PROJECT REPORT

HØGSKOLEN I ÅLESUND
Aalesund University College

TITLE:
Synergic grasping with a modular hand

CANDIDATE(S):			
2225		**2226**	
2223			
DATE:	COURSE CODE:	COURSE TITLE:	RESTRICTION:
06.05.2013	IP304812	Innføring I Mekatronikk	
STUDY PROGRAME:		PAGES/APPENDIX:	LIBRARY NO.:
AUTOMATISERINGSTEKNIKK		27/1	

SUPERVISOR(S):
Houxiang Zhang, Filippo Sanfilippo

SUMMARY:

This report gives a brief overview over the concept of synergies and relates the subject to control of modular robotic hands, with the practical example of a gripper based on the Barret Hand, and makes probable the viability of synergies as a simplifying mechanism in the control of such hands.

This report is submitted by students for evaluation at Ålesund University College.

Postal adress:
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Visit adress
Larsgårdsvegen 2
Internett
www.hials.no

Telephone
70 16 12 00
E-mail
postmottak@hials.no

Fax
70 16 13 00

Bank
7694 05 00636
Enterprise no.
NO 971 572 140

Preface

When we first were introduced to this assignment, none of us had ever really heard of the term synergies or underactuation. As we read more and more papers about the subject, the concept seemed more and more abstract. Much time was invested in trying to understand the principles, as there were a lot of confusing terminology. We'd therefore like to thank Filippo Sanfilippo for helping out in this process, and giving us the opportunity to have a Skype call with Ph.D. student Guido Gioioso at the Department of Advanced Robotics of the Italian Institute of Technology (Genova, Italy) and at the Department of Information Engineering of the University of Siena, who cleared up much of the confusion around the subject. A special thanks to him aswell.

Contents

Chapter 1

Summary

This report gives a brief overview over the concept of synergies and relates the subject to control of modular robotic hands, with the practical example of a gripper based on the Barret Hand, and makes probable the viability of synergies as a simplifying mechanism in the control of such hands.

Chapter 2

Terminology

2.1 Terms

Synergy
Throughout this report, except where otherwise stated, the term *synergy* will be used to refer to linearily scaled sets of angles that together define a state controlled by a single variable.

Synergy matrix
A matrix that contains a set of synergies, which transforms a vector into a linear combination of all the system's defined synergies, scaled by the degree of activation.

Activation vector
A vector that contains coefficients for scaling each individual synergy in a synergy matrix, describing to which degree each synergy is activated.

Underactuation
Controlling a mechanical system with less control parameters than it has degrees of freedom.

2.2 Abbreviations

DOF
Degrees of Freedom
The number of independant variables or rotations/translations in a system

PCA
Principal Component Analysis

I^2C
Inter Integrated Circuit

UART
Universal asynchronous receiver/transmitter

DH-Table
Denavit-Hartenberg table

Chapter 3

Introduction

Controlling robotic hands have always been a big challenge for robotic researchers. Since objects come in a vast variety of shapes and sizes, and since the kinematic structure of a robotic hand can be quite complex, it can be challenging to develop uncomplicated control systems. This is where the concept of synergies can be useful.

This report describes the main project in the course Introduction to Mechatronics. The project was to control a modular robotic hand based on the Barret Hand [1], using synergies. Synergies enables the use of few parameters to control many degrees of freedom (underactuation), and thus it can simplify the control mechanisms for robotic grippers.

Chapter 4

Theory

4.1 Synergy

4.1.1 Definition

The word synergy is derived from the greek word *sunergos*, which means "working together" [2]. It is defined as *"the interaction or cooperation of two or more organizations, substances, or other agents to produce a combined effect greater than the sum of their separate effects."* [2]. So what does this means for human and robotic hands?

On the human hand, the joints in all the fingers are working together to achieve a greater goal, whether it's grasping an object or manipulating it. Studies also indicate that even though the human hand has 20 *DOF*, it's largely controlled by very simple parameters [3]. Where robotic hands were controlled by moving each joint separately, the synergic movement of the joints in the hand allows it to be controlled by only a few parameters. This means that it might be possible to simplify the control mechanisms of robotic hands as well.

There are currently several different terms being used when describing synergies in relation to grasping. In [4] they are referred to as *eigengrasps* and *eigengrasp matrixes*, where in [3] both *synergy* and *principal components* have been used. In most reports it refers to the same thing, but some reports use *synergy* explicitly about *Santello's* work on grasping-postures of the human hand. This distinction will be done explicitly here where necessary, and the terms *eigengrasp* and *synergy* on their own will be treated as equivalent.

4.1.2 Santello's work on mapping human synergies

In 1998, Marco Santello, Martha Flanders and John F. Soechting performed a study where five subjects where asked to grasp 57 different objects [3]. However, instead of grasping the actual object, the subjects were given a description of the object and were then asked to perform the grasp in the air. Santello and the other researchers noticed that certain patterns emerged, and by using a procedure called principal component analysis (see description below), they were able to identify these patterns [3]. These patterns were labeled as *postural hand synergies* [3] and the study indicated that the first two of the components could account for >80% of the varience in our gripping techniques [3]. This information inspired researcher to explore new design- and control strategies for robotic hands [5].

4.1.3 Mapping synergies onto a robotic hand

There are several approaches to mapping synergies onto a robotic hand. One way is to use a simulator to create a digital representation of the hand, and then simulate a series of different grasps on a diverse set of objects. The quality of the grip is then analyzed by looking at the contact points between the hand and the object, and using algorithms to determine how stable these are. Once enough data have been accumulated one may perform a procedure called *Principal Component Analysis (PCA)*. This technique is used to analyze large sets of data, and expressing the data in such a way as to highlight their similarities

and differences [6]. This will define a set of *principal components* [6], which can be used to determine the postural synergies.

Synergic gripping may also be achieved using Santello's work on the human hand [3]. The postural synergies found in his studies is described by a 20x15 matrix. This matrix describes the 20 DOF for the human hand, and 15 different synergies. This will have to be done by naming the joints on the robotic gripper after joints on the human hand, and finding the values from Santello's work and map it to a new matrix for the robotic hand.

An algorithm suggested by Monica Malvezzi, Guido Gioioso, Gionata Salvietti and Domenico Prattichizzo in [7] is another way to map synergies onto a robotic hand. This algorithm is based on using a virtual hand controlled by synergies to manipulate an object, and calculating the movements of the physical robotic hand that would result in the equivalent manipulation of the object.

4.2 Mathematical background

A useful way to describe points and how they are moved in relation to each other is to use vectors to describe points in space and matrixes to define transformations from one coordinate system to another [8].

4.2.1 Matrix transformations

Two important types of transforms for describing physical systems are rotation around the coordinate axes and translation along the coordinate axes. Given a transformation matrix T and a vector v, the transformed vector v^* is given by $v^* = v \cdot T$ [8].

