



Designing, implementing and integrating a controller for the MRI-compatible robotic breast biopsy system

M. (Mohamed) Essam Mohamed Kassem Abdelaziz

MSc Report

Committee : Prof.dr.ir. S. Stramigioli V. Groenhuis, MSc

> Dr.ir. B. ten Haken Dr. F.J. Siepel

> > August 2016

037RAM2016 Robotics and Mechatronics EE-Math-CS University of Twente P.O. Box 217 7500 AE Enschede The Netherlands

UNIVERSITY OF TWENTE.



Abstract

One of the breast cancer screening and diagnosis methods is MRI-guided breast biopsy. Current techniques have limitations in accuracy and efficiency, which may cause lesions to be missed. In order to tackle this problem, a pneumatically actuated 5 DOF MRI-compatible robot is integrated with an electronic valve manifold and a graphical user interface to target fish oil capsules (mimicking lesions) in breast phantoms inside a 0.25T MRI scanner. Measurements with the scanner have shown that the robot without needle has no measurable influence on MRI scans. When equipped with an off-the-shelf titanium needle, artifacts are present in the vicinity of the needle when inserted into tissue. Preliminary needle positioning measurements show that the accuracy is in the order of (4.7 - 7.3 mm) and that there is a decrease in the time of the MRI-guided localization procedure.

Contents

1	Intr	roduction 1					
	1.1	Problem Statement					
	1.2	Hypothesis					
	1.3	Sketch of the proposed solution					
	1.4	Experiments and evaluations					
	1.5	Backg	round	1			
		1.5.1	Breast imaging techniques	1			
		1.5.2	Magnetic Resonance Imaging (MRI) Working Principles	2			
		1.5.3	Existing solutions for MRI-guided breast biopsy	3			
		1.5.4	MRI-guided biopsy procedure	5			
	1.6	Prior V	Work (MRI-compatible robotic biopsy systems)	5			
2	MR	l-comp	atible robotic breast biopsy system	6			
	2.1	Storm	ram 1	6			
	2.2	Storm	ram 2	7			
		2.2.1	Parts/Components	8			
			2.2.1.1 Base	8			
			2.2.1.2 Actuator	8			
			2.2.1.3 Needle Holder	9			
			2.2.1.4 Stormram 1 vs Stormram 2:	9			
		2.2.2	Inverse Position Kinematics	10			
		2.2.3	Forward Position Kinematics	18			
		2.2.4	Robot Kinematic Constraints	23			
			2.2.4.1 Lengths constraints	23			
			2.2.4.2 Central shaft and sockets constraint	24			
			2.2.4.3 Sockets constraint	30			
			2.2.4.4 Ball joint (Rack) and base constraint	31			
			2.2.4.5 Colliding Racks Constraint	33			
			2.2.4.6 Rack and NeedleHolder Constraint	35			
		2.2.5	Robot Workspace	36			
3	Pne	umatic	Distributors	38			
	3.1	.1 General Overview					
	3.2	Pressu	re Distribution Modules	38			
		3.2.1 DC motor controlled pneumatic distributor (double pole double throw (DPDT) switches controlled) (V.Groenhuis and S.Stramigioli, 2016) 38					
		3.2.2	Solenoid valves pneumatic distributor (Computer-controlled)	40			

vi	Desią	gning, implementing and integrating a controller for the MRI-compatible robotic bre biopsy syste	east em					
		3.2.3 Pneumatic Directional Control Valves	41					
		3.2.4 Manifold Electronics	43					
		3.2.5 Detailed Overview	46					
4	Stor	rmram 2 Computer Control	48					
	4.1	Control Approach	48					
		4.1.1 General Overview	48					
		4.1.2 Pneumatic Linear Stepper Motor Control	49					
		4.1.3 Inverse Position Kinematics and PWM Control Integration	50					
	4.2	Stormram 2's Control Interface	53					
		4.2.1 Software & Hardware Interface	53					
		4.2.2 Graphical User Interface Control Panel	55					
5	Exp	Experiments and Results						
Ū	r 5.1	Experimental Setup	62					
	5.2	Experiment Preparations	63					
	5.3	Experimental Procedure	65					
	5.4	Experiments, Results and Discussion	69					
		5.4.1 Further Analysis and Discussion	75					
6	Conclusions and Future Work 77							
	6.1	Final Discussion and Conclusion	77					
	6.2	Future Work	77					
A	Appendix 1 79							
	A.1	InverseKinematics.m	79					
	A.2	ForwardKinematics.m	81					
	A.3	ThetaSolver.m	86					
	A.4	PCB Schematic	89					
	A.5	GUI - step by step	90					
	A.6	Differential Kinematics - Joystick Control	91					
Bi	bliog	graphy	93					

