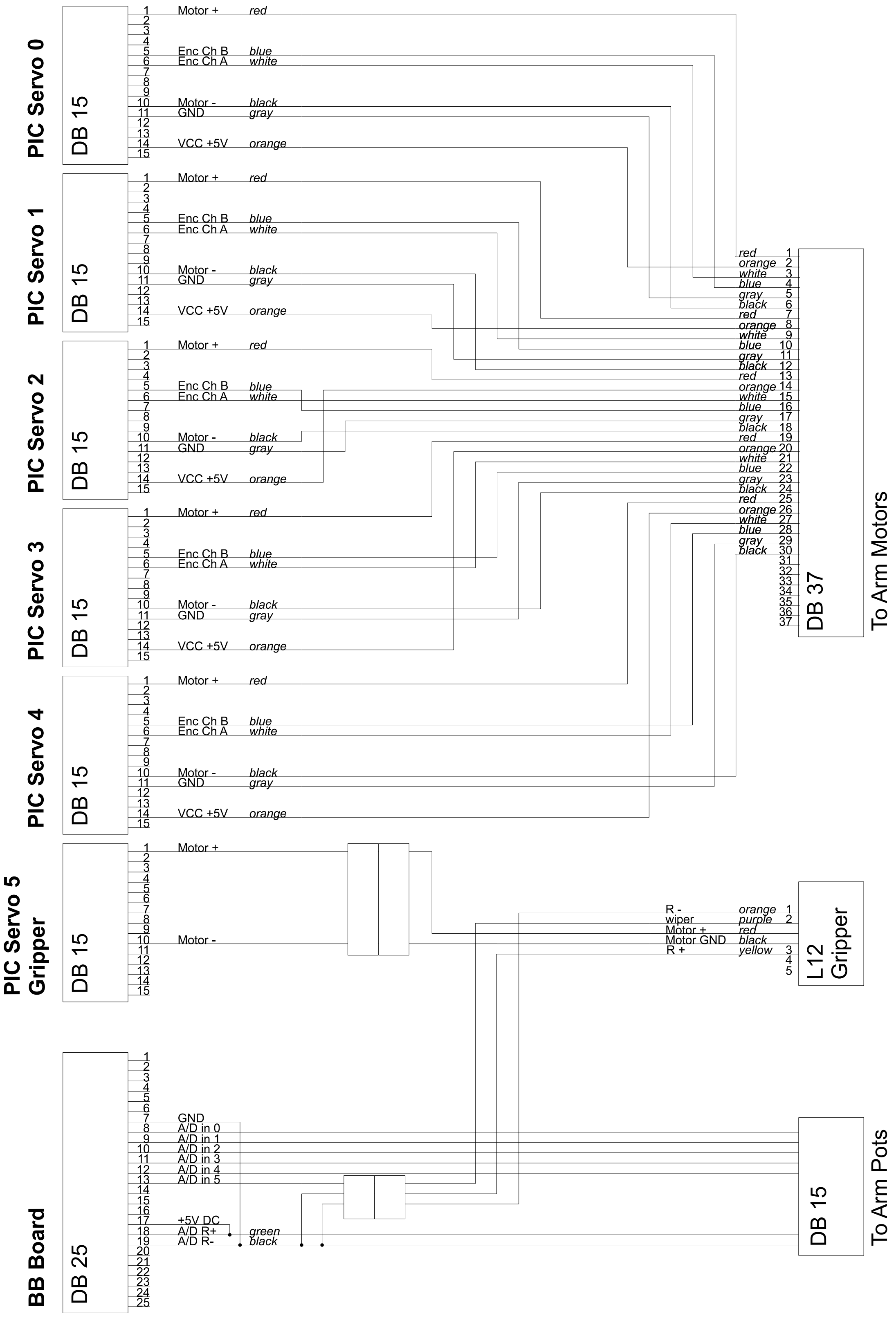
(a) DC Power supply.

(b) Stack of PIC-Servo Boards.

(c) BB Board and connection.

(d) Connectors.

Figure 43: *Hardware components and connections.*


PIC Servo 0
DB 15
PIC Servo 1
DB 15
PIC Servo 2
DB 15
PIC Servo 3
DB 15
PIC Servo 4
DB 15
PIC Servo 5
Gripper
DB 15
BB Board
DB 25
Motor +
red
Enc Ch B
blue
Enc Ch A
white
Motor -
black
GND
gray
VCC +5V
orange
DB 37
To Arm Motors
R -
wiper
Motor +
Motor GND
R +
orange
purple
red
black
yellow
L12
Gripper
A/D in 0
A/D in 1
A/D in 2
A/D in 3
A/D in 4
A/D in 5
+5V DC
A/D R+
A/D R-
green
black
DB 15
To Arm Pots

B.4 Code Overview

This Section only aims at giving an overview on how the code is organized rather than providing a complete description of all classes and members.

B.4.1 Parameters

`k10ArmParameters.xml` contains all the default parameter values as initialized in the automatically generated parameter classes (see `k10ArmParameters.h`) in the `build` directory. You may use the parameter configuration framework for parameter tuning rather than changing `k10ArmParameters.xml` and recompiling.

B.4.2 Most Important Classes

- `class Gripper`
  This is the Gripper driver: notice that the `moveTo`, `open`, and `close` functions are to be used in a loop; always call the `init` function before to initialize measurement filtering and the controller.

- `class ArmKinematics`
  `ArmKinematics` objects have the main functionality of performing forward and inverse kinematics of the arm. It is mainly used for trajectory generation.

- `class Trajectory`
  Generate either a joint space (`generateJ`) or Cartesian space trajectory (`generate`).

- `class Arm`
  This is the Arm driver: look at `class BrickPickGui` to see how it may be used. Notice that malfunctions are logged and accessible.

- `class BrickFinder`
  This class is an interface to the usage of the stereo camera; the `.tsai` files store the individual camera models. Notice that the extrinsic parameters of the left camera are set to zero while the right extrinsic parameters define the calibrated relative position of the right camera with respect to the left one.

- `class BrickPickGui`
  This is the main QT widget. It also defines the workflow of the Arm operation.

B.4.3 Motion Planning Library

The most important folders are `K10ArmRobot/` containing the Arm specific source files as well as `K10ArmConfig/` containing settings as well as the triangulated meshes of Arm, Brick and environment.

The implementation of the friction model and different collision checking modes are found in `k10ArmRobot.cpp`. The necessary adaptions, however also affected the `Robotics` Library `Library/Robotics/`: major changes were necessary to `SBL2.cpp` as well as to `SBLTree.cpp/`.

In conjunction with switching to `freeglut` (Open GL Viewer), adaptions to the `KrisLibrary` were necessary as well.

B.5 Useful Hints and Known Problems of Hard- and Software

The following three subtleties are addressed in order to prevent the future user/developer from going through some of the writer's troubles.

