Modular Robots for Making and Climbing 3-D Trusses

by

Yeoreum Yoon

B.S. Mechanical and Aerospace Engineering
Seoul National University, February 2004

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Department of Mechanical Engineering
May 12, 2006

Certified by.....
Daniela Rus
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by.....
Kamal Youcef-Toumi
Professor of Mechanical Engineering
Thesis Reader

Accepted by .....
Lallit Anand
Chairman, Department Committee on Graduate Students

Modular Robots for Making and Climbing 3-D Trusses

by

Yeoreum Yoon

Submitted to the Department of Mechanical Engineering
on May 12, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

A truss climbing robot has been extensively investigated because of its wide range of promising applications such as construction and inspection of truss structures. It is designed to have degrees of freedom to move in three-dimensional truss structures. Although many degrees of freedom allow the robot to reach various position and orientation, it causes complexity of design and control.

In this thesis, the concept of modular robots is suggested as a solution to reconcile a trade-off between the functionality and the simplicity of a truss climbing robot. A single module has fewer degrees of freedom than required to achieve full 3-D motion, but it can move freely in a 2-D plane. For full 3-D motion, multiple modules connect to and cooperate with each other. Thus, modular truss climbing robots can have both properties: functionality and simplicity.

A modular truss climbing robot, called Shady3D, is presented as the hardware implementation of this concept. This robot has three motive degrees of freedom, and can form a six-degree-of-freedom structure by connecting to another identical module using a passive bar as a medium. Algorithms to move the robot in a 3-D truss structure have been developed and tested in hardware experiments. The cooperation capability of two modules is also demonstrated.

As a next step beyond truss climbing robots, the concept of a self-assembling truss robot with active and passive modules is presented. In this system, multiple Shady3D robots are employed as active modules and they become an active truss structure using passive bars. The procedure of self-assembling such a truss is demonstrated in computer simulations, which show a potential application in robotic truss assembly.

Thesis Supervisor: Daniela Rus
Title: Associate Professor of Electrical Engineering and Computer Science

Thesis Reader: Kamal Youcef-Toumi
Title: Professor of Mechanical Engineering

Acknowledgments

First of all, I would like to thank my advisor—Daniela Rus—for all her guidance and encouragement throughout this research. I would also like to thank my thesis reader—Kamal Youcef-Toumi—for his time and advice. I am deeply grateful to Keith Kotay, Carrick Detweiler, Marsette (Marty) Vona, and Iuliu Vasilescu for their invaluable help in all aspects of the research, from hardware design to software development. Without them, I could never have done this project. Additionally, I thank other members of the Distributed Robotics Laboratory—Peter Osagie, Paulina Varshavskaya, Mac Schwager, Kyle Gilpin, and Johnny Russ—for their assistance. Most of all, I would like to thank my parents—Semoon Yoon and Chuntaek Chung—and my sister—Yeokyung Yoon—for all their loving care and support.

The research described in this thesis was supported by Boeing, NSF, and Intel. We are grateful for their assistance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A truss is a structure of many straight, rigid bars and beams connected together at structural nodes. This type of structure is a common form in many large civil and industrial structures such as bridges, towers, communication antennas, and construction scaffolds. It is also widely used for in-space structures such as space station components and solar panel supports. Tasks related to these trusses are often dangerous or difficult for human workers. Because bars in truss structures are narrow, workers on high towers or construction scaffolds have more risk of falling than those on wide and flat floors. In space, the construction or maintenance tasks of structures outside a spacecraft require dangerous extravehicular activity (EVA) missions by astronauts. Therefore, there have been attempts to develop autonomous mobile robots that move on trusses and perform various tasks instead of human workers.

Mobile robots that climb and traverse trusses have been extensively investigated. For example, Amano et al developed a handrail-gripping robot for fire fighting [2]. Balaguer et al present a climbing autonomous robot that can move in complex 3-D metallic-based structures such as bridges [4]. Nechyba et al's SM2 robot was designed to walk along the I-beam truss structure of a space station [23, 24]. Although these truss climbing robots vary in locomotion and gripping mechanisms according to their specific applications, they have a common characteristic in that all degrees of freedom to perform a full range of required locomotion are implemented in a single robot. For example, Balaguer et al's robot has six motive degrees of freedom to reach an arbitrary

pose in 3-D space [4].

A single robot equipped with enough degrees of freedom required for a complete set of motion is advantageous in terms of the independence of the robot. However, having many degrees of freedom makes the design and control of the robot more complicated. Moreover, in many cases, movement on trusses does not require full degrees of freedom. Many truss structures can be understood as a collection of faces, or planes, composed of bars. If the robot moves only in a certain two-dimensional plane, some degrees of freedom for three-dimensional motion are wasted. Therefore, there is a trade-off between complete locomotion functionality and simplicity.

This thesis introduces the concept of modular robots to reconcile this trade-off. Instead of a single, full-degree-of-freedom robot, multiple simpler modules can be used. A single module has fewer degrees of freedom than required for complete 3-D motion, but it can move in a 2-D plane and reach a goal position in many cases. If complete 3-D motion is necessary, multiple modules can connect to and cooperate with each other to reach a goal position and orientation. In this thesis, a modular robot system that implements this concept is presented.

The robot we present is the extension of a specific truss-climbing application our group has been working on: window shading. We have developed a robot, called Shady, that climbs 2-D window frames and deploys a fan to create shading at an optimal location [34]. Shady is a 2-D truss climbing robot.

This 2-D Shady robot concept has been extended to a 3-D truss climbing modular robot system called Shady3D. The design of the 2-D Shady robot has been modified to be able to escape from a 2-D plane. Based on the modified design, robot hardware including both the mechanical parts and electronics has been developed. We have also developed low-level control algorithms that control joint rotation and gripper operation, and high-level planning algorithms that enable the robot to navigate in a 3-D truss structure. Experiments have been carried out to test the algorithms in hardware. In addition, the cooperation of two modules to achieve full degrees of freedom for 3-D locomotion is demonstrated, using a passive bar as a medium of connection.

As a next step beyond truss climbing modular robots, a self-assembling truss robot with active and passive modules is presented. This robot becomes an active truss by combining many active modules and passive units. Active modules are Shady3D truss climbing robots, and passive units are simple, rigid bars that can be gripped and moved by active modules. These two types of modules self-assemble a truss structure and cooperate to move it. An algorithm for self-assembling and moving a truss tower is described in a computer simulation. This is a special class of heterogeneous self-reconfiguring robots with active and passive units. It has many potential application such as in-space self-assembly of truss structures and dynamic scaffolds for construction [8].

This thesis is organized as follows: Chapter 2 summarizes related research on truss climbing robots and self-reconfiguring modular robots, and introduces the preliminary Shady window shading project that led to this project. Chapter 3 discusses the optimal design model for an individual robot and explains the Shady3D hardware design. Chapter 4 and Chapter 5 discuss low-level control for the hardware and high-level planning algorithms, respectively. In Chapter 6, experiments based on the algorithms are described. Finally, Chapter 7 presents our conclusions.

Chapter 2

Related Work

Our work presented in this thesis is related to the fields of truss climbing robots and self-reconfiguring modular robots. Specifically, it was led by the Shady robotic window shading project carried out in our research group. In this chapter, related research on truss climbing robots and self-reconfiguring modular robots is described, and the Shady robotic window shading project is introduced.

2.1 Truss climbing robots

Mobile robots that climb and traverse truss-like structures have been extensively explored by many researchers. Many different hardware systems have been designed and built for various applications. A variety of locomotion and gripping mechanisms have been employed according to desired motion characteristics and applications.

Amano et al [2] developed a vertically moving robot for fire-fighting purpose. Their robot consists of three links and two grippers connected by four rotational joints. The grippers were designed to grip the balcony handrails of buildings. The robot uses the balcony handrails as steps to climb a building floor by floor. Its motion is restricted in a two-dimensional vertical plane by its kinematic topology.

Kotay et al's Inchworm robot [18] was designed to navigate 3-D steel web structures. The body of the Inchworm robot consists of four links and the first and last link have attaching mechanisms utilizing electromagnets. The robot has four degrees

of freedom. Three degrees of freedom allows the robot to move linearly by extending and contracting like an inchworm. The fourth degree of freedom was employed to enable the robot to turn. The robot can climb steel structures with flat surfaces using electromagnets as an attaching mechanism. Its kinematic topology of linear linkage allows the robot to climb a relatively narrow truss with flat faces.

Aracil et al proposed the climbing parallel robot (CPR), which climbs along tubular and metallic structures [3]. An outstanding feature of their work is that they applied a parallel robot concept to truss climbing robots. They designed the robot based on Stewart-Gough's platform (S-G), which is widely used for flight simulators. The robot consists of two circular rings and six linear actuators that are connected by spherical and universal joints as a standard S-G platform. The ability to climb a tubular structure like a palm tree using grippers installed in the two rings was demonstrated in experiments [1]. In this case, the rings of the robot surrounded the tubular structure that it climbs, and grips on the structure were made inside the rings. Because the robot cannot move in this configuration if there is an obstacle such as a structural node on a tubular structure, they have also adapted the robot to move across structural nodes [30]. The adapted robot used grippers stretched outside of the rings so that it could grip on a particular side of trusses.

Ripin et al designed a modular pole climbing robot [27]. The robot was designed for climbing a pole through grasp-push-grasp motion. It has two grippers connected by a two-part central arm. Pneumatic cylinders were employed for actuation. This robot was developed for agricultural applications, such as climbing palm trees for harvest instead of human workers.

Nechyba et al present a truss climbing robot for in-space inspection tasks [24, 23]. They designed a robot named the Self-Mobile Space Manipulator (SM2) that is capable of walking along the pre-integrated I-beam truss structure of a space station. It has seven degrees of freedom, which allows arbitrary movement in three-dimensional space. Its gripper was designed to hold I-beams with various thickness and width. Because it was developed for the non-gravity environment in the space, a gravity compensation system was employed for testing of hardware. This non-gravity condition

alleviated the restriction of the size and weight of the robot.

Balaguer et al developed a climbing autonomous robot that can move in a complex three-dimensional metallic-based truss structures such as bridges [4]. Their robot, named ROMA, has two grippers that can grip on a truss in a structure. It has six motive degrees of freedom, which are implemented by five rotational joint and one prismatic joint, so that it can move freely in 3-D space. They also developed a path planning algorithm for navigation in a three-dimensional truss structure.

Greenfield et al [11] suggested a model of a hyper-redundant robot climbing by bracing. In this model, the robot composed of many links climbs inside a pipe-like structure by bracing, or by pressing outward to induce friction. This research explored algorithms for climbing, but hardware implementation was not done.

Truss climbing is a special type of structure climbing. Many researchers have studied other types of climbing such as wall climbing. Pack et al [25] present an inspection robot that can climb a 2-D planar surface. Two vacuum fixtures were employed as an attaching mechanism. Bretl et al [5] developed a free-climbing four-limbed robot, named Legged Excursion Mechanical Utility Rover (LEMUR). This robot was designed to climb a vertical wall that has pegs. It climbs the wall by putting its feet on the pegs. It uses friction between its feet and the pegs to climb the wall instead of using a gripping mechanism. The concept of the LEMUR has been also considered for a robot for in-space maintenance and inspection [13].

As shown above, there have been various attempts in terms of the locomotion and gripping (or attaching) mechanisms of truss climbing robots. Our truss climbing robot described in this thesis is close to robots using grippers and rotational degrees of freedom. A major difference in our robot from these other robots is that the concept of modular robots has been applied to the design. While the full degrees of freedom for a complete set of motion are implemented in a single robot in other systems, we have designed a system in which a single robot module has fewer degrees of freedom than required for a complete set of motion and many such modules are incorporated for more complicated motion. This system can achieve both simplicity of the robot design and locomotion functionality. In addition, it has advantages of versatility and

extensibility.

2.2 Self-reconfiguring modular robots

A self-reconfiguring modular robot is a robotic system composed of many physically connected modules that can change their structural configuration for various purposes [15]. A huge amount of research has been done in this field both on hardware design and on algorithm development.

Self-reconfiguring modular robots can be classified by some criteria. One criterion for classification is the model of a system: lattice or chain [15, 6]. In the lattice model, modules form a lattice structure composed of identical cells. Each module occupies a cell and is connected to other modules occupying neighboring cells. Examples of the lattice-based systems are the Fracta robots [21, 20], the Metamorphic Hexagons and Squares [26], the Molecule robot [17, 16, 19], the Crystal robot [28, 29], the I-Cube [32, 33], and the Telecube [31]. In the chain model, modules do not fill a fixed cells in the space. Instead, they can have arbitrary position and orientation. Connected modules form a structure like a serial chain, tree, or ring. The Polybot [35] and the CONRO robot [7] are some examples of chain model systems. Some systems have some of the characteristics of both lattice and chain models. Murata et al's M-TRAN robot [22] and Hamlin et al's TETROBOT [12] are examples of such systems.

Another criterion for classifying self-reconfiguring modular robots is the type of modules: homogeneous or heterogeneous [15]. In the homogeneous system, all modules are identical. For example, the M-TRAN robot [22] and the Crystal robot [29] are homogeneous systems. Each module has the same structure and actuation mechanism. Because only one type of module exist in the homogeneous system, all modules are active, that is, actuated. The heterogeneous system, however, consists of different types of modules. Usually, the number of types is two. For example, Kotay and Rus' Molecule robot (Version 3) has the male modules and female modules [15, 16]. Ünsal et al's I-Cube is composed of the cube modules and link modules [32, 33].

The heterogeneous self-reconfiguring robot system can be classified more: (1)

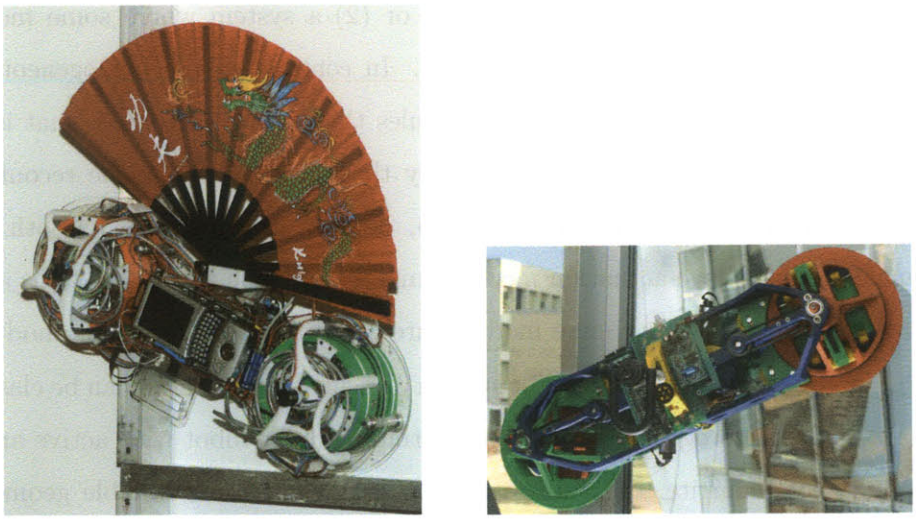
a system where all modules are active, or (2) a system where some modules are active and others are passive (*bipartite*). In contrast to the homogeneous system, the heterogeneous system can have modules that are not actuated, that is, passive. Though passive modules cannot move by themselves, they can be reconfigured by active modules. The Molecule robot [15, 16] falls in category (1). Both male and female modules have actuation mechanisms. The I-Cube robot [32, 33] is a bipartite system where the passive cube modules are carried by the active link modules.

The self-assembling truss robot system presented in this thesis can be classified as a chain-based, heterogeneous self-reconfiguring modular robot with active and passive modules, or bipartite. It is also parallel with research on variable geometry truss systems such as Hamlin et al's TETROBOT [12] and Williams et al's NASA/DOE "SERS DM" [14]. Such self-assembling and self-reconfiguring modular truss robot systems have many potential applications such as in-space construction. Everist et al's research on the SOLAR robotic self-assembly system [10] and Doggett's overview of automatic structural assembly for NASA [9] suggest such applications.

2.3 Shady robotic window shading project

The truss climbing robot presented in this thesis is the extension of a specific truss climbing application our group has been working on: window shading. Our group works in a lab with a large wall-window, about 4m tall and 8m wide. The window has a grid of aluminum frames. Because there is no shade to block the sun, the bright sunlight coming through the window often prevents lab members from looking at computer screens. Instead of traditional shades that block the whole window, we have developed a robot, called *Shady*, that climbs the window frames and deploys a fan to create shading at an optimal location [34]. Therefore, the Shady is a truss climbing robot that can move in a two-dimensional plane.

The Shady robot hardware has been developed. It has two rotating "barrels" and two grippers that are installed in the barrels. The grippers were designed to grip on a window frame. With one of the grippers closed on the frame, the barrels are



(a) (b)

Figure 2-1: Two versions of the Shady window shading robot. (a) Version 1. The robot holds a truss that simulates the modified frame. A hand fan is deployed as a shade. The image is adopted from [8]. (b) Version 2. The robot grips on the actual window frame. The fan is removed in this image. The image is adopted from [34].

rotated to move the robot body over the window frame. By alternatively gripping and rotating, the robot can travel in the 2-D plane of the window.

Figure 2-1 shows two versions of Shady hardware. The grippers of the first version were not able to hold the window frame firmly and required a slight modification to the frame for a reliable grip [8]. The second version features a new gripper design utilizing kinematic singularity, which solves the problem of the first version grippers. The grippers of the second version are able to hold the unmodified window frame positively. Another feature of the second version is mechanical compliance implemented by springs, which enables the grippers to grip on the frame firmly even when they are misaligned with the frame [34].

Control software for the Shady robot has also been developed. These control software is used not only for the hardware but also computer simulations. Specifically, a path-planning algorithm has been developed to determine the shortest path to the goal location in the window [8].

The concept of the Shady model has been extended to the self-assembling and self-

reconfiguring active 2-D truss structure composed of many Shady (active) modules and passive bars. The algorithms for this active truss has been demonstrated in computer simulations [8].

The Shady robotic window shading project led to the project described in this thesis. We have extended the concept of the 2-D Shady to a truss climbing robot that can move in a 3-D truss environment.

2.4 Summary

Many truss climbing robots with various locomotion and gripping (or attaching) mechanisms have been studied and developed. Though our robot is closely related to some of these robots using grippers and rotational degrees of freedom, it is different in that the concept of modular robots has been applied. Among various types of self-reconfiguring modular robots, the self-assembling truss robot system presented in this thesis can be classified into a chain-based, heterogeneous system with active and passive modules (bipartite). Our work has been extended from the Shady robotic window shading project, which our group has been working on for the practical purpose.

Chapter 3

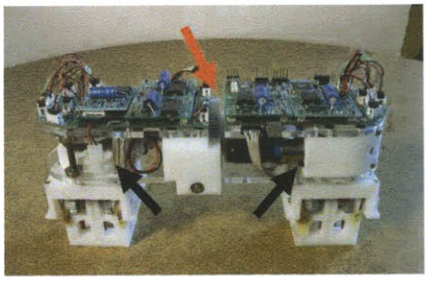
Shady3D Truss Climbing Robot

This chapter describes the concept and the hardware design of our Shady3D truss climbing robot. The design concept of the Shady3D robot is described, followed by discussion on several design alternatives. Next, the detailed description of the Shady3D hardware is presented.

3.1 Concept overview

The Shady3D is a robot capable of gripping and moving on trusses. It can climb trusses by gripping on a truss element and using its rotating degrees of freedom. It is the extended version of the two-dimensional Shady window shading robot [34] to a three-dimensional one.

The overall shape of the Shady3D robot resembles a stick (See Figure 3-1). It is composed of two rotating *grippers* linked by a two-part *arm*. The grippers can grip and release a truss bar by closing and opening its gripper paddles. Each gripper is connected to the arm by a rotating joint (the gripper joint), which enables the gripper to align with a truss in various orientation. Two parts of the arm are connected by another rotating joint (the middle joint), which creates a relative angle between the directions of two grippers. The Shady3D robot has total five degrees of freedom: three rotational degrees of freedom for locomotion and two degrees of freedom for gripper opening and closing.



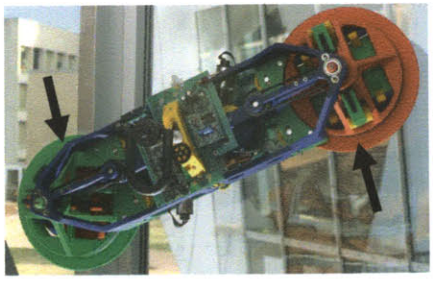


Figure 3-1: This figure shows the degrees of freedom of the Shady3D robot. The left image shows the Shady3D robot. The degrees of freedom of the gripper joint are indicated by the black arrows. The degree of freedom of the middle joint is indicated by the red arrow. Compare the Shady3D robot with the 2-D Shady window shading robot shown in the right image. The 2-D Shady robot does not have the degree of freedom in the middle of the body. The right image is adopted from [34].

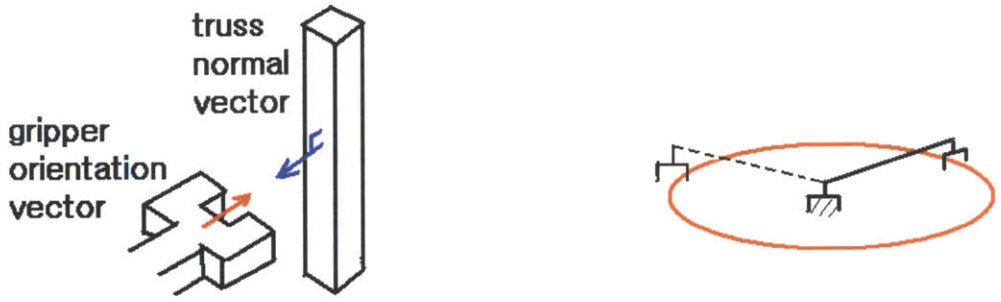
The robot can move within a 2-D plane by using two gripper joints. The middle joint is not used in this case. When the robot needs to move from one plane to another in different orientation, the middle joint is used in order to direct grippers toward different planes. If more degrees of freedom are required than a single robot has, more than one robot can be connected using a rigid bar as a medium (See Section 3.2.5 for details).

3.2 Design alternatives

We designed the Shady3D robot as the extension of the Shady window shading robot which climbs a two-dimensional window frame.[1] However, the motion of the 2-D Shady robot is confined in a 2-D plane because of its kinematic configuration. Additional degrees of freedom are needed to enable the robot to move freely in 3-D space.

Although many degrees of freedom give a robot capability to reach a wide range of poses in various position and orientation, they make hardware design and control complicated. Because many actuators need to be installed, packaging them efficiently in a restricted space becomes hard, particularly in the case where the size of the robot

[1]For more detailed description of the Shady window shading robot, see Section 2.3.



(a) (b)

Figure 3-2: Terms used to analyze design models. In (a), the truss normal vector and the gripper orientation vector are depicted. (b) shows the workspace of the robot (represented as a red circle). One gripper is fixed.

is small. In order to make hardware design and control simple, it is desirable to keep the degrees of freedom of the robot as low as possible.

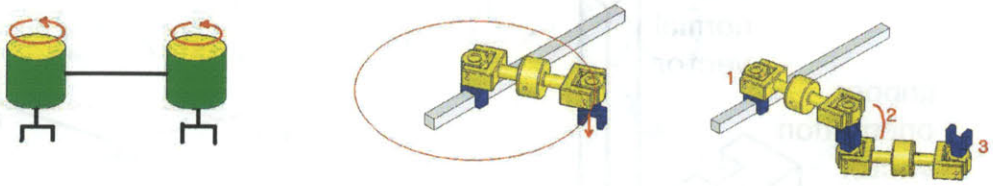
We considered and analyzed several design models with different numbers of degrees of freedom to find the optimal number. Some geometric terms used for this analysis are defined below:

- A **truss normal vector** is defined as a vector perpendicular to a truss surface on which a gripper grips. This vector represents the orientation of a truss. In the 2-D case, truss normal vectors for all trusses direct the same direction. In the 3-D case, however, the direction of truss normal vectors vary because trusses can have arbitrary orientation.

- A **gripper orientation vector** is defined as a vector pointing toward a surface on which the gripper grips on. In order for a gripper to hold a truss, its gripper orientation vector must be in the opposite direction to the truss normal vector of the truss.

- The **workspace** of the robot is defined as a set of the reachable points of one gripper when the other gripper is fixed.

Figure 3-2 illustrates these terms.



(a) (b) (c)

Figure 3-3: This figure shows the 2-DOF design model. (a) Each gripper joint has one rotational DOF. (b) The workspace of the 2-DOF model is a planar circle, which is shown as the red circle. The gripper orientation vector is illustrated as the red arrow. (c) Two modules are connected to make a 3-DOF linkage. One DOF is reduced in gripper-to-gripper connection. The motion of the whole linkage is still confined to a 2-D plane.

For each design model, the workspace and possible gripper orientation were examined. In addition, we investigated the number of modules needed to be connected in order to obtain six degrees of freedom. Six degrees of freedom are minimum required for the arbitrary position and orientation of the free gripper in 3-D space.

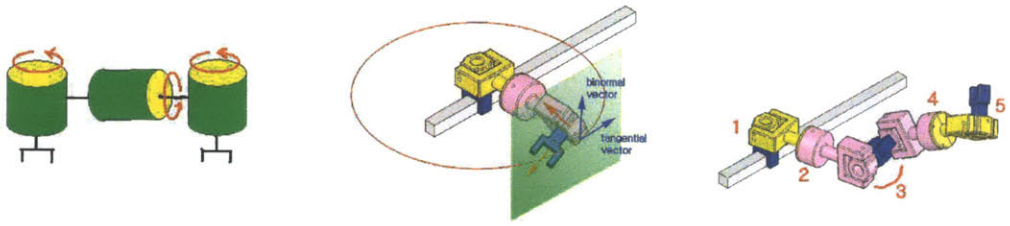
When we discuss the degrees of freedom of design models, degrees of freedom for gripper opening/closing are not included because they do not affect the kinematic configuration of the robot. Only motive degrees of freedom are counted.

3.2.1 2-DOF model (2-D Shady for window shading)

The 2-DOF design model is used for the 2-D Shady robot for the window shading application. It has two rotational degrees of freedom, one for each gripper rotation. The axes of rotation for both gripper joints are parallel (See Figure 3-3(a)).

The workspace of the 2-DOF robot is a planar circle, as shown in Figure 3-3(b). The direction of the gripper orientation vector is fixed to one direction, which means that the robot can grab a truss in one orientation. The motion is confined to the 2-D plane.

Figure 3-3(c) shows two modules connected by gripping each other's gripper. Notice that loss of DOF occurs when two modules are connected. When two grippers



(a) (b) (c)

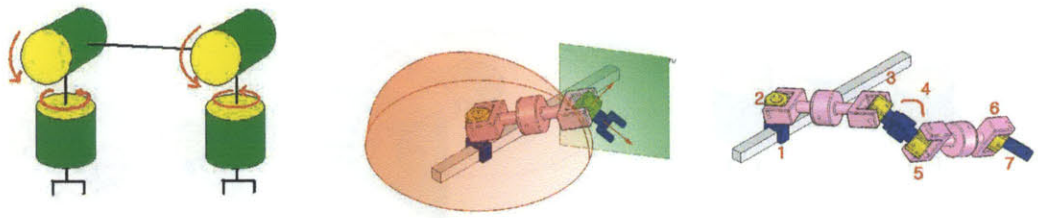
Figure 3-4: This figure shows the 3-DOF design model. (a) The rolling joint is added in the middle axis of the 2-DOF model. (b) The workspace of the 3-DOF model is a planar circle (the red circle), but the gripper can have any orientation on the gripper orientation plane (depicted as a greed plane). (c) Two modules are connected to make a 5-DOF linkage.

grip on each other, the axes of two gripper joints lie along the same line and two degrees of freedom become redundant. Therefore, one degree of freedom is wasted. This reduction of DOF always happens when two grippers are directly connected.