For rotation with an angle θ around the coordinate axes exist the following transformation matrixes.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \qquad R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \qquad R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To express translations in 3-space, a 4-space vector and matrix is needed, with the 4th element of the vector being 1. This gives the transformation-matrix along x, y and z as[8]

$$T_{xyz} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In addition to this, for visualization purposes two more transformations are necessary; The view and projection matrix transform the 3D models to the perspective normally seen[9]. These transformations are explained by OpenGL sdk documentation , and will be used directly as explained there[10], e.g.

http://www.opengl.org/sdk/docs/man2/xhtml/gluLookAt.xml

http://www.opengl.org/sdk/docs/man2/xhtml/gluPerspective.xml.

4.2.2 Denavit-Hartenberg

Denavit and Hartenberg described a systematic method for describing chains of attached coordinate systems, in which each transition from one joint to the next can be described as only four parameters [8]. The shortest possible distance a between the joint axes, the angle α between the axes measured around the axis of the distance, the distance d along the joint axis of the second link and the angle θ of rotation around the same axis[8].

Describing all the links in this way results in a table of parameters, and for each entry in the DH-Table a transformation matrix can be set up, transforming coordinates from the coordinate system of one joint to that of the previous[8]

$${}^{i-1}_{i}T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & a \\ \sin\theta\cos\alpha & \cos\theta\cos\alpha & -\sin\alpha & -\sin\alpha d \\ \sin\theta\sin\alpha & \cos\theta\sin\alpha & \cos\alpha & \cos\alpha d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Chapter 5

Method and Materials

5.1 Materials

Hardware

1	Computer	Capable of running Processing and the Arduino IDE
1	Arduino Uno	http://arduino.cc/en/Main/arduinoBoardUno
3	Arduino Nano	http://arduino.cc/en/Main/arduinoBoardNano
11	HiTEC HS-85MG $^{+}$ servos	
3	Lynxmotion Aluminum Long "C" Micro Servo Bracket	
14	Lynxmotion Aluminum "C" Micro Servo Bracket	
13	Lynxmotion Aluminum Multi-Purpose Micro Servo Bracket	
22	1Ω - 0.6 Watt Resistors	
2	1.5kΩ Resistors	
2	Power supply	
1	USB Cable	Standard USB A → B connector
1	USB Cable	Standard USB A → mini-B connector
1	Soldering iron	
	Breadboards	
	Assorted wires	
	Assorted strips	
	Vulcanizing tape	
	Electrical tape	
	Knife	

Software

Arduino IDE
Netbeans
Blender
OpenRAVE
GraspIt!
MatLAB with SynGrasp

Software libraries

Java	LWJGL (Lightweight Java Gaming Library, has OpenGL bindings)
	Kahlua (Lua scripting engine)
	RXTX (Communication library for serial and parallel ports)
Arduino	Wire
	EEPROM
	SoftwareServo
	EasyTransfer

5.2 Assembly

5.2.1 Modifying the hand

Some modifications were made to enhance the overall grip quality of the modular gripper. Instead of having the fingertips loose and able to move, it was decided that it would be better to position the fingertips in a rigid way. A bit of soft material was taped to the fingertips using vulcanized tape to increase the friction.

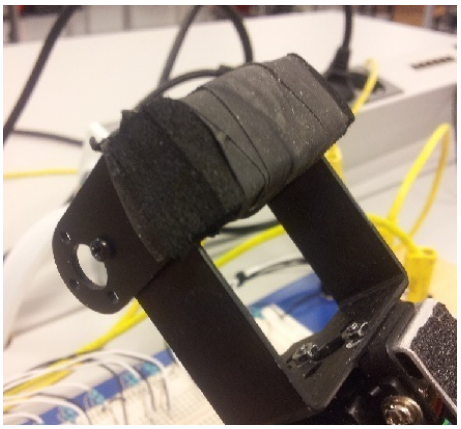


Figure 5.1: Rigid fingertip

5.2.2 Constructing a base for the hand

In order to test the gripper in a practical way it was decided to mount it onto a small base. The base was made big enough to fit the gripper and also the controll circuit that later would controll it. Wood was chosen as construction material since it is very quick and easy to work with. It was constructed using a 500x500x15mm wooden plate and a 70x70x50mm cut off part of a wooden beam. The small cut off beam was screwed to the wooden plate using a couple of metal brackets, and the gripper was then mounted on top of the small beam with the purpose of lifting the gripper from the ground.

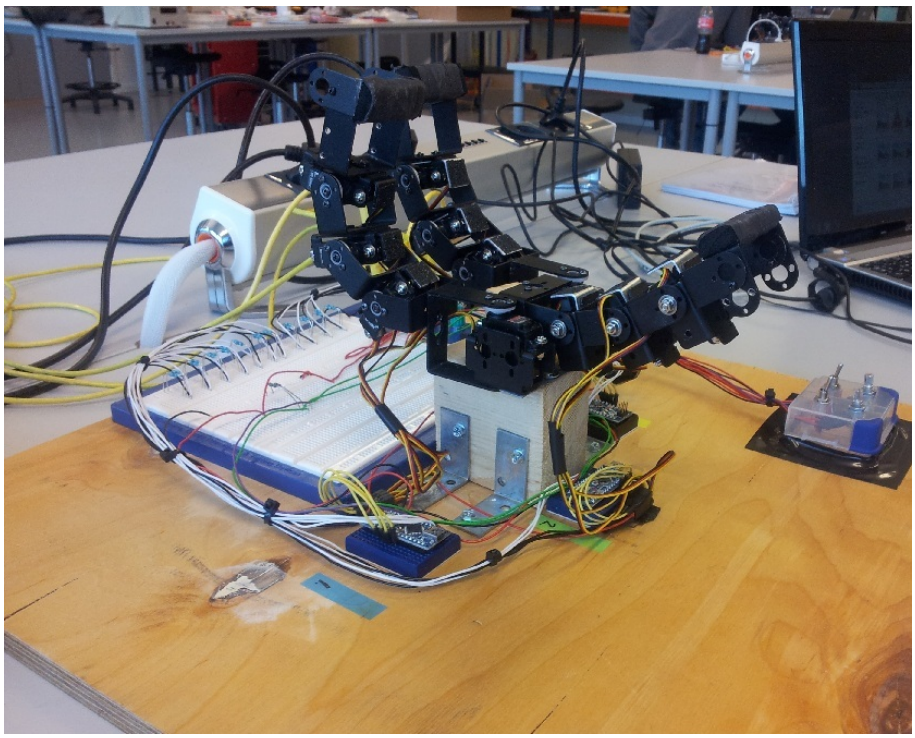


Figure 5.2: Hand mounted on the base

5.2.3 Control circuit

Controll

Four arduinos are used to control the gripper, one Arduino Uno as master and one Arduino Nano for each of the three fingers acting as slaves. The nanos each have its own small breadboard which is mounted on the wooden plate. There is also a bigger breadboard where the master Arduino, and the rest of the controll circuit is located. I^2C is used for communication and this means connecting the A4 and A5 together for all the Arduinos. A 1.5kΩ resistor is placed between A4 and 5V and one is placed between A5 and 5V. This is to ensure that the Arduinos won't interpret noise as an actual high or low value. To control the gripper, two potensiometers are used. Two switches were also added, such that it would be easy to turn off the power to the master Arduino or the slave Arduinos.