1 Introduction

Breast cancer is the most occurring invasive cancer among women, accounting for about 25% of all cases worldwide and 29% of all cases in the Netherlands (cijfersoverkanker, 2015). Early diagnosis is essential for successful treatment (Lu et al., 2009). An extensive screening program is available in many countries such as in The Netherlands (Fracheboud et al., 2001). When a lesion (a suspicious abnormality) is found in the breast, then it is necessary to perform a biopsy to extract tissue from the lesion for examination. This gives important information about the type of lesion, and whether further treatment (e.g. excision) is needed. Therefore, it is crucial that the biopsy is taken from the right site. In this research, Magnetic resonance imaging (MRI) is used to localize the lesion and guide the biopsy needle towards it.

1.1 Problem Statement

Most MRI-guided breast biopsies are successful, but in about 10% of the cases the lesion is missed (13.8% according to (Hauth et al., 2008), 5%-10% according to four interviewed experts radiologists). In most cases, a lesion miss can be detected on the scan after which the biopsy is immediately redone, at the cost of extra time and additional tissue damage. But sometimes the lesion miss is unnoticed, potentially resulting in false-negative diagnosis.

Efficiency can also be improved. The entire MRI-guided breast biopsy procedure takes about 45-55 minutes on average. If the procedure could be shortened by 5 minutes, by automatizing certain parts or reducing the number of missed lesions during breast biopsy, then more patients could be helped in the same time. So the technical problem statement is as follows: how can the efficiency and accuracy of MRI-guided breast biopsy be improved?

1.2 Hypothesis

The hypothesis is that a computer-controlled MRI-compatible robotic breast biopsy system **makes** MRI-guided breast biopsy more accurate and efficient.

1.3 Sketch of the proposed solution

The MRI-compatible robot is placed inside the MRI scanner with a breast phantom positioned in front of it. A pressure distributor is assembled and placed outside the MRI scanner. The distributor is connected to the robot through long pneumatic tubes (5m). The aforementioned components are integrated with a graphical user interface GUI. The operator has control of the device through the GUI. Scans are made using the MRI and the lesion is localized. After localizing the lesion, the operator actuates the robot using the GUI and the needle moves towards the lesion. After targeting the lesion, the radiologist substitutes the needle in the robot with the biopsy device to take breast tissue samples.

1.4 Experiments and evaluations

Experimental setups are created to practically test the computer-controlled MRI-guided biopsy system. Phantoms filled with fish oil capsules and with different stiffnesses are utilised. The accuracy and efficiency of the computer-controlled MRI guided biopsy are measured (quantit-atively if possible, qualitatively otherwise), and finally discussed.

1.5 Background

1.5.1 Breast imaging techniques

There are several ways of detecting lesions inside the breast. Some lesions can be detected by palpation (examining the body using one's hands), but it's not always possible to do so. Other

ways include the utilisation of medical imaging techniques such as mammography, ultrasonography and MRI-imaging. The main goal of these imaging techniques/modalities is to **detect** and **localize** lesions and to investigate further the tissue structure inside the breast. The imaging techniques are described further:

- Mammography 1.1 (a), is a cheap and simple imaging technique. It basically uses the X-ray imaging modality to produce flattened, 2D images of the breast. The sensitivity of this technique ranges from 83% to 95%, and specifity from 94% to 99% for the general population (Lee et al., 2009).
- Ultrasonography 1.1 (b), is an ultrasound-based imaging technique, allowing to make2dimensional sections of the breast. It is sometimes used to confirm the location of a lesion seen on the mammogram, or to screen high-risk patients in addition to mammography.
- MRI imaging 1.1 (c), is the most advanced technique. It makes three-dimensional scans of the breast with high resolution and detail, and uses no harmful radiation (as X-ray does). A MRI scanner is a huge and costly device, compared with mammography and ultrasonography equipment. The sensitivity is very good (90%-99%), but specificity is rather low: values from 37% to 81% are mentioned in literature (An et al., 2013).