When one module is added, two DOFs of the new module are added but one DOF is lost due to the DOF reduction. As a result, the total degrees of freedom of the whole mechanism increase by one. As shown in Figure 3-3(c), the number of DOFs of the two-module mechanism becomes three. In order to obtain six degrees of freedom, we need to connect five modules serially. The serial chain of five modules, however, is not capable of reaching an arbitrary position and orientation in 3-D space. No matter how many modules are connected, all axes of rotational degrees of freedom are in the same direction. Therefore, the mechanism of multiple modules is still confined to a 2-D plane. Three-dimensional motion is impossible with the 2-DOF design model.

3.2.2 3-DOF model

The 3-DOF design model has an additional joint between two gripper joints of the 2-DOF model (See Figure 3-4(a)). This joint, called the rolling joint, enables two gripper joint axes to have different directions. Therefore, the grippers are able to hold trusses with different truss normal vectors, that is, in different planes.

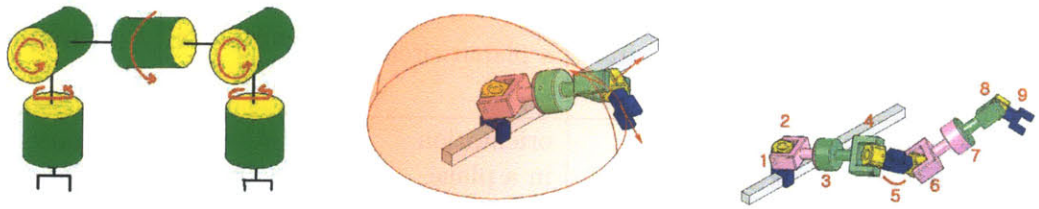


(a) (b) (c)

Figure 3-5: This figure shows the 4-DOF design model. (a) In addition to gripper joints in the 2-DOF model, two pitch joints are added. (b) The workspace of the 4-DOF model is a hemisphere shell (the shaded area in red). The gripper orientation plane is illustrated by the green plane. (c) Two modules are connected to make a 7-DOF linkage.

The workspace of the 3-DOF model is a planar circle as the 2-DOF one. Because of the rolling joint, however, the gripper orientation vector can have more than one orientation (See Figure 3-4(b)). It can be any vector on a plane spanned by the tangential and binormal vectors of the workspace circle. This plane is named the gripper orientation plane. The robot can grip on any truss whose truss normal vector is included in the gripper orientation plane. By gripping a truss in a different plane from the plane the other gripper grips on, it can move from one plane to another plane, which means 3-D motion. However, this 3-D motion is not complete because the robot cannot grip on a truss that does not intersect with the workspace circle or whose truss normal vector does not lie on the gripper orientation plane.

In Figure 3-4(c), the linkage of two connected 3-DOF modules is shown. The total degrees of freedom of the whole linkage are five because one degree of freedom is lost in gripper connection (DOF reduction). This mechanism can implement 3D motion because axes of rotation vary. In order to obtain six degrees of freedom which is necessary for arbitrary position and orientation, we need to add one more module. With three modules connected serially, we have seven degrees of freedom.



(a) (b) (c)

Figure 3-6: This figure shows the 5-DOF design model. (a) This model has two gripper joints, two pitch joints, and a rolling joint. (b) The workspace of the 5-DOF model is a hemisphere shell (the shaded area in red). The gripper can have arbitrary orientation. (c) Two modules are connected to make a 9-DOF linkage.

3.2.3 4-DOF model

For the 4-DOF design model, we added rotational degrees of freedom that lift up grippers (See Figure 3-5(a)). These joints, named pitch joints, enable the gripper joints to have angles other than the right angle with respect to the robot body line.

The workspace of the 4-DOF model is a hemisphere shell as shown in Figure 3-5(b). This model has a 3-D workspace by itself. The gripper orientation plane is a plane spanned by the normal vector to the hemisphere and the tangential vector to the longitude line. Trusses whose truss normal vector lies on this plane can be grabbed by the gripper. However, the robot is not able to hold a truss whose truss normal vector has a component perpendicular to the plane.

Two connected modules form a 7-DOF linkage as shown in Figure 3-5(c). Six degrees of freedom can be obtained with two modules. This linkage can reach arbitrary position and orientation in 3-D space.

3.2.4 5-DOF model

The 5-DOF design model is a combination of the 3-DOF and 4-DOF models. It has two rotational degrees of freedom for gripper joints, two pitch joints and a rolling joint (See Figure 3-6(a)).

Model	2-DOF	3-DOF	4-DOF	5-DOF
Workspace of the single module	a planar circle	a planar circle	a hemisphere shell	a hemisphere shell
Gripper orientation	only one direction	orientation in a plane	orientation in a plane	arbitrary orientation
No. of modules for 6-DOF	5 (2-D)	3	2	2

Table 3.1: Summary of design model analysis

The workspace of the 5-DOF model is a hemisphere shell as the 4-DOF model (See Figure 3-6(b)). In contrast to the 4-DOF model, however, the gripper can have arbitrary orientation. The robot can hold a truss in any direction as long as it intersects with the workspace shell.

In Figure 3-6(c), a 9-DOF manipulator composed of two 5-DOF modules is illustrated. We can obtain six degrees of freedom necessary for arbitrary 3D motion with two modules connected in serial.

3.2.5 Summary and discussion

Table 3.1 summarizes the results of design model analysis. The 2-DOF model is not suitable for a robot climbing 3-D trusses because it cannot make a kinematic structure capable of reaching arbitrary 3-D poses, even with multiple modules. This means that the design of the 2-D Shady robot for the window shading application needs to be modified for 3-D truss climbing.

All the 3-DOF model through 5-DOF model can make a 6-DOF structure that is able to have arbitrary position and orientation in 3-D space. In order to make it, at least two modules are needed in each case. Both 4-DOF and 5-DOF models can form a 6-DOF structure with two modules. Therefore, the 5-DOF model has no significant advantage over the 4-DOF model in terms of the number of modules needed for arbitrary 3-D motion. Because including more degrees of freedom causes

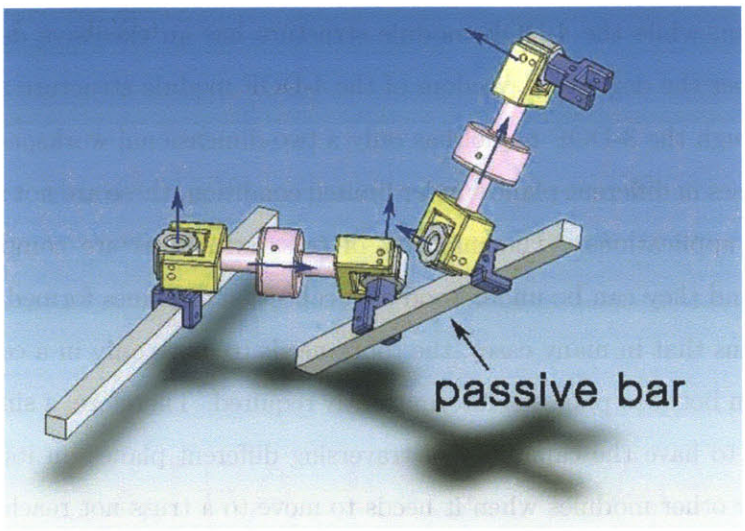


Figure 3-7: A 6-DOF manipulator composed of two 3-DOF modules and a passive truss element. Note that the axes of two grippers holding the passive truss are not in the same line. No DOF reduction occurs in this configuration.

complexity of design and control, the 5-DOF model is ruled out.

For the 3-DOF model, three modules are required to make a 6-DOF structure. Two-module assembly has only five degrees of freedom because of the DOF reduction at the connection of two grippers. If we can avoid this reduction, we can obtain six degrees of freedom with two 3-DOF modules. The reason for the DOF reduction is that the axes of two gripper joints lie on the same line. This problem can be solved by introducing a free truss element as a medium of connection. Figure 3-7 shows such configuration. In this configuration, two modules grip on the different parts of the truss element and the axes of gripper joints are not on the same line. Therefore, we can avoid the DOF reduction and the total number of degrees of freedom becomes six. We can build a 6-DOF manipulator with two 3-DOF modules and a passive truss element, or a passive bar.

By using a passive bar, we can reduce the number of modules for a 6-DOF structure to two for the 3-DOF model. As a result, the advantage of the 4-DOF model over the 3-DOF model disappears in terms of the number of modules to achieve arbitrary 3-D motion. Moreover, the structure of 3-DOF modules have no excessive degrees

of freedom, while the 4-DOF module structure has an excessive degree of freedom (Remember the degrees of freedom of the 4-DOF module structure are seven).

Although the 3-DOF model has only a two-dimensional workspace and is able to grab trusses in different planes under limited condition, these are not serious problems in many applications. The majority of truss structures are composed of *straight* trusses, and they can be understood as a collection of planes formed by such trusses. This means that in many cases, the robot needs to move only in a certain 2-D plane. Transition between planes is intermittently required. Therefore, a single module does not have to have the capability of traversing different planes on its own. It can be helped by other modules when it needs to move to a truss not reachable by itself.

For the reasons above, we concluded that the 3-DOF model is the optimal design option among design models considered. It has only one additional degree of freedom compared with the 2D Shady robot. It can grip a truss in a different plane. In addition, we can obtain a 6-DOF manipulator with two modules and a passive bar, without loss and waste of degrees of freedom.

3.3 Design

In the previous section, we explained the conceptual design of the Shady3D truss climbing robot. This section describes the physical implementation of the Shady3D robot hardware.

The Shady3D hardware consists of two grippers and a two-part arm linking them. Each gripper is installed in the arm using a rotating joint. Two parts of the arm is connected by the middle rotating joint, which corresponds to the rolling joint of the 3-DOF design model (See Section 3.2.2).

3.3.1 Structure

Each gripper structure is composed of four main components–the housing, the housing cover, the housing top cover, and two gripper paddles. The gripper housing forms the cubical gripper shape and supports shafts on which gears for the gripper mechanism

are assembled. It also provides a contact surface for a truss gripped by the gripper. The housing cover and the housing top cover feature mounting places for the gripper motor. The motor is mounted between these two covers. A worm gear used for the gripper joint mechanism is also placed between two covers. Each cover has a groove for gripper joint bearings around its circumference. The gripper paddles are used to make a grip on a truss. For the detailed description of the gripper paddles, see Section 3.3.2.

The arm structure features two gripper joints, a middle joint, and a base structure for these joints. The base structure consists of four acrylic plates, two for each side. In each side of the arm, joint assemblies are assembled between two plates. Joint structures support motors, bearings, and shafts for joints. For the detailed description of the joints, see Section 3.3.3.

All parts for the structure except for the arm plates are fabricated with the FDM2000 Fused Deposition Modeling (FDM) rapid prototyping machine manufactured by Stratasys, Inc. This machine makes high-strength, lightweight ABS plastic parts from 3D CAD models we made [15]. The strength of FDM material has proved to be strong enough to support forces and torques exerted on them during experiments. We have had no failure in structural parts. The arm plates are fabricated from acrylic plate with a laser cutter.

The overall size of the Shady3D robot is 250mm long and 80mm wide. The height from the bottom of gripper paddles to the top of the robot is about 140mm, when the grippers are fully closed. The width and the length of the gripper housing are 60mm and 70mm, respectively. The distance between the axes of two gripper joints (center-to-center distance) is 180mm. This distance determines the length of one step of the robot. See Appendix A for the detailed specifications of the Shady3D robot.

3.3.2 Grippers

Grippers are essential parts for a truss climbing robot. In order to achieve reliable movement on trusses, the gripper of the robot must hold trusses firmly. Incomplete grip can cause the wobble and slippage of the robot. In the worst case, the robot

can fall off trusses. Therefore, designing a gripper that grips a truss reliably is very important.

In addition to designing for a firm grip, there are other issues to consider regarding gripper design for the Shady3D robot. One issue is *compliance*. Because it is possible that a gripper is misaligned on a truss, compliance for such misalignment is necessary. In addition, although the gripper joint axis and the middle joint axis are supposed to orthogonal according to the model, errors are produced in real implementation because of gravity. When one gripper grips on the upper side of a truss and the other side is stretched to empty space, the other side of the robot slightly tilts downward. This tilt can cause collision between the disconnected gripper and the truss when the robot moves over the truss. On the other hand, in the case that the robot hangs under or on the side of a truss with one gripper, the other gripper is slightly off from the truss. To adjust this error, the gripper need to have capability to draw itself to the truss by pulling it during gripping procedure.

Another capability required of the Shady3D gripper is *retractability*. When the gripper is open, the gripper paddles need to be fully retracted behind the contact surface of the gripper.[2] If the gripper paddles are extended beyond the contact surface, they would collide against a truss while the robot moves the opened gripper over the truss.

We designed a gripper that meets these design requirements. First of all, the gripper was designed to encompass a truss in order to ensure a firm grip. It can hold a truss with a 3/4in-by-3/4in square cross section. When it is closed onto a truss, the four faces of the truss are fully encompassed by the gripper paddles and the contact surface of the housing. This encompassing grip prevents the gripper from wobbling on the truss. In addition, rubber pads were attached on the contact surface of the gripper paddles in order to prevent slippage along the truss. These rubber pads provide a large friction on aluminum and acrylic surfaces.

Several features are implemented to accomplish compliance for misalignment and

[2]The contact surface is the surface of the gripper housing which contacts with a truss surface. See Section 3.3.1.

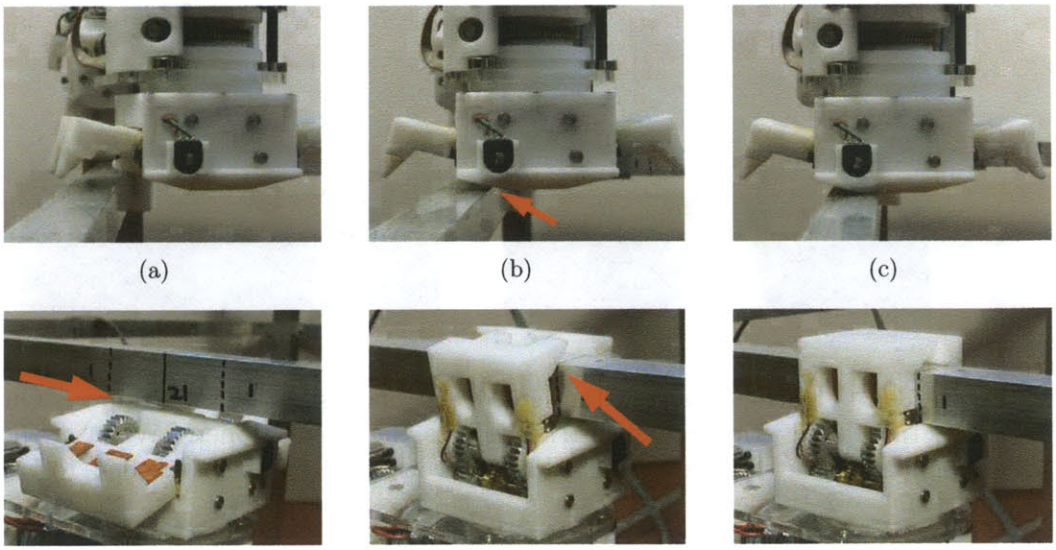


Figure 3-8: The compliance of the gripper for tilt. (a)-(c) The slope in the housing, marked with the red arrow in (b), guides the gripper along the edge of the truss. (d)-(f) The slopes of the gripper paddles (highlighted in (e)) help the gripper pull the truss and compensate the gap (marked in (d)) between the contact surface and the truss.

tilt. To correct misalignment between the gripper and a truss, four detector switches were installed on gripper paddles. The states of these switches are checked during gripping procedure. If the gripper is not aligned with the truss, one switch or two switches in diagonal position are pushed before others. Based on this information, the corresponding gripper joint is rotated in direction to resolve the misalignment. For the detailed description of misalignment correction procedure, see Section 4.3.2.

Compliance for tilt caused by gravity is accomplished in two ways. First, in order to prevent collision caused by tilt toward a truss (in the case where the robot is on the upper side of the truss), a slope around the edges of the housing was made. This slope guides the housing to the right position and compensates the error caused by the tilt. Next, we also made slopes in the gripper paddles. These slopes enable the gripper paddles to reach the far edges of a truss that does not contact with the contact surface of the housing. As gripper paddles are closed, the truss is guided and pulled

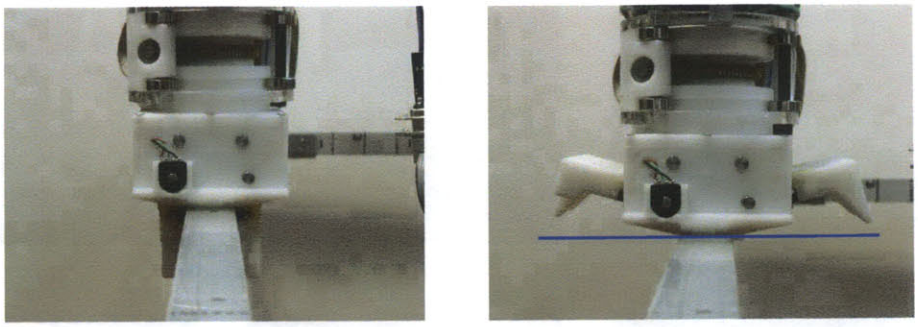
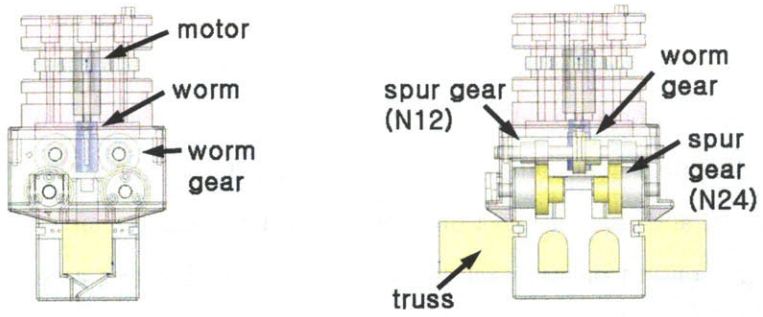


Figure 3-9: This figure shows the closed and open state of the gripper. On the left is the closed gripper and on the right is the open gripper. When open, the gripper paddles are fully retracted behind the contact surface (the blue line) to prevent collision.

to the contact surface by the slopes. Equipped with this compliance, the gripper can hold a truss 10mm away, which is much greater than observed during experiments. Figure 3-8 shows these two types of compliance for tilt.

In addition, the gripper paddles are fully retracted when open. They are rotated around the shafts mounted on the housing, and the range of rotation is about 100 degrees. Because of this wide range of rotation, the gripper paddles can be retracted behind the contact surface of the housing, allowing an opened gripper to move over a truss without collision. The retracted position of the gripper paddles is shown in Figure 3-9.

The actuation of the gripper was implemented with gear power transmission. The actuation mechanism is shown in Figure 3-10. The motor is installed vertically between two housing covers. A worm connected to the motor shaft drives two worm gears for both grippers. Each worm gear transmits power from the worm to small spur gears assembled on the same shaft. The small gears, in turn, drive engaged large spur gears, which are fixed to the gripper paddles. Since the gripper paddles are fixed to the large spur gears, they rotate at the same rate as the large spur gears. Because power transmission routes for two paddles are symmetric, both grippers moves symmetrically, like a mirror image. In this manner, the gripper can be opened and closed.



(a) (b)

Figure 3-10: CAD drawings of the gripper assembly. The housing and the covers has been rendered transparent in order to show the internal configuration. The left image is the front view and the right one is the right view.

The torque of the motor is amplified by gear reduction. The gear reduction occurs in two steps: in the worm-worm gear part and in the spur gear part. The worm is single-threaded and the number of teeth of the worm gear is 30. Therefore, the gear reduction of the worm-worm gear part is 1:30. As for the spur gear part, the small one has 12 teeth and the large one has 24 teeth, resulting in 1:2 reduction. Consequently, overall reduction becomes 1:60. The torque of the gripper motor, which is 20mN-m, is amplified to 1.2N-m. Through experiments, this torque proved to be strong enough to close the gripper completely even with errors of truss position. For example, as shown in Figure 3-8, the torque of the gripper is enough to lift the weight of robot while the gripper pulls a truss on the lower side.

An advantage of using a worm and a worm gear for actuation is non-back-drivability: a worm can drive a worm gear but the reverse direction is impossible. This means that once the gripper is closed, the motor can be turned off because the force to open the gripper paddles cannot drive the motor backward.[3] Moreover, since the motor do not need to provide large torque to keep the gripper paddles closed, its size can be reduced. For this reason, we could use a mini geared motor whose rated torque is

[3]The force trying to open the closed gripper is produced by the weight of the robot itself. If another module is connected to one robot through a passive bar, the weight of the module also tries to open grippers gripped on the passive bar or on the environment truss.

only 20mN-m.

Since the worm is non-back-drivable, however, the force trying to open the gripper pushes the worm upward and generates considerable stress on the gripper structure. Though the FDM material is quite strong, it can fail if the thickness of a part is thin. In addition, connection between structural parts are especially weak. In our design, connection between the housing and the housing cover turned out to be the weakest part. We originally used 3-mm thickness for the gripper housing cover and 4 screws to fix it to the housing. This connection was broken during an experiment under heavy load. The 3-mm plate of the housing cover was broken and the threads made on the housing to fix screws were torn off. After experiencing this failure of the structure, we strengthened the connection between the two parts. The thickness of the housing cover plate was increased to 4mm, and the number of screws was doubled to eight. This modification resulted in stronger connection. The structure has not failed after the modification. The gripper is able to deal with the gripper-opening force successfully.

3.3.3 Joints

The Shady3D robot has three rotational joints for three motive degrees of freedom. Two gripper joints connect the grippers and the arm. The middle joint allows two parts of the arm rotate relatively.

Gripper joint

The gripper joint is a gear drive system that rotates the gripper with respect to the arm. Figure 3-11 shows the gripper joint assembly. It consists of a motor and a worm mounted on the arm and a worm gear mounted on the gripper. The worm is connected to the motor shaft. The motor is installed in the motor mounting block that is placed between two arm plates. The worm gear is fixed between the gripper housing cover and the gripper housing top cover. The cylindrical part formed by the two covers of the gripper is placed inside the holes of the arm plates. To keep the

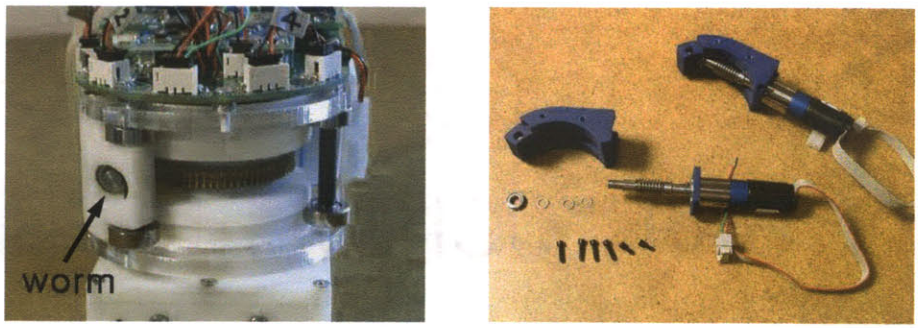
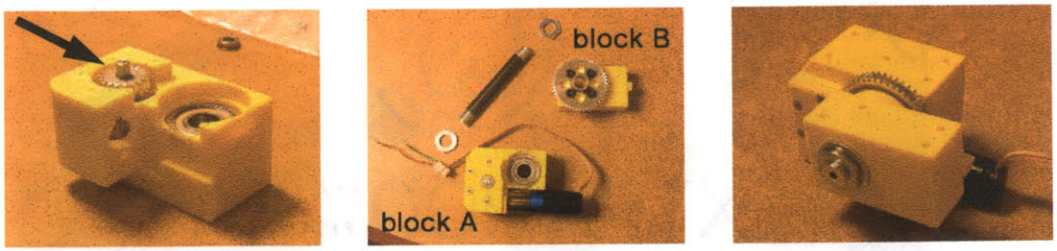


Figure 3-11: This figure shows the gripper joint assembly of the Shady3D robot. The left image is a close-up view of the gripper joint. The worm is placed in the motor mounting block. Two arm posts with four bearings are shown. One post is behind the gripper covers. The bearings roll along the grooves around the circumference of the cylindrical gripper covers. The right image shows motor mounting blocks for gripper joints.

axis of the gripper in the center of the arm plate holes, we use six bearings installed on three shafts. These shafts play a role not only as axes of rotation for bearings but also as a supporting structure between the upper and lower arm plates. On the gripper side, the cylindrical covers have grooves in which the bearings can roll.

The worm used in the gripper joint is the same as one used for the gripper—a single-threaded, 48-pitch worm. A worm gear with 75 teeth was chosen for the worm gear of the gripper. Because the worm drives the worm gear directly, we obtain 1:75 gear reduction. The maximum continuous output torque after the gearhead of the motor is 277.2mN-m (according to the catalogue). This torque is amplified to 20.79N-m by the gear reduction. This is enough to lift the weight of one robot, approximately 1.34kg.

The use of a worm drive mechanism provides non-back-drivability. Since the worm is not back-driven, the motor does not have to provide torque to resist the torque exerted on the gripper joint by the weight of the robot or other factors. It can be turned off after rotating the joint to a desired position. Non-back-drivability, however, has a disadvantage in terms of gripper misalignment. If the motor is back-drivable, a misaligned gripper can be passively aligned as the gripper paddles are closed. The non-back-drivable joint cannot be aligned in this manner. To compensate



(a) (b) (c)

Figure 3-12: This figure shows the middle joint assembly. (a) The partial assembly of the block A. The intermediate gearing is indicated. (b) The assemblies of the block A and B are shown with the middle shaft. The motor is installed in the block A. (c) The completed middle joint assembly.

this disadvantage, we have implemented active correction procedure for misalignment. See Section 3.3.2 and 4.3.2 for detailed information on misalignment correction.

The design of the gripper joint allows limitless rotation. However, too much rotation makes wires connecting the gripper and the arm be twisted excessively. Excessively twisted wires can cause physical damage on the circuitry by pulling connectors and hinder the rotation of the joint. Therefore, we limited the range of rotation by software.

Middle joint

The middle joint realizes the rolling degree of freedom[4] that enables the robot to perform 3-D motion. It is motorized by a gear drive system, like the gripper joint. While a worm drives a worm gear directly in case of the gripper joint, however, an intermediate gearing has been inserted between a worm and a worm gear. The reason for this insertion of additional gearing is packaging. The issue of packaging is explained in Section 3.3.6.

Figure 3-12 shows the middle joint assembly. The joint consists of two main blocks and other auxiliary parts. One block (block A) contains a motor, a worm, a worm gear, and a small spur gear along with five bearings for shafts. The other block (block B) has a large spur gear. These two blocks are connected by a 3/8-in aluminum shaft.

[4]See Section 3.2.2 for the definition of the rolling joint.

Nuts at the ends of the shaft press two blocks against each other so that two parts do not move in the axial direction. By this, we can ensure that the distance between two grippers is constant.

The motor of the middle joint drives the worm connected to its shaft. Next, power is transmitted to the worm gear that is assembled with the small spur gear on one shaft. Finally, the small spur gear drives the large spur mounted on the block B. Since the large spur gear is fixed on the block B and the middle aluminum shaft, the block A rotates around the shaft. Each block is fixed to each side of the arm. Consequently, relative rotation between two sides of the arm is obtained.[5]

We use the same worm used for the gripper joints. The worm gear has 30 teeth. Therefore, the gear reduction in the worm-worm gear pair is 1:30. The numbers of teeth of the small and large spur gears are 18 and 48, respectively. This generates gear reduction of 1:2.67, and overall gear reduction becomes 1:80. Since we use the same motor with the maximum continuous torque of 277.2mN-m as the gripper joint, the resulting torque of the joint becomes 22.18N-m, which is strong enough to lift the weight the robot itself.