B.5.1 Calibration of the Stereo Camera Location

The calibration of the stereo camera frame location and orientation is a time consuming process that has not been gone through yet at the extent necessary for precise point localization[7]. You may use the `brickpicker` application to localize arbitrary single points by toggling the option 'Locate Points': this will allow comparing their position measured by the stereo pair to their precisely known position with respect to the Arm Base frame. Choose a reasonable amount of points within the Arm workspace. Notice that points selected near the edge of either left or right picture are likely to yield inaccurate positions due to the individual camera distortion models being inaccurate there.

[7] Only few calibration points close to the camera were measured and the parameters were estimated approximately and adapted rather than rigorously applying a least square error minimization.

B.5.2 Closing of Motion Preview Window

Unfortunately, `glut` must not be restarted during the execution of an application. Therefore it is not possible to close the motion preview window (terminating `glut`) and reopen it. To be precise, it can be done, but some memory allocation problems will occur (also affecting the actual display). Apparently, there is also a related bug in the XWindow system.

A QT GL Viewer alternative is currently being developed as part of the ATHLETE project. Hence it is recommended to switch to a similar embedded preview widget in the long run.

B.5.3 Cartesian Space Trajectory Generation Failure

The different termination conditions in the Cartesian space trajectory generation are somewhat fragile; consider an unsuccessful generation step to be the cause in case the application stops responding. The most recent version has not shown related problems, however. Also the inverse kinematics have not been tested with an extensive amount of singular cases, therefore related problems have the potential of causing trajectory generation to fail as well.

B.6 Gripper Potentiometer Occasional Short

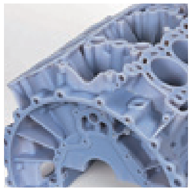
Inside the L12 actuator, a short had been found between the reference voltage inputs. A quick fix was applied involving some isolation tape. In order to protect the BB Board from future possible occurrences, a series resistor was inserted.

For the reason of comparable problems when the actuator is fully retracted, the washer between barrel nut and actuator shaft was removed and the parameters were adapted; hence, it will never be totally retracted and the corresponding position signal inaccuracy is avoided.

C Specs Sheets

C.1 FullCure Rapid Prototyping Material

FULLCURE® FullCure® Materials Datasheet



Properties	Standard Procedure	FullCure® 720	VeroWhite	VeroBlue	VeroBlack	TangoBlack	TangoGray
Tensile Strength MPa	D-638	60.3	49.8	55.1	50.7	-	-
Elongation at break, %	D-638	15%-25%	15%-25%	15%-25%	17.7%	-	-
Modulus of Elasticity, MPa	D-638	2,870.0	2,495.0	2,740.0	2,192.0	-	-
Flexural Strength, MPa	D790	75.8	74.6	83.6	79.6	-	-
Flexural Modulus, MPa	D790	1,718.0	2,137.0	1,983.0	2,276.0	-	-
Izod Notched Impact, J/m	D256	39.6	37.5	42.5	-	-	-
Compression Strength, MPa	D695	84.3	-	79.3	-	-	-
SHORE	Scale D	83.0	83.0	83.0	83.0	-	-
Rockwell	Scale M	81.0	81.0	81.0	-	-	-
Heat Distortion Temperature, °C	D648 @ 0.45Mpa	48.4	47.6	48.8	47	-	-
	@ 1.82Mpa	44.4	43.6	44.8	42.9	-	-
Tg, °C	DMA, E”	48.7	58.0	48.7	62.7	-	-
A sh Content		‹0.01%	‹0.40%	‹0.30%	-	-	-
Tensile Strength MPa	ASTM D - 412	-	-	-	-	2.0	4.36
Elongation at break, %	ASTM D - 412	-	-	-	-	47.7	47.0
Compression set, %	ASTM D - 395	-	-	-	-	0.8	1.0
SHORE A Hardness, Deg	ASTM D - 2240	-	-	-	-	61.0	75.0
Tensile Tear Resistance, Kg/cm	ASTM D - 624	-	-	-	-	3.8	9.5
Tg, °C	DSC (-80 °c +100 °c)	-	-	-	-	-10.7	+2.6

Objet Geometries Ltd.
Headquarters
2 Holtzman St.,
Science Park,
P.O. Box 2496,
Rehovot 76124, Israel
Tel: +972 8 931 4314
Fax: +972 8 931 4315

Objet Geometries Inc.
North America
5 Fortune Drive
Billerica,
MA 01821
USA
Tel: 1-877-489-9449
Fax: 1-866-676-1533

Objet Geometries AP Ltd.
Asia Pacific
Unit 52/A 13/F HITEC,
1 Trademart Drive,
Kowloon Bay,
Hong Kong
Tel: +852-217-40111
Fax: +852-217-40555

Objet Geometries Ltd.
Europe
Leuvensesteenweg 388
1932 Sint-Stevens-Woluwe
Belgium
Tel: +32 2 717 6502
Fax: +32 2 717 6500

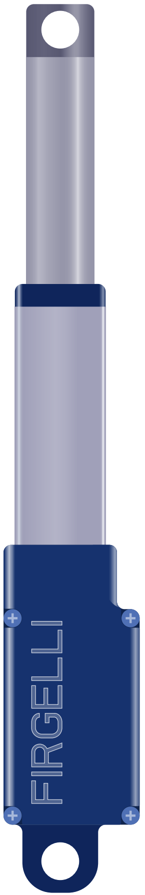
www.2objet.com info@2objet.com

C.2 Firgelli L12 Gripper Motor

Miniature Linear Motion Series • **L12**

Firgelli Technologies' unique line of Miniature Linear Actuators enables a new generation of motion-enabled product designs, with capabilities that have never before been combined in a device of this size. These small linear actuators are a superior alternative to designing with awkward gears, motors, servos and linkages.

Firgelli's L series of micro linear actuators combine the best features of our existing micro actuator families into a highly flexible, configurable and compact platform with an optional sophisticated on-board microcontroller. The first member of the L series, the L12, is an axial design with a powerful drivetrain and a rectangular cross section for increased rigidity. But by far the most attractive feature of this actuator is the broad spectrum of available configurations.