(b)



(c)

Figure 1.1: (a) Mammography, (b) Ultrasonography, (c) MRI

1.5.2 Magnetic Resonance Imaging (MRI) Working Principles

The basic working principle of the MRI-scanner is described below:

- MRI use protons, which are abundant in the human body
- All protons spin creating a small magnetic charge
- When a strong magnetic field is introduced, as in the case of the MRI machine, the protons align with that field

- The MRI technician / radiologists introduces a radiofrequency pulse that disrupts the proton and forces it into either a 90 degree or 180 degree realignment with the static magnetic field
- Since the radio-frequency pulse pushed the proton against it's nature, once this pulse is turned off, the protons realign with the magnetic field, releasing electromagnetic energy along the way.
- The MRI is able to detect this energy and is able to differentiate various tissues based on how quickly they release energy after the pulse is turned off



Figure 1.2: How MRI works

1.5.3 Existing solutions for MRI-guided breast biopsy

In this section, a few existing systems for performing MRI-guided breast biopsy are described. A complete system consists of a patient rest, a breast coil, a breast fixation system, a needle positioning grid (or other needle guidance system), a breast biopsy device (gun) and a software suite. Some parts such as the biopsy needle or gun can be made by different vendors, as long as they are compatible.

4





Figure 1.3: MRI-guided breast biopsy solutions: (a) Machet MICS, (b) Invivo Sentinelle, (c) Philips Mammo Trak, (d) Noras biopsy unit

The Machnet MICS system (figure 1.3 (a)) has a biopsy/fixation unit that can rotate all around the breast, allowing to perform the biopsy from any side. A fine-spaced, rigid grid system allows to insert a biopsy needle (figure 1.4 (a)) with minimal error (1-2mm) in the breast. Some regions (close to chest wall or around nipple) are difficult to reach with this system, which is an important disadvantage. Hologic developed the ATEC vacuum-assisted breast biopsy system (figure 1.4 (b)). This device is easy to operate and allows to take multiple biopsies in one run, so it is frequently used by radiologists. Invivo Corporation, a Philips Healthcare business, developed the Invivo Breast Coil, and acquired the Sentinelle line of breast coils (figure 1.3 (b)). Philips itself developed the Mammotrak (figure 1.3 (c)). Noras, a Siemens Healthcare business, developed complete MRI-guided breast biopsy solutions (figure 1.3 (d)), but also biopsy units that can be used with Invivo systems.



Figure 1.4: Breast biopsy devices: (a)Core biopsy needle (b)Vacuum-assisted biopsy device

From interviews with radiologists, it is learned that MRI-guided solutions from Invivo and Noras are currently most used in hospitals, in combination with the Hologic ATEC vacuum-assisted biopsy device.

1.5.4 MRI-guided biopsy procedure

The current MRI-guided biopsy procedure is as follows:

- 1. Patient lies down on table, breast is put through hole in table.
- 2. Breast is fixated by compressing it between two plates.
- 3. Table with patient moves into the MRI scanner and a first scan is made.
- 4. Contrast is applied and a scan is made.
- 5. Radiologist examines scans and pinpoints the location of the lesion.
- 6. Computer calculates grid coordinates and depth of the lesion.
- 7. Table comes back out of the scanner, breast is anaesthetized.
- 8. Radiologist inserts stylet plus needle guide in correct hole at the right depth.
- 9. Inner stylet is removed and obturator is inserted inside the needle guide.
- 10. Table slides back into MRI scanner and a new scan is made.
- 11. Radiologist validates obtuator position with respect to lesion. If not correct, then repeat from step 5 or 8.
- 12. Radiologist removes obturator and inserts vacuum-assisted biopsy device.
- 13. 6-12 tissue samples are extracted with biopsy device.
- 14. A new scan is made to verify that the lesion has been accurately sampled. If not, then repeat from step 5 or 8.
- 15. Biopsy device is removed, and a biopsy clip marker is inserted for future reference.
- 16. A last scan is made to get the location of the clip marker with respect to the lesion.
- 17. Needle guide and plates are removed, wound is cleansed and patient leaves the table.