Measuring current

To measure the current, two 1Ω resistors in parallell are used for each servo. The resistors are connected from the servos ground to common ground, such that the current will run through that point. Since the resistors are very small, the current can be read as the change in voltage over those resistors. Wires are therfore connected from those resistors to the analog input on the respective Arduinos. The full schematics can be seen in figure 5.3.

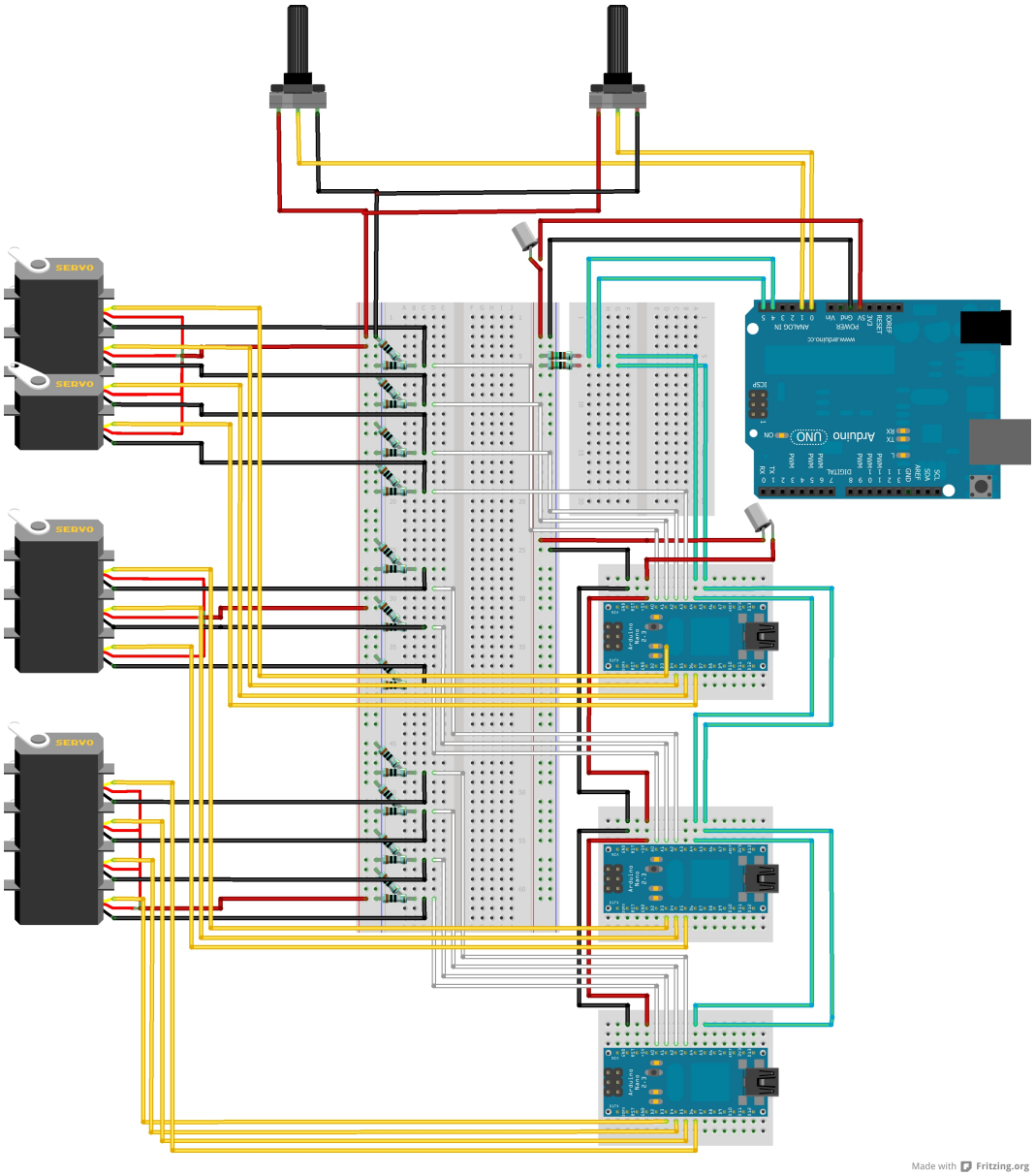


Figure 5.3: Connections

5.3 Communication

5.3.1 I²C

For communication, this project uses the *EasyTransfer* library for Arduino, written by Bill Porter [11]. This library uses I²C as a basis, but it provides an extra interface which makes it easier to use. The library allows for all sorts of data types to be transfered without having to decode it at a binary level. Figure 5.4 shows a diagram explaining the information flow in the system.