This joint is also able to rotate without limit by its design. However, the issue of twisted wires needs to be considered again. Wires connecting two sides of the arm are necessary for power transmission and communication. Too much rotation of the joint leads to the excessive twist of these wires, which causes problems mentioned previously. Therefore, we limited the range of rotation by software, as we did for the gripper joint.

3.3.4 Actuators

The nature of a truss climbing robot imposes considerable load on actuators. One case imposing large load on actuators is cantilevered configuration in a vertical plane. The robot must be able to lift up its own weight to move upward. If an additional module is connected along with a passive bar, the torque required of the joint holding an

[5]Because a worm drive mechanism is used, we can take advantage of the non-back-drivability as in case of the gripper joints.

Figure 3-13: The motors used for the Shady3D robot. The upper is a Sanyo NA4S mini gearmotor with a 298:1 gearhead. This is used for the gripper opening/closing mechanism. The lower is a MocroMo 1724T006SR DC micromotor combined with a 66:1 gearhead. This is used for joint actuation.

environment truss increases drastically. For example, in our system with the weight of 1.340kg and the gripper-to-gripper distance of 180mm, the torque required to lift up one robot is 1.18N-m. In order to lift up a robot-passive bar-robot linkage in cantilevered configuration, however, 7.17N-m is required.[6] Actuators for joints need to provide torque to meet this requirement.

In addition to required torque, the size of an actuator should be considered. Generally, the torque of an actuator increases as the size increases. However, a large actuator is heavy and increases required torque. Moreover, since a large actuator takes more space, the overall size of robot increases. This leads to the increase of a moment arm and results in larger required torque. Therefore, we need to balance between the size and the torque of an actuator.

After a thorough search for suitable actuators for joints, we decided to use a MicroMo 1724T006SR DC micromotor combined with a 66:1 gearhead (shown in Figure 3-13). This motor provides 277.2mN-m at the output of the gearhead. Since we use 1:75 or 1:80 gear reduction for the joints, the torque generated by them becomes 20.79N-m or 22.18N-m, assuming no friction loss. This is strong enough to perform

[6]For this calculation, we assumed that the distance between the grippers of two modules on the passive bar is the same as the center-to-center distance of the module, which is 180mm. A acrylic tube with square cross section is used as a passive bar. The weight of the bar is 0.03kg.

rotation in cantilevered configuration, even when an additional module is connected with a passive bar. The motor is 50mm long (excluding the length of the shaft), and 17mm in diameter. This size fits in the dimension of the robot.[7]

As for motors for the grippers, Sanyo NA4S mini gearmotors with 298:1 gearheads are used. This motor provides 20mN-m torque, and is 25mm long and 12mm wide (See Figure 3-13). Because of the non-back-drivability of the gripper gearing mechanism, we do not need large torque to resist the force trying to open the gripper (See Section 3.3.2 for detailed discussion on non-back-drivability). This motor proved to be good enough to make a firm grip in various situations.

In regard of mounting, motors for the joints are mounted with auxiliary mounting plates made of ABS plastic. First, the mounting plate is fixed on the front face of the motor with M2 screws. Then, the mounting plate is installed on a block in a joint assembly with #2-56 screws. The reason for using auxiliary plates is that it is impossible to mount the motor on the block directly because the block is bulky in the direction of the motor shaft, so a screwdriver cannot access to drive screws. Because the relatively thin plates are more prone to failure, we used five or six screws to distribute stress throughout the parts. The gripper motors are mounted between the housing cover and the housing top cover of the grippers, as described in Section 3.3.1.

We had some challenges with assembling worms with motors. The first is that the dimension of worms is in inch units while that of motor shafts in metric. The second is that the motor shaft is too short to connect a worm reliably. To resolve these problems, we machined auxiliary shafts that have a hole fitting the motor shaft and outer diameter fitting the worm bore. These shafts were made long enough to cover the length of the worm and to reach a hole on the opposite side of the joint block.

[7]In the beginning, we assumed the center-to-center distance as 250 mm. Since the motor we chose required less space, we could reduce the distance to 180 mm.

3.3.5 Sensors

Joint position

The joint of the Shady3D robot needs position feedback to rotate to a certain angle. This position information must be measured with respect to a reference. Therefore, we need to obtain information on a reference position, or an absolute zero position, and relative position referenced to it.

To obtain relative position information, we use 512 counts/rev encoders installed in the joint motors. These encoders are connected to counters in motor control boards. We can read counter values to know the current joint angles. Because the resolution of encoders is very fine,[8] the counter value is divided by 64 in software before it is used for position feedback.

A reference position is necessary to determine the exact angle of a joint. Because the counter for the encoder is reset when power is off, the encoder reading is only relative and not sufficient to determine the exact angle. A sensor to indicate an absolute zero position is required. For this purpose, detector switches are installed for each joint. In the gripper joint, a detector switch is mounted under the circuit board that covers the top of the joint. It is triggered by an extrusion on the gripper housing top cover. For the middle joint, another detector switch is installed on the edge of the border between two parts of the arm. It is triggered by an extrusion on the block B of the joint when two grippers point to the same direction. The counters of joint motors are reset to zero when these switches are triggered during absolute position referencing procedure.[9] In this manner, we can set references for the joint positions. Figure 3-14 shows the absolute position referencing switch for the middle joint.

[8]Because the encoder is attached the motor before the gearhead, it counts very frequently as the joint rotates. For example, as the middle joint rotates 360 degrees, the counter counts 2703360 times.

[9]Absolute position referencing procedure is described in detail in Section 4.3.4.

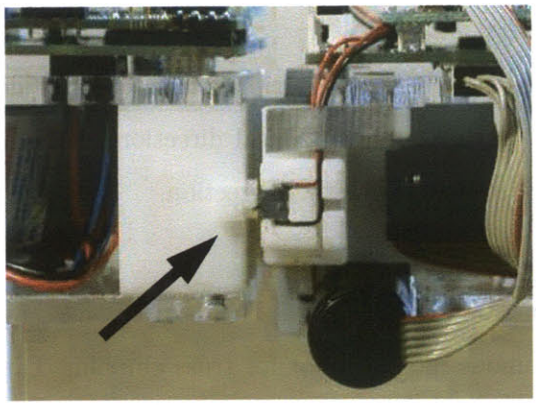


Figure 3-14: The switch for the absolute position referencing of the middle joint. This switch is turned on by an extrusion on the block B (indicated by the arrow) when both grippers point to the same direction.

Gripper state

To sense the state of the gripper, we employ a potentiometer connected to one of the gripper paddle shafts. The output voltage of the potentiometer is read by an analog-to-digital converter in a motor control board. The potentiometer is adjusted to indicate the middle value when the gripper paddles are fully closed. Since it is hard to adjust exactly to the middle value by hand, the value is precisely tuned in software.

Misalignment correction

Because the gripper joint is non-back-drivable, an active correction process for a misaligned gripper is necessary. To do this, the robot need to sense whether the gripper is misaligned or not.

Four detector switches are employed for each gripper for this purpose. Each switch is mounted on the side of each gripper paddle. Information on misalignment can be obtained by checking the states of these four switches during the closing procedure of the gripper. If the gripper is not properly aligned with a truss, one switch or two in diagonal position are triggered before others. We can adjust the corresponding gripper joint based on this information. A detailed discussion on misalignment correction is

in Section 4.3.2.

A feature of the switch used for gripper misalignment detection is that it can be triggered both in vertical and horizontal direction. It is useful because it happens that a truss push the switch in either direction.

Sensors for environment

For a fully autonomous mobile robot for truss climbing, capability to sense an environment where the robot moves is needed. With this capability, the robot can build a map of the environment, avoid obstacles, and find a path to a goal location. One of key sensing capability for a truss climbing robot is ability to find a truss. In 2-D case, some sorts of proximity sensors installed in grippers can be used for the purpose since a gripper is guaranteed to pass over a truss as the robot rotates. For example, the first version of the 2-D Shady window shading robot is equipped with two infrared proximity sensors for each barrel [8]. In 3-D case, however, finding a truss is much more complicated. We can not guarantee that a robot will meet a truss by rotating a certain degree of freedom. Moreover, because the number of degrees of freedom is much more than 2-D case (six for a fully-capable 3-D manipulator), probability to come in the vicinity of a truss by arbitrarily rotating its degrees of freedom is very low. One possible solution is using vision. With a vision system, a robot can take and process the image of the environment to determine the position and orientation of trusses. Though vision is a novel and attractive solution, it requires powerful computing capability of the microprocessor of the robot.

Unfortunately, an 8-bit microcontroller used for our current Shady3D robot is not suitable for image-processing. Though we have a 32-bit miniature computer for high-level computing, it still lacks memory to save a large amount of image data. Therefore, the Shady3D robot does not include sensors to detect the environment. Instead of gaining information with sensors, we made the robot have the information on the environment and on its location in it. Although it is obviously not ideal, we have chosen this option to make the control of the robot simple. We leave the sensing ability of environment for future work.

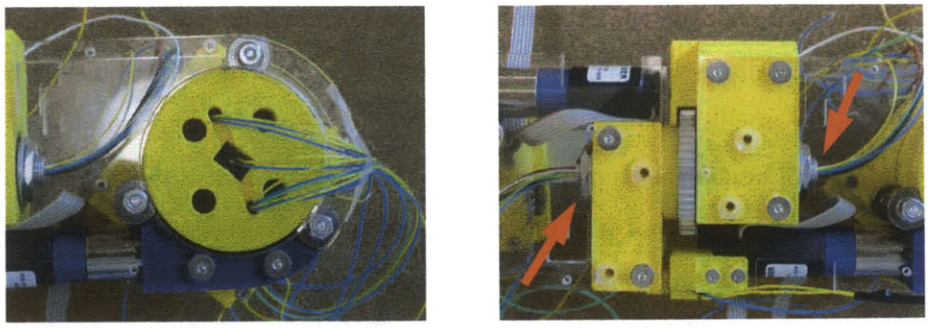
Because we assume that the robot knows the environment *a priori*, it can avoid collision with trusses without sensing. However, we implemented a function to prevent motor failure possible to occur as a result of unexpected collision or impediment. Each motor control board has a current sensor and a current limit is defined for each motor in software. If the current of a motor exceeds the limit, then an unexpected collision or impediment can be assumed and the motor is turned off. Although this is not an accurate method to detect an obstacle, it can function as a safety device for actuators.

3.3.6 Packaging

Packaging is one of important design issues. Placing many components within a restricted space is challenging especially when the size of a robot is small. Since our Shady3D robot is relatively small (250 mm long and 80 mm wide), allocating space efficiently to motors, gears, batteries, and electronics required much time and revision of design.

As mentioned in Section 3.3.3, the packaging problem led to the difference in gearing between the gripper joint and the middle joint. At the beginning, we tried to use a direct worm-worm gear drive mechanism for the middle joint like the gripper joint. For this configuration, however, the worm needed to be placed in the center of the arm width and the motor connected to the worm was extended about 30 mm out of the side boundary of the arm. This extension of the motor body might hinder the rotation of the middle joint. In order to place the motor-worm assembly, which is 80 mm long overall, the use of gearing to mediate the worm and the gear on the middle shaft was unavoidable. For this reason, we have introduced an additional shaft with a worm gear and a small spur gear as the intermediate gearing.

Wires created another packaging problem. Many wires are used in the Shady3D robot to connect sensors, motors, and batteries to the circuit boards. Since electronics are distributed on two sides of the arm, wires for communication and power transmission between two parts are also necessary. Often, these wires lie across the joints. Though we limited the rotation angle of the joints, they still need to rotate more than 360 degrees for truss climbing motion. If wires are placed outside of the joints,



(a) (b)

Figure 3-15: This figure shows the packaging of wires lying across joints. (a) Wires from a gripper are placed through holes in the housing covers. (b) Wires connecting two sides of the arm are placed through the hollow middle shaft. The circuit boards on the arm are removed for a clear view.

they would easily be entangled when the joints rotate in large angles. To prevent such a problem, wires need to be placed inside the joint assembly, as close as possible to the axis of rotation. This requirement is satisfied for both the gripper joint and the middle joint. For the gripper joint, we made holes for wires through the housing cover and the housing top cover and placed wires through them (See Figure 3-15(a). Then, the wires go through a hole on the circuit board and are plugged into connectors. In regard of the middle joint, the middle aluminum shaft was machined to have a hole through it. Wires crossing the joint were placed through the hole (See Figure 3-15(b)). Thus, the possibility of entanglement of wires is removed.

3.3.7 Electronics

The electronics of the Shady3D robot is composed of three layers: the motor control boards, the Robostix control board, and the Gumstix miniature computer. The motor control boards and the Robostix board are used for low-level motor control. The Gumstix computer is dedicated to high-level control and planning algorithms. All these components are installed on two base circuit boards on top of two parts of the arm.

The **motor control board** is a general-purpose motor controller developed in

address	motor	AtoD	sensor
0	left joint	0	left joint position referencing switch
		1	left gripper switch 0
		2	left gripper switch 1
1	right joint	0	right joint position referencing switch
		1	right gripper switch 0
		2	right gripper switch 1
2	middle joint	0	middle joint position referencing switch
		1	none
		2	none
3	left gripper	0	left gripper switch 2
		1	left gripper switch 3
		2	left gripper potentiometer
4	right gripper	0	right gripper switch 2
		1	right gripper switch 3
		2	right gripper potentiometer

Table 3.2: Organization of motor control boards

our lab. It has its own microcontroller (AVR Atmega8) to run a control code. It also has a counter to count encoder inputs from a motor, a RS485 transceiver for serial communication, and an H-bridge to drive a motor. The control code generates PWM signals according to commands given through serial communication. Not only direct PWM control but also position, velocity, torque, and current feedback control functions are implemented. In addition, there are a current sensor and three built-in analog-to-digital (AtoD) converters which can be used for various feedback. These AtoD converters can also be used as digital input pins by adjusting the configuration code of the motor control board.

In our Shady3D robot, five motor controller boards are employed. Table 3.2 shows the organization of motor control boards in the Shady3D robot. In the table, the left side means the side of the arm in which the middle joint motor is installed, and the right side means the opposite side. Each motor control board is given an address in order that the Robostix can select a motor to control. Boards with addresses 0, 2, and 3 are installed on the left circuit board and those with addresses 1 and 4 on the right circuit board. AtoD converters on them are assigned to different sensors. The

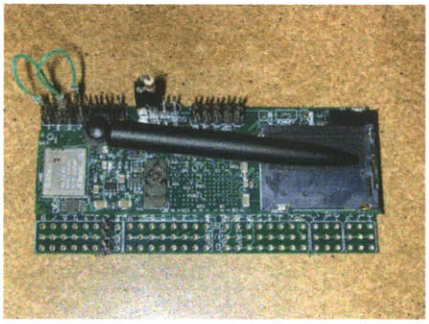


Figure 3-16: The Gumstix miniature computer mounted on the Robostix board.

AtoD converter No. 0 of each joint-controlling board is connected to the switch for absolute position referencing. The AtoD converter No. 2 of each gripper-controlling board is used to read the potentiometer value of the gripper.

Each gripper has four switches on its paddles to correct its misalignment. These switches are numbered from 0 to 3. The switch closest to the potentiometer of the gripper is assigned the number 0. Other switches are numbered clockwise from the switch 0 when the gripper is viewed from the top. These switches are connected to the boards for the joint and the gripper on each side as shown in Table 3.2. AtoD converters for switches for misalignment correction and joint position referencing are set as digital input pins.

The **Robostix** control board is a microcontroller board manufactured by Gumstix, Inc. It is equipped with an Atmega128 microcontroller. We use this board to incorporate five motor control boards in a network of RS485 serial communication. A control program running in the Atmega128 processor selects and sends commands to the motor control boards according to the message from the higher level.

The **Gumstix** (shown in Figure 3-16) miniature computer is a small Linux computer manufactured by Gumstix, Inc. It is used for high-level algorithms and planning. It communicates with Robostix through the serial port. In addition, the Gumstix computer has Bluetooth wireless communication function, which can be used for exchanging messages among multiple modules.

The circuit boards serve as a base structure on which all these components are

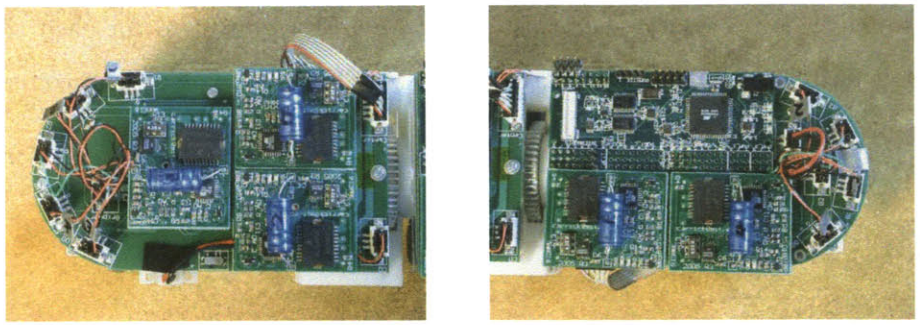


Figure 3-17: This figure shows the circuit boards of the Shady3D robot. The left image shows the circuit board on the left side, containing three motor control boards. The right image shows the circuit board on the right side, containing two motor control boards and the Robostix board. The Gumstix computer is removed to show the Robostix board clearly.

installed. All motors, sensors, and batteries are also connected to them. They provide wiring for power and communication among them. Each circuit board is installed on top of each part of the arm. On the left circuit board, three motor control boards (address 0, 2, and 3) are mounted. The right circuit contains two motor control boards (address 1 and 4) and the Robostix board. The Gumstix computer is mounted on the Robostix board. Figure 3-17 shows these circuit boards.

Two circuit boards are connected with six lines of wires. Two wires are used for RS485 serial communication, and other two are used as power lines. The remaining two wires are devoted to power switching and to interrupt for motor control boards, respectively. See Appendix B for the Shady3D schematics.

3.3.8 Power

Four 3.7V, 750mAh Polymer Li-ion battery cells are employed to provide power to the Shady3D robot. In each side of the arm, two batteries are connected to the circuit board in serial, providing 7.4V. Then, batteries in two sides are wired in parallel via wires connecting two sides. Therefore, even if batteries on one side does not function, the robot can still run.

Since batteries are divided in two sides, a method to switch both sides syn-

chronously is necessary. For this, power MOSFETs are used for each side, and a single switch is connected to gates of two MOSFETs. A line of wire connecting two circuit boards is used for this purpose.

3.4 Summary

The Shady3D truss climbing robot is the 3-D extension of the 2-D Shady window shading robot. Among several design alternatives, the 3-DOF design model, which has two gripper joints and one rolling joint in the middle, is selected. Two 3-DOF modules can form a 6-DOF manipulator using a passive bar as a medium for connection. The Shady3D hardware has been developed based on the 3-DOF design model. The grippers of the Shady3D robot feature retractibility and compliance for misalignment and tilt. The two gripper joints and the middle joint substantiate the three motive degrees of freedom of the 3-DOF model. The sensors are used for joint positioning, gripper opening/closing, and gripper misalignment correction. The electronics of the Shady3D robot is composed of three layers and divided on two circuit boards.

Chapter 4

Control

This chapter describes the hardware control of the Shady3D truss climbing robot. First, an overview of the control system of the Shady3D robot is presented. Next, the software representation of the state of the Shady3D robot is presented. Finally, the basic motion primitives of the robot are described. All control algorithms except for joint absolute position referencing are implemented in Java codes running on high-level control computer (the Gusmtix miniature computer or a workstation).[1]

4.1 Overview

The control system of the Shady3D robot is composed of three layers: the motor controllers, the Robostix microcontroller, and the Gumstix computer (or a workstation). Figure 4-1 shows a simplified block diagram of the control system. Sensor information is transferred from the low-level layer (the motor controllers) to the high-level layer (the Gumstix or the workstation). Five motor controllers collect information from the sensors and send this information to the Robostix microcontroller. The Robostix organizes these sensor values and sends them to the Gumstix or the workstation via serial communication. The control program running on the highest layer evaluates the state of the Shady3D robot based on these sensor readings. On the other hand,

[1]The motion of joint absolute position referencing is implemented in the low-level control code running on the Robostix. The reason why it is not included in the Java code is explained in Section 4.3.4.

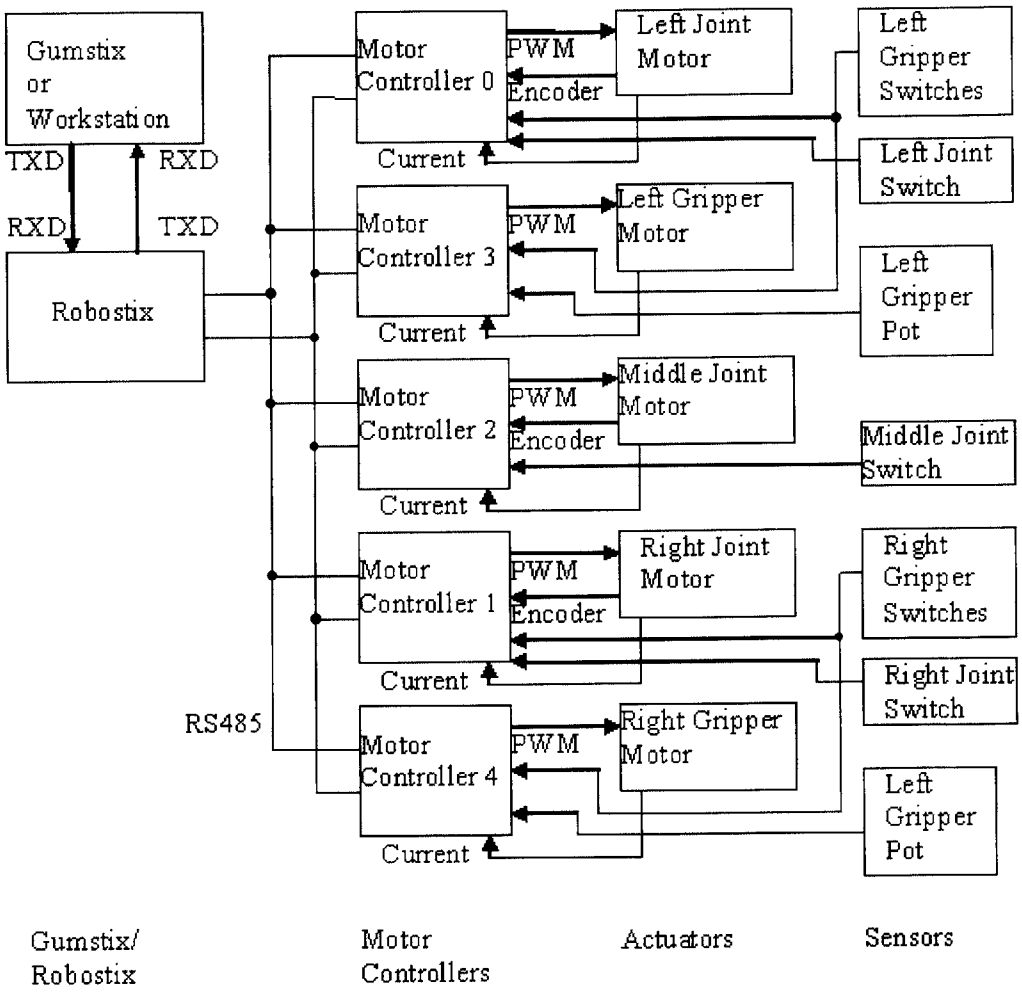


Figure 4-1: A simplified block diagram of the Shady3D robot control system.

the control commands flow from the high level to the low level. To perform basic motion primitives, the control program sends necessary commands to the Robostix. According to the commands, the Robostix selects an appropriate motor controller and transmits commands to it. Then, the motor controller generates a PWM signal to drive the motor accordingly.

4.2 State of the Shady3D robot

The state of the Shady3D robot needs to be clearly defined in order for the robot to move to a desired location and to have a necessary pose. Based on the state, the robot can compute a path to follow and a sequence of actuation for steps. It can also check any obstacles in its way and avoid them.

The representation of the state of the Shady3D robot includes both the internal state and the external state. The internal state represents information that is obtained from the sensors of the robot. It does not show a relation between the robot and its environment. For example, the joint angle and the gripper state are the elements of the internal state. On the other hand, the external state describes the relation of the robot with the environment. Information such as the position and the orientation of the robot are included in the external state. The elements of these two states are interrelated. Figure 4-2 illustrates elements to represent the state of the Shady3D robot.

4.2.1 Internal state

The internal state describes the state of the robot based on the sensor values. This state provides information on the configuration of the joints and grippers.

Joint angles The joint angle is the angle of a joint with respect to its reference position. These angles determine the kinematic configuration of the robot.

To define the joint angles, some geometric terms need to be defined. The **body line** is a line connecting the centers of rotation of the left and right joints. This line

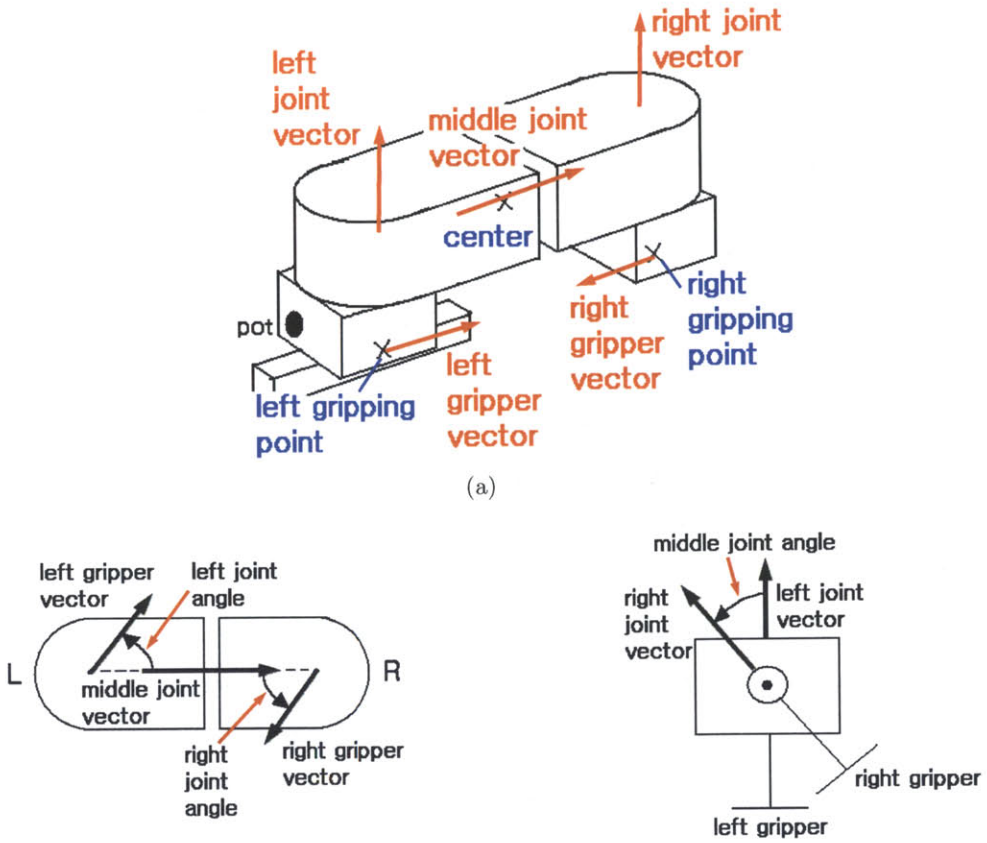


Figure 4-2: This figure shows the elements of the state of the Shady3D robot. (a) The vectors and points of the state are illustrated. Note that the gripper vector directs toward the opposite direction to the potentiometer of the gripper. (b) The top view of the Shady3D robot. The relation between the vectors and gripper joint angles is shown. (c) The right view of the robot. The relation between the middle joint angle and two gripper joint vectors is shown. Refer to the text for the definitions of the elements.

is parallel to the axis of rotation of the middle joint. The **top** of the Shady3D robot is defined as the side on which circuit boards are installed. Because two parts of the arm can rotate relative to each other, the top for the left part can differ from that for the right part. This notion is used to define the positive direction of the left and right joint angles.