1.6 Prior Work (MRI-compatible robotic biopsy systems)

Different MRI-compatible biopsy robots have been developed. (Su et al., 2012) developed different piezo/pneumatic-driven robots for prostate biopsy, of which the control electronics are placed in a shielded box inside the MRI room. (Yang et al., 2014) developed a piezo/pneumatic robotic system for breast biopsy. In the case of (T. et al., 2004), the device is actuated by means of telescoping shafts using ultrasonic motors. The operator has control over the device through a graphical user interface (GUI). Even though the aforementioned devices have shown potential, they also have serious drawbacks such as high complexity and slow operation and thus are not yet used in practice. In the next chapter, two MRI-compatible robotic breast biopsy systems are presented.

2 MRI-compatible robotic breast biopsy system

In this chapter, two functional prototypes of MRI-compatible robotic breast biopsy systems **Stormram 1** and **Stormram 2**, are presented. For Stormram 1, a brief overview of the robot's construction and different parts is given. This is followed by a detailed overview of Stormram 2's construction and parts. In addition, a brief comparison between both prototypes is presented.

In this thesis, the controller is designed, implemented and integrated for stormram 2. Therefore, the inverse and forward position kinematics of Stormram 2 are derived. The kinematic model describes the static relationships between the linear stepper actuators and the Cartesian position and orientation of the end-effector. Furthermore, Stormram 2's constraints are thoroughly discussed and analysed. Lastly, the reachable workspace ¹ of Stormram 2's end-effector is investigated.

2.1 Stormram 1

6

Stormram 1 is a 7-DOF (Degree Of Freedom) MRI-compatible biopsy robot designed and built using 7 linear pneumatic stepper motors as actuators (made out of acetal/Delrin). The kinematic design consists of a 6-DOF hexapod (also known as a Stewart platform - see figure 2.1(a)), plus a single DOF needle insertion mechanism on top of the platform as the end-effector (see figure 2.1(b)).



Figure 2.1: (a) Stewart Platform, (b) **Stormram 1**, MRI-compatible breast biopsy robot driven by pneumatic linear stepper motors

The pneumatic linear stepper motors work with three toothed pistons which can move up and down individually, acting on a rack or gear. As illustrated in figure 2.2, the rack is locked in its current position by the middle piston (T2). When the right piston (T3) is moved up and the middle one retracted, then the rack moves one step to the left. By moving the three pistons (T1, T2 and T3) in 3-phase fashion, the position of the rack or gear can be controlled. Note: In order for the rack to move left and right, there should be an offset between the teeth and the rack as illustrated in figure 2.2.

¹workspace is the set of points that can be reached by the robot's end-effector

Stormram 1 consists of 12 3D-printed passive ball joints. They are printed with the Objet Eden 250 printer in Vero transparent material and each joint consists of a ball and a socket. Out of the 12 joints, 6 are connected to the base, with a diameter of 40 mm and the remaining 6 joints are connected to the platform with a diameter of 15 mm.

The 40 mm joint connected to the base has a hole all the way through the ball for the rack, and also pin and groove to eliminate the rudimentatry degree of freedom of the link between both ball joints, turning it into a universal joint. This way, the orientation of the stepper motor can be fixed to avoid collisions between the stepper motors of adjacent legs of the Stewart platforms.



Figure 2.2: Labeled SOLIDWORKS assembly of **Stormram 1** showing the different components, labeled sketch of the inside of the pneumatic linear stepper actuator and the positions of both the groove and pin in the 40 mm ball joint. Note: This Solidworks assembly doesn't include the single DOF needle insertion mechanism

2.2 Stormram 2

Stormram 2 is the second functional prototype of MRI-compatible robotic breast biopsy systems. It is a five-link parallel manipulator.



Figure 2.3: (a) **Stormram 2**, MRI-compatible breast biopsy robot driven by pneumatic linear stepper motors, (b) **Stormram 2** with the MRI-compatible needles and needle holder

2.2.1 Parts/Components

It consists of mainly: one base, five ball joints with integrated pneumatic linear stepper motors, needle holder and needle. The design and implementation of the different parts, motors and joints are described in this section.



Figure 2.4: Labeled disassembled SOLIDWORKS model of **Stormram 2** showing the different components with an inside look of the ball joint (actuator)

2.2.1.1 Base

The base is 3D printed in PLA (Polylactic acid or polylactide). It has five mounting points for 45 mm ball joints, which consist of a 3D-printed socket and a 45mm ball with an integrated pneumatic linear stepper motor.