Benefits

- Compact miniature size
- Simple control using industry standard interfaces
- Low voltage
- Equal push / pull force
- Easy mounting

Applications

- Robotics
- Consumer appliances
- Toys
- Automotive
- Industrial automation

L12 Specifications

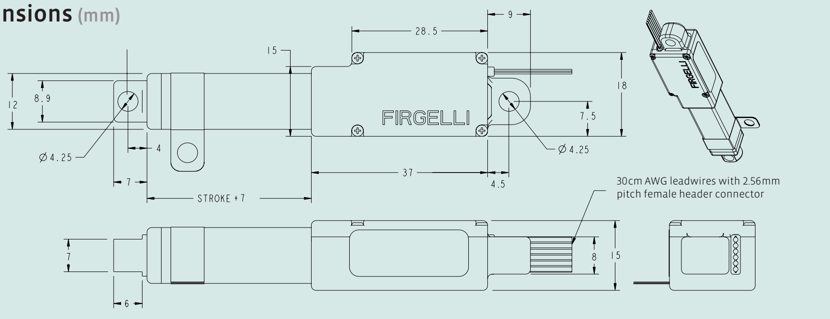
Gearing Option	30	63	100	210	298
Peak Power Point [1]	8 N @ 16 mm/s	14 N @ 8 mm/s	23 N @ 6 mm/s	45 N @ 2.5 mm/s	67 N @ 2 mm/s
Peak Efficiency Point	4.5 N @ 23 mm/s	7.5 N @ 12 mm/s	12 N @ 8 mm/s	18 N @ 4 mm/s	30 N @ 3 mm/s
Max Speed (no load)	33 mm/s	16 mm/s	12 mm/s	5 mm/s	4 mm/s
Backdrive Force [2]	27 N	55 N	80 N	150 N	230 N
Positional Accuracy	0.2 mm	0.2 mm	0.1 mm	0.1 mm	0.1 mm

Stroke Option	10 mm	30 mm	50 mm	100 mm
Weight	35 g	37 g	39 g	43 g
Max Side Force (fully extended)	50 N	40 N	30 N	15 N

Mechanical Backlash	0.1 mm
Feedback Potentiometer	2.75 kΩ/mm ± 30%, 1% linearity
Duty Cycle	20%
Lifetime	1000 hours at rated duty cycle
Operating Temperature	–10°C to +50°C
Storage Temperature	–30°C to +70°C
Ingress Protection Rating	IP–54
Audible Noise	55 dB at 45 cm
Stall Current	450 mA at 5 V & 6 V, 200 mA at 12 V

[1] 1 N (Newton) = 0.225 lb_f (pound-force)
[2] a powered-off actuator will statically hold a force up to the Backdrive Force

Dimensions (mm)


28.5
9
15
18
12
8.9
7.5
FIRGELLI
Ø 4.25
4
Ø 4.25
37
7
4.5
STROKE + 7
30cm AWG leadwires with 2.56mm pitch female header connector
7
8
15
6

Firgelli Technologies Inc.

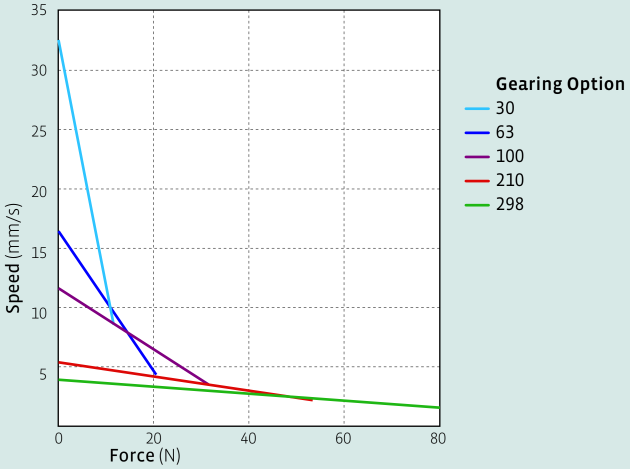
3020 152nd Street, 2nd Floor
Surrey, BC V4P 3N7
Canada

1 (206) 347-9684 *phone*
1 (888) 225-9198 *toll-free*
1 (206) 347-9684 *fax*

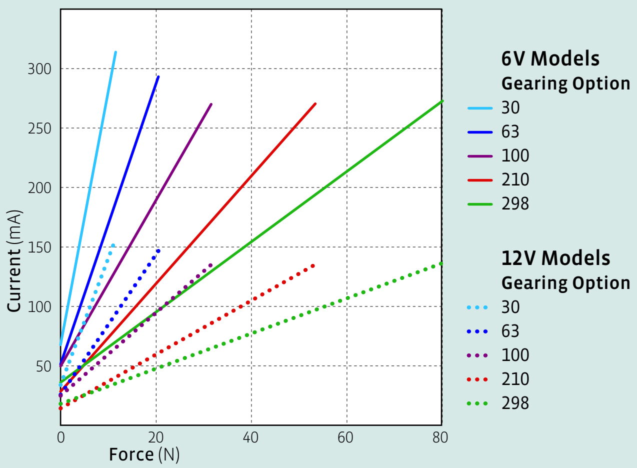
sales@firgelli.com
www.firgelli.com

L12 Specifications

Load Curves



Current Curves



Model Selection

The L12 has five configurable features. L12 configurations are identified according to the following scheme:

L12-SS-GG-VV-C-L

feature	*options*
SS: Stroke Length (in mm)	**10, 30, 50, 100** Any stroke length between 10 and 100 mm is available on custom orders, in 2 mm increments.
GG: Gear reduction ratio (refer to force/speed plots)	**30, 63, 100, 210, 298** Other gearing options may be possible on custom orders.
VV: Voltage	**06** 6 V (5 V power for Controller options B and P) **12** 12 V
C: Controller	**B** Basic 2-wire open-loop interface, no position feedback, control, or limit switching. Positive voltage extends, negative retracts. **S** 2-wire open-loop interface (like B option) with limit switching at stroke endpoints. **P** Simple analog position feedback signal, no on-board controller. **I** Integrated controller with Industrial and RC servo interfaces (see L12 Controller Options section). Not available with 10mm stroke length configurations.
L: Mechanical or electrical interface customizations	Custom option codes will be issuedby Firgelli for custom builds when applicable.

Basis of Operation

The L12 actuator is designed to move push or pull loads along its full stroke length. The speed of travel is determined by the gearing of the actuator and the load or force the actuator is working against at a given point in time (see Load Curves chart on this datasheet). When power is removed, the actuator stops moving and holds its position, unless the applied load exceeds the backdrive force, in which case the actuator will backdrive. Stalling the actuator under power for short periods of time (several seconds) will not damage the actuator. Do not reverse the supply voltage polarity to actuators containing an integrated controller (I controller option).

Each L12 actuator ships with two mounting clamps, two mounting brackets and two rod end options: a clevis end and a threaded end with nut (see drawing on page 4). When changing rod ends, extend the actuator completely and hold the round shaft while unscrewing the rod end. Standard lead wires are 28 AWG, 30 cm long with 2.56 mm (0.1") pitch female header connector (Hi-Tec™ and Futaba™ compatible). Actuators are a sealed unit (IP–54 rating, resistant to dust and water ingress but not fully waterproof).

Ordering information

Sample quantities may be ordered with a credit card directly from www.firgelli.com.

Please contact Firgelli at sales@firgelli.com for volume pricing or custom configurations.

Note that not all configuration combinations are stocked as standard products. Please refer to the current **L12 configuration sheet** at www.firgelli.com/L12 for available configurations.