The **left and right joint angles** are defined so that they are zero when the corresponding gripper can grip a bar that is coincident with the body line. However, the gripper can hold such a bar with joint angles in 180 degree difference. Therefore, a configuration where the potentiometer of the gripper points outward direction is chosen to define the zero joint angle. A positive angle means that the corresponding joint is rotated counterclockwise about the axis of rotation when viewed from the top.

The **middle joint angle** is defined as an angle that the axes of rotation of the left and right joints make. It is zero when the two axes are parallel and the top sides for both parts of the arm face the same direction. A positive angle means that the right axis is rotated counterclockwise from the left axis when viewed from the right.[2]

All the joint angles are represented in degrees. Because the position feedback from the counters of joint motors are not in degrees, conversion factors are used to relate raw position feedback units with degrees. The gripper joints (left and right) and the middle joint use different conversion factors since their gear reduction ratios are different (1:75 for the gripper joints and 1:80 for the middle joint).

Gripper states The gripper state of each gripper is described in the normalized gripper unit (NGU) in the range of zero and one. Zero NGU means that the gripper is fully opened, and one NGU means that it is fully closed. Sometimes, a gripper state can have a value larger than one if the gripper is closed more than the point where the potentiometer indicates fully-closed state. It occurs when the gripper is commanded to close as firm as possible regardless of the position feedback from the potentiometer. This gripper state larger than one NGU is also considered as the fully-closed state.

The position feedback from the potentiometer of the gripper is not represented in

[2]The positive direction of joints is discussed again in the description of the external state in Section 4.2.2.

NGU. Therefore, a conversion factor is introduced to change raw position feedback values to values in NGU.

States of switches The states of the switches of the Shady3D robot are included in the state for the purpose of the correction of gripper misalignment and the absolute position referencing of joints. The **gripper switch states** for each gripper represent the states of four switches mounted on its paddles. The **joint switch state** for each joint reflects the state of its absolute position referencing switch. The switch states are implemented as boolean variables. The values are true when the corresponding switches are on, false when off.

Current The current each motor draws is checked through the current sensor embedded in the motor control board. This value is an element of the internal state and used to protect the motor from the damage caused by high current flow. Raw current feedback values from the motor control boards are converted to ampere by the conversion factor.

4.2.2 External state

The external state of the Shady3D robot identifies its relation with an environment where it moves. The state includes information on the position and orientation of the robot in the environment.

Joint vectors The joint vectors represent the direction of the joints of the robot. They also define the direction of positive joint angles.

The **middle joint vector** is defined as a unit vector from the center of the left joint to the center of the right joint. It is parallel to the axis of rotation of the middle joint and the body line of the robot. The **left joint vector** is a unit vector parallel to the axis of rotation of the left joint. It points toward the top of the left part of the arm. In the same manner, the **right joint vector** is defined as a unit vector parallel to the axis of rotation of the right joint. It also directs toward the top of the right

part of the arm.

The positive direction of the joint angles can be defined with these joint vectors. For the gripper joints, the positive direction is counterclockwise direction about the corresponding joint vector. A positive middle joint angle means the counterclockwise rotation of the right joint vector from the left joint vector about the middle joint vector. The relation between the joint vectors and joint angles is illustrated in Figure 4-2.

Gripper vectors The gripper vector of each gripper is defined as a unit vector that makes an counterclockwise angle with the body line of the robot. It is parallel with a bar on which the corresponding gripper can grip on.

The gripper vectors can be computed from the joint vectors and joint angles. The left gripper vector is obtained by rotating the middle joint vector by the left joint angle about the left joint vector. As for the right gripper vector, the vector opposite to the middle joint vector is used instead of it. The right gripper vector is obtained by rotating the *inverted* middle joint vector by the right joint angle about the right joint vector. See Figure 4-2 for the pictured description.

Gripping points The gripping point is a representative point for each gripper. This point is defined as an intersection of the axis of the corresponding joint of each gripper and the centerline of a bar gripped by the gripper. The centerline of a bar is a line going through the center point of its cross section.

Center The center of the Shady3D robot is defined as the midpoint of the two gripper joint centers. The gripper joint centers are the intersections of the axis of the middle joint and the axes of the gripper joints. These gripper joint centers are not explicitly included in the state representation. They are computed by other state variables when necessary.

In-Grip information This element of the external state is used to indicate what is in the gripper. When the gripper is open, it is set to `null` in software. It is also `null`

in the case where the gripper is closed but it holds nothing. If the gripper grips on a truss, the information of the truss is assigned to this variable. In our implementation, the environment has a set of nodes on trusses where the robot can be located.[3] When one of the grippers of the robot grips on a certain node in the environment, the node is assigned to the corresponding in-grip variable.[4]

Anchor The anchor indicates which side of the robot is the starting point to compute the whole external state of the robot. An open gripper cannot be the anchor gripper. It cannot be a solid basis on which the position and orientation of the robot is evaluated because its gripping point is more likely to be affected by the errors of the joints and grippers than a gripper closed on a truss. If both grippers are gripping trusses, either side can be the anchor. The case where both grippers are open is not considered because the robot is not connected to any trusses so its position and orientation cannot be defined.

Fault The fault state variable indicates any situation that the robot does not operate normally or a specified action would cause a failure of the robot. The robot stops operation to protect its hardware when it comes into the fault state. Table 4.1 lists possible faults.

4.2.3 Evaluation of the state

To keep track of the movement, the state of the Shady3D robot needs to be updated after every motion primitive is performed. Updating the state is carried out in two steps. First, the internal state is updated with sensor readings. Then, the external state is computed from the updated internal state and the anchor information.

The update of the internal state is straightforward. The software simply goes through all sensors and reads their values. To change raw sensor readings to values

[3]For the detailed description of the environment representation, see Section 5.2.

[4]The gripper can hold a passive bar instead of a truss fixed in the environment. The abstract representation of passive bars has not been developed yet.

Fault name	Description
no fault	No faults.
unknown fault	An unknown fault occurred.
timeout	Communication with the hardware is timed out.
over current	The current in an actuator is too high.
would collide	The specified action might cause a collision.
collision	A collision may have occurred.
would motion limit	The specified action might cause a mechanism DOF to go over its motion limit.
motion limit	The specified action may have caused a mechanism DOF to exceed its motion limit.
unaligned	The gripper may not be properly aligned with an environment truss.
would duel	A joint can not be rotated because both grippers are closed on trusses in an environment.
comm error	Communication with the lower level hardware has an error.
interrupted	An ongoing motion has been interrupted in software.
following error	A position controlled actuator has strayed outside of its allowed error.
would following error	A position command would have been farther from the current actuator position than the allowed following error.

Table 4.1: Fault list of the Shady3D robot

in units used for the state representation, appropriate conversion factors are applied according to the type of sensors.

To compute the external state, information not only on the internal state but also on the anchor is required. The anchor information is composed of three external state variables—the gripping point of the anchor gripper, the gripper vector of the anchor gripper, and the joint vector of the anchor joint. This information is given by a user at the initialization stage of the robot. Once initialized, the robot updates it automatically as the state is updated.

The evaluation of the external state is implemented in several steps. First, the software checks what the anchor gripper grips on. It goes through all nodes in the environment and checks whether its anchor gripper is placed on a certain node or not. In order to check this, the anchor gripping point, the anchor gripper vector, and the anchor joint vector are compared with the position, the direction vector, and the surface normal vector of the node, respectively.[5] To conclude that the gripper is really located on a node, the gripping point and the joint vector must be equal to the position and the surface normal vector of the node, respectively. In addition, the gripper vector must be parallel to the direction vector of the node. Because of errors in real implementation, approximate equality and parallelism with tolerances are used instead of exact comparison. If all three anchor variables are approximately equal or parallel to the corresponding elements, the node is assigned to the in-grip variable of the anchor.[6] Then, the elements of the node information are copied and assigned to the corresponding anchor variables.

The next step of the evaluation is computing other vectors and points of the state from the anchor information and the internal state. The middle joint vector is first computed by rotating the anchor gripper vector about the anchor joint vector. Then,

[5]Every node has information on its position, direction, and surface normal direction. The definition of a node is presented in Section 5.2.

[6]Tolerances are set to decide approximate equality and parallelism. A gripping point is considered equal to the position of a node if the distance between two points is less than 0.001m. A joint vector and the surface normal vector of a node are approximately equal if differences between the corresponding components of the vectors are less than 0.001m. A gripper vector is considered approximately parallel to the direction vector of a node if the angle between two vectors is smaller than 2.0 degrees.

the joint vector and gripper vector on the opposite side to the anchor are computed from the middle joint vector and joint angles. The center point and the gripping point on the opposite side to the anchor are computed by translating the anchor gripping point along the joint vectors. This procedure is basically the same both for the left anchor and the right anchor. However, detailed computation is slightly different for the two cases. This difference comes from the direction of the middle joint: from the left joint to the right joint. The detailed procedure for each case is explained below.

The left anchor case In the case where the anchor is the left side, the middle joint vector is obtained by rotating the left gripper vector about the left joint vector by the negative of the left joint angle. For example, if the left joint angle is 45 degrees, the left gripper vector is rotated by -45 degrees to gain the middle joint vector. The sign of the joint angle is inverted because the joint angle is measured from the middle joint vector to the gripper vector and, in this case, the middle joint vector is computed backward from the gripper vector.

After the middle joint vector, the right joint vector and the right gripper vector are computed successively. The right joint vector is gained by rotating the left joint vector about the middle joint vector by the middle joint angle. Then, the right gripper vector is computed by rotating the middle joint vector about the right joint vector by the right joint angle plus 180 degrees. The additional 180-degree rotation is needed because the reference vector for the right joint angle is the *inverted* middle joint vector.

The center point of the robot is computed by translating the left gripping point along the left joint vector and middle joint vector. The gripping point is translated in the direction of the joint vector by the distance between the joint center and the gripping point.[7] Then, it is translated in the direction of the middle joint vector by the half of the center-to-center distance.

Finally, the right gripping point is obtained by translating the center point along the middle joint vector and the *inverted* right joint vector. The distances of translation

[7]This distance is 69.5 mm in our Shady3D robot.

are the half of the center-to-center distance and the gripping-point-to-joint-center distance, respectively.

The right anchor case The idea of state computation in case of the right anchor is basically the same as the left anchor case. However, there are subtle differences caused by the direction of the middle joint vector.

The middle joint vector is computed by rotating the right gripper vector about the right joint vector by the negative of the right joint angle plus 180 degrees. The additional 180-degree rotation is required since the middle joint vector is in the opposite direction to the reference vector for the measurement of the right joint angle. For example, if the right joint angle is 45 degrees, the right gripper vector must be rotated by 135 degrees (-45 plus 180) to obtain the correct middle joint vector.

Next, the left joint vector and the left gripper vector are computed successively. The left joint vector is evaluated by rotating the right joint vector about the middle joint vector by the *negative* of the middle joint angle. The reason for the use of the inverted sign for the middle joint angle is that the angle is measured from the left joint vector to the right joint vector. Therefore, if the middle joint angle is 45 degrees, the left joint vector is obtained by rotating the right joint vector by -45 degrees. Then, the left gripper vector is computed by rotating the middle joint vector about the left joint vector by the left joint angle.

The center point and the left gripping point are evaluated in the same way for the left anchor case. The difference is that the *inverted* middle joint vector is used instead of the middle joint vector.

The final step of the evaluation of the Shady3D state is to determine what is in the gripper on the opposite side of the anchor. If the opposite gripper is open, the in-grip variable for it is set to `null`. If it is closed, the software goes through the list of nodes in the environment and checks if the gripper is located on a certain node. The criteria for the check is the same as those used for the anchor gripper. If the gripper is proved to be on a certain node, the node is assigned to the in-grip variable

for it.[8]

Through the procedure described above, both the internal state and the external state of the Shady3D state are evaluated. This evaluation is performed automatically after each basic motion primitive. It can also be manually done by a user.

4.3 Basic motion primitives

The Shady3D robot performs four basic motion primitives: gripper opening motion, gripper closing motion, joint rotation, and joint absolute position referencing. All motion primitives except for joint absolute position referencing are implemented in the high-level Java code. The motion primitive of joint absolute position referencing is performed by the low-level control code running on the Robostix. These motion primitives can be combined to achieve higher-level motion.

4.3.1 Gripper opening motion

This motion primitive is to open the paddles of the designated gripper so that it can release a truss. After the motion primitive is performed, the gripper paddles are fully retracted behind the contact surface of the gripper housing in order to prevent collision while the gripper moves over a truss.

Before the gripper is actually opened, we need to make sure that the opposite gripper is closed on a truss. If not, opening the designated gripper cause the robot to fall off a truss. Therefore, the preliminary check on the opposite gripper state is executed before the actual opening motion of the designated gripper. If the opposite gripper is not closed properly, procedure is stopped and the failure of opening motion is reported.

If the opposite gripper is closed properly, the actual opening procedure is carried out. It is performed by sending a position control command to the corresponding gripper motor. If the gripper is partially open, the control program simply sends

[8]Since the gripper can hold a passive bar as well as a truss fixed in the environment, this procedure should include investigation for such bars. This check procedure for passive bars will be implemented in the future.

a position command indicating the fully open position, and waits until the gripper paddles reach the fully-open state. Then, it turns the gripper motor off by sending a zero PWM command.

If the gripper to open is currently closed, some additional operations are performed to maximize the stability of the robot. The fact that the designated gripper is currently closed means that both grippers are gripping on trusses. In such a configuration, the grippers may not hold trusses as firm as possible because of slight misalignment of grippers. Even though gripper misalignment is corrected during gripper closing motion as described in Section 4.3.2, there is still possibility that small misalignment remains not corrected. Such misalignment may result in a grip which is not completely firm even though the gripper is closed fully and does not slip on a truss. To solve this problem, the opening gripper is first opened to 99% of the fully closed state and the other gripper is closed as firm as possible by driving its motor at the maximum power during a period of time.[9] The opposite gripper can conform to a truss because the loosened gripper allows the body of the robot to move slightly. This procedure contributes to the stable attachment of the robot to environment trusses. After making a firm grip for the opposite gripper, the opening gripper is fully opened through the same procedure mentioned in the previous paragraph.

It is possible that the movement of the gripper paddles is blocked for some reason. For example, an unexpected obstacle can come into the path of the paddles. In this case, the motor of the gripper needs to be shut down to protect it from heat caused by excessive current. Therefore, current drawn by the motor is checked throughout the actual opening procedure. If the current exceeds the limit (0.3A in our case), the motor is forcefully stopped and the over-current fault is set.

If the gripper paddles are successfully opened, the control program updates the state of the robot. If the gripper that is the anchor of the robot is opened, the anchor is switched to the opposite gripper because an open gripper cannot be the anchor by definition. After reassigning the anchor, the full state of the robot is updated by the procedure shown in Section 4.2.3. Finally, this motion primitive is completed by

[9]In our implementation, we assigned one second.

reporting success or failure based on the final state of the gripper.

4.3.2 Gripper closing motion

The purpose of this motion primitive is to make a reliable grip on a truss. A major challenge to be handled to reach this goal is the correction of gripper misalignment. This task is performed by incorporating four detector switches mounted on gripper paddles and the corresponding joint.

Gripper paddles are closed by sending a motor for the gripper a position control command. A position value corresponding to the closed state is given to the motor. While the gripper paddles are closing, the control program continuously check the states of the gripper switches. If at least one of the four switches is turned on, which means an edge of a gripper paddle touches a truss, the gripper motor is stopped and the program investigates in which direction the gripper is misaligned.

Figure 4-3 shows possible combinations of the gripper switch states. If switch 0 or switch 2 is turned on and other switches are not, we can assume that the gripper is misaligned counterclockwise with respect to the truss. On the other hand, if switch 1 or switch 3 is turned on and other switches are not, the gripper can be assumed to be misaligned clockwise with respect to the truss. Other combinations are interpreted as the gripper is properly aligned with the truss.

If gripper misalignment is detected, the joint corresponding to the gripper is rotated in the direction to correct the misalignment. For example, it is rotated clockwise if the gripper is misaligned counterclockwise. The joint is rotated until the misalignment condition is resolved. If the gripper is misaligned counterclockwise, for instance, the joint is turned until switch 1 or 3 is triggered or both switch 0 and 2 are turned off. After the misalignment condition is resolved, the gripper paddles continue closing motion. Figure 4-4 shows the procedure of correcting gripper misalignment. During the whole procedure, current flowing through the gripper motor and the joint motor is checked to protect them from damage caused by excessive current.

This procedure is repeated until the gripper state reaches 98% of the fully-closed state, and then, the gripper motor is stopped. Next, the joint angle after the gripper

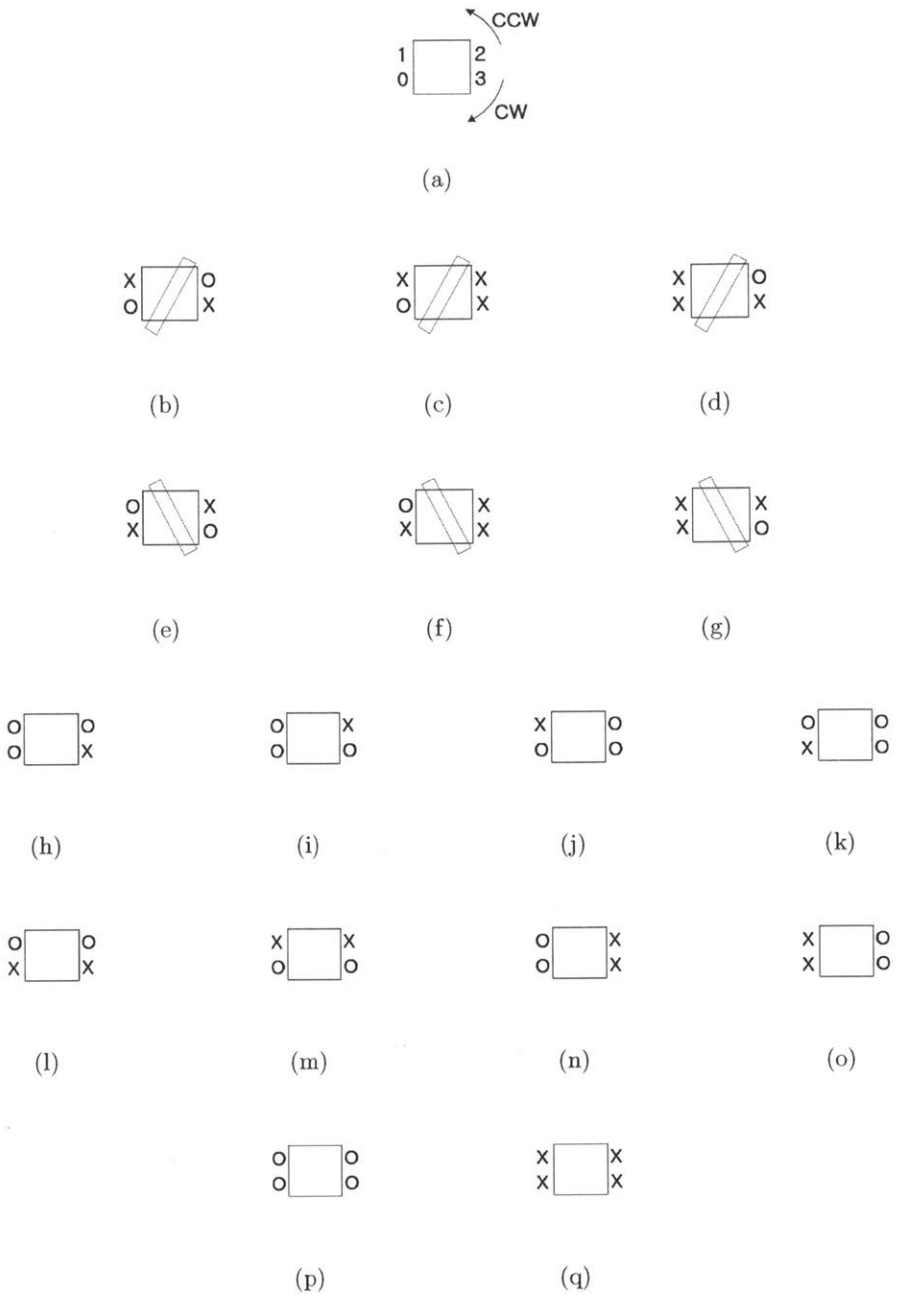


Figure 4-3: This figure shows all combinations of the gripper switch states. O means the switch is on, and X means off. (a) The numbering of gripper switches. (b)-(d) These combinations indicate that the gripper is misaligned counterclockwise. (e)-(g) These combinations indicate that the gripper is misaligned clockwise. (h)-(q) For these combinations, the gripper is assumed aligned properly.

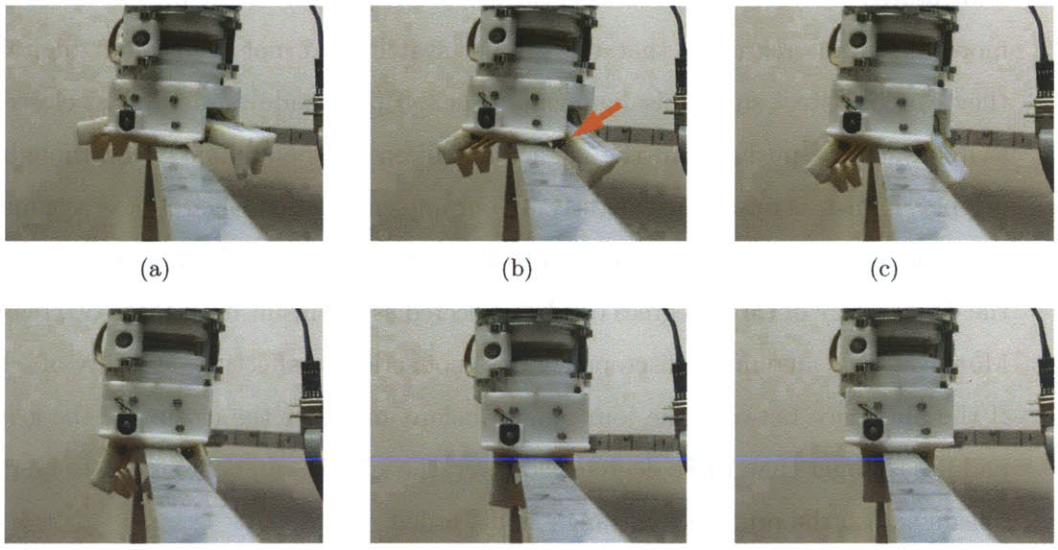


Figure 4-4: This figure shows the procedure of gripper misalignment correction. The gripper misaligned clockwise in (a) is aligned with the truss during closing motion. When a switch on the gripper paddle (switch 4) is pushed by the truss, as indicated by a red arrow in (b), clockwise misalignment is detected and the joint is rotated counterclockwise to correct it (c). The correction procedure is repeated as the gripper paddles continue to close as shown (d) and (e). Finally, the gripper becomes properly aligned with truss when fully closed (f).

closed is compared with that before the close motion, which is saved at the beginning of the gripper close motion. If the difference between them is negligible, specifically smaller than three degrees, the joint is rotated to the original joint angle. This calibration is performed for two reasons. The first is that the gripper joint angle has an error caused by gear backlash and gaps between the joint bearings and gripper housing covers. Because of this error, the program may have detected misalignment and rotated the joint when, in fact, the gripper is properly aligned. The second and more important reason is that the gripper switches are not analog but digital, so they cannot detect small misalignment. When gripper paddles are almost closed, all switches can be pushed and the aligned condition can be assumed even though the gripper, in fact, is misaligned slightly. In our implementation, the robot is assumed to have the map of the environment and to be aligned correctly with a truss. Therefore, the discrepancy of three degrees can be neglected as the result from these two reasons. Moreover, this amount of discrepancy does not seriously affect the stability of a grip. If the difference between the joint angles before and after the closing of the gripper paddles is larger than three degrees, however, the gripper can be assumed to be really misaligned so the original joint angle is discarded.

After calibrating a negligible error, the gripper paddles are closed as firm as possible to maximize the stability of the grip. This is done by driving the gripper motor at the maximum power for two seconds. Then, the motor is turned off.

The final stage of gripper closing motion is the update of the Shady3D state. Finally, the control program examines the final state of the gripper and reports success or failure accordingly.

4.3.3 Joint rotation

Rotating the joints of the Shady3D robot is a fundamental motion primitive to move it from one position to another in a truss structure. The rotating motion is implemented in two ways. One is rotating a joint *by* a given angle, and the other is rotating a joint *to* a desired joint angle (goal position).

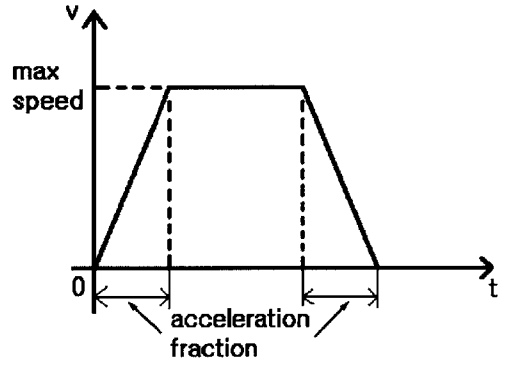
An important thing needed to be checked prior to actual motion is whether both

grippers are closed on trusses fixed in the environment. If any joint is rotated when both grippers grip on fixed trusses, the robot hardware will be broken. Therefore, rotation is not executed and the "would duel" fault is set in such a case. Otherwise, actual rotating motion is executed as described in the following.

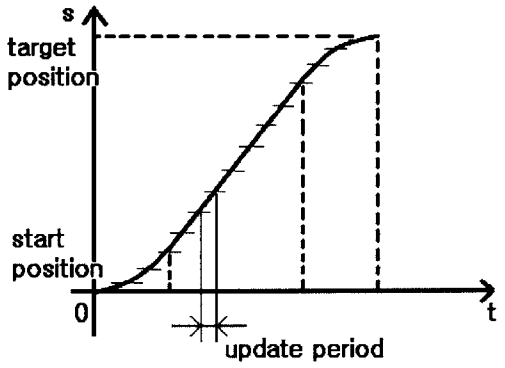
Rotation *by* a given angle Rotating motion is executed using a position profile generated by a trapezoid velocity profile. With a given angle and the current joint angle, the control program computes the position profile for the joint to follow. Figure 4-5 shows position and velocity profiles used for rotation motion. The duration of motion is computed from the given angle to rotate, the maximum speed, and the fraction of duration for acceleration. The maximum speed of rotation and the fraction for acceleration have been determined as 12 degrees per second and 5%, respectively. As shown in Figure 4-5(a), the velocity is increased linearly to the maximum speed during the acceleration fraction of the duration and is kept constant at the maximum speed. Then, it is decreased to zero during the acceleration fraction of the duration as the joint approaches the target position. This trapezoid velocity profile generates smooth acceleration and deceleration of motion. Based on the velocity profile, the position profile is obtained as shown in Figure 4-5(b).