2.2.1.2 Actuator

The Stormram 2 actuator is a pneumatic linear stepper motor. Being a miniaturized version of Stormram 1's actuator design, it is embedded inside a 45mm ball joint. The stepper motor's internal mechanism (see Figure 2.4) consists of laser-cut acetal parts, following the design principles stated in (V.Groenhuis and S.Stramigioli, 2016). The cylinder case consists of seven plates that are stacked, forming three cavities in which toothed pistons (green) can slide, sealed by silicone seals.

By pressurizing either chamber, the piston is pushed to the other side, engaging or disengaging a rack. The bore's cross-sectional area *A* is $12 \text{ mm} \times 5 \text{ mm} = 60 \text{ mm}^2$ and operating pressure *P* is 0.3 MPa, so the theoretical force *F* exerted by the pistons (ignoring friction losses) is $F = P \cdot A = 0.3 \cdot 10^6 \cdot 60 \cdot 10^{-6} = 18 \text{ N}$. The three pistons interact with a rack having teeth pitch '*P* = 3 mm and teeth depth '*D* = 3 mm. The interaction is by means of a wedge mechanism: a piston displacement of 3 mm causes a rack displacement of 1 mm, so the wedge factor is $\alpha = \frac{2D}{P}$. Assuming ideal transfer of work, the theoretical force exerted by the rack becomes $\alpha \cdot F = \frac{2\cdot3}{3} \cdot 18 = 36 \text{ N}$. The actual force is lower, due to friction losses in the sliding parts. The principle of operation of the whole motor is given part (b) of figure 2.5, similar to Stormram 1's actuator (see figure 2.2). When the green piston is pushed up and the yellow one retracted, the rack



Figure 2.5: (a) **Stormram 2**'s MRI-compatible ball joint with an integrated pneumatic linear stepper motor, (b) Principle of operation of the pneumatic linear stepper motor

moves one step to the right. This happens because the three piston's jaws are phased 120° apart. By pressurizing the six chambers with appropiate waveforms, the position of the rack can be controlled in steps of 1 mm.

2.2.1.3 Needle Holder

The needle holder consists of six pieces that are printed with the Stratasys Objet Eden 250 in FullCure720 material. The central shaft, connecting points *A*, *C* together, consists of two parts that are connected by a bayonet mount and accomodates a 12-gauge (2.1mm) needle.



Figure 2.6: Disassembled needle holder and the step by step procedure of assembling the sockets

Joint *A* and *C* are three-way ball joints, that consist of three parts, shown in figure 2.6. The ball part is enclosed by a pair of identical socket parts that are interlocked by a revolute joint inspired by the bayonet mount. Because each socket part features both a rim and a groove, these can revolute over more than 180° in the interlocked state. Each socket part is rigidly attached to the racks of the motors. The joint connecting rack 5 to the needle holder is a revolute joint, using a 3 mm screw as the pin.

2.2.1.4 Stormram 1 vs Stormram 2:

The main differences between Stormram 1 and Stormram 2 are: (1) Stormram 2's rack pitch is 3mm instead of 4mm (Stormram 1), resulting in smaller step size. (2) Stormram 2's actuators are miniaturized, into 45mm ball joints with a much smaller overall frame. (3) Stormram 2 is 5 DOF, on other hand, Stormram 1 is 7 DOF because it is a 6 DOF stewart platform with an additional DOF on top of it. (5) Stormram 2 uses a different needle: 2.1mm instead of \approx 5mm (although Stormram 2 has multiple needle holders, allowing 5mm needles as well. (6) Storm-

ram 1 is mainly laser-cut from acetal whereas Stormram 2 is mainly 3D-printed (but stepper motor mechanisms still laser-cut).

2.2.2 Inverse Position Kinematics

In this section, the inverse position kinematics of (**Stormram 2**) is derived. The basic idea here is to determine the lengths (denoted by L_1, L_2, L_3, L_4 and L_5) between the critical points of the needle holder (*A*, *B*, *C*) and the centre of the ball joints denoted by *B*1, *B*2, *B*3, *B*4 and *B*5). The linear stepper actuators are integrated within the spherical ball joints. Lengths L_1, L_2, L_3, L_4 and L_5 provide a desired end-effector position and orientation (end-effector is denoted by point *E*). **Stormram 2** consists of five linear stepper actuators and can therefore control the end-effector in five degrees of freedom, i.e. translation in *x*, *y*, *z*, rotation around *y* (denoted by γ) and rotation around *x* (denoted by α). So in short, the end-effector position and orientation (*x*, *y*, *z*, γ, α) are provided and the lengths (L_1, L_2, L_3, L_4 and L_5) are calculated. To illustrate the idea more clearly, **Stormram 2** is labeled and sketched out (as shown in figures 2.7, 2.8 and 2.9), and the equations are derived based on the given geometry, followed by the computation of the lengths using MATLAB and verification using the SOLIDWORKS model.