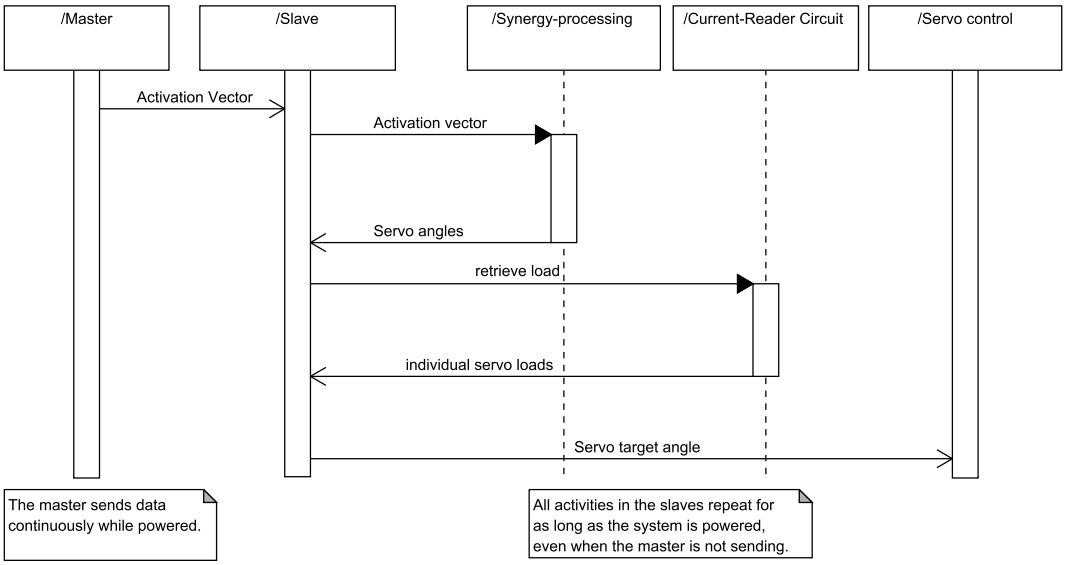
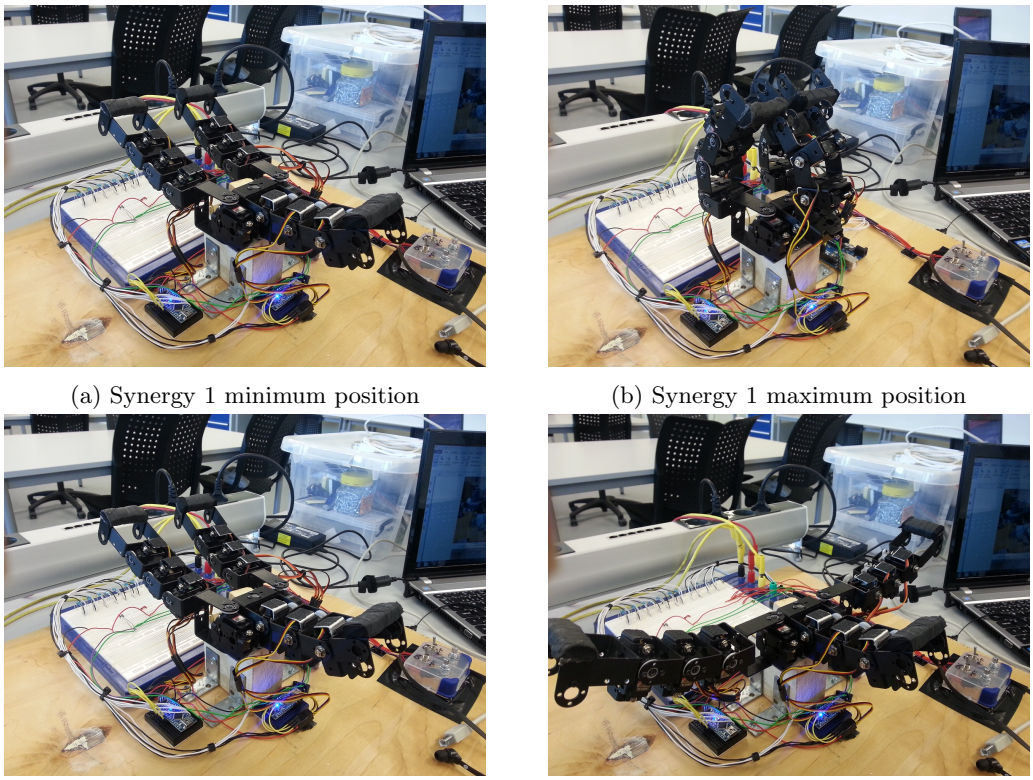


Figure 5.4: Information flow diagram

5.4 Mapping the synergies

The synergies was developed based on the findings in [4]. In [4] the author proposed two synergies for the Barret Hand, and since the modular gripper used in this project is based on the Barret Hand it seemed logical that the synergies had to be similar to the ones found in [4]. The author proposes two synergies: one for the flexion of the fingers, and one to control the spread angle. By combining these two synergies the hand is able to grasp a wide variety of objects. The mapped synergies are shown in figure 5.5.



(a) Synergy 1 minimum position

(b) Synergy 1 maximum position

(c) Synergy 2 minimum position

(d) Synergy 2 maximum position

Figure 5.5: Two mapped synergies

5.5 Control

The program that runs on the master Arduino is very simple. It joins the I^2C network as the master, and read and send the values from the potentiometers to the slave Arduinos. The slave Arduinos each have their own program that interprets the values sent by the master, and then map them differently for each servo that it is controlling.

Load is measured for each servo, and when the load reaches a certain limit the servos are programmed to "give in", which means it moves a little to reduce the strain. This is to ensure that no servo gets overloaded, and to provide flexibility to the hand.

5.6 Simulation

To simulate the gripper a fairly basic simulation program was developed in Java, utilizing the LWJGL library for visualizing by OpenGL and the Kahlua library to make the simulator scriptable and dynamic. To provide communication-abilities, the RXTX-library was also included, and the libraries were bound together with the base clsases in the core of the simulator, as shown in the simplified class diagram in figure 5.6.