L12 Controller options

Option B—Basic 2-wire interface

WIRING:

1 (red)	**Motor V+** (5 V or 12 V)
2 (black)	**Motor ground**

The –B actuators offer no control or feedback mechanisms. While voltage is applied to the motor V+ and ground leads, the actuator extends. If the polarity of this voltage is reversed, the actuator retracts. The 5 V actuator is rated for 5 V but can operate at 6 V.

Option S—Basic 2-wire interface

WIRING:

1 (red)	**Motor V+** (5 V or 12 V)
2 (black)	**Motor ground**

When the actuator moves to a position within 0.5mm of its fully-retracted or fully-extended stroke endpoint, a limit switch will stop power to the motor. When this occurs, the actuator can only be reversed away from the stroke endpoint. Once the actuator is positioned away from it's stroke endpoint, normal operation resumes. For custom orders, limit switch trigger positions can be modified at the time of manufacture, in 0.5mm increments.

Option P—Position feedback signal

WIRING:

1 (orange)	**Feedback potentiometer negative reference rail**
2 (purple)	**Feedback potentiometer wiper** (position signal)
3 (red)	**Motor V+** (5 V or 12 V)
4 (black)	**Motor ground**
5 (yellow)	**Feedback potentiometer positive reference rail**

The –P actuators offer no built-in controller, but do provide an analog position feedback signal that can be input to an external controller. While voltage is applied to the motor V+ and ground leads, the actuator extends. If the polarity of this voltage is reversed, the actuator retracts. Actuator stroke position may be monitored by providing any stable low and high reference voltages on leads 1 and 5, and then reading the position signal on lead 2. The voltage on lead 2 will vary linearly between the two reference voltages in proportion to the position of the actuator stroke.

Option I—Integrated controller with industrial and RC servo interfaces

WIRING:

1 (green)	**Current input signal** (used for 4–20 mA interface mode)
2 (blue)	**Voltage input signal** (used for the 0–5V interface mode and PWM interface modes)
3 (purple)	**Position Feedback signal** (0–3.3 V, linearly proportional to actuator position)
4 (white)	**RC input signal** (used for RC-servo compatible interface mode)
5 (red)	**Motor V+** (+6 Vdc for 6 V models, +12 Vdc for 12 V models)
6 (black)	**Ground**

The –I actuator models feature an onboard software-based digital microcontroller. The microcontroller is not user-programmable. Custom controller software development may be available as a service from Firgelli.

The six lead wires are split into two connectors. Leads 4, 5 and 6 terminate at a universal RC servo three-pin connector (Hi-Tec™ and Futaba™ compatible). Leads 1, 2 and 3 terminate at a separate, similarly sized connector.

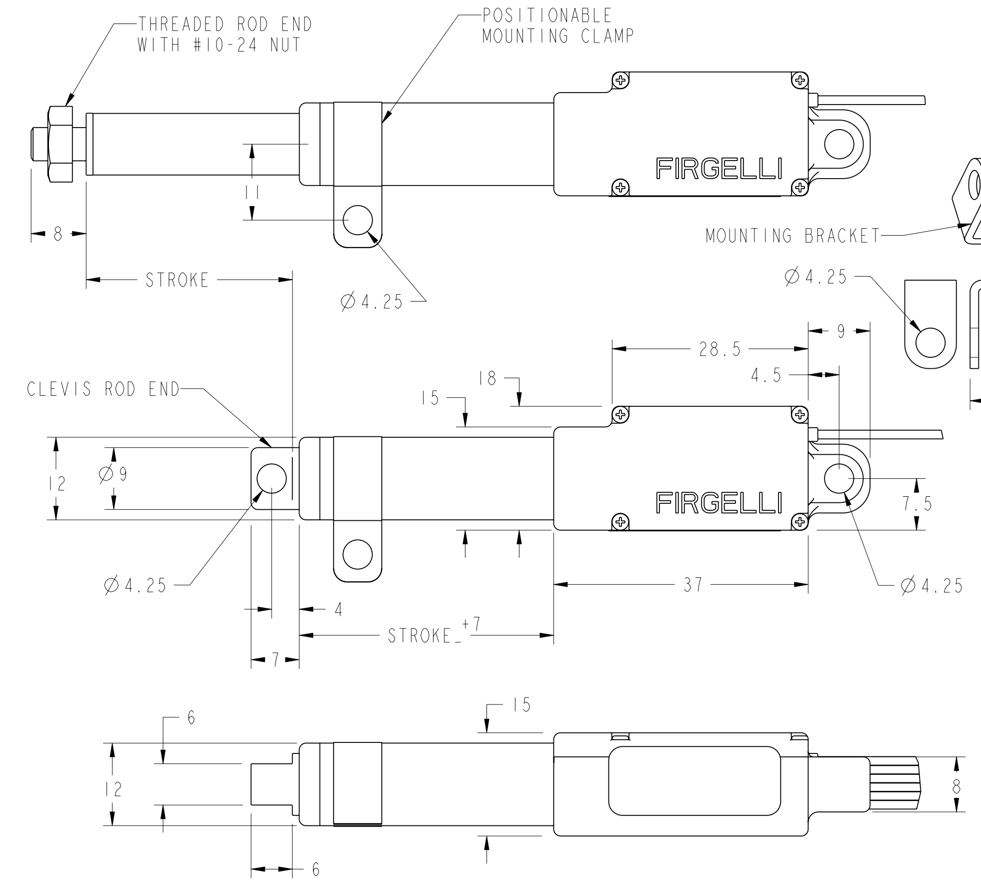
When the actuator is powered up, it will repeatedly scan leads 1, 2, 4 for an input signal that is valid under any of the four supported interface modes. When a valid signal is detected, the actuator will self-configure to the corresponding interface mode, and all other interface modes and input leads are disabled until the actuator is next powered on.

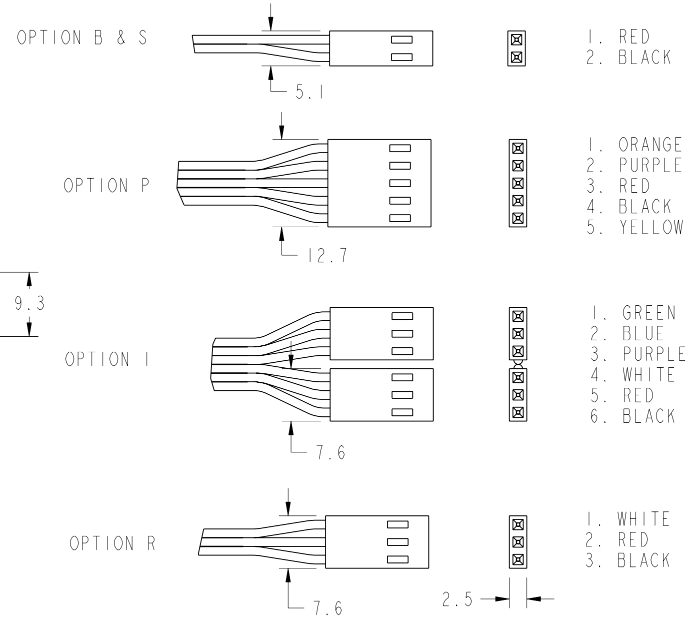
0–5 V Interface Mode: This mode allows the actuator to be controlled with just a battery, and a potentiometer to signal the desired position to the actuator – a simple interface for prototypes or home automation projects. The desired actuator position (setpoint) is input to the actuator on lead 2 as a voltage between ground and 5V. The linear relationship between the input voltage and the actuator position is determined by the formula $I = 5P \div S$, where I is the input voltage (V), P is the desired actuator stroke position (mm) and S is the full stroke length of the actuator model (mm). The setpoint voltage must be held on lead 1 until the desired actuator stroke position is reached. Lead 2 is a high impedance input.