Sectional Side View

Figure 2.7: Labeled SOLIDWORKS assembly of **Stormram 2** showing all the critical points of the needle holder, end-effector (A, B, C and E), the centre of the ball joints (B1, B2, B3, B4 and B5) and the connections between the points in different views. Note: Point B is located on the line connecting points E, A and C. The actual joint connecting the rack 5 is situated at a lower point B'. Therefore, L_5 should be the length between B' and B5, however, to simplify the geometry of the robot, L_5 is approximated to the distance between points B and B5. This difference is compensated for in the software implementation of the inverse position kinematics



Figure 2.8: Labeled sketch of **Stormram 2**'s back view (a) showing all the critical points of the needle holder, end-effector (A, B, C and E), the centre of the ball joints (B1, B2, B3, B4 and B5) and the connections between the points. The connection between point B and B5 is not added to avoid confusion caused by crossing of multiple lines (b) showing the missing connection between B and B5, (c)-(e) showing the distances (L_1 , L_2 , L_3 , L_4 and L_5) between the needle holder points (A, B and C) and their corresponding joints (B1, B2, B3, B4 and B5)





The positions (*B*1, *B*2, *B*3, *B*4 and *B*5) of the centres of the ball joints, with respect to the **Robot Coordinate System** (Ψ^R) are given by:

$$\begin{bmatrix} x_{B1} \\ y_{B1} \\ z_{B1} \end{bmatrix} = \begin{bmatrix} 45.37 \\ 0 \\ -22.5 \end{bmatrix} \qquad \begin{bmatrix} x_{B2} \\ y_{B2} \\ z_{B2} \end{bmatrix} = \begin{bmatrix} -45.37 \\ 0 \\ -22.5 \end{bmatrix} \qquad \begin{bmatrix} x_{B3} \\ y_{B3} \\ z_{B3} \end{bmatrix} = \begin{bmatrix} 45.37 \\ 0 \\ 22.5 \end{bmatrix}$$
$$\begin{bmatrix} x_{B4} \\ y_{B4} \\ z_{B4} \end{bmatrix} = \begin{bmatrix} -45.37 \\ 0 \\ 22.5 \end{bmatrix} \qquad \begin{bmatrix} x_{B5} \\ y_{B5} \\ z_{B5} \end{bmatrix} = \begin{bmatrix} 0 \\ -9.08 \\ -63.95 \end{bmatrix}$$
(2.1)

The distance between two points (x_1, y_1, z_1) and (x_2, y_2, z_2) in 3 dimensional space, is given by: Distance = $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ (2.2)

Similarly, based on equation 2.2, the distance between the needle holder critical points $A = (x_A, y_A, z_A)$, $B = (x_B, y_B, z_B)$ and $C = (x_C, y_C, z_C)$ and their corresponding centers of the ball joints are obtained as shown below:

Distance between point *A* and ball joint center *B*3:

$$(x_A - x_{B3})^2 + (y_A - y_{B3})^2 + (z_A - z_{B3})^2 = (L_3)^2$$
(2.3)

Distance between point *A* and ball joint center *B*4:

$$(x_A - x_{B4})^2 + (y_A - y_{B4})^2 + (z_A - z_{B4})^2 = (L_4)^2$$
(2.4)

Distance between point *C* and ball joint center *B*1:

$$(x_C - x_{B1})^2 + (y_C - y_{B1})^2 + (z_C - z_{B1})^2 = (L_1)^2$$
(2.5)

Distance between point *C* and ball joint center *B*2:

$$(x_C - x_{B2})^2 + (y_C - y_{B2})^2 + (z_C - z_{B2})^2 = (L_2)^2$$
(2.6)