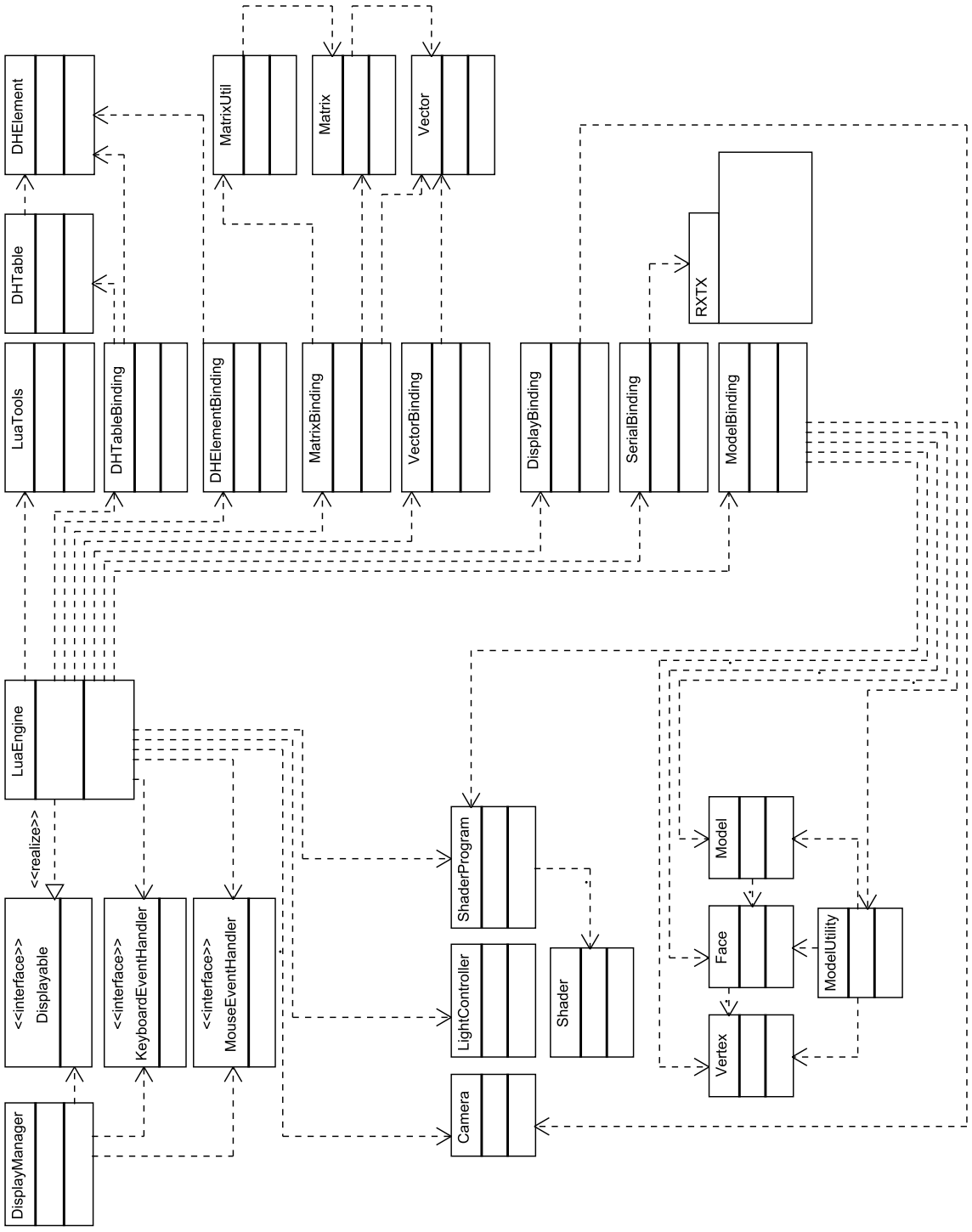


Figure 5.6: Simulator Class Diagram

5.6.1 Core

Since transformations were provided in the form of matrixes, basic support for matrix- and vector-math was added in the form of addition, subtraction, multiplication and translation, and for vectors the scalar product and cross product. Since the simulator made distinctions between matrixes and vectors, a more specialized method for transforming a vector with the matrix was added. Together, these formed the basis for both normal transformations, as well as adding the option to process synergy-matrixes directly in the simulator.

To provide a simple way to process the kinematics of the fingers of the gripper, two classes were added; DHTable provides a way to manage an entire DH-Table, and DHElement represents a set of the 4 Denavit-Hartenberg parameters, as well as information about whether the joint is prismatic or rotational. Each DHEntry-object was made capable of generating it's own transformation-matrix from it's link to the previous, as well as dynamically deciding which value to update when provided new variable-data. The DHTable class was then updated to utilize these functions to provide transformations from any joint to global coordinate space, and each DHTable was for convenience given a transformation-matrix to re-position the base coordinate system statically.

To visualize the simulated behavior, OpenGL was utilized on a low level, using raw OpenGL-calls through LWJGL's OpenGL bindings, and convenience classes were added to hold on to Model-data, Face-data and Vertex-data. Each model was decided to consist of Faces, where each face is a triangle consisting of 3 vertexes. Each vertex represents a point in 3-space, but is kept separate from the Vector-class due to it's direct relation to building graphical models. To help build models, a convenience class was created, containing static methods for loading a model from file, or to automatically generate a box. A model can consist of several loaded files and boxes in any combination, and each generated or loaded part can have it's own coloring.

To manage the displaying of the scene, a simple DisplayManager class was written, which can track Displayable objects. Each Displayable was through DisplayManager made capable of recieving data about mouse and keyboard events, so basic input could be provided. In addition, lightweight classes for managing camera-placement and light were created, each contributing either data to the shader, or transformation-matrixes. All of these require the presence of a shader-program, which manages the actual display, and a default shader was bundled into the simulator, to ease it's use.

5.6.2 Scriptability

To make it possible to dynamically create and test models without having to recompile the core each time, the core functionality was bound to a scripting engine: Lua. Through use of the Kahlua library, basic functions for creating vectors, matrixes, dhtables and models from within a simple script were created, and access to the most necessary functions of each object were mapped from Java to Lua. This allowed the direct creation of DH-Tables and transformation matrixes from a dynamically called script.

Convenience methods were added to give the ability to split vectors and perform transformations from the script, as well as the ability to apply a vector of numbers to a DHTable, setting all the appropriate variables in one command. This was done to give the ability to transform an activation vector with a synergy matrix, and split the results over the three DH-tables for the gripper's fingers.

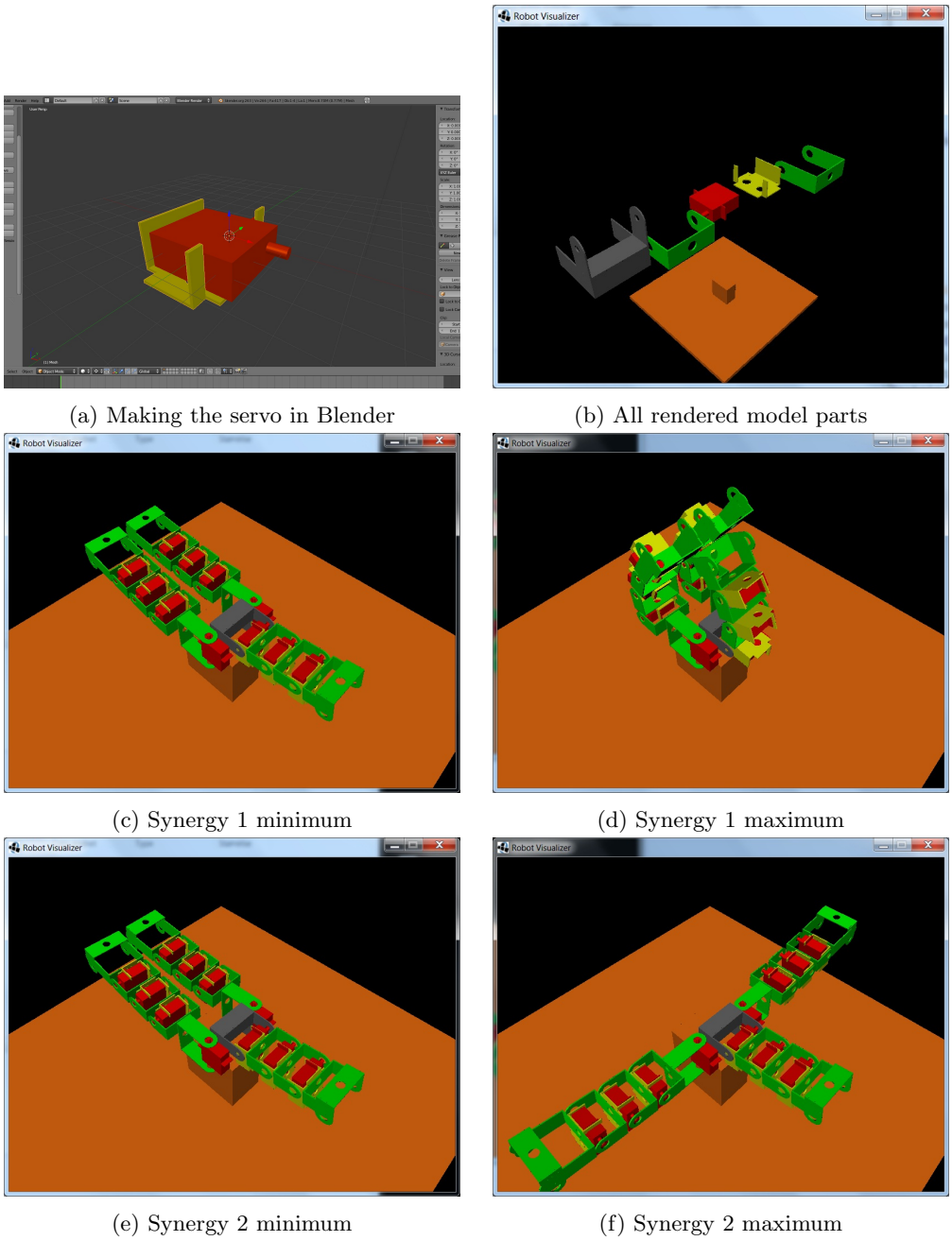
Finally, the lua core was turned into a Displayable-compatible class, and it was given the ability to manually call specially selected functions from within the lua-core, allowing lua to control the display process. The specific functions looked for were to initialize,render and destroy the scene, as well as provide event-data from mouse and keyboard.