The control program extracts an intermediate way point (shown as the short horizontal line segments in Figure 4-5(b)) every update period of motion. Then, it sends a position control command for that way point to the joint motor and waits for the update period. After the update period, it extracts the next way point and repeats the procedure. This iteration continues until the joint reaches the target angle, which is obtained by adding the given rotation angle to the angle before the motion.

At each step of the iteration, the program checks whether the joint follows the position control command for each way point. This checking is performed twice: before and after sending the control command. Before sending the command, the program checks whether the way point is too far from the current angle for the joint to reach. It does this by comparing the difference between two values with



(a)



(b)

Figure 4-5: This figure shows examples of velocity and position profiles used for joint rotation. Given an angle to rotate, the control program generates a trapezoid velocity profile and a corresponding position profile. It assigns way points to the joint motor every update period. The way points are marked with the horizontal line segments.

the maximum rotation error, which is set to 10 degrees in our implementation. If the difference is larger than the allowable error, the "would following error" fault is set and the motion fails. Otherwise, the program sends out the command and waits for the update period. After the update period, it checks whether the joint has actually followed the position control command during the period. If not, the "following error" fault is set and the motion fails. Otherwise, the program moves on to the next iteration step.

After the iteration finishes, the control program delays for a period of time to allow the joint to be settled in the final angle. We assigned 1.5 seconds for this settling period. Finally, the state of the robot is updated and the rotating motion is completed.

Rotation *to* a goal angle This type of rotation is implemented by computing the difference between the target angle and the current angle and performing "rotation *by* a given angle" motion with it. However, one additional preliminary check is carried out. The program checks if the target angle is out of the allowed range for joint angles. If it is, the "would motion limit" fault is set and the motion fails. The motion limits of the joint angles are ± 270 degrees for the gripper joints and ± 180 degrees for the middle joint.

4.3.4 Joint absolute position referencing

As mentioned in Section 3.3.5, each joint has a detector switch to set a reference for its angle. The position referencing switch for the gripper is triggered when the potentiometer of the gripper points to the opposite direction to the center of the robot. This position agrees with the definition of the zero joint angle of the gripper joints. For the middle joint, the switch is turned on when two gripper joint vectors (left and right) points toward the same direction.

The procedure of joint absolute position referencing is straightforward. The joint is rotated in one direction until the referencing switch is turned on. When the switch is triggered, the joint motor is stopped and the counter of the motor is set to zero.

This function was implemented first in the Robostix code. The Robostix code has worked well. We also tried to implement this function in the high-level Java code. However, the method in the Java code for joint absolute position referencing disrupted the control system and finally caused the failure of the control program. The cause of this error has not been clarified yet.

In addition, there is room for improvement for this motion primitive. First of all, the position where the detector switch is triggered, in fact, is not the exact zero position. The switch can be placed on a circuit board or on a switch mounting block with some tolerance. Moreover, the switch has *ranges* for on and off states, not a precise point. For these reasons, the position where the counter is set to zero may not be the exact zero position in fact. This error needs to be calibrated by setting a specific offset value for each joint.

Second, rotating the joints in one direction to find the zero position can cause wires in the joints to be twisted excessively. This results from the fact that the joint angles after the previous run are not known. For example, if the joint angle was 270 degrees when the previous run finished and the direction of rotation for position referencing is counterclockwise, the joint angle is set to zero in a position that is, in fact, 360 degrees. If this kind of situation occurs many times, the wires will be twisted counterclockwise more and more. One possible solution for this problem is to rotate the joint in the opposite direction by some angle and then to start the search for the zero position. However, this cannot be a perfect solution because this method can cause the same problem in the opposite direction.

Finally, because the joint angles are not correct, the state of the robot is not determined when joint absolute position referencing is performed. As a result, collision against trusses in the environment cannot be predicted during the procedure. Search for a reliable solution for this problem is left for the future research.

For the reasons above, this motion primitive has not been used for the autonomous operation of the robot. Currently, it is used only for manual position referencing, not on trusses.

4.4 Summary

The control system of the Shady3D robot is composed of three layers: the motor controllers, the Robostix microcontroller, and the Gumstix computer. The sensor information flows from the low level to high level, and the commands flows from the high level to low level. The state of the Shady3D robot includes the internal state and external state. Both states are updated after every motion primitive is performed. The Shady3D robot performs four basic motion primitives: gripper opening motion, gripper closing motion, joint rotation, and joint absolute position referencing.

Chapter 5

Planning

In the previous chapters, the hardware design and basic motion primitives of the Shady3D robot have been presented. This chapter explores planning algorithms for the locomotion of the robot in a 3-D truss environment. First, an algorithm for the robot to make a single step is presented. Then, this single step is consecutively combined by a path planning algorithm to move the robot from a certain location to a goal location in the environment along the most efficient path between them. In addition to algorithms for the navigation of a single robot (module), algorithms for multiple-module cooperation for locomotion and self-assembling are also studied.

5.1 Assumptions

Before developing planning algorithms, some fundamental assumptions were made about the characteristics of the robot and the environment. These assumptions provide a basis for the representation of the truss environment where the robot navigates and the planning algorithms of the robot.

First of all, we have assumed that the robot knows the environment *a priori*. This assumption is necessary because our Shady3D robot is not equipped with sensors to obtain information on the environment, as mentioned in Section 3.3.5. If the robot has the map of the environment, it can decide where to move next without sensing. It can also avoid collision to trusses in the environment. To implement this assumption,

the robot is given an instance of the environment in software. The instance of the environment has information on the position and orientation of trusses and on the nodes where the robot can be located. The representation of the environment is described in detail in Section 5.2.

In addition to the information on the environment, the robot is assumed to know its current location in the environment. The initial location is given by a user at the initialization stage. After that, the robot updates its location after every motion primitive using its information on the environment.

Next, discrete gripping points are assigned on trusses in the environment. In reality, the robot can grip on any points on a truss. For simplicity of algorithm development, however, it has been assumed that the robot can grip on some specific points on trusses. These discrete gripping points are implemented as *nodes* in the environment representation.

Finally, it has been assumed that passive bars used for the cooperation of multiple modules are placed in designated positions in the environment. The environment representation includes information on such bars. Because the robot knows the environment, it also has knowledge on the position of the bars and can pick them up according to its need.

5.2 Environment

Based on the assumptions stated in the previous section, the representation of a truss environment has been developed. The environment representation is composed of two parts: the abstract graph for gripping points and the representation of physical trusses.

5.2.1 Abstract graph for gripping points

The abstract graph of the environment is used for the path planning of the robot. The graph is composed of *nodes*, which represent discrete gripping points on trusses, and *edges*, which represent connection between two nodes. The robot is located on

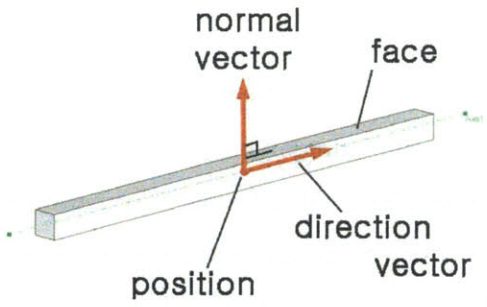


Figure 5-1: This figure shows the elements of the node. The dashed line is the centerline of the truss.

one of nodes. Given a goal location, which corresponds to another node, the robot computes the path in the graph from its current node to the goal node. The path becomes a sequence of nodes for the robot to follow.

Node A node represents a gripping point on which the robot can grip on. It is specified by three elements: the position of the node, the direction of the node, and the face of the node. Figure 5-1 illustrates these elements.

The position of a node determines where it is located in the environment. It is defined as a point on the centerline of a truss on which the node is. When a gripper holds a node, the position of the node coincides with the gripping point of the gripper.

The direction of a node is the direction of a truss on which the node is located on. This element is represented by the direction vector of the node, which is a vector parallel to the truss. When a gripper is closed on a node, the direction vector of the node is parallel to the gripper vector of the gripper.

The face of a node determines on which side of a truss a gripper can be located. Because a truss for the Shady3D robot has a square cross section, a gripper can approach the truss in four different directions—one for each face. Therefore, it is necessary to include the information of the face in the node representation to single out one specific node. The information of the face of a node is specified by the normal

vector. The normal vector is perpendicular to the face that contacts with the contact surface of a gripper, and points outward from the face. When a gripper grips on a node, the normal vector coincides with the joint vector corresponding to the gripper.[1]

In addition to these three fundamental elements to specify a node, each node has its own index as an identifier. It also has information on its neighbor nodes. The neighbor nodes of a node are nodes that are reachable from the node by the single step of an individual Shady3D robot or by the cooperation of two Shady3D robots. Each node can have more than one neighbors. Connection to the neighbor nodes are represented by edges between the node and them.

Edge An edge shows connection between two nodes. If two nodes are neighbors, an edge is assigned between them. For each edge, abstract cost between two nodes is assigned. This cost indicates if an individual Shady3D robot can move from one node to another without other's help. If it can, the *single-cost* is assigned to the edge. If the robot needs the help of another Shady3D robot to move from one node to another, the *multi-cost* is given to the edge connecting those two nodes. Because the movement by the cooperation with two Shady3D robots is more difficult, the multi-cost is set larger than the single-cost. For our implementation, the single-cost is one and the multi-cost is five.

Node information is given by a user according to an environment. For each node, the position, the direction vector, the normal vector, the index, and the information on neighbor nodes need to be provided. With this information, the software adds edges between nodes neighboring each other and assigns appropriate cost to each edge. To evaluate the cost of an edge, it investigates if an individual Shady3D robot can traverse two neighboring nodes. The process of investigation is like the following. First, it computes two points, one for each node, by translating the position point of each node in the direction of the normal vector by the distance between the joint center and the gripping point of the Shady3D robot. These two points correspond

[1]For the definition of the gripping point, gripper vector, and joint vector, see Section 4.2.2.

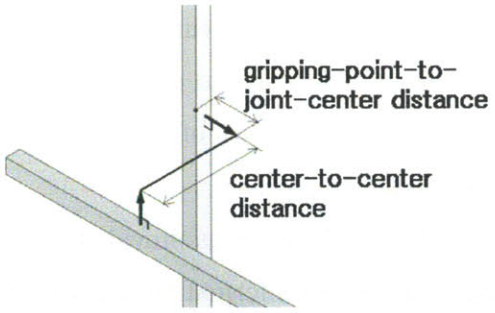


Figure 5-2: This figure visualizes the process of cost evaluation of an edge connecting two nodes. The cost of the edge connecting two nodes shown in this figure has the single-cost. Refer to the text for the detailed description of cost evaluation.

to the joint centers of the robot when it grips on these nodes. Next, the distance between two points is compared with the center-to-center distance of the Shady3D robot. If they are the same, it means that an individual Shady3D robot can grip both nodes at the same time, so it can move between two nodes without other's help. In this case, the software assigns the single-cost to this particular edge. If two distances are different, it means that the help of another robot is necessary to move between these two nodes, so the multi-cost is assigned to the edge. Figure 5-2 visualizes this process.

By generating all edges and assigning appropriate cost to them, the software creates the abstract graph composed of nodes and edges. As mentioned above, this graph is employed to compute the path of the Shady3D robot.

5.2.2 Physical trusses

Though the abstract graph of nodes and edges of the environment is useful for the path finding of the Shady3D robot, it does not represent the real configuration of trusses in the environment. Information on physical trusses in the environment is necessary to avoid collision between the robot and the trusses. Therefore, this information is included in the environment representation.

Each truss is represented as a line segment with the start point and end point. If two trusses intersect, each part of a truss divided by the intersection point is counted

as one truss in the representation. By this, it can be guaranteed that there is no obstacles between the start point and end point of each truss. In addition to setting the start point and end point, the width of trusses is also provided. In our Shady3D system, the width of trusses is 3/4in.

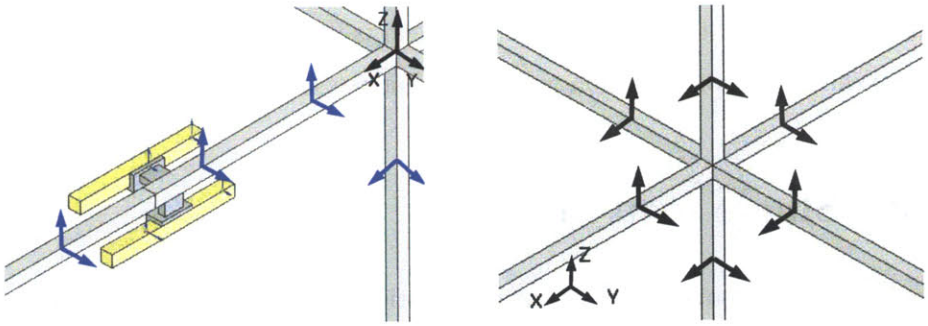
The information of physical trusses is provided by a user according to an environment. The software keeps this information as a list of trusses in the environment representation. It uses this information to predict and avoid collision between the robot and trusses during the motion of the robot.

Information on passive bars used for the connection of multiple Shady3D robots needs to be included in the environment representation. In our current software, the representation of passive bars has not established yet. It can be realized in a similar way to fixed trusses. A bar can be represented as a line segment with the start point and end point as a fixed truss is. Because a passive bar can be picked up and moved by Shady3D robots, however, additional information is necessary, such as its representative position, its orientation, and its state indicating whether it is grabbed by a robot or not. The representation of passive bars will be developed in the future.

5.2.3 Shady3D environment

The environment representation presented above can represent any types of truss environments. For our Shady3D experiments, however, we restricted an environment to a truss structure consisting of perpendicularly intersecting bars with square cross section. Each truss bar is parallel to one axis of the three-dimensional Cartesian coordinates. Each of its four faces is perpendicular to one of other two axes. For example, a truss bar parallel to the x-axis has two faces perpendicular to the y-axis and two faces perpendicular to the z-axis. Two faces perpendicular to an axis can be classified into the negative face and positive face. The positive face has a surface normal vector directing the same direction as the perpendicular axis. For the negative face, the surface normal vector points the opposite direction to the direction of the perpendicular axis.

In the Shady3D environment, nodes are assigned on the positive faces of each



(a) (b)

Figure 5-3: This figure shows the configuration of nodes in the Shady3D environment. (a) For each truss in the Shady3D environment, the positive faces are used for nodes and the negative faces are used for passive bar mounting stations. In this image, nodes, represented by their normal vectors depicted as the blue arrows, are on the positive faces of the truss parallel to the x-axis. The passive bars are placed on the negative faces of the truss. (b) Node configuration around the intersection of six trusses. The black arrows represent the normal vectors of the nodes. There are twelve nodes around the intersection of six trusses.

truss. The negative faces are spared for passive bar placement. They are attached to mounting stations installed on the negative faces of a truss. Figure 5-3(a) shows the configuration of nodes and passive bars for a truss. In this configuration, the movement of the Shady3D robot among nodes are not blocked by a passive bar because the robot moves on the positive faces and the passive bars are placed on the negative faces. When the robot needs to pick up a bar, it can move its gripper on the bar and grab it.

Nodes on a truss are separated by the space equal to the center-to-center distance of the Shady3D robot. On the corner where two trusses intersect perpendicularly, the distance between two nodes closest to the intersection point is set to the center-to-center distance of the Shady3D robot. By placing nodes in this way, the robot is able to move from one truss to another without other's help if they are in the same plane, or in other words, two nodes on them have the same normal vectors. The cost between such two nodes is set to the single-cost. In the case where two nodes closest to the intersection point of trusses are not in the same plane, that is, their normal vectors are different, the robot cannot move by itself and needs other's help. For these

nodes, the cost of the edge connecting them is set to the multi-cost. Figure 5-3(b) shows the nodes configuration around the intersection of six trusses.

5.3 Single step move

The single step move is a fundamental motion for truss climbing. A truss climbing robot can locomote from one position to the next position on trusses by this motion. It can move from one location to another in the environment by performing this motion consecutively.

The goal of the single step move of the Shady3D robot is to move reliably from the current node to the next, avoiding collision with trusses. The current node is defined as the node on which the anchor gripper is located. The next node is one of the neighbor nodes of the current node. Neighbor nodes that cannot be reached by an individual Shady3D robot—edges for those nodes have the multi-cost are excluded because, in that case, the help of other robot is required.[2] After the single step move is completed, the side on the next node becomes a new anchor of the robot.

The single step move is performed in several steps: body rotation angle evaluation, collision prevention, joint rotation angle evaluation, actual movement execution, and verification. These steps are described in the following. In the following description, the anchor gripper means the gripper on the anchor side, and the opposite gripper means the gripper on the opposite side to the anchor side.

5.3.1 Body rotation angle evaluation

First of all, the software checks if the opposite gripper is already located on the next node to move to. For this, it checks the in-grip variable of the state of the robot. If the opposite gripper already grips on the next node, the anchor is switched to the side of the next node and the move is completed.

[2]The single step move described in this section is for the individual Shady3D locomotion. An algorithm for a move from one node to its neighbor by means of the cooperation of two Shady3D robots has not been implemented in the Shady3D hardware yet. The concept of the two-Shady3D cooperation is presented in Section 5.5.

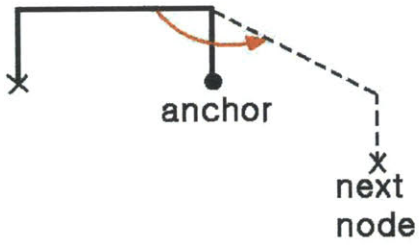


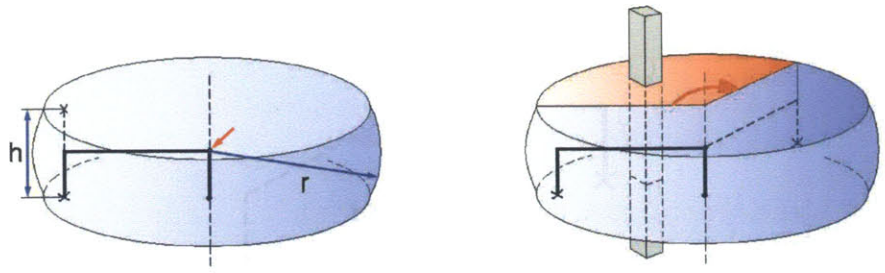
Figure 5-4: This figure shows the definition of the body rotation angle. The solid line represents the current state of the Shady3D robot. The dashed line represents the state after the robot moves to the next node. The body rotation angle is depicted as the red curved arrow.

If the opposite gripper is not located on the next node, the software computes an angle between the current body line and the body line that the robot will have after the move (next body line).[3] The current body line is represented as a vector from the anchor joint center to the opposite joint center. The next body line is represented as a vector from the anchor joint center to the point gained by translating the next node position point by the gripping-point-to-joint-center distance in the direction of the next node normal vector. The angle these two body lines make, called the body rotation angle, is used to compute the rotation angle of the anchor joint. Figure 5-4 illustrates the body rotation angle.

5.3.2 Collision prevention

To reach the next body line, the robot body can be rotated around the anchor joint in either counterclockwise or clockwise direction. Mostly, both directions are possible. However, in some cases, there could be an obstacle—a truss in the environment—in one direction. In such a case, a direction free of obstacles should be selected to prevent collision during actual movement. Because the next node must be one of the

[3]The body line of the robot is the line connecting two gripper joint centers. Refer to Section 4.2.1.



(a) (b)

Figure 5-5: This figure illustrates the collision space used for collision prevention. (a) The center of the collision space is indicated by the red arrow. Its radius, r, and height, h, are also shown. (b) In this figure, the truss is blocking the clockwise body rotation (the region shaded in red). Refer to the text for details.

neighbor nodes of the current node, at least one direction is guaranteed to be free of obstacles.

To predict potential collision, the collision space is defined. Figure 5-5(a) illustrates it. The collision space is a space swept by the robot when it rotates around the anchor gripper joint. The center of this space is the center point of the anchor gripper joint. The height of this region is two times of the distance from the gripper joint center to the contact surface of the gripper because the opposite gripper can direct the opposite direction to the anchor gripper with the 180 degrees of the middle joint angle. The distance between the gripper joint center and the contact surface of the Shady3D robot is 60mm. Therefore, the height of the collision space is 120mm. The radius of the space is the sum of the distance between the anchor joint center to the farmost point of the opposite side of the robot, the half of the truss width, and some clearance. The center-to-farmost-point distance is 245mm and the truss width is 19.05mm. The clearance is set to 10mm. Thus, the radius is 265.525mm.

The software goes through the list of physical trusses in the environment representation and checks if any truss is located within the collision space. If a certain truss is found in the collision space, it investigates which direction of movement the truss blocks. For example, in Figure 5-5(b), the truss is blocking the clockwise rotation of

the robot body. When one direction is proved to be blocked by a truss, the other direction is chosen as the direction of body rotation. If both directions are free of obstacles, the direction in which the body rotation angle is smaller is selected.

5.3.3 Joint rotation angle evaluation

After deciding the direction of body rotation, the software computes by what angle each joint needs to be rotated. First, the angle for the anchor joint is evaluated according to the direction of body rotation. The anchor joint needs to be rotated in the opposite direction to the direction of body rotation. For example, if the body rotation angle is 90 degrees, which means counterclockwise body rotation, the anchor joint rotation angle is set to -90 degrees, which implies clockwise rotation.

The anchor joint rotation angle is adjusted in order not to cause the final joint angle to exceed the motion limit of the joint. The software precalculates the final anchor joint angle and check if it would exceed the limit, which is ± 270 degrees. In the case where the violation of the motion limit is anticipated, the anchor gripper is opened and rotated by 180 degrees in the opposite direction to the required rotation in order to secure more operation range. For example, if the current anchor joint angle is 180 degrees and the required anchor joint rotation is 180 degrees, the final angle would be 360 degrees, which violates the limit of 270 degrees. Therefore, the anchor gripper is opened and rotated by -180 degrees. Consequently, the current joint angle becomes zero and the intended anchor rotation would not violate the motion limit.

After the anchor joint rotation angle is computed, the rotation angles for the middle joint and the opposite joint are evaluated. In contrast to the anchor joint, the direction of rotation does not matter for these joints because their motion does not change the body line. Therefore, instead of computing the angle *by which* the joints are rotated, angles *to which* they need to be rotated are evaluated. To calculate such angles, the expected state of the robot after it moves to the next node is evaluated based on the normal vector and direction vector of the next node. The expected middle joint angle is adjusted not to exceed the limit of ± 180 degrees. For the

opposite joint angle, two or three solutions are possible because the gripper vector can be parallel to the direction vector of the next node for every 180 degrees of the joint angle. To single out one solution, therefore, the range of the expected opposite joint angle is restricted within ±90 degrees. Keeping the gripper joint angle as close to zero as possible in this way also reduces possibility that the over-limit condition occurs during the next single step move.

5.3.4 Actual motion execution

The step of actual motion execution is to rotate each joint to its expected angle. An important issue in this step is the point where the middle joint is rotated. The middle joint cannot be rotated if both grippers touches trusses because it will cause collision between some robot parts, such as the gripper paddles and housings, and the trusses. The rotation of the middle joint needs to be done when the opposite gripper does not touch any trusses.

In our Shady3D environment where all trusses are configured rectilinearly, the anchor joint rotation angle is always larger than 90 degrees and there is no obstacles for the middle joint rotation at a point 45 degrees before any expected anchor joint angle. Therefore, the middle joint rotation can be performed at that point. The actual motion is executed in steps described below:

1. Open the opposite gripper.

2. Rotate the anchor joint by the anchor joint rotation angle reduced by 45 degrees in magnitude. For example, if the anchor joint rotation angle is -135 degrees, rotate -90 degrees.

3. Rotate the middle joint to its expected angle.

4. Rotate the anchor joint by the remaining anchor joint rotation angle, whose magnitude is 45 degrees.

5. Rotate the opposite joint to its expected angle.

5.3.5 Verification

After the rotation of all joints is completed, the opposite gripper is supposed to be located on the next node. However, there is possibility that it has not reached the next node for some reasons such as unexpected obstacles or low battery. To ensure that it is really located and able to grip on the next node, the software compares the state of the robot and the information of the next node. It checks if the position and the normal vector of the node are approximately equal to the opposite gripping point and the opposite joint vector, respectively. It also checks if the direction vector of the node is approximately parallel to the opposite gripper vector.[4] If all conditions to conclude that it is able to grip on the next node are satisfied, the opposite gripper is closed. Finally, the anchor is switched to the side on the next node (the opposite gripper side), and the single step move is completed.

5.4 Navigation in the environment

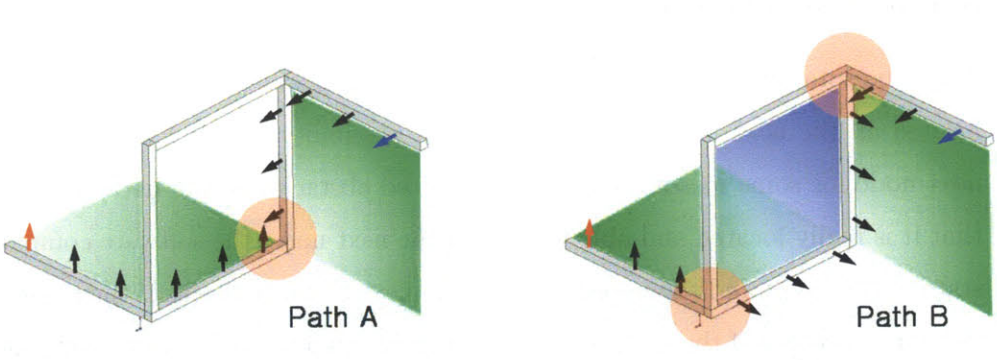
The Shady3D robot can move from one node in the environment to another by combining the single step move consecutively. To reach a goal location quickly and with the minimum consumption of energy, the robot needs to find the most efficient path from its current location to the goal location. In this section, a path planning algorithm to find the most efficient path and an algorithm to follow the path are described.

5.4.1 Path planning

The *path* of the Shady3D robot is a sequence of abstract nodes for the robot to follow successively. Given a goal node, there can be a large number of such sequences to reach the goal from the current location of the robot. The path planning algorithm determines a sequence with minimum cost among them.

The *cost* of the path does not mean only the physical length of the path. It also includes the difficulty of movement between nodes included in the path. Even though

[4] Tolerances for the check of approximate equality and parallelism are the same as those used in the evaluation of the robot state. See Section 4.2.3.



(a) (b)

Figure 5-6: This figure shows two paths from the start node (the red arrow) to the goal node (the blue arrow). The nodes are represented by their normal vectors. The physical lengths of the two paths are the same, but the cost is different. The shaded circles in red represent inter-plane transition that requires two-Shady3D cooperation. Planes covered by each path are highlighted in green and blue.

the physical lengths of two paths are the same, their cost can be different from each other. Figure 5-6 shows an example of such a case. In this figure, two different paths from the start node to the goal node are shown. The physical lengths of the Path A and Path B are the same. However, while the Path A includes one transition between different planes (indicated by the red-shaded circle in Figure 5-6(a)), the Path B requires two transitions between different planes (indicated by the red-shaded circles in Figure 5-6(b)). Transition between different planes requires the cooperation of two Shady3D robots. Such cooperation takes more time and computation and consumes more energy than the step movement of an individual robot. Therefore, the Path B can be considered more difficult, or more costly, though its physical length is the same as that of the Path A.

The path planning algorithm of the Shady3D software evaluates cost of paths in terms of both the length and difficulty of movement, and finds the most efficient path with the minimum cost. It employs Dijkstra's shortest path algorithm on the abstract graph of nodes and edges of the environment. Because edges connecting two nodes that can be traversed only by the cooperation of two robots have larger cost

(multi-cost) than those connecting two nodes reachable by an individual robot have (single-cost), Dijkstra's algorithm can compute the most efficient path that includes a minimum number of inter-plane transitions. Consider the example of Figure 5-6 again. Assuming that the single-cost is one and the multi-cost is five, the cost of the Path A is 15 (ten single-cost edges and one multi-cost edge) and that of the Path B is 19 (nine single-cost edges and two multi-cost edges). Thus, Path A is selected as the most efficient path connecting the start node and end node.