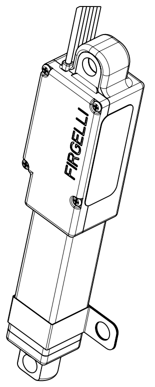
4–20 mA Interface Mode: This mode is compatible with PLC devices typically used in industrial control applications. The desired actuator position (setpoint) is input to the actuator on lead 1 as a current between 4 mA and 20 mA. The linear relationship between the input current and the actuator position is determined by the formula $I = (16P \div S) + 4$, where I is the input current (mA), P is the desired actuator stroke position (mm) and S is the full stroke length of the actuator model (mm). The setpoint current must be held on lead 3 until the desired actuator stroke position is reached.

RC Servo Interface Mode: This is a standard hobby-type remote-control digital servo interface (CMOS logic), compatible with servos and receivers from manufacturers like Futaba™ and Hi-Tec™. The desired actuator position is input to the actuator on lead 4 as a positive 5 Volt pulse width signal. A 1.25 ms pulse commands the controller to fully retract the actuator, and a 1.75 ms pulse signals full extension. The linear relationship between the input signal and the position setpoint is determined by the formula $I = (0.5P \div S) + 1.25$, where I is the input pulse width (ms), P is the desired actuator stroke position (mm) and S is the full stroke length of the actuator model (mm). If the motion of the actuator, or of other servos in your system, seems erratic, place a 1–4Ω resistor in series with the actuator's red V+ leadwire.

PWM Mode: This mode allows control of the actuator using a single digital output pin from an external microcontroller. The desired actuator position is encoded as the duty cycle of a 5 Volt 1 kHz square wave on actuator lead 2, where the % duty cycle sets the actuator position to the same % of full stroke extension. 100% duty cycle represents full extension, and 0% duty cycle represents full retraction. The waveform must be 0V to +5V in order to access the full stroke range of the actuator.







SCALE 1:1

NOTES:

1) WHEN CHANGING ROD ENDS, EXTEND ACTUATOR COMPLETELY AND HOLD THE ROUND SHAFT WHILE UNSCREWING THE ROD END PLUG

2) ∅4.25 MOUNTING HOLES WILL FIT M4 OR UN #8 SIZE SCREWS

3) TWO MOUNTING CLAMPS, TWO MOUNTING BRACKETS, ONE EXTRA THREADED ROD END WITH NUT AND #8-32 HARDWARE, IS INCLUDED WITH EACH ACTUATOR.

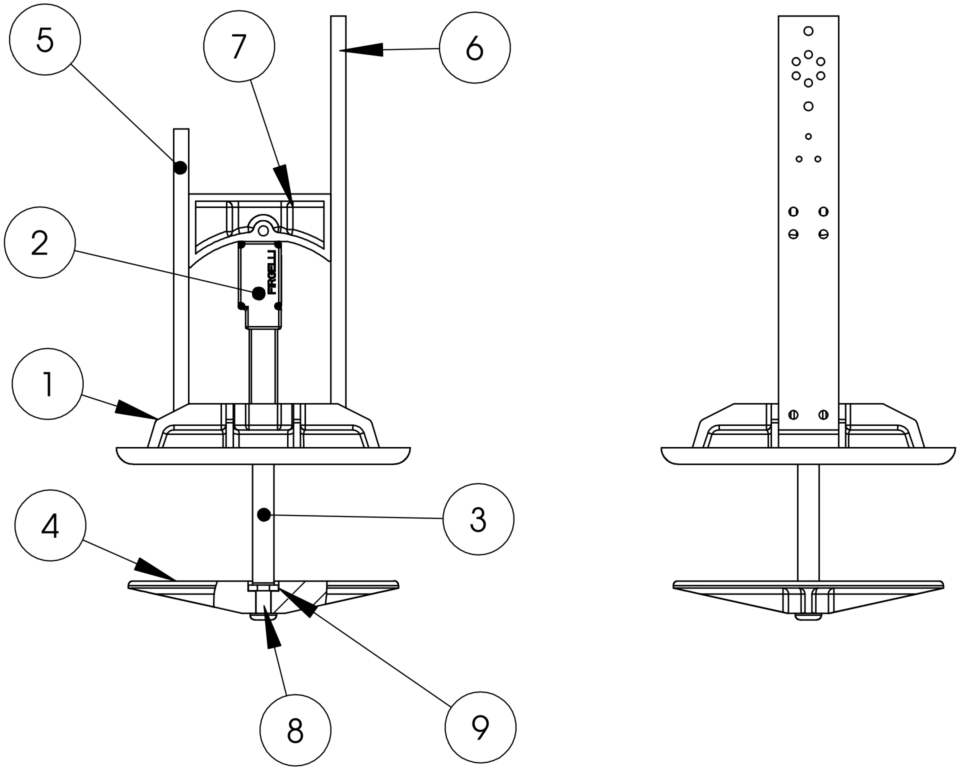
4) STANDARD LEAD WIRE LENGTH IS 300mm. 28AWG STRANDED RIBBON WIRE. CONNECTORS ARE 2.54mm PITCH FEMALE HEADER CONNECTORS

FIRGELLI

L12 ACTUATOR FAMILY
INTERFACE DIMENSIONS

SCALE
3:2

C.3 Gripper Drawings



ITEM NO.	PART NUMBER	QTY.
1	Upper	1
2	I12-30-i-ret	1
3	L12Rod	1
4	LowerShell	1
5	MountL	1
6	MountR	1
7	Fixation	1
8	BarrelNut	1
9	Washer	1

		NAME	DATE
	DRAWN	S.L.	04/07/08
	CHECKED		
	ENG APPR.		
	MFG APPR.		
	Q.A.		
	COMMENTS:		

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

INTERPRET GEOMETRIC
TOLERANCING PER:

MATERIAL

FINISH

DO NOT SCALE DRAWING

NEXT ASSY	USED ON
APPLICATION	

IRG

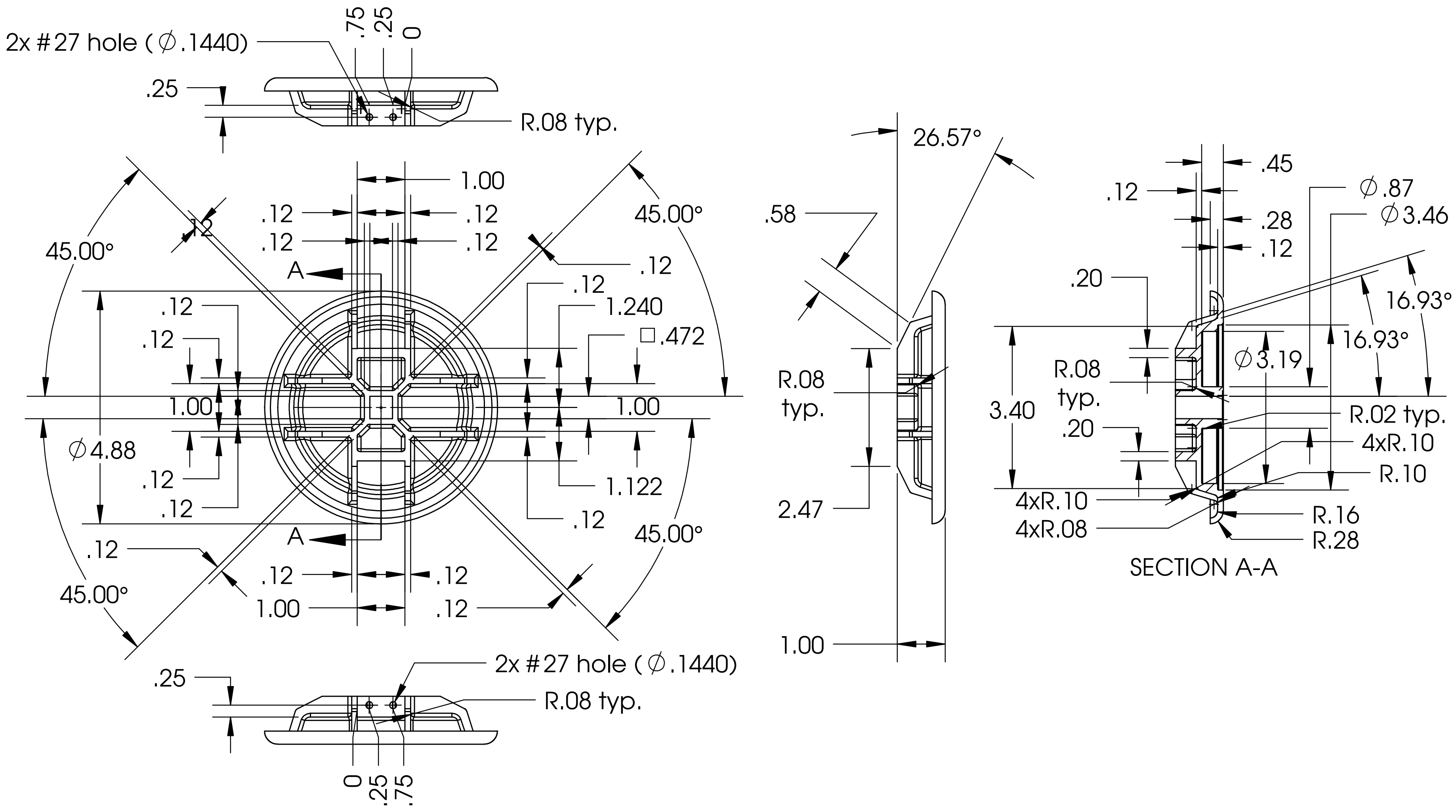
TITLE:

Gripper Assem.

SIZE A DWG. NO. Gripper REV

SCALE: 1:3 WEIGHT: SHEET 1 OF 1

5 4 3 2 1



	NAME	DATE
DRAWN	S.L.	04/05/08
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

COMMENTS:

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

INTERPRET GEOMETRIC
TOLERANCING PER:

MATERIAL

FINISH

DO NOT SCALE DRAWING

NEXT ASSY	USED ON
APPLICATION	

IRG

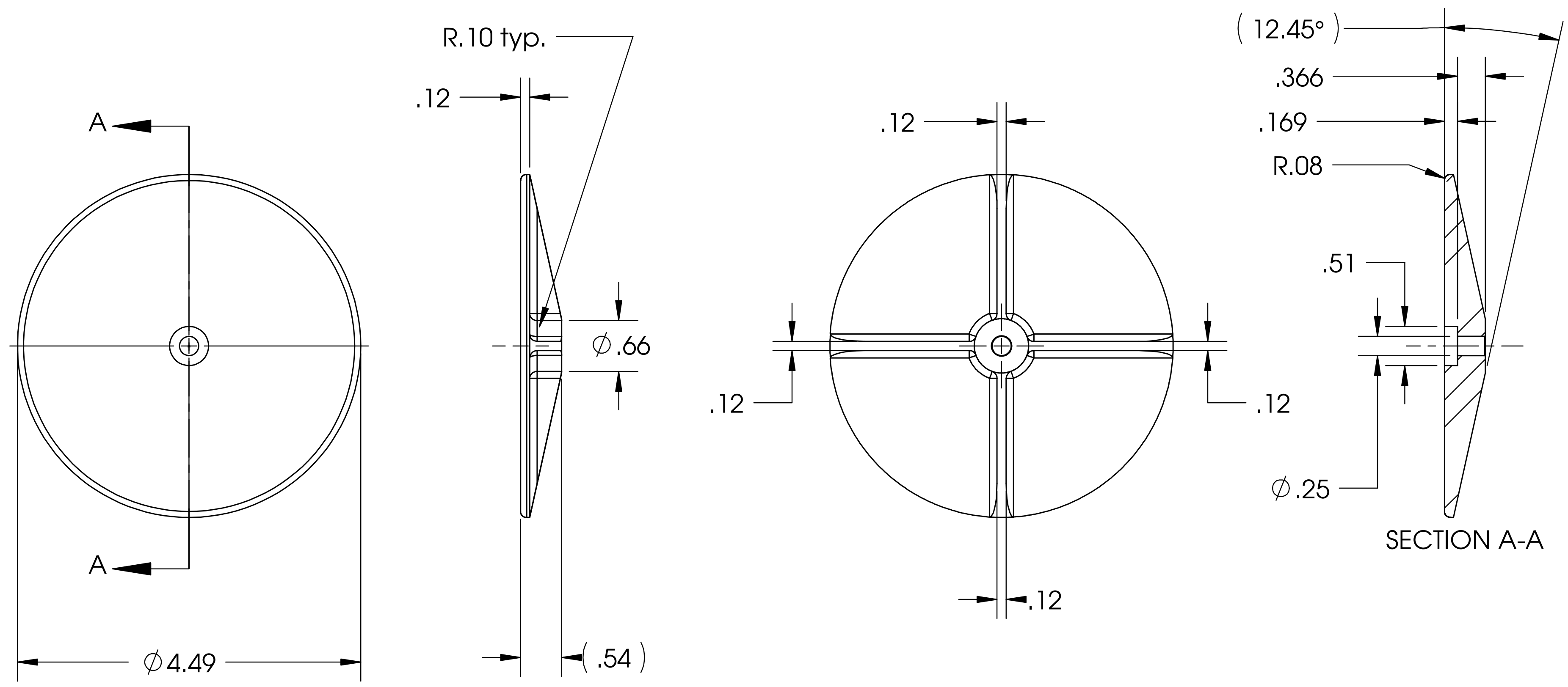
TITLE:

Upper Grip. Shell

SIZE A DWG. NO. Upper REV

SCALE: 1:3 WEIGHT: SHEET 1 OF 1

4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	S.L.	04/06/08
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL Nylon	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

IRG

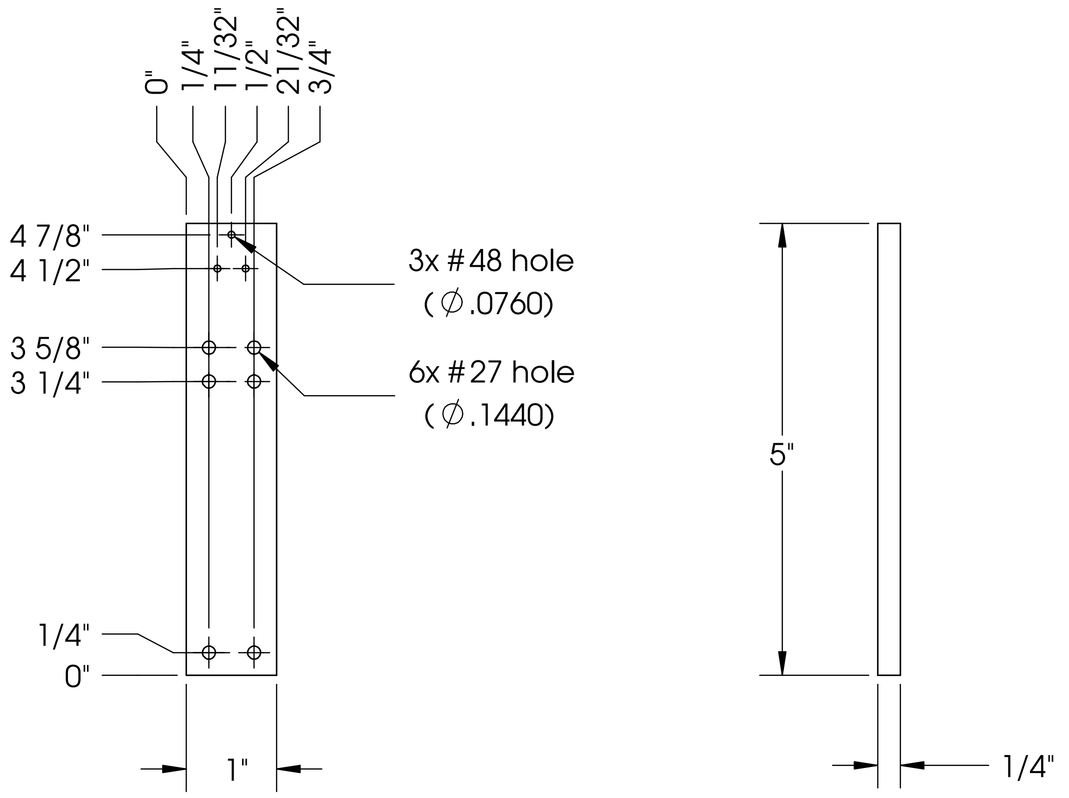
TITLE:

Lower Grip. Shell

SIZE A DWG. NO. LowerShell REV

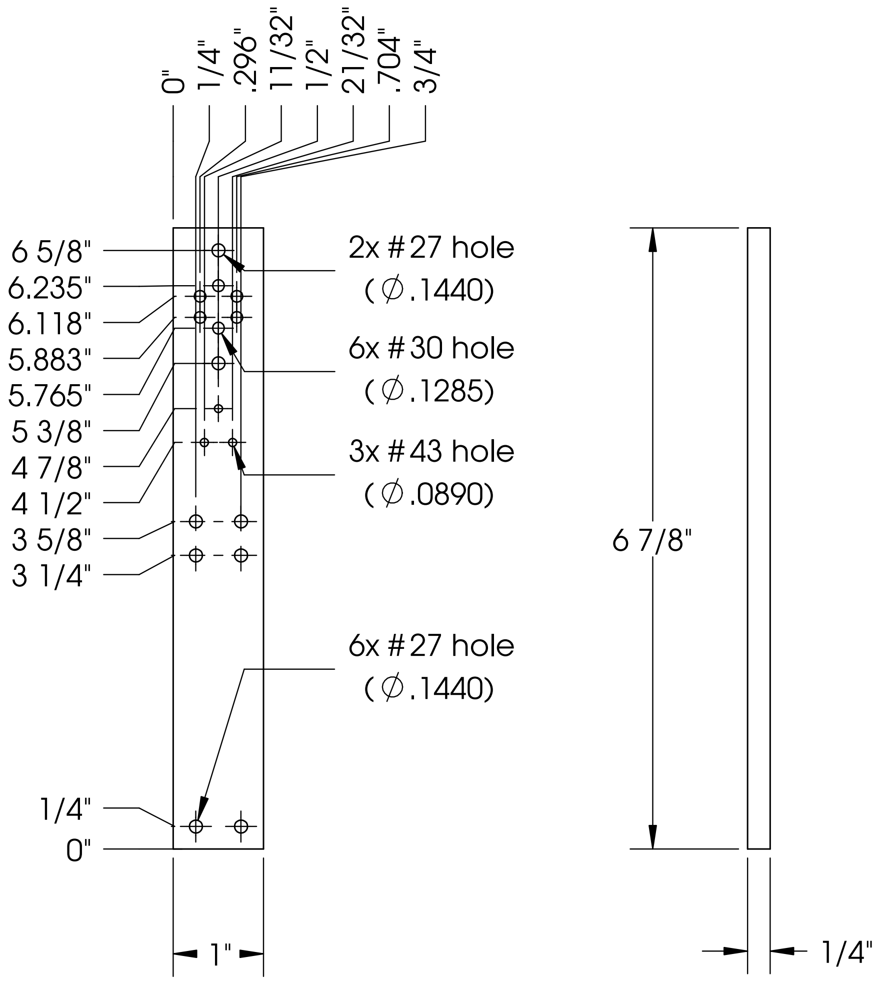
SCALE: 1:2 WEIGHT: SHEET 1 OF 1

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	IRG
		DIMENSIONS ARE IN INCHES TOLERANCES: ± 0.005	DRAWN	S.L.	04/05/08	TITLE:
			CHECKED			Left Mount Bar
			ENG APPR.			
			MFG APPR.			
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.			
		MATERIAL 1/4"x1" Al Flat	COMMENTS:			SIZE A DWG. NO. Mount L REV
NEXT ASSY	USED ON	FINISH (None)				
APPLICATION		DO NOT SCALE DRAWING				SCALE: 1:2 WEIGHT: SHEET 1 OF 1