For convenience, a small binding was also added to the RXTX-library, giving a way to feed serial data into the script. It was made so that once the connection was established, every time a line of text was recieved, a user-selected function would be called with the line as parameter.

5.6.3 Use

When the simulation engine itself was created, models of the brackets and motors were created using Blender, and exported to the Wavefront obj-format, specifying it to export the normals for each face, as well as making all surfaces triangles. In addition, all models were exported specifying Z to be the up-axis to have the models conform to the convention that was used when describing the gripper.

The models were loaded into the script, along with the script-generated DH-Tables and a transformation matrix, initially mapping the x and y coordinates to the mouse to basic synergies, and later on recieving the same information from the arduino over the serial interface. A basic camera control was added to the script, to make it possible to look closer at the movements while fine tuning the values.



(a) Making the servo in Blender

(b) All rendered model parts

(c) Synergy 1 minimum

(d) Synergy 1 maximum

(e) Synergy 2 minimum

(f) Synergy 2 maximum

Figure 5.7: Images from the visualisation process

Chapter 6

Result

From the examples and data provided in [4] and the experiments done with finetuning the initially guessed values a synergy-matrix for two synergies was created. The largest values for the first synergy are at the base of each finger, the smallest values at the outermost link. For other grippers of similar topology some values may have to be flipped in the case of differing definitions for positive rotation.

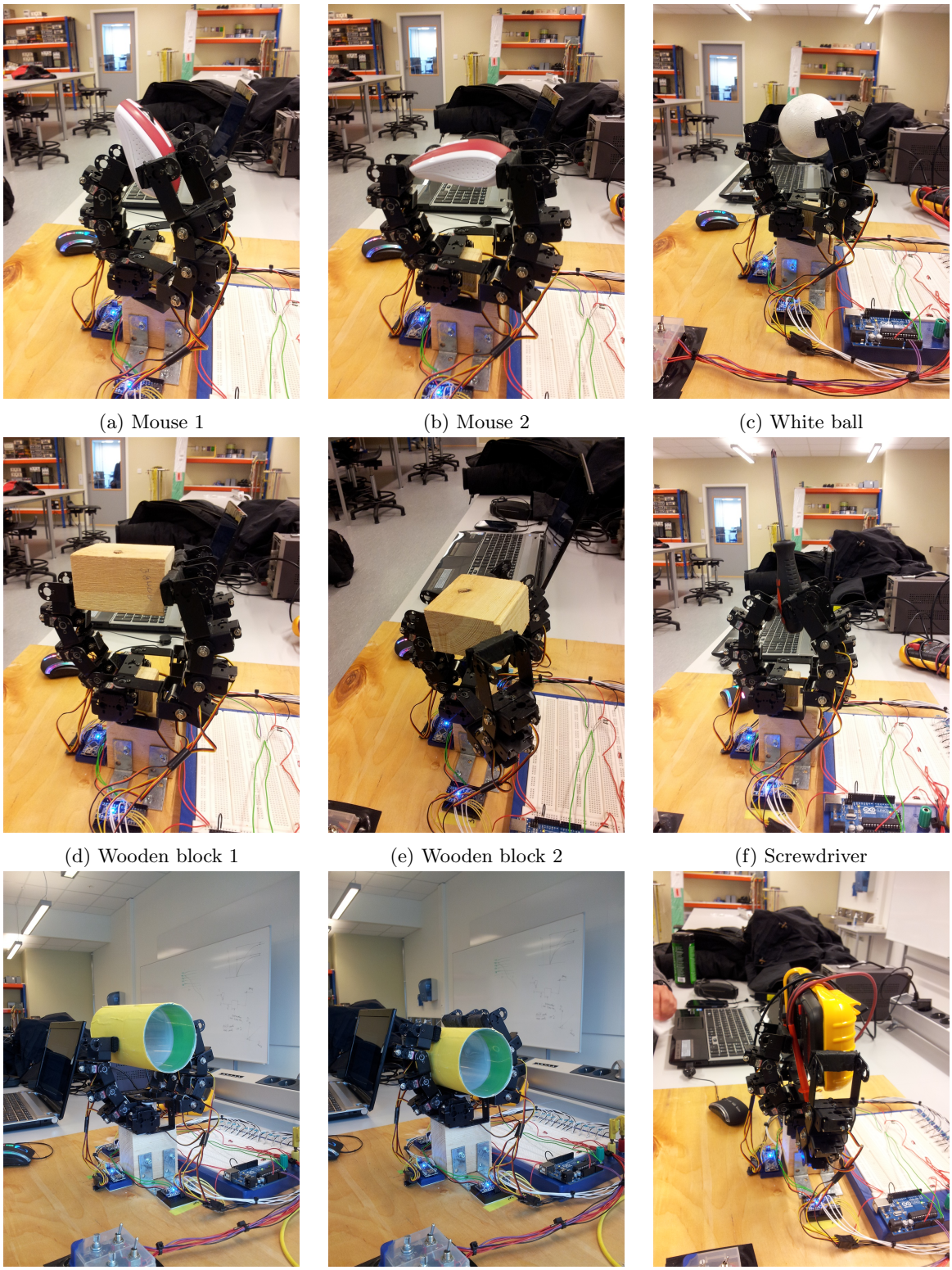
$$S = \begin{bmatrix} -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \\ 0 & -1.6 \\ -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \\ 0 & 1.6 \\ -0.7 & 0 \\ -0.2 & 0 \\ -0.1 & 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\end{matrix}}\right\} Thumb \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} Finger1 \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} Finger2 \end{matrix}$$

To describe the behavior of the joints for simulation purposes, the measurements lead up to the following DH-tables

DH-Table, thumb				
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	3.2	0	θ_1
2	0	3.2	0	θ_2
3	0	3.2	0	θ_3

DH-Table, finger				
i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	1.5	θ_1
2	$\frac{\pi}{2}$	5.9	0	θ_2
3	0	3.2	0	θ_3
4	0	3.2	0	θ_4

Figure 6.1 shows how the modular gripper is able to grasp different objects using synergy.