The nodes on the most efficient path determined by Dijkstra's algorithm are saved in a sequence of nodes representing the path. The robot follows each node in the sequence successively to reach the goal.

5.4.2 Path following

The Shady3D robot follows the path by repeating movement from a node to the next node in the path. For every iteration, the software retrieves the first element of the path sequence (the next node) and checks cost between the current node and the next node. If the cost is the single-cost, which means the robot can move to the next node by itself, the robot performs the single step move to that node. The node the robot has reached is removed from the path. After the robot moves to the next node successfully, the software repeats the same process for the next node, which is the first element of the path sequence. The iteration continues until the sequence of the path becomes empty, which implies that the robot has reached the goal node.

In the case where the cost between the current node and the next node in the path is the multi-cost, the Shady3D robot needs the help of another robot to move to the next node. An autonomous algorithm to traverse such nodes by means of the cooperation of two Shady3D robots has not been implemented in the Shady3D hardware yet. Therefore, the current Shady3D robot cannot actually follow a path including a pair of nodes with the multi-cost. However, the concept of the two-Shady3D cooperation has been studied and is described in Section 5.5. When this concept is implemented in the near future, the Shady3D robot will be able to follow any path, regardless cost between nodes in them.

5.5 Cooperation of two Shady3D robots

The Shady3D robot is designed to have three motive degrees of freedom.[5] In order to achieve the six degrees of freedom required for arbitrary three-dimensional motion, therefore, two robots, or modules, need to be connected. As described in Section 3.2.5, two Shady3D robots connected by means of a passive bar realize six degrees of freedom.

In our Shady3D environment, this two-Shady3D structure is needed when a robot is to move from a node in one plane to a node in a different plane around the intersection of trusses.[6] The relation of such nodes is represented by the multi-cost between them. An individual Shady3D robot cannot perform such movement because of the lack of degrees of freedom. It can achieve the goal only with the help of other robot. The other robot, called a *helper*, moves to one of nodes around the intersection and picks up a bar stationed nearby. Then, it takes an appropriate pose in order that the *helped* robot can hold the other end of the bar. The helped robot grips the other end of the bar and releases the fixed truss. In this way, two robots form a six-degree-of-freedom structure, which is able to reach the goal node. The helper robot and helped robot cooperate to locate the free gripper of the helped robot on the goal node. After the gripper reaches position and orientation to grip on the node, the helped robot closes the gripper and releases the bar. Thus, the helped robot traverses nodes with the multi-cost. Figure 5-7 shows some snapshots of the process described above.

5.5.1 Feasibility of connection

To realize the cooperation of two Shady3D robots, two robots must be connectable to each other by means of a passive bar. Therefore, the feasibility of connection around the truss intersection has been checked with the Shady3D CAD model. In this test, the distance between two grippers on the passive bar is set to be the same as the

[5]It has two more degrees of freedom for the opening/closing of two grippers. However, they are not included in motive degrees of freedom because they does not affect the pose (position and orientation) of the robot.

[6]Nodes around the intersection of trusses are illustrated in Figure 5-3(b).

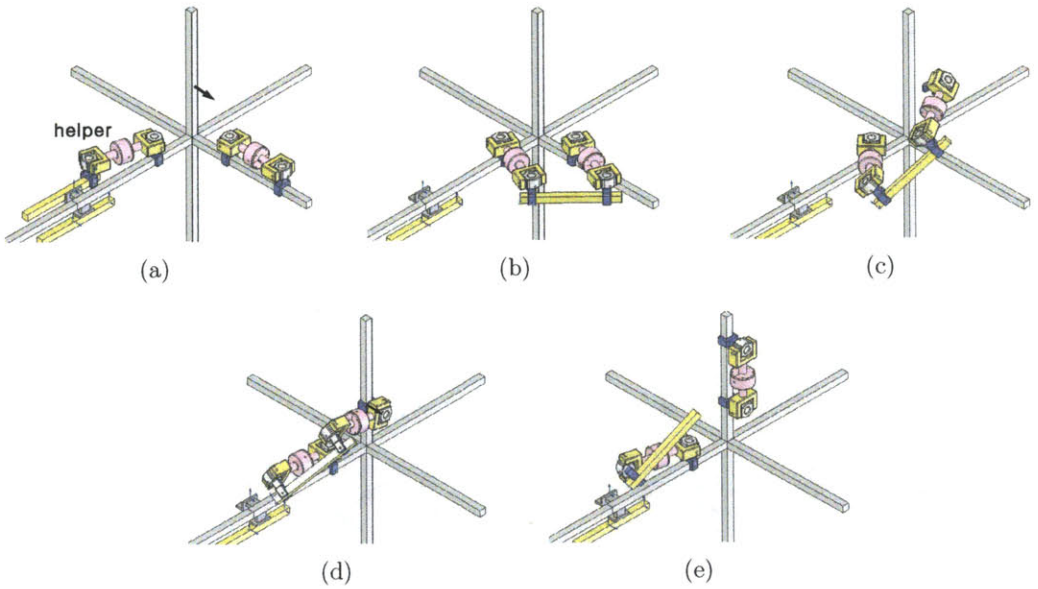
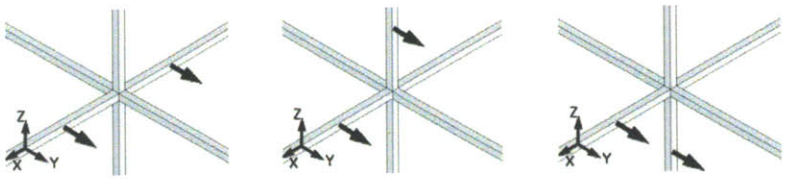
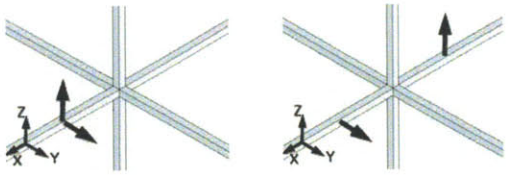


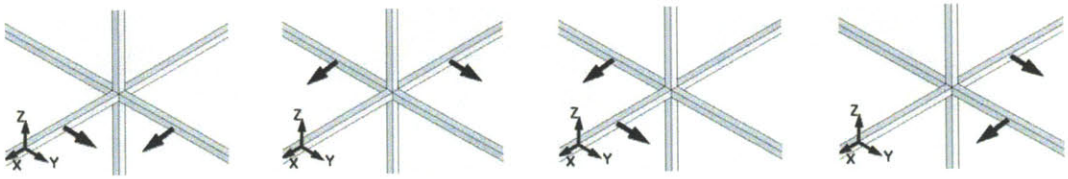
Figure 5-7: This figure shows some snapshots of the cooperation of two Shady3D robots around the intersection of trusses. (a) The robot on the right cannot move to the goal node (represented by the black arrow) by itself. The helper robot picks up a bar for cooperation. (b) Two robots connect to each other by means of the bar. (c) The helped robot release the fixed truss. Two robots cooperate to reach the goal node. (d) The gripper of the helped robot reaches the goal node. (e) Two robots break connection after the helped robot moved to the goal node successfully.



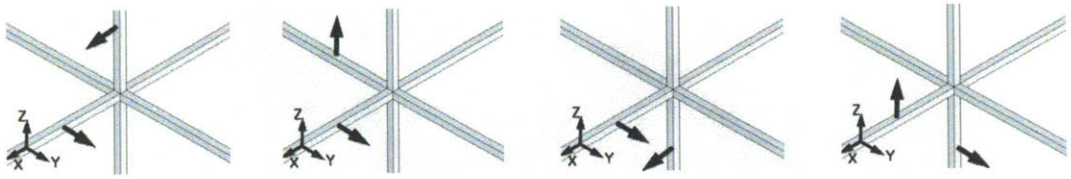
(a) Case 1



(b) Case 2



(c) Case 3



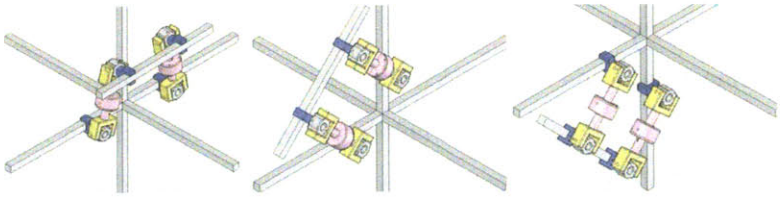
(d) Case 4

Figure 5-8: This figure shows four cases of possible node pairs around the intersection of six trusses. Refer to the text for details.

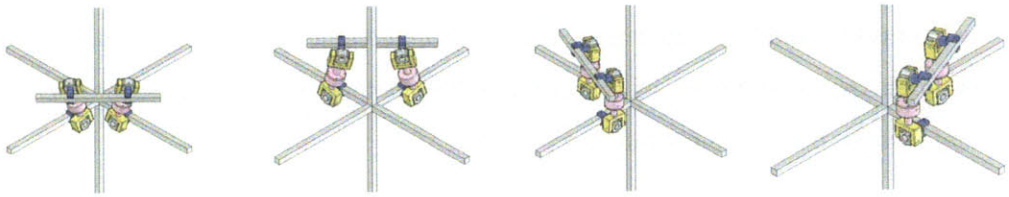
center-to-center distance of the Shady3D robot. Since two robots can be located any two nodes out of 12 nodes around the intersection of trusses, total 66 ($_{12}C_2$) pairs of nodes are possible. These pairs can be classified into four cases. Figure 5-8 shows them. The four cases are described below:

- **Case 1 (Figure 5-8(a), 18 pairs):** The normal vectors of two nodes are the same. Two robots are in the same plane.

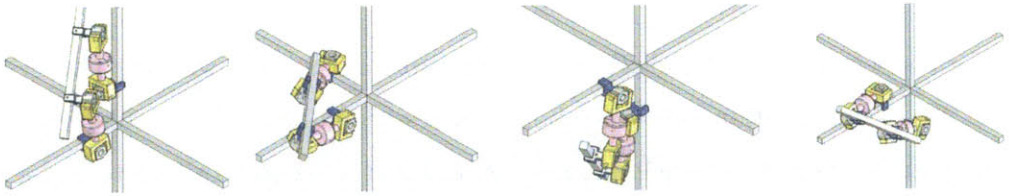
- **Case 2 (Figure 5-8(b), 12 pairs):** The normal vectors of two nodes are different, but the direction vectors of them are the same.



(a) Case 1



(b) Case 3



(c) Case 4

Figure 5-9: This figure shows some snapshots from the test for feasibility of connection for various cases. In all cases except for Case 2, pairs of nodes are connectable. Each snapshot in each case is a test for a corresponding node pair in Figure 5-8.

- **Case 3 (Figure 5-8(c), 12 pairs):** The normal vectors of two nodes are different, and the direction vectors are also different. However, the normal vectors are in the same plane. Therefore, the normal vector of each node is parallel to the direction vector of another node.

- **Case 4 (Figure 5-8(d), 24 pairs):** The normal vectors of two nodes are different, and the direction vectors are also different. In contrast to Case 3, however, two normal vectors are not in the same plane. Thus, this case represents the skewed configuration of the normal vectors of nodes with different direction vectors.

Testing with the CAD model showed that pairs of nodes are connectable in all cases except for Case 2. Figure 5-9 shows some snapshots from the test. In Case 2,

the kinematic constraints prevent connection.

Feasibility of connection needs to be checked twice for each two-Shady3D cooperated movement. First, the *start* node of the helped robot and the node on which the helper robot is located (helper node) must be connectable in order that the helped robot and helper robot can connect for cooperation. Second, the *goal* node of the helped robot and the helper node must be connectable in order to reach the goal node through the cooperation of two robots. The helper robot needs to be located on a node where both connections are possible.

5.5.2 Communication between robots

For cooperation, two Shady3D robots need to communicate with each other. First of all, communication between robots is necessary in order that a robot that needs help finds a helper. The helped robot broadcasts the request for help to find a helper robot. A robot that is able to help the helped robot responds to the broadcast message and moves to an appropriate helper node. Communication is also required during the process of cooperation. The helped robot needs to inform the helper robot that it has grabbed the passive bar held by the helper and released a truss fixed in the environment. They also have to communicate with each other to move to the goal node and check if the helped robot has reached it.

A detailed communication algorithm for two-Shady3D cooperation will be developed in the future. As for the method of communication, the Bluetooth wireless communication embedded in the Gumstix miniature computer will be employed.

5.6 Self-assembly of a truss structure

The Shady3D truss climbing robot can be extended to a *self-assembling* truss robot system. A large number of Shady3D robots connect to one another using passive bars to form a large truss structure. The structure built in this way is an active truss that is able to move on a fixed truss structure or to change its shape by reorganizing its components. This system of Shady3D robots and passive bars can be a special type

of three-dimensional heterogeneous self-reconfigurable robots with active and passive modules (bipartite).

Algorithms for building and moving a truss tower through the cooperation of multiple Shady3D modules and passive bars have been developed. In this algorithm, pairs of Shady3D robots form six-degree-of-freedom structures with passive bars. Then, these 6-DOF structures connect to one another and arrange according to the goal shape to assemble a larger tower structure. The completed active truss tower can be moved and rotated by synchronously rotating the joints of the Shady3D robots in the structure. These algorithms were implemented in computer simulations. The detailed description of the simulations is presented in Section 6.5.

5.7 Summary

The planning algorithms of the Shady3D robot have been described in this chapter. For algorithm development, we have assumed the *a priori* knowledge of the robot on the environment and its current location, and the discrete gripping points and designated bar locations in the environment. The environment representation includes the abstract graph of nodes and edges, which is used for path planning, and the representation of physical trusses, which is used for collision prevention. The algorithm for the single step move has been implemented to move the robot from the current node to the next node avoiding collision with trusses. The algorithm for navigation finds the most efficient path from the current node to the goal node and moves the robot according to the path. Multiple Shady3D robots can cooperate using passive bars to move between nodes with the multi-cost and to self-assemble an active truss structure.

Chapter 6

Experiments

This chapter describes experiments performed on the Shady3D truss climbing robot. For experiments, a testing truss environment (Shady3D environment) was built. In this environment, algorithms for (1) the single step move and (2) the navigation of the Shady3D robot were tested with the Shady3D hardware. Algorithms for multi-Shady3D cooperation for an active truss structure were implemented in computer simulations. The following sections present the experimental results and discussion.

6.1 Environment

Experiments on the Shady3D hardware were performed in a custom-designed truss environment, called the Shady3D environment.[1] Figure 6-1 shows the Shady3D environment. It was built with 3/4-inch-wide square-cross-section aluminum tubes. Each truss is perpendicular to adjacent ones. The main parts of the environment is the square horizontal frame and the vertical frame surrounded by it. The vertical frame is located so that an individual Shady3D robot can move from the horizontal frame to the vertical frame without other's help by performing the single step move. The horizontal frame is supported by four leg trusses.

The nodes, which are spaced by the center-to-center distance (180mm) of the robot, are marked on the positive faces of the trusses. Each truss in the horizontal

[1]The conceptual features of the Shady3D environment are presented in Section 5.2.3.

Figure 6-1: This figure shows the Shady3D testing environment. The origin is the left-rear corner of the horizontal frame in the image (marked by the XYZ frame). Indices and gripper boundaries are marked for the nodes. The vertical frame in the middle is located so that an individual Shady3D robot can move from the surrounding horizontal frame to the vertical frame by itself.

frame has eight nodes—four for each positive face—so the number of nodes in the horizontal frame is 32. The vertical frame contains 14 nodes, and four leg trusses have eight nodes in total. Thus, the total number of nodes in the environment is 54. For each node, its index is written on its face, and expected gripper boundaries are marked to check a gripper is located on the node correctly.

As mentioned in Section 5.2.3, passive bars for multi-Shady3D cooperation are supposed to mounted on the negative faces of the trusses. These passive bars, however, have not been installed because a cooperation algorithm was not implemented for the Shady3D hardware. They will be added to our environment in the future when the algorithm is implemented and ready for a test.

6.2 Single step move

The single step move of the Shady3D robot was tested for various situations as follows:

- Straight movement in a horizontal plane.
- Transition between trusses in a horizontal plane.
- Straight movement in a vertical plane, in the horizontal direction.
- Straight movement in a vertical plane, in the vertical direction.
- Transition between trusses in a vertical plane.
- Transition between a horizontal plane and vertical plane.

Straight movements in the horizontal plane and the vertical plane are the same kinematically. However, they were separately tested because gravity affects the performance of the robot in different ways according to its pose. In addition to separating tests in the horizontal plane and vertical plane, we tested straight movement in the vertical plane in two different directions (horizontal and vertical). Likewise, though transition between trusses in two planes are the same kinematically, separate tests were performed for both cases.

We originally intended to use the Gumstix miniature computer to run our Java high-level control software. However, it has been proved that available Java Virtual Machines (JVM) for the Gumstix are not capable of executing our code properly. We tried two different JVMs, but both of them had problems with either floating point computation or file input stream handling for serial communication with the Robostix. Therefore, we were not able to use the Gumstix for our experiments. As an alternative, we connected the Robostix directly to the workstation via a serial cable and executed the Java control software in the workstation. Though this resulted in the robot tethered to an on-ground controller, we could perform experiments without problems. We expect to remove a tether from the robot when a more completed JVM for the Gumstix is available.

6.2.1 Initialization

Before presenting the results from experiments, the procedure of initialization of the robot for tests in the Shady3D environment is described in this section. The Shady3D robot needs to be properly initialized before it executes commands. The initialization procedure is performed by setting two properties of the state of the robot: (1) a node on which the anchor gripper grips on and (2) the joint angles.

First, a node for the anchor gripper is set by a user. Because the anchor gripper of the robot must be closed by definition, at least one gripper must be closed on a node in the environment. When a user manually locates the robot on a certain node and starts the software, the software first asks which side the anchor is (left or right). Then the user is requested to specify the node on which the anchor gripper is located and to decide whether or not the gripper vector points the same direction as the direction vector of the node. Given this information, the software computes the gripper vector, gripping point, and joint vector for the anchor.

Second, the joint angles of the robot need to be set. Ideally, this task can be done automatically using the absolute position referencing functionality. However, it has been proved that this function needs improvement to be employed for autonomous operation on trusses for reasons described in Section 4.3.4. Therefore, we currently set joint angles manually at the initialization stage. A user is requested to input each joint angle and the software set a corresponding counter according to the input.

In addition to specifying each angles, we implemented a semi-autonomous joint angle setting option using the *default configuration*. The default configuration is defined as a configuration that all joints are in their absolute zero position. It can be assumed when both grippers of the robot are closed on the same straight truss with their potentiometers pointing outside. If the software is informed that the robot is in such a pose, it sets each joint angle to zero. We can use this more convenient method by leaving the robot in the default configuration when it is turned off.

Intended Steps	Successful Steps	Time Consumed	Failure Modality
5	5	6 min 26 sec	
10	10	13 min 8 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
15	15	100.0%	78.3 sec

Table 6.1: Experimental results for straight movement in a horizontal plane.

6.2.2 Results

As previously described, the single step move algorithm was tested for six different situations. The purpose of these tests was to verify that the robot is capable of moving on trusses reliably in various situations. Each experiment was performed by manually setting the path of the robot. Then, the robot was commanded to follow the assigned path.

Straight movement in a horizontal plane

This experiment demonstrated the ability of the Shady3D robot to move along a straight truss in a horizontal plane. The experiment was performed using one of trusses in the horizontal frame of the Shady3D environment. The robot was placed on the upper face of the truss and commanded to move back and forth along it. It repeated the sequence of two forward steps followed by two backward steps. Total 15 steps were attempted, and all steps were performed successfully. Ten consecutive steps were performed reliably. The data for the test trials are summarized in Table 6.1.

During this test, the body of the robot was slightly tilted downward because of gravity. This tilt error was successfully compensated by the slope in the gripper housing, so no failure by collision occurred.[2]

Transition between trusses in a horizontal plane

This experiment was carried out to verify the capability of the robot to make transition from one truss to another in a horizontal plane. For this test, a corner in the

[2]See Section 3.3.2 for more details about the tilt compensation of grippers.

Intended Steps	Successful Steps	Time Consumed	Failure Modality
12	12	15 min 36 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
12	12	100.0%	78.0 sec

Table 6.2: Experimental results for transition between trusses in a horizontal plane.

Intended Steps	Successful Steps	Time Consumed	Failure Modality
16	7	11 min 3 sec	joint motor failure
16	16	25 min 15 sec	

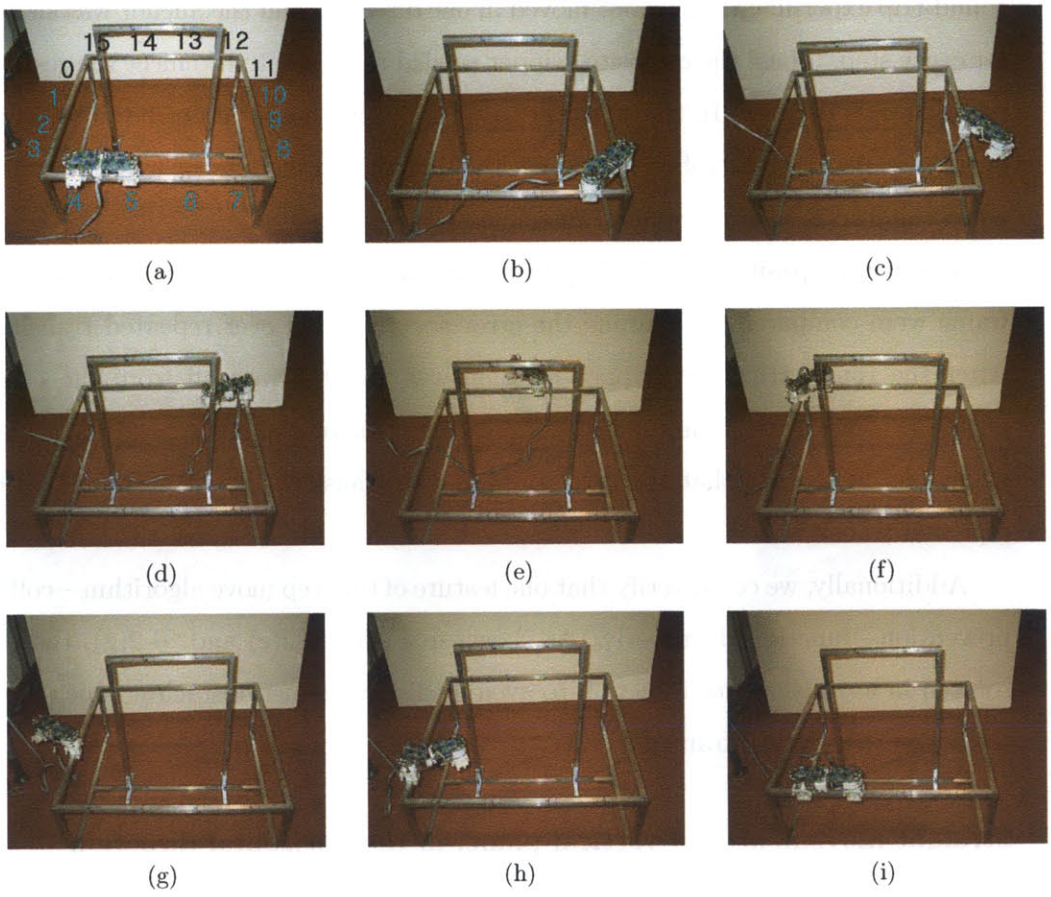
Total Attempted	Total Successful	Success Rate	Avg. Time/Step
24	23	95.8%	94.7 sec

Table 6.3: Experimental results for a round trip in a horizontal plane.

horizontal frame of the Shady3D environment was used. As in case of the straight movement experiment presented above, the robot was placed on the upper face of trusses. It was initially located on the closest node to the corner and commanded to move to the adjacent truss at the corner. Six consecutive transitions—composed of 12 steps—were attempted and all of them were successful. Table 6.2 summarizes the results from the test.

During the test, errors in the gripper position were observed after transition between trusses at a corner. When the robot moved to an orthogonal truss, the opposite gripper was closed about 4 mm farther than the expected position. This error was generated by the error in the gripper joints. The error in the anchor joint affected more than that in the opposite joint because the length of the robot amplifies a small error in the angle to a large displacement in the opposite gripper position. However, this amount of errors did not affect the reliability of a grip for most cases.

After testing straight movement and transition between trusses separately, the overall performance of horizontal-plane movement was tested by moving the robot around the horizontal frame of the Shady3D environment. Figure 6-2 shows some



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 6-2: Nine snapshots from the second trial of the experiment for a round trip in a horizontal plane. The robot started from Node 4 and traveled in a loop counterclockwise. Nodes covered by the robot are indicated in (a). As shown in (c) and (g), the robot rotated in a direction where it could avoid collision to an obstacle (the truss in the vertical frame).

snapshots of this round-trip experiment. As the robot traveled around the frame, twelve straight steps and four inter-truss transitions were successfully performed. The results of the round-trip experiments are presented in Table 6.3. Note that the average time per step is longer than that of separate tests. The reason is that in the round-trip experiment, the robot moved in one direction and the anchor was switched for every step. Thus, the opposite gripper needed to be rotated to meet the restricted range of ± 90 degrees. In the separate tests, however, this rotation did not occur for every step because the robot moved back and forth and the anchor was not switched when the direction of movement was changed.

The gripper positions on Node 4 before and after the travel around the horizontal frame were compared to examine the error accumulation over repeated transitions. After the round trip, the gripper position was about 5 mm off from the original position. This error is almost the same as the error of a single transition, which is negligible. It seemed that the error produced in transition is not accumulated over repeated transitions.

Additionally, we could verify that one feature of the step move algorithm—collision prevention—functioned properly. As shown in Figure 6-2(c) and 6-2(g), the robot rotated in a collision-free direction to avoid collision against obstacles, which are the trusses in the vertical frame.[3]

Straight movement in a vertical plane, in the horizontal direction

This experiment demonstrated the capability of the Shady3D robot to move in a vertical plane along a horizontal truss. The robot was placed and moved back and forth on the side face of one of the trusses in the horizontal frame. As in the experiment for straight movement in a horizontal plane, it repeated two forward steps followed by two backward steps. The step movement in a vertical plane in the horizontal direction was performed successfully for most cases. The results are presented in Table 6.4.

Though almost all attempted steps were successful, however, the movement in a vertical plane along a horizontal truss proved to be more difficult and more prone

[3]See Section 5.3.2 for more details about the collision prevention of the single step move.

Intended Steps	Successful Steps	Time Consumed	Failure Modality
4	4	5 min 43 sec	
12	2	—	excessive current in gripper motor
12	1	—	large gripper misalignment
12	12	17 min 40 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
21	19	90.5%	87.7 sec

Table 6.4: Experimental results for straight movement in a vertical plane, in the horizontal direction.

to failure than that in a horizontal plane. Compared with the results in Table 6.1, the average time per step is about ten seconds longer than that for horizontal-plane movement despite the same sequence of motion primitives. The difference in time for joint rotation was negligible. However, gripper closing motion took longer time than the horizontal-plane case. Because of the error in the anchor joint, the body of the robot was tilted downward when it was horizontally stretched to reach the next node. As a result, the opposite gripper was located about 5 mm lower than its expected position. The gripper had to lift up the weight of the robot to make a firm grip on the truss. This process imposed large load on the gripper motor, so the closing speed of the gripper paddles was slowed down.