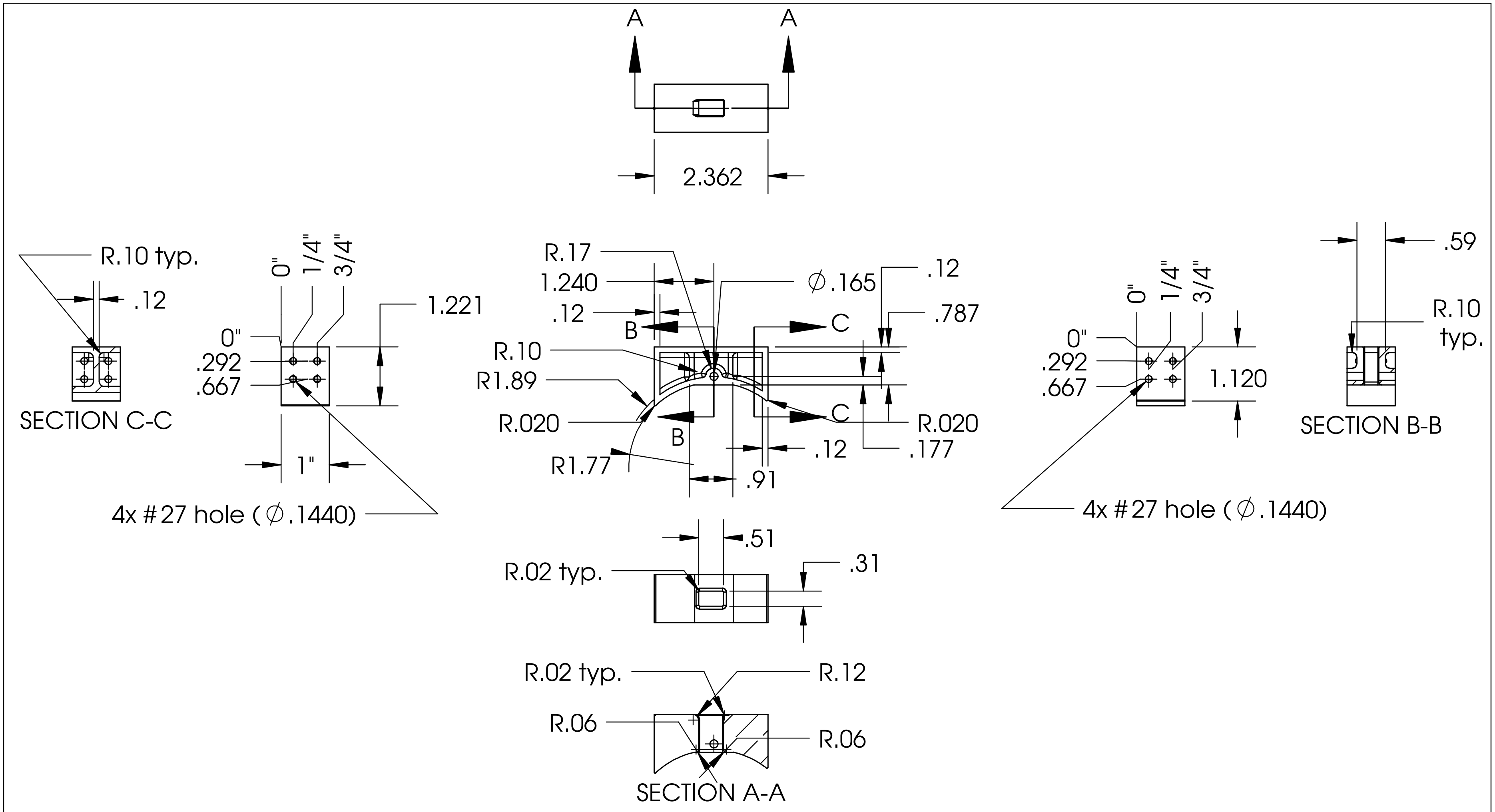
5 4 3 2 1



UNLESS OTHERWISE SPECIFIED:		NAME	DATE	IRG
DIMENSIONS ARE IN INCHES TOLERANCES: ± .002	DRAWN	S.L.	04/05/08	TITLE:
	CHECKED			Right Mount Bar
	ENG APPR.			
	MFG APPR.			
INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.			
MATERIAL 1/4"x1" Al Flat	COMMENTS:			SIZE A DWG. NO. MountR REV
FINISH (none)				
DO NOT SCALE DRAWING				SCALE: 1:2 WEIGHT: SHEET 1 OF 1

NEXT ASSY	USED ON
APPLICATION	

5 4 3 2 1



		UNLESS OTHERWISE SPECIFIED:		NAME	DATE	IRG
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	S.L.	04/05/08	TITLE:
			CHECKED			Motor Fixation
			ENG APPR.			
			MFG APPR.			
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.			
		MATERIAL	COMMENTS:			SIZE A DWG. NO. Fixation REV
NEXT ASSY	USED ON	FINISH				
APPLICATION		DO NOT SCALE DRAWING				SCALE: 1:3 WEIGHT: SHEET 1 OF 1

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, 2005.

[2] T. Fong and I. R. Nourbakhsh. Interaction challenges in human-robot space exploration. *Interactions*, 12(2):42–45, 2005.

[3] H. Geering. *Optimal Control with Engineering Applications*. Springer-Verlag Berlin Heidelberg, 2007.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.

[5] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 3rd edition edition, 2006.

[6] S. C. Jacobsen, E. K. Iversen, D. F. Knutti, R. T. Johnson, and K. B. Biggers. Design of the utah/MIT dextrous hand. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1520–1532, 1986.

[7] C. Ljungström. Manipulator, mounting and integration, and communication brick collection. Master's thesis, Royal Institute of Technology (KTH), 2008.

[8] P. J. McKerrow. *Introduction to Robotics*. Addison-Wesley Publishing Company, 1991.

[9] G. Sánchez-Ante and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In R. Jarvis and A. Zelinsky, editors, *ISRR*, volume 6 of *Springer Tracts in Advanced Robotics*, pages 403–417. Springer, 2001.

[10] V. SunSpiral, D. Chavez, M. Broxton, P. Mihelich, L. Keely, and D. Mittman. Footfall: A ground based operations toolset enabling walking for the athlete rover. In *AIAA SPACE 2008*. American Institute of Aeronautics and Astronautics, AIAA-2008-7889, 2008.

[11] R. Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, FL, 1986.

[12] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.