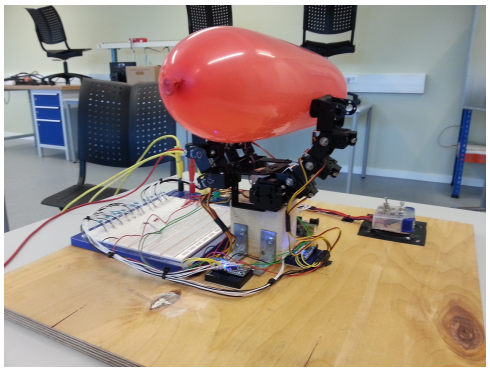
(a) Mouse 1 (b) Mouse 2 (c) White ball

(d) Wooden block 1 (e) Wooden block 2 (f) Screwdriver

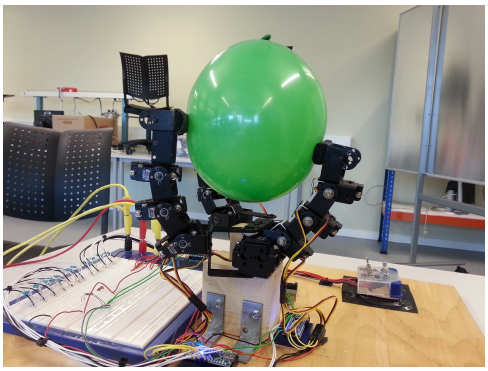
(g) Sylinder (h) Sylinder closed (i) Multimeter

Figure 6.1: Grasping different objects

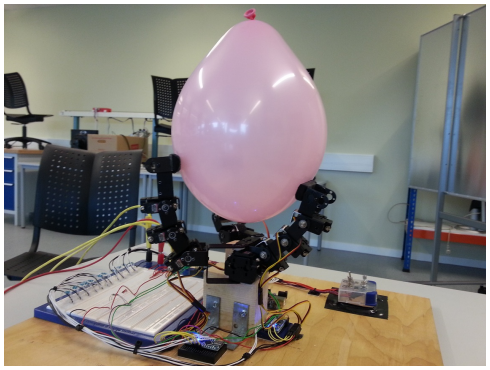
To demonstrate the flexibility provided by the load restrictions, an experiment was done with some balloons. The goal was simply to grasp the balloons without bursting them, and at the same time provide stable and flexible grasps. Figure 6.2 shows the results.



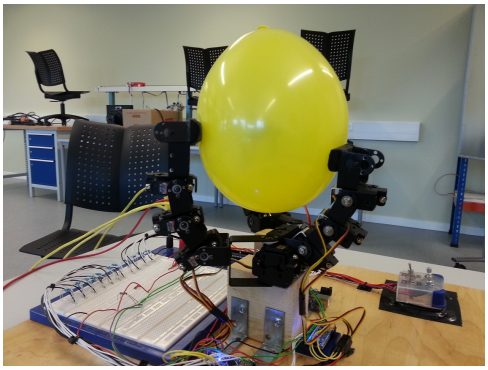
(a) Balloon 1



(b) Balloon 2



(c) Balloon 3



(d) Balloon 4

Figure 6.2: Grasping different balloons

Chapter 7

Discussion

7.1 Assembly

7.1.1 Modifying the hand

The modular hand used in this project is a design based on the Barret Hand made by Barret Technology Inc [1]. The hand was constructed in an earlier project by other students. This hand was chosen instead of constructing a new one since the project is more focused on the approach of *how to control* rather than what the topology of the gripper is, and due to a lot of missing knowledge at the beginning of the project, a lot of time was set aside to understanding and finding out how to work with the principles.

7.1.2 Controll circuit

The switches that controls the power to the Arduinos were added because it was suspected that the order of which the master or the slaves boots up was important. It was later discovered that this was probably not necessary, but the switches were not removed since they were not doing any harm.

7.2 Communication

7.2.1 I^2C

I^2C was chosen as communication protocol since it is relatively easy to set up. It also supports slaves having different addresses, which is great for this projects purpose. At first there were an intolerable amount of shaking on the servos, and I^2C seemed to be causing this, therefore, other communication options were explored. Regular serial communication (UART) through the RX/TX lines on the arduinos was one of these options. However, the master device used in this project is an Arduino Uno, and it only supports one serial line of communication, therfore we would need an Arduino Due or Arduino Mega, which supports up to three lines of communication. Then there could have been a line between each of the three Arduino Nanos and the master device.

After implementing the EasyTransfer library, the shaking issues seemed to be resolved.

7.3 Mapping the synergies

There were a lot of discussion on how to map synergies onto the robotic hand. An attempt was made using Santello's work and cross-mapping the synergies found in [3] onto the modular gripper. This could esentially be done by naming the joints on the gripper so they would correspond to the proximal phalanx, middle phalanx and distal phalanx on the human hand. The spread angle opening on the gripper could correspond to the abduction/adduction of the human fingers, but since the modular gripper used in this project does not resemble a human hand, it proved difficult to use this kind of approach. The synergy matrix that was developed using this method seemed illogical and produced strange movements, therefore, other methods were used.

The main goal of synergy is to be able to grasp a wide variety of objects, and producing stable grasps without using too many parameters to control the hand. Results from the grasping tests done in this project indicate that the synergies that was developed for the modular gripper are well suited for these types of hands.

7.4 Control

Originally, the software that controls the modular gripper had a very different design. The master device stored a synergy matrix internally, and the master decided which part of that matrix should be used for each slave. Values from the potentiometers were stored as an activation vector, and then sent out to each of the slaves. This is in theory a better design, as it allows for different synergies to be tested on the modular gripper without having to reprogram all the devices.

There were several reasons that the original control software was changed. The first design did not seem to work as expected, and it was very hard to debug to find out exactly what was wrong. Also, some of the servos broke during the project and had to be switched with other cheaper servos. This meant that the range that the servos operated on now varied from servo to servo. Some of the new servos even had an inverted movement than what was expected. This made it difficult to use the original intended design, without having to spend a lot of extra time and effort.

Since the project was nearing an end, the control program ended up being a simplified version of the original design, where the slaves handles the operating range for each servo internally. This is a less modular solution, but for the purpose of this project it is good enough since the main goal is to demonstrate how synergies can be implemented.