The error of the joint and consequent tilt of the robot body was the main cause of failure. Because a large load imposed on the gripper motor caused large current, the motor current was more likely to reach the limit. In addition, gripper misalignment larger than three degrees occurred more often, which sometimes caused failure in reaching an appropriate position in the next step.

Straight movement in a vertical plane, in the vertical direction

Straight movement in a vertical plane was also tested in the vertical direction. For this test, one of the vertical trusses in the vertical frame in the Shady3D environment was employed. Because the vertical trusses have only three nodes for each face, the

Intended Steps	Successful Steps	Time Consumed	Failure Modality
10	10	12 min 38 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
10	10	100.0%	75.8 sec

Table 6.5: Experimental results for straight movement in a vertical plane, in the vertical direction.

Intended Steps	Successful Steps	Time Consumed	Failure Modality
4	4	—	
12	12	15 min 35 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
16	16	100.0%	77.9 sec

Table 6.6: Experimental results for transition between trusses in a vertical plane.

robot was commanded to take an upward step and a downward step in turn, instead of moving two steps in one direction before returning. Total 10 steps were attempted, and all steps were performed successfully. The data for the test are summarized in Table 6.5.

A key issue in movement in a vertical plane in the vertical direction is the slippage of the robot along a truss. The robot is more likely to slip in the middle of step move process when only one gripper grips the truss. In addition, an incomplete grip of a gripper can result in slippage. After ten consecutive steps in the vertical direction, about 1-mm slippage was observed, which is negligible.

Transition between trusses in a vertical plane

This experiment was done to verify that the Shady3D robot is able to move from a vertical truss to horizontal truss in a vertical plane, and vice versa. For this experiment, a corner in the vertical frame in the Shady3D environment was used as a testing place. Total four vertical-to-horizontal transitions and four horizontal-to-vertical transitions were attempted, and all of them were done successfully. The results of the experiment were presented in Table 6.6. As in case of transition in a

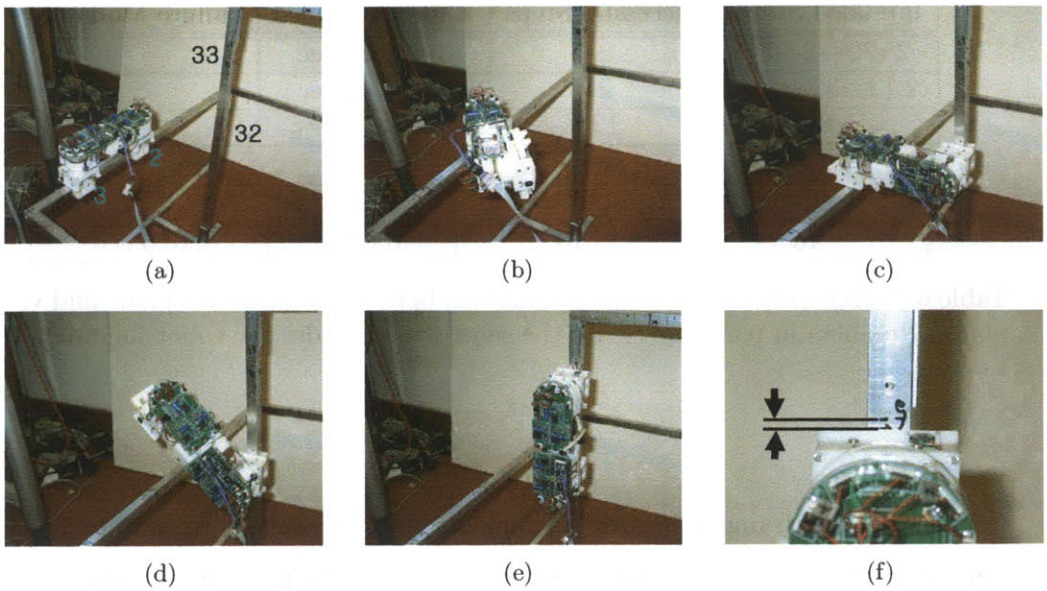


Figure 6-3: Six snapshots of transition from the horizontal frame to the vertical frame. Nodes covered by the robot are indicated in (a). (f) shows the error in gripper position generated by the downward tilt during the transition.

horizontal plane, an error in gripper position, which was about 2mm, was generated after several transitions. However, this error did not affect the overall process.

Transition between a horizontal plane and vertical plane

The experiment for transition between a horizontal plane and vertical plane was carried out to test the capability of the robot to move in a three-dimensional space using its middle joint. In contrast, experiments described above were performed in a certain two-dimensional plane—horizontal or vertical—without using the middle joint. This experiment was done by moving the robot between the horizontal frame and vertical frame of the Shady3D environment. Figure 6-3 shows some snapshots of transition from the horizontal frame to vertical frame. As shown in the figure, the robot moved from Node 2 on the horizontal frame to Node 32 on the vertical frame (indicated by their indices) to perform the transition.[4] Total eight transitions—four

[4]Node 2 on the horizontal frame and Node 32 on the vertical frame are located so that an individual Shady3D robot can traverse two nodes by itself. Therefore, the cost between the two

Intended Steps	Successful Steps	Time Consumed	Failure Modality
2 (32→2→3)	2	3 min 50 sec	
2 (2→32→33)	2	4 min 40 sec	
12	12	20 min 55 sec	

Total Attempted	Total Successful	Success Rate	Avg. Time/Step
16	16	100.0%	110.3 sec

Table 6.7: Experimental results for transition between a horizontal plane and vertical plane. Numbers in parentheses are the sequences of nodes the robot followed.

from the vertical frame to horizontal frame and four from the horizontal to vertical—were attempted, and all attempts succeeded. Table 6.7 summarizes the results.

The middle joint was operated reliably. The algorithm to rotate the middle joint at a point 45 degrees before the expected anchor joint angle worked well and caused no collision. However, when the robot moved from the horizontal frame to the vertical frame, downward tilt by gravity resulted in the error of the gripper position on the vertical truss (See Figure 6-3(f)). The amount of the error was 5mm. In contrast to the case of movement in a horizontal plane, where such downward tilt was compensated by the slope in the gripper housing, there was nothing to push up the opposite gripper in transition to the vertical frame (transition from Node 2 to Node 32). Because our Shady3D robot does not have a degree of freedom to lift its body to compensate such an error, the error remained and affected the position on the vertical truss. Fortunately, this amount of error did not seriously affect the next movement in the vertical plane, such as transition to the horizontal truss in the vertical frame. However, the robot placed slightly lower than the expected position increased the possibility of failure.

nodes is the single-cost.

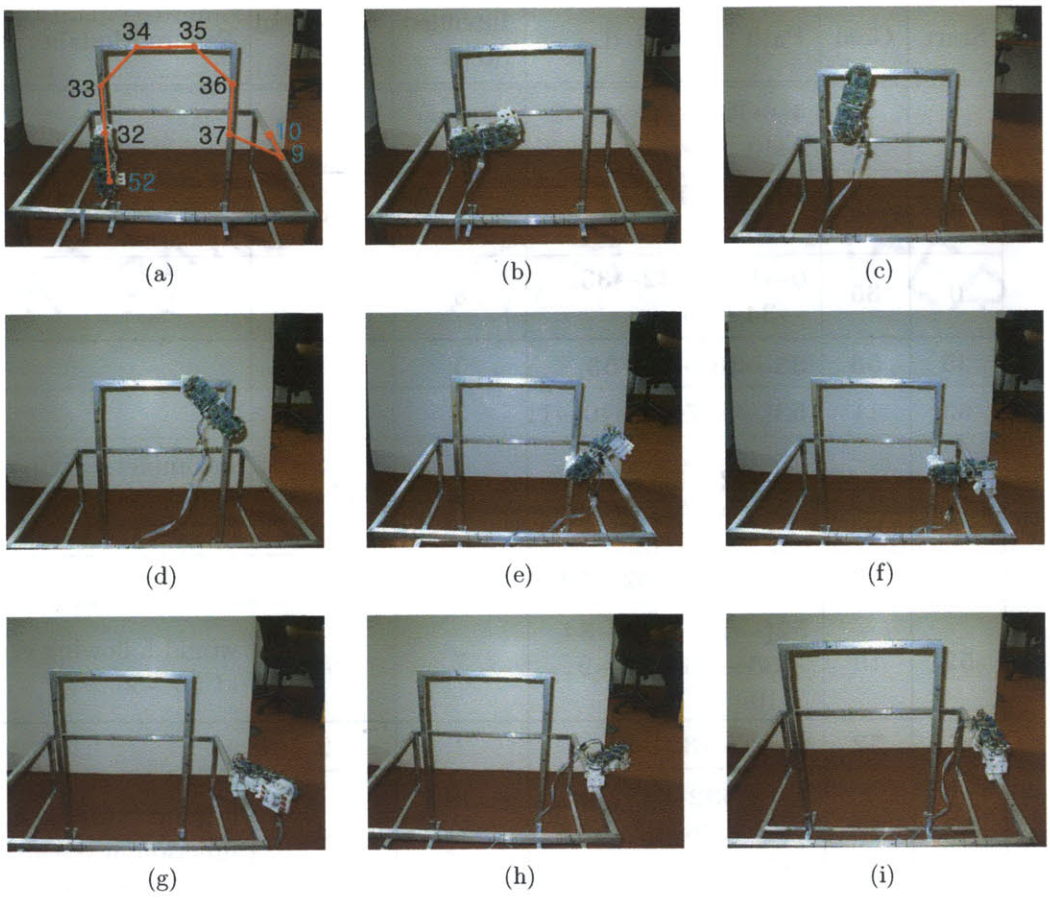


Figure 6-4: Nine snapshots of navigation. The robot computed and followed the most efficient path (highlighted in (a)).

6.3 Navigation

6.3.1 Results

After testing the single step move function in various situations, we tested the navigation capability of the Shady3D robot. The path planning algorithm was used to find the most efficient path to a given goal node. Then, the obtained path was followed by performing the single step move sequentially.

Various combinations of the start node and goal node were attempted. For all attempts, the Shady3D software successfully computed the most efficient paths. The

Start	Goal	Path Computed	Intended Steps	Successful Steps	Failure Modality
4	10	4(→)5→6→7→8→9 →10	5	5	
10	0	10→11→12→13→14 →15→0	6	6	
0	**35**	0→1→2→**32→33 →34→35**	6	6	
35	**53**	**35→36→37→53**	3	3	
53	11	**53(→)37**→9→10→11	3	3	
11	**52**	11→12→13→14→15 →0→1→2→**32→52**	9	0	incomplete grip in transition between trusses at 11→12
0	**52**	0(→)1→2→**32→52**	3	3	
52	10	**52(→)32→33→34 →35→36→37**→9 →10	7	3	would following error at 35→36
3	**52**	3(→)2→**32→52**	2	2	
52	10	**52(→)32→33→34 →35→36→37**→9 →10	7	3	gripper switch malfunction at 35→36
52	10	**52(→)32→33→34 →35→36→37**→9 →10	7	7	
9	**52**	9→**37→36→35→34 →33→32→52**	7	2	excessive current in gripper motor at 36→35
9	**52**	9→**37→36→35→34 →33→32→52**	7	2	incomplete grip in transition between trusses at 36→35
9	**52**	9→**37→36→35→34 →33→32→52**	7	7	
52	6	**52(→)32**→2→3→4 →5→6	5	5	

Table 6.8: Experimental results for navigation in the Shady3D environment. Numbers in Start, Goal, and Path Computed columns are node indices. The nodes in the vertical frame are written in boldface. The arrows in parentheses in the path indicates that the opposite gripper was already located at the next node so an actual step move was not necessary.

Gripper closing	Gripper opening	Joint rotation
24.0 sec	20.8 sec	14.4 sec / 90 deg

Table 6.9: Average time consumed for the motion primitives. The time consumed for the joint rotation motion is normalized as time used for 90-degree rotation.

computed paths varied according to the start node and goal node. Paths were only in a certain plane in some cases, while they included transition between the horizontal and vertical planes in other cases. These computed paths were actually followed by the robot. Figure 6-4 shows one of trials in which the robot moved from the vertical frame in the middle of the Shady3D environment to the surrounding horizontal frame.

Table 6.8 summarizes the results of trials. As seen in the table, not all paths were successfully followed because of the occasional failure in hardware. Most failures occurred during transitions between trusses in a vertical plane (between Node 35 and Node 36). Except for the case where the mechanical malfunction of one gripper switch caused a failure, the original cause for these failures was the error in the anchor joint and consequent downward tilt of the robot body. Particularly, the downward tilt occurred during transitions from the horizontal frame to vertical frame, which was not compensated, was the main cause of the problems. Though such errors were overcome in most cases, this problem needs to be handled for more robust locomotion.

For each step of the path following procedures, time consumed for the motion primitives was measured. Table 6.9 shows the average time used for the gripper closing motion, gripper opening motion, and joint rotation. As shown in the table, the gripper closing motion took longer time than the gripper opening motion. The reason was that in the closing motion, the procedure of the misalignment correction caused additional rotation of the gripper joints. The joint rotation motion took 14.4 seconds per 90-degree rotation on the average. From these results, we can see that the gripper closing and opening motions take a large portion of time in a step of the robot. For example, if the robot moves straight on a single truss and the total angle of rotation during the step is 360 degrees (180 degrees for the anchor joint and 180 degrees for the opposite joint), the time for joint rotation is 57.6 seconds and

the portion of time for the gripper opening/closing motions becomes 43.8%, which is almost half. This implies that we can reduce the time for a step considerably by reducing time taken by the gripper motions.

6.4 Discussion on hardware experiments

The results of experiments both for the single step move and the navigation of the Shady3D robot have showed that algorithms implemented for these functions worked well. For the single step move, each step of the move—body rotation angle evaluation, collision prevention, joint rotation angle evaluation, actual movement execution, and verification—was performed reliably. Particularly, the collision prevention algorithm, which decides the direction of the body rotation avoiding collision, functioned properly, and no collision between the robot and trusses in the environment occurred during the whole experiments. Furthermore, the path planning algorithm, which finds the most efficient path from the current node of the robot to a given goal node, accomplished its objective successfully.

However, the experiments revealed that there is room for improvement for the hardware of the Shady3D robot. To investigate failure-causing factors in the hardware, all steps made in the whole experiments, both for the single step move and for navigation, have been classified according to their types, as shown in Table 6.10. The robot could successfully and reliably perform straight movement in a horizontal plane, straight movement in a vertical plane in the vertical direction, and transition between a horizontal plane and vertical plane.

Failures occurred in transitions between trusses and straight movements in a vertical plane in the horizontal direction. Some of these failures occurred because one of the mechanical parts occasionally did not function properly. For example, a failure in transitions in a horizontal plane, whose failure modality is described as joint motor failure, took place because the set screw fixing the worm on the joint motor shaft was loosened. The cause of a failure in transitions in a vertical plane was that one of the gripper switches was stuck in a pushed state, which is described as gripper switch

Type	Attempted	Successful	Success Rate	Failure Modality
straight, horizontal plane	60	60	100.0%	
transition, horizontal plane	17	15	88.2%	joint motor failure imcomplete grip
straight, vertical plane, horizontal direction	30	28	93.3%	excessive current in gripper motor larger gripper misalignment
straight, vertical plane, vertical direction	32	32	100.0%	
transition, vertical plane	21	17	81.0%	would following error gripper switch malfunction excess current in gripper motor incomplete grip
transition between horizontal and vertical plane	16	16	100.0%	
Total	176	168	95.5%	

Table 6.10: The summary of the results of hardware experiments. All steps made in experiments are classified according to their type.

malfunction. As a result, the gripper joint kept rotating to correct false misalignment based on the wrong switch state. Because these types of hardware failures rarely happen, we did not consider this seriously.

If failures by occasional hardware problems are ruled out, the almost all failures can be attributed to the error in the gripper joints. When the robot was stretched out in a horizontal plane, the body of the robot was tilted downward because of gravity. This downward tilt caused an error in the robot position on a vertical truss when it moved from the horizontal frame to vertical frame, as shown in Figure 6-3(f). This error affected the following steps, such as transition to the horizontal truss in the vertical frame. Because the robot is slightly off from the expected node position, the opposite gripper was not placed on the exact position on the horizontal truss. Though this amount of error was overcome in many cases, it caused the majority of failures in transitions in a vertical plane. The error in the gripper joints also produced problems during straight movement in a vertical plane in horizontal direction because the opposite gripper was placed lower than its goal position.

The bearing joint mechanism used in the current gripper joints were not good enough to prevent tilt errors. Even though the parts used for gripper joints –the gripper housing covers, arm posts, and arm plates—were designed so that the gripper is tightly held by the six bearings mounted on the arm posts, a gap, about 1mm, was observed between the bearings and the groove on the gripper housing covers. Particularly, the bearing support structure was not good to support an axial load. Because the axial load is exerted on a certain point of each bearing by grooves on the gripper housing covers, the bearing was tilted with respect to the arm post when a load was imposed axially. The gap between the bearings and gripper and the tilt of the bearings by an axial load resulted in the tilt of the body with respect to the gripper.

The best way to solve this problem of the gripper joint errors is to make the joint mechanism more robust. Currently, three arm posts are used to hold the gripper at three points of its circumference. More arm posts may be added for more robust installation. Furthermore, a new joint mechanism can be developed instead of the

bearing joint mechanism. The joint mechanism for the middle joint, which employs two mounting blocks holding the connecting shaft, can be referred as a candidate. No significant error was observed in the middle joint. The improvement of the gripper joints can be one direction of the future research.

6.5 Self-assembly simulation

A self-assembling truss structure can be a fascinating extension of the Shady3D truss climbing robot. In this system, multiple Shady3D robots connect to one another using passive bars to form a large truss structure, called an active truss. This active truss can move and change its shape by synchronously actuating the active Shady3D modules. This system can be viewed as a heterogeneous self-reconfigurable robot system with active (the Shady3D robots) and passive modules (passive bars).

The concept of self-assembly of an active truss was implemented in computer simulations. Figure 6-5 through 6-7 show several snapshots from the simulations. In the snapshots, the segments in blue and cyan represent the active Shady3D modules. The blue part corresponds to the left side of the robot and the cyan part to the right side. Closed grippers are indicated by the short red segments. Passive bars are represented by the magenta line segments. The green segments simulate the ground structure where the modules move around.

For these simulations, MatLab was used as a development tool. Each simulation was performed in the following steps:

1. Initialize the simulation. The initialization process is executed based on the initialization data implemented as cell arrays in MatLab. The objects of the Shady3D modules, passive bars, and ground are created. Each Shady3D module and passive bar object is given a specific ID. The Shady3D modules that holds the ground structure are added to the `onGround` variable of the ground object.

2. Read the instruction for a step of iteration. The instruction is also implemented as a cell array in MatLab. A set of instruction for a step of iteration is composed

of three components: the ID of the Shady3D object to move, the parameter to modify (the joint or gripper), and the state of the parameter (the angle or gripper state). Multiple Shady3D modules can be modified at one step of iteration.

3. Update the object values according to the instruction. If one of the joint angles is modified, the corresponding joint angle of the designated Shady3D object is set to the commanded value.

   If one of the grippers is opened, the simulation program checks what object the gripper has held previously. If it has held the ground and the other gripper does not hold the ground, this Shady3D object is removed from the `onGround` variable of the ground object. If the gripper has held one of the passive bars, this Shady3D object is removed from the `ShadyList` variable of the bar. In addition, the information of the bar is removed from the Shady3D object.

   If one of the grippers is closed on the ground, this Shady3D object is added to the `onGround` variable of the ground object. If it is closed on one of the bars, this Shady3D object is added to the `ShadyList` of the bar, and the information of the bar is added to the Shady3D object. Because the Shady3D module grips on one of the four faces of the bar, the simulation program investigates which face the gripper grips on and assign the index of the face to the Shady3D object.

4. Rebuild the Shady3D and bar objects. In this step, the positions and orientations of the Shady3D modules and passive bars are computed based on the updated information in Step 3. First, all the Shady3D and bar objects are set to be "floated," which means they are not rebuilt, or computed. Then, the simulation program starts the rebuilding process from the ground structure. The Shady3D objects that are directly connected to the ground object are rebuilt and set to be "grounded." Next, the bars connected to the "grounded" Shady3D objects are rebuilt and also grounded. In turn, the Shady3D objects that are connected to the grounded bars and not grounded are rebuilt. This rebuilding process continues until all the Shady3D and bar objects are grounded,

or computed based on the updated information.

The way in which the Shady3D object is rebuilt is almost the same as the state evaluation procedure of the Shady3D robot, described in Section 4.2.3. The rebuilding method for the bar object is also similar to that of the Shady3D object.

5. Draw all the objects in an animation frame and save the frame in a movie file (`avi` format).

6. Repeat Step 2 through 5 with the next set of the instruction. Continue the iteration until all sets of the instruction are executed.

Simulations were carried out for three types of behaviors: tower building, tower moving, and tower rotating. The description of each simulation is presented below.

Tower building Figure 6-5 shows the procedure of the self-assembling of a truss tower. Twelve active modules and eight passive bars were employed to build a three-dimensional tower. The tower building is performed through the following steps.

- Two active modules form a six-degree-of-freedom manipulator by connecting to each other using a passive bar. Eight active modules form four manipulators. These manipulators can reach arbitrary position and orientation.

- Four 6-DOF manipulators move to the base location of the tower and approach remaining four active modules to pick them up. Remaining modules hold passive bars to connect to the 6-DOF manipulators.

- The 6-DOF manipulators connect to four remaining active modules. The active modules held by the manipulators release the grippers gripping the ground trusses. Thus, each 6-DOF manipulator becomes a structure consisting of three active modules and two passive bars.

- The four structures formed in the previous stage arrange themselves in the desired poses. Then, they connect to their neighbors to complete the tower structure.

Tower moving Figure 6-6 demonstrates the tower moving procedure. The tower built in the previous simulation is moved to the neighbor grid on the ground trusses. This is done through the following steps.

- The upper part of the tower is tilted toward the direction in which it moves. For this, two legs in diagonal position are rotated so that all gripper joints in the middle of legs have parallel joint vectors.
- The legs of the tower are moved to their desired position on the next grid. One leg is moved at a time to maximize the stability of the structure during the process.
- After all legs are moved to the next grid, gripper joints in the middle of legs are rotated synchronously to recover the original shape of the tower.

Tower rotating Figure 6-7 shows the rotating motion of the active truss. In this simulation, the tower is rotated by 90 degrees clockwise. Because of the kinematic limitations of the structure, it cannot rotate 90 degrees at a time. The rotation is divided into two stages. The procedure was described below.

- The top face of the tower is rotated by 30 degrees with respect to the grid on the ground by cooperatively rotating the joints of modules in the legs.
- Each leg, one at a time, is relocated at the intermediate position. This relocation allows the tower to rotate by remaining 60 degrees.
- The top face of the tower is rotated by 60 degrees to complete 90-degree rotation.
- Each leg, one at a time, is moved to the final position. Then, the original shape of the tower is restored.

These simulations demonstrated the possible motion of the truss structure consisting of active Shady3D modules and passive bars. The current Shady3D hardware is likely not appropriate for this type of system because it is not capable of lifting a

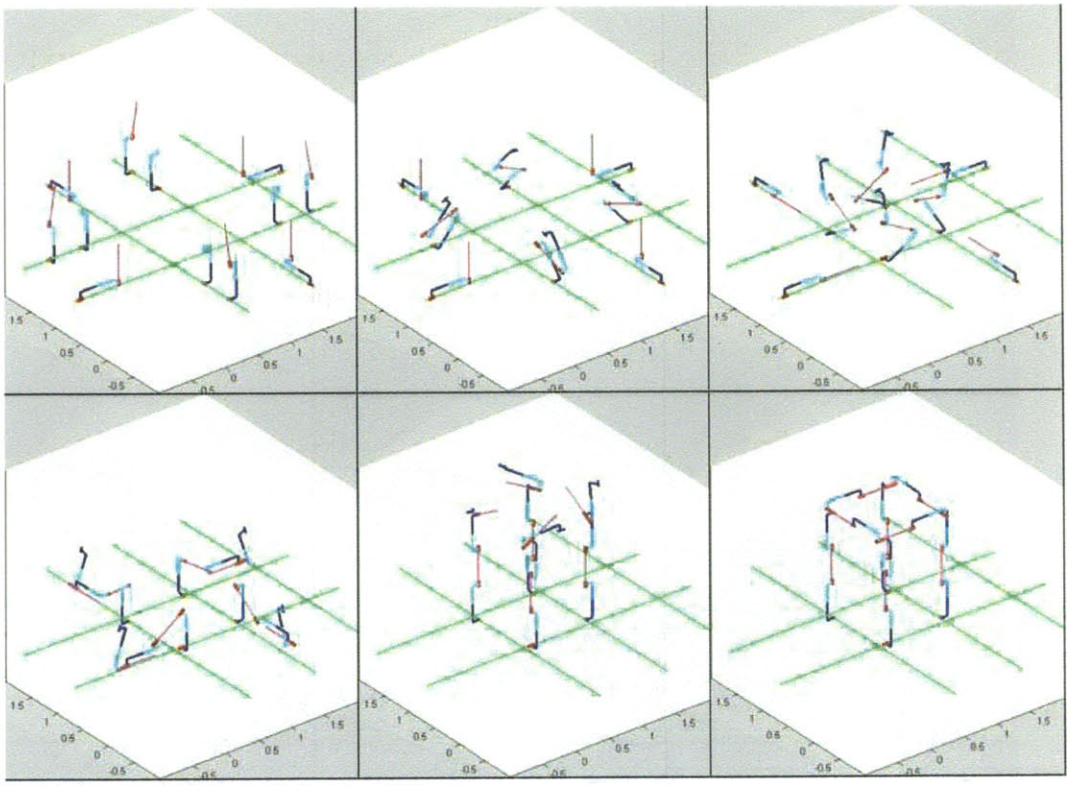


Figure 6-5: Six snapshots of the tower building simulation.

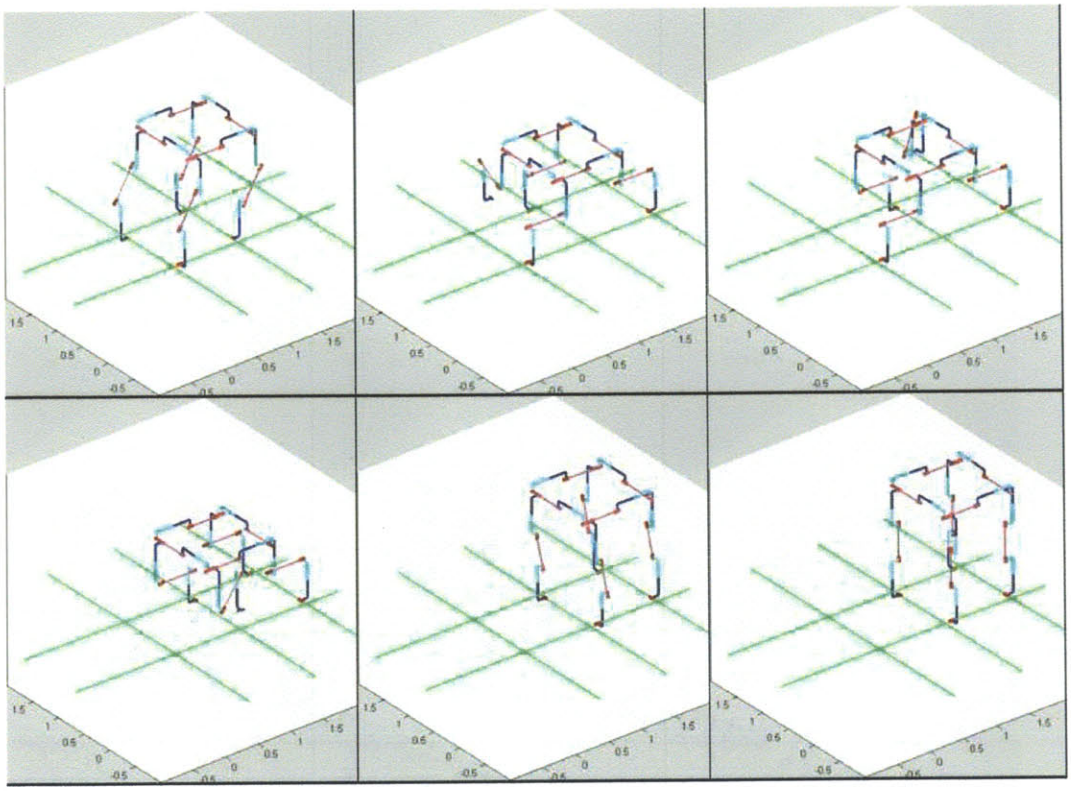


Figure 6-6: Six snapshots of the tower moving simulation.