7.5 Simulation

7.5.1 Early attempts

Visualisation of the gripper turned out to be a big challenge, and a lot of different options were explored. OpenRAVE [12] was one of the first programs that were tested. In theory this program should be easy to use and can be a simple way to display graphics and simulate a robotic system. There were however a lot of issues, and the program was crashing for unknown reasons, and the necessary parts to control it from MatLab refused to compile with very unhelpful error messages. It was later pointed out that OpenRAVE has a lot of trouble running on Windows, and that it would run better in a Linux environment, but by then other solutions had already been found, and it was decided to spend time on those solutions instead of making more attempts with OpenRAVE.

Since GraspIt![13], another robotic simulator that was tested, was the foundation for OpenRAVE it was expected that it would share a lot of the same feature set. The hope was that it would be more stable, but allow similar testing scenarios, but the reality was that exacly the same problems were encountered. With fewer features and as many problems, it was ranked under OpenRAVE as an option.

The project largely being based on existing work with synergies, a third option was tested; SynGrasp[14], which is a toolbox for MatLAB[15] developed for displaying and simulating synergy in any hand. The toolbox did, as hoped, provide functions for drawing hands, defining synergies and simulating motions, amongst other useful features[16]. The toolbox turned out quite helpful throughout the project, providing a more intuitive understanding of how the theory lined up with practical use. What it was lacking, however, was real-time visualization of motion and placement. It was therefore decided that writing custom software instead would be the right way to go for this project.

7.5.2 Making a new simulator

Since a new visualization program was to be made, decisions had to be made regarding what platforms and libraries to use to facilitate the visualization/simulation. As a foundation for the engine, Java was chosen for it's cross-platform support, previous experience with the language and the availability of good documentation. There were two major libraries available providing direct access to the OpenGL API, JOGL and LWJGL. Previous experiences with JOGL and knowing that LWJGL is available for several

major platforms (Windows, Linux, OSX, Solaris and BSD) lead to LWJGL being chosen as the major display library.

It was fairly early decided that locking the visualization to only work for this gripper would be a large waste of effort, and it was therefore necessary to make the visualization-engine itself more accessible for more models. There are several scripting languages available for Java, but Lua was known in the group, and it was known amongst group members that a pure-java version of Lua was available, and it was therefore chosen as the language to provide dynamic functionality. Events that occured with mouse and keyboard would be propagated to the lua-core, and hooks were added to make it so that the lua engine was made capable of running the code to render models and move the camera.

A lot of the functionality in this simulator was left in a very basic state out of time-constraints, and some functions like serial communication were added as quick fixes with little work done to ensure their stability, only testing that they work for the needs of this project. Despite this, a lot of time was put into ensuring that the core of the engine remained expandable and flexible, so that it could be further developed later on.

Currently the simulator completely lacks a physics engine, which puts it at a disadvantage when compared to more established software like OpenRAVE. In return it is a much more consolidated package, relying less on arbitrary software to control it, instead running scripts internally. Due to the basic support for vectors and matrixes it was also fully capable of at least visualizing the actuation of synergies and processing it internally, which allowed the combination of some of the good features of Syngrasp's existence within Matlab and OpenRAVE's real time view.

Chapter 8

Conclusion

The experiments with grasping utilizing synergies gave very intuitive controls for the gripper, while maintaining ability to grasp every object that could be find to try, which indicates that using synergies is a very good way to control even modular hands. Despite difficulties with the communication protocol and current lack of flexilibity in the system, the experiments still show that modular robotic hands are very much a viable platform for synergic grasping, and it is an area that deserves more attention in the future, preferably with a module design geared towards the use of synergies and a more powerful master controller.

While the development of new software to visualize the gripper was time consuming and required a lot of extra work, it was rewarding in that it provided a relatively flexible solution in one single program, and with some further development it could become a very viable tool for simulation and visualization. At it's current state of development it still provided a clear model of the hand moving with the use of synergies.

References

[1] Barret Technology Inc [Website]; [cited 2013 April 18]. Available from: http://www.barrett.com/robot/index.htm.

[2] Oxforddictionaries [Website]; [cited 2013 April 18]. Available from: http://oxforddictionaries.com/definition/english/synergy.

[3] Santello M, Fl M, Soechting JF. F.: Postural hand synergies for tool use. The Journal of Neuroscience. 1998;.

[4] Ciocarlie MT. Low-Dimensional Robotic Grasping: Eigengrasp Subspaces and Optimized Underactuations. Columbia University; 2010.

[5] Prattichizzo D, Malvezzi M, Bicchi A. On motion and force controllability of grasping hands with postural synergies;.

[6] Smith LI. A tutorial on Principal Component Analysis; 2002. PDF/Tutorial.

[7] Gioioso G, Salvietti G, Malvezzi M, Prattichizzo D. An Object-Based Approach to Map Human Hand Synergies onto Robotic Hands with Dissimilar Kinematics;.

[8] Zhang PH. Lecture notes, Mechatronics; 2013. PDF/Powerpoint.

[9] Songho, OpenGL [Tutorials]. Song Ho Ahn; [cited 2013 May 2]. Available from: http://www.songho.ca/opengl/index.html.

[10] OpenGL Documentation [API Documentation]. OpenGL.org; [cited 2013 May 4].

[11] billporter.info [Website]; [cited 2013 April 29]. Available from: http://www.billporter.info/2011/05/30/easytransfer-arduino-library/.

[12] openrave.org [Website]; [cited 2013 May 02]. Available from: http://openrave.org/.

[13] GraspIt! [Website]; [cited 2013 May 02]. Available from: http://www.cs.columbia.edu/~cmatei/graspit/.

[14] SynGrasp [Website]; [cited 2013 May 02]. Available from: http://sirslab.dii.unisi.it/syngrasp/.

[15] MatLAB [Website]; [cited 2013 May 02]. Available from: http://www.mathworks.se/products/matlab/.

[16] Malvezzi M, Gioioso G, Salvietti G, Prattichizzo D. SynGrasp MATLAB Toolbox User Guide 1.2; 2012.

[17] Gabiccini M, Bicchi A. On the Role of Hand Synergies in the Optimal Choice of Grasping Forces;.

[18] N/A. Modular Grasping Hand; 2012. Student project. Høgskolen i Ålesund.

[19] Connections I2C [Tutorial]. hacknmod.com; [cited 2013 April 18]. Available from: http://hacknmod.com/hack/how-to-connect-multiple-arduino-microcontrollers-using-i2c/.

Appendix

	Path	***Description***
A	ControlSoftware/Arduino Sketches	Arduino source code for the controllers
B	ControlSoftware/Libraries	The libraries used by the arduino source code
C	Visualizer (32bit) Visualizer (64bit)	Visualizing software with scripts, 32/64 bit
D	src/RobotSimulator	NetBeans project folder with source code for the visualization engine
E	lib	Java libraries used in the development of the simulator
F	report.pdf	A digital copy of the report
G	schematics.jpg	Schematics for connecting the arduinos, for communication and control of the gripper