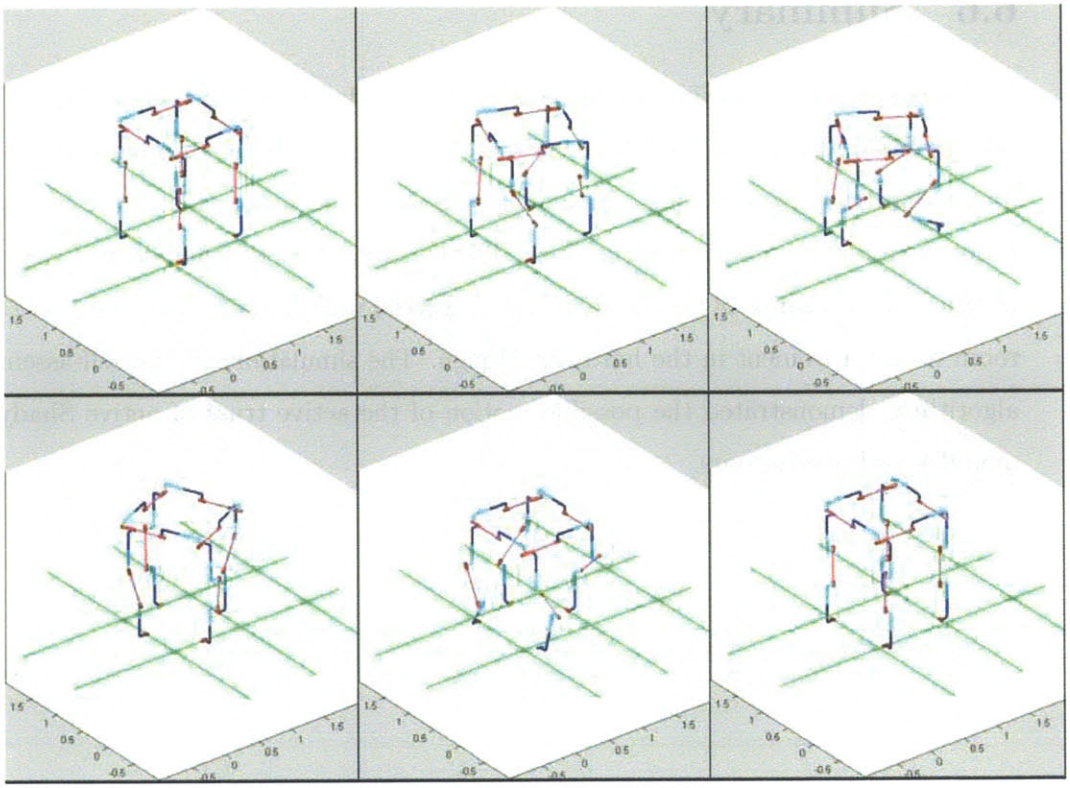


Figure 6-7: Six snapshots of the tower rotating simulation.

long chain of connected modules. However, we expect that hardware with the same kinematic configuration as the Shady3D robot can be developed to implement cooperative behaviors demonstrated in the simulations. To realize this goal, we need to reduce the size and weight of robot hardware and increase the power-to-mass ratio than the current Shady3D hardware.

6.6 Summary

In this chapter, the experiments on the Shady3D hardware and the simulations of the self-assembly algorithms have been presented. The Shady3D environment was built for the hardware experiments. The single step move and navigation of the Shady3D robot were tested in various situations. The hardware experiments showed that the design and algorithms of the Shady3D robot worked well in most cases, but there was room for improvement in the hardware design. The simulations of the self-assembly algorithms demonstrated the possible motion of the active truss of active Shady3D modules and passive bars.

Chapter 7

Conclusion

This thesis has presented the hardware design, low-level control algorithms, and high-level planning algorithms of the Shady3D modular truss climbing robot. Experiments on the hardware and simulations of the algorithms have also been described.

The 3-DOF design of the Shady3D robot has proved to be an optimal option to achieve both simplicity and locomotion capability. The hardware experiments in our test environment showed that the design and algorithms of the Shady3D robot work successfully and reliably for navigation in the 3-D truss structures. The simulations demonstrated the feasibility of the cooperation of multiple modules and the ability of the Shady3D system to self-assemble active trusses using passive bars. However, some shortcomings that require further improvement have been found, particularly in the hardware design.

7.1 Lessons learned

We have learned many lessons throughout the development of the Shady3D system. While many decisions we made were right, some of them were wrong and required repeated trials in different ways. Through this process of trial and error, we could learn many things. Some of the lessons learned are described below.

Design We tried to keep the design of the Shady3D robot as simple as possible. Out of several design alternatives, the 3-DOF design model was chosen, and it has worked well as an optimal option to achieve simplicity and locomotion functionality. When it comes to hardware implementation, however, errors in the hardware caused unanticipated problems. For example, the downward tilt caused by gravity and the error in the gripper joint could not be compensated in transition from a horizontal plane to vertical plane. The uncompensated error became the main cause of failure in the following steps. To compensate such an error, an additional degree of freedom such as the pitch joint in the 4-DOF model is necessary. However, adding more degrees of freedom requires the overall modification of the design, which takes a huge amount of time and effort. It is more desirable to make the hardware design more robust to minimize errors.

This error in the gripper joint resulted from the bearing joint mechanism used for the gripper joint. As discussed in Section 6.4, it was not as robust as expected. It was particularly weak for the axial load, which had not been considered seriously in the design stage. If we had tested the robustness of the bearing joint mechanism more rigorously in the early phase of the design, we could have chosen a better mechanism. The lesson learned regarding the design of the Shady3D robot is that we need to consider all possible errors and test the robustness of any suggested design for such errors from the beginning.

Packaging Packaging was a critical issue during the design of the Shady3D robot because the size of the robot is small. We had to redesign many times to configure components in a restricted space. For example, when the gripper was designed first, the switches for gripper misalignment correction were not taken into consideration. As a result, we spent much time to figure out the way to mount these switches on the gripper that did not have enough space for them. Finally, the design of the gripper paddle was modified to have mounting places for the switches. Another example related to the packaging issue is the installation of the electronics. Because the size and configuration of the electronics components were not considered sufficiently

at the early stage of the design, the circuit boards containing the electronics were installed on the top of the arm plates, covering the screws of the arm assembly. As a result, disassembling particular parts of the robot for repair or other purpose became difficult and painful. The lesson learned is that space allocation and configuration for *all* components of the robot—motors, sensors, structural parts, and electronics—must be considered early in the design process.

Sensors Originally, we wanted to implement the sensing ability for the environment in the Shady3D robot. With such ability, the robot could localize its position and find a truss to grab. A vision system was suggested as a solution. However, the computing capability of our current Shady3D robot was not suitable to implement vision. Moreover, interpreting images taken by a camera is complicated. Some patterns might be attached on trusses so that the robot could recognize their distance and orientation. For these reasons, we realized that vision is much more complicated than we had thought, and used the alternative solution, a priori knowledge of the robot on the environment. Vision can be investigated in the future.

7.2 Future work

Future work can be divided into research on the Shady3D hardware and research on planning algorithms. In the hardware-related research, a more robust gripper joint mechanism needs to be explored to reduce errors observed in experiments. Though the current bearing joint mechanism can be strengthened, designing a new, more robust mechanism would be more desirable. Next, sensors for external information can be considered for more autonomous navigation ability. Vision can be a promising and attractive option for this purpose. Thirdly, the Gumstix needs to be improved to implement the Java control software. When this is done, we could remove the tether from the robot and run multiple robots at the same time. Finally, the way to reduce the size and weight of the robot and to increase the power-to-mass ratio should be explored to enable the multi-module cooperation.

In the algorithm-related research, algorithms for the communication and cooperation of multiple robots would be implemented. The implementation of these algorithms will include establishing a rule for the message interchanged between robots and developing a program that utilizes the Bluetooth function of the Gumstix. The software representation of passive bars also needs to be developed. In addition to algorithms for communication and cooperation of multiple robots, algorithms for self-assembling and self-reconfiguring active trusses would be studied in detail.

Appendix A

Shady3D robot specifications

Table A.1 presents the specifications of the Shady3D robot.

Item	Value
Total arm length	250 mm
Center-to-center distance	180 mm
Arm width	80 mm
Gripper length	70 mm
Gripper width (closed)	60 mm
Gripper width (open)	118 mm
Total height (with grippers closed)	133 mm
Gripping-point-to-joint-center distance	69.5 mm
Weight	1.34 kg
Power methodology	4 Polymer Li-ion battery cells
Power voltage	7.4 V
High-level control	Gumstix or Linux workstation
High-level communication	Serial
Low-level control	AVR Atmega128 microcontroller in Robostix
Low-level communication	Serial RS485
Motor control	AVR Atmega8 microcontrollers
Joint actuators	MicroMo 1724T006SR + 16/7 66:1 gearmotor
Gripper actuators	Sanyo NA4S mini gearmotor with 298:1 gearhead

Table A.1: Shady3D robot specifications

Appendix B

Schematics

As described in Section 3.3.7, the electronics of the Shady3D robot is divided on two circuit boards. Each circuit board is installed on top of each part of the arm. The left circuit board contains three motor control boards (address 0, 2, and 3), and the right circuit board contains two motor control boards (address 1 and 4) and the Robostix board. Figures B-1 and B-2 show the schematics for the left and right circuit board, respectively.[1] Figure B-3 shows the schematics for the motor control board.

[1]Figure B-2 includes a voltage regulator (LP2992) and an accelerometer (MMA7260). These components are not used in the current Shady3D robot. They will be used in the future to detect the direction of the gravitational acceleration.

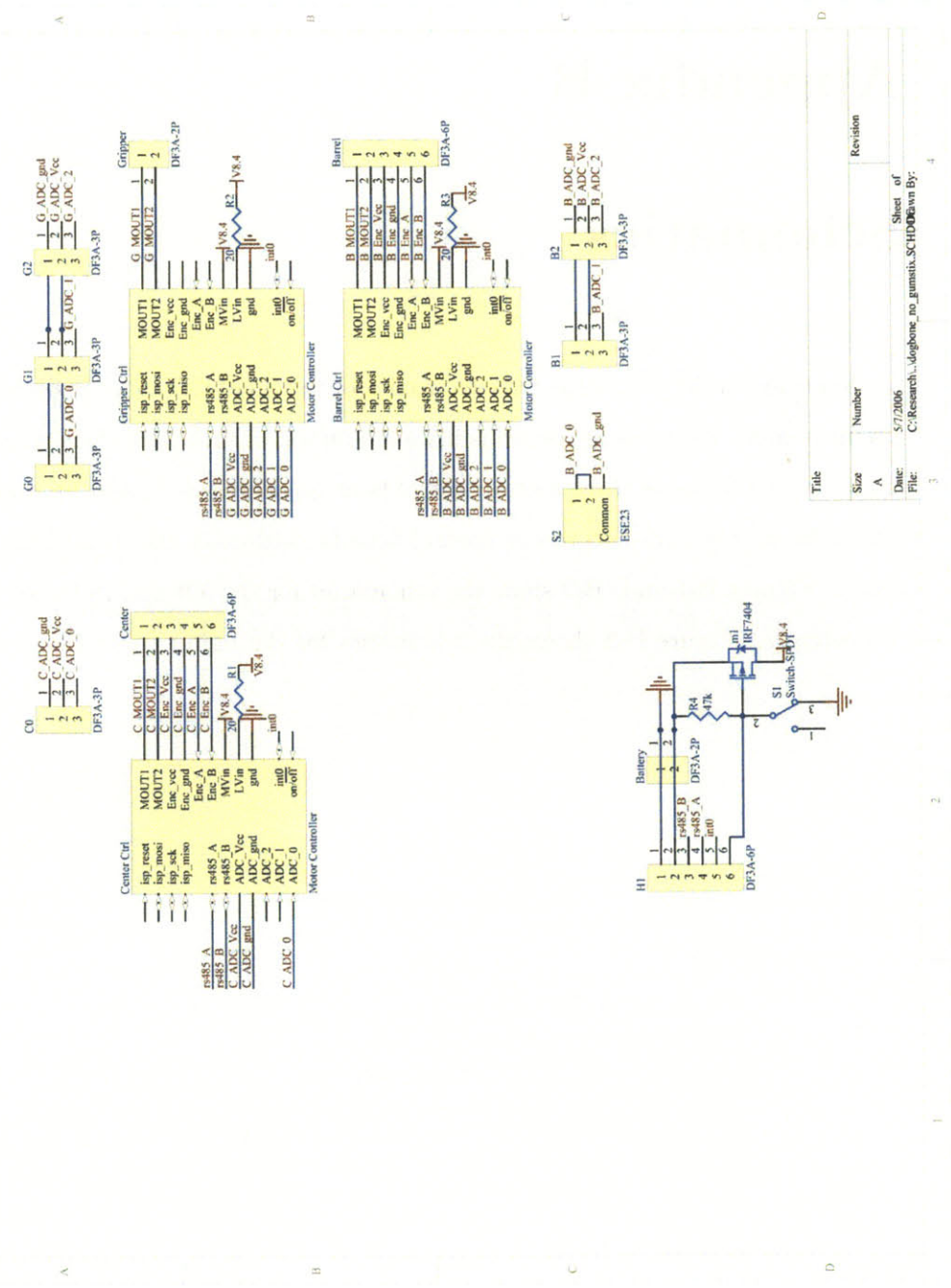


Figure B-1: Schematics for the left circuit board.

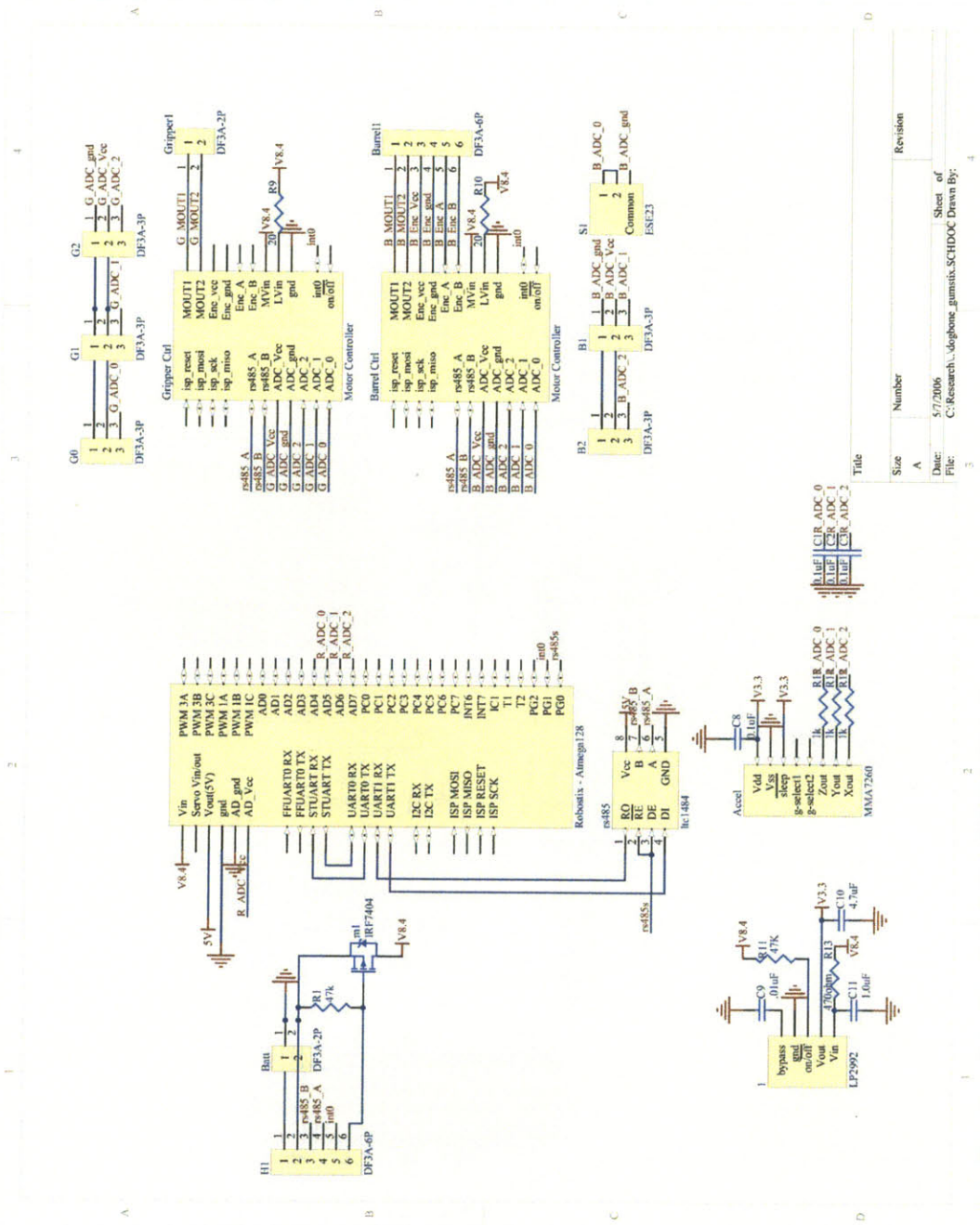


Figure B-2: Schematics for the right circuit board.

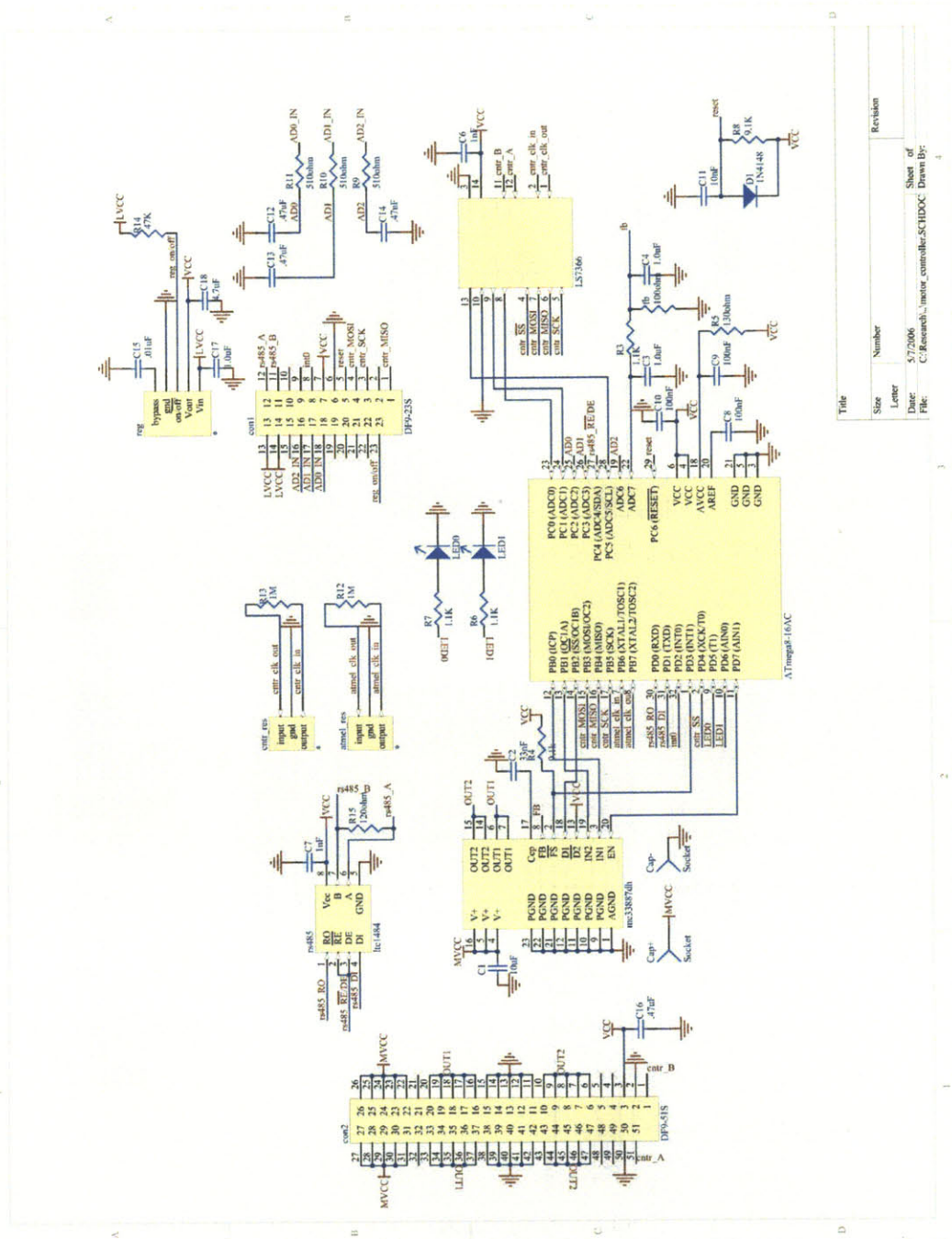


Figure B-3: Schematics for the motor control board.

Bibliography

[1] M. Almonacid, R. J. Saltarén, R. Aracil, and O. Reinoso. Motion planning of a climbing parallel robot. *IEEE Transactions on Robotics and Automation*, 19(3):485–489, June 2003.

[2] Hisanori Amano, Koichi Osuka, and Tzyh-Jong Tarn. Development of vertically moving robot with gripping handrails for fire fighting. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 661–667, Maui, Hawaii, USA, 2001.

[3] Rafael Aracil, Roque J. Saltarén, and Oscar Reinoso. A climbing parallel robot. *IEEE Robotics and Automation Magazine*, pages 16–22, March 2006.

[4] C. Balaguer, A. Giménez, J.M. Pastor, V.M. Padrón, and M. Abderrahim. A climbing autonomous robot for inspection applications in 3d complex environments. *Robotica*, 18:287–297, 2000.

[5] Tim Bretl, Stephen Rock, Jean-Claude Latombe, Brett Kennedy, and Hrand Aghazarian. Free-climbing with a multi-use robot. In *Proceedings of the International Symposium of Experimental Robotics*, 2004.

[6] Zack Butler and Daniela Rus. Distributed planning and control for modular robots with unit-compressible modules. *International Journal of Robotics Research*, 22(9):699–715, 2003.

[7] A. Castano and P. Will. Mechanical design of a module for reconfigurable robots. In *Proceedings of International Conference on Intelligent Robots and Systems*, pages 2203–2209, 2000.

[8] Carrick Detweiler, Marsette Vona, Keith Kotay, and Daniela Rus. Hierarchical control for self-assembling mobile trusses with passive and active links. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 2006.

[9] William Doggett. Robotic assembly of truss structures for space systems and future research plans.

[10] Jacob Everist, Kasra Mogharei, Harshit Suri, Nadeesha Ranasinghe, Berok Khoshnevis, Peter Will, and Wei-Min Shen. A system for in-space assembly. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2356–2361, Sendai, Japan, 2004.

[11] Aaron Greenfield, Alfred A. Rizzi, and Howie Choset. Dynamic ambiguities in frictional rigid-body systems with application to climbing via bracing. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1959–1964, Barcelona, Spain, April 2005.

[12] G. Hamlin and A. Sanderson. TETROBOT: A modular approach to reconfigurable parallel robotics. *Kluwer Academic Publishers*, 1997.

[13] Ben Iannotta. Creating robots for space repairs. *Aerospace America*, pages 36–40, May 2005.

[14] Robert L. Williams II and James B. Mayhew IV. Cartesian control of vft manipulators applied to doe hardware. In *Proceedings of the Fifth National Conference on Applied Mechanisms and Robotics*, Cincinnati, OH, October 1997.

[15] Keith Kotay. *Self-reconfiguring robots: Designs, algorithms, and applications.* PhD dissertation, Dartmouth College, December 2003.

[16] Keith Kotay and Daniela Rus. Efficient locomotion for a self-reconfiguring robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[17] Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 424–432, 1998.

[18] Keith D. Kotay and Daniela L. Rus. Navigating 3D steel web structures with an inchworm robot. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 1996.

[19] Keith D. Kotay and Daniela L. Rus. Algorithms for self-reconfiguring molecule motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, 2000.

[20] S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 442–448, 1994.

[21] Satoshi Murata, Haruhisa kurokawa, Eiichi Yoshida, Kohji Tomita, and Shigeru Kokaji. A 3-d self-reconfigurable structure. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 432–439, Leuven, Belgium, May 1998.

[22] Satoshi Murata, Eiichi Yoshida, Kohji Tomita, Haruhisa Kurokawa, Akiya Kamimura, and Shigeru Kokaji. Hardware design of modular robotic system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and System*, 2000.

[23] Michael C. Nechyba and Yangsheng Xu. SM2 for new space station structure: Autonomous locomotion and teleoperation control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1765–1770, May 1994.

[24] Michael C. Nechyba and Yangsheng Xu. Human-robot cooperation in space: SM2 for new space station structure. *IEEE Robotics and Automation Magazine*, 2(4):4–11, December 1995.

[25] Robert T. Pack, Joe L. Christopher Jr., and Kazuhiko Kawamura. A rubbertuator-based structure-climbing inspection robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Albuqurque, New Mexico, April 1997.

[26] A. Pamecha, C.-J. Chiang, D. Stein, and G chirikjian. Design and implementation of metamorphic robots. In *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, 1996.

[27] Zaidi Mohd Ripin, Tan Beng Soon, A.B. Abdullah, and Zahurin Samad. Development of a low-cost modular pole climbing robot. In *TENCON*, pages 196 200, Kuala Lumpur, Malaysia, 2000.

[28] D. Rus and M. Vona. Self-reconfiguration planning with unit compressible modules. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2513–2520, 1999.

[29] D. Rus and M. Vona. A physical implementation of the Crystalline robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[30] Roque Saltaren, Rafael Aracil, Oscar Reinoso, and Maria Antonieta Scarano. Climbing parallel robot: A computational and experimental study of its performance around structural nodes. *IEEE Transactions on Robotics*, 21(6):1056–1066, December 2005.

[31] J. Suh, S. Homans, and M. Yim. Telecubes: Mechanical design of a module for self-reconfiguring robotics. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2002.

[32] Cem Ünsal and Pradeep K. Khosla. Mechatronic design of a modular self-reconfiguring robotic system. In *Proceedings of the IEEE International conference on Robotics and Automation*, pages 1742–1747, San Francisco, CA, 2000.

[33] Cem Ünsal, Han Kiliççöte, and Pradeep K. Khosla. A modular self-reconfigurable bipartite robotic system: Implementation and motion planning. *Autonomous Robots*, 10(1):23–40, 2001.

[34] Marsette Vona, Carrick Detweiler, and Daniela Rus. Shady: Towards robust truss climbing with mechanical compliances. In *Proceedings of the 2006 International Symposium of Experimental Robotics*, 2006.

[35] M. Yim, D. Duff, and K Roufas. Polybot: a modular reconfigurable robot. In *Proceedings of IEEE International Conference on Robotics and Automation.*