Distance between point *B* and ball joint center *B*5:

$$(x_B - x_{B5})^2 + (y_B - y_{B5})^2 + (z_B - z_{B5})^2 = (L_5)^2$$
(2.7)

where L_1 , L_2 , L_3 , L_4 and L_5 are the defined lengths between the critical points as stated earlier. Next, the mathematical relations between the end-effector (*E*) and the needle holder critical points (*A*, *B* and *C*) are derived. As mentioned above, the end-effector *E* position (x_E , y_E , z_E) and orientation (γ , α) are provided. The relation between the needle holder critical points (*A* and *C*) and the end-effector (*E*) is derived and illustrated in 2 basic steps (see figure 2.11). Namely, the link is rotated about the y-axis, followed by a rotation about the x-axis. Note that the only part of **Stormram 2** considered in the derivation in figure 2.8 is the connection between the 3 points (*A*, *C* and *E*) (see figure 2.10). Point *B* is neglected in this derivation.



Figure 2.10: Connection for the derivation of the relations between points (A, B and C)



Figure 2.11: 2 steps illustrating the derivation of the positions of the needle holder critical points $A = (x_A, y_A, z_A)$, $C = (x_C, y_C, z_C)$ from the provided end-effector position and orientation $E = (x_E, y_E \text{ and } z_E), \gamma, \alpha$

$$x_A = x_E - L_{AE}\sin(\gamma) \tag{2.8}$$

$$y_A = y_E + L_{AE}\cos(\gamma)\sin(\alpha) \tag{2.9}$$

$$z_A = z_E - L_{AE} \cos(\gamma) \cos(\alpha) \tag{2.10}$$

$$x_C = x_A - L_{AC}\sin(\gamma) \tag{2.11}$$

$$y_C = y_A + L_{AC}\cos(\gamma)\sin(\alpha) \tag{2.12}$$

$$z_C = z_A - L_{AC} \cos(\gamma) \cos(\alpha) \tag{2.13}$$

Next, the position of point *B* is derived. From the sketch below, point *B* lies between points *A* and *C*.



Figure 2.12: Point B is computed in terms of point A and C

In general, the position of a point (in this case *B*) on a line segment (*AC*) is given by:

$$B = \lambda C + (1 - \lambda)A \tag{2.14}$$

i.e.

$$x_B = \lambda x_C + (1 - \lambda) x_A \tag{2.15}$$

$$y_B = \lambda y_C + (1 - \lambda) y_A \tag{2.16}$$

$$z_B = \lambda z_C + (1 - \lambda) z_A \tag{2.17}$$

where λ is the ratio of the known length between points *A* and *B* (i.e. L_{AB}) to the known length between points *A* and *C* (i.e. L_{AC}). The lengths: $L_{AB} = 15.07$ mm, $L_{AC} = 45.833$ mm.

$$\lambda = \frac{L_{AB}}{L_{AC}} = \frac{15.07}{45.833} = 0.3288 \tag{2.18}$$

The following figure (2.16) summarises the algorithm of calculating all the positions of the critical needle holder points (*A*, *B* and *C*) based on a given/provided end-effector *E* position.

Next, the inverse kinematics algorithm is implemented in MATLAB and tested for three different cases 2

- **Case 1** (Positioning): $x_E = 15$, $y_E = 80$, $z_E = 90$, $\gamma = \alpha = 0(0^\circ)$
- **Case 2** (Positioning + Rotation About *y*): $x_E = 10$, $y_E = 65$, $z_E = 75$, $\gamma = \frac{\pi}{36}(5^\circ)$, $\alpha = 0(0^\circ)$
- **Case 3** (Positioning + Rotation About *x* and *y*): $x_E = 9$, $y_E = 62$, $z_E = 73$, $\gamma = \frac{\pi}{18}(10^\circ)$, $\alpha = \frac{-45\pi}{4}(-8^\circ)$

²The end-effector positions (x_E , y_E , z_E) and orientations (γ , α) are with respect to the **Robot Coordinate System** (Ψ^R)

This is followed by verifying the results using the SOLIDWORKS model, i.e. the lengths computed using the MATLAB script (IKM.m) are used to adjust the lengths in the SOLIDWORKS model and the end-effector position and orientation are compared with the set end-effector position and orientation. The MATLAB script consists of 2 basic steps, firstly, the algorithm deriving the positions of all critical points (see figure 2.16) is implemented, followed by solving the five equations (2.3, 2.4, 2.5, 2.6 and 2.7) to obtain the five unknown lengths (L_1, L_2, L_3, L_4 and L_5). Refer to appendix A.1 in the appendix for the complete commented MATLAB InverseKinematics.m. For each case, the MATLAB results are presented alongside labeled screenshots of the adjusted SOLIDWORKS model (*based on the newly computed lengths*). To begin with, the computed lengths for positioning the end-effector with no orienting [i.e. **case1**] are:



Figure 2.13: Case 1 : Positioning with no orientation Next, the computed lengths for positioning the end-effector and orienting it about the y-axis [i.e. **case 2**], are:

>> L1 = 79.6239 >> L2 = 80.2648 >> L3 = 77.3616 >> L4 = 82.5363 >> L5 = 96.4969



Figure 2.14: Case 2 : Positioning and orienting about y-axis Finally, the computed lengths for positioning the end-effector and orienting it in both axes (y and x) [i.e. **case 4**], are:

>>	L1	=	73.2796
>>	L2	=	59.9282
>>	L3	=	72.0642
>>	L4	=	69.7076
>>	L5	=	86.2679

Top View (Orientation Measurement) Rotation about y-axis



Figure 2.15: Case 3: Positioning and orienting about both y and x axes



Figure 2.16: Algorithm summarising the derivation process of the positions of all critical points from the provided end-effector position and orientation. (MATLAB implementation A.1)

2.2.3 Forward Position Kinematics

In this section, the forward position kinematics of **Stormram 2** is derived. The basic idea here is to determine the end-effector position based on given lengths L_1, L_2, L_3, L_4 and L_5 . In order to obtain the forward kinematic relations, a direct approach would be to simply utilise the equations obtained in the previous section (Inverse Position Kinematics) and reformulate them to obtain 5 equations in terms of the end-effector Cartesian position and orientation ($x_E, y_E, z_E, \gamma, \alpha$). This is done by substituting Equations (2.8)–(2.35) into Equations (2.3)–(2.7), resulting in five highly non-linear simultaneous equations as shown below:

$$\begin{aligned} (x_E - L_{AE}\sin(\gamma) - x_{B3})^2 + (y_E + L_{AE}\cos(\gamma)\cos(\alpha) - y_{B3})^2 + (z_E - L_{AE}\cos(\gamma)\cos(\alpha) - z_{B3})^2 &= (L_3)^2 \\ (x_E - L_{AE}\sin(\gamma) - x_{B4})^2 + (y_E + L_{AE}\cos(\gamma)\cos(\alpha) - y_{B4})^2 + (z_E - L_{AE}\cos(\gamma)\cos(\alpha) - z_{B4})^2 &= (L_4)^2 \\ (x_E - L_{AE}\sin(\gamma) - L_{AC}\sin(\gamma) - x_{B1})^2 + (y_E + L_{AE}\cos(\gamma)\cos(\alpha) + L_{AC}\cos(\gamma)\sin(\alpha) - y_{B1})^2 \\ &+ (z_E - L_{AE}\cos(\gamma)\cos(\alpha) - L_{AC}\cos(\gamma)\sin(\alpha) - z_{B1})^2 &= (L_1)^2 \\ (x_E - L_{AE}\sin(\gamma) - L_{AC}\sin(\gamma) - x_{B2})^2 + (y_E + L_{AE}\cos(\gamma)\cos(\alpha) + L_{AC}\cos(\gamma)\sin(\alpha) - y_{B2})^2 \\ &+ (z_E - L_{AE}\cos(\gamma)\cos(\alpha) - L_{AC}\cos(\gamma)\sin(\alpha) - z_{B2})^2 &= (L_2)^2 \\ (\lambda(x_E - L_{AE}\sin(\gamma)) + (1 - \lambda)(x_E - L_{AE}\sin(\gamma) - L_{AC}\sin(\gamma)) - x_{B5})^2 \\ &+ (\lambda(y_E + L_{AE}\cos(\gamma)\cos(\alpha)) + (1 - \lambda)(y_E + L_{AE}\cos(\gamma)\cos(\alpha) - L_{AC}\cos(\gamma)\sin(\alpha)) - y_{B5})^2 \\ &+ (\lambda(z_E - L_{AE}\cos(\gamma)\cos(\alpha)) + (1 - \lambda)(z_E - L_{AE}\cos(\gamma)\cos(\alpha) - L_{AC}\cos(\gamma)\sin(\alpha)) - z_{B5})^2 &= (L_5)^2 \end{aligned}$$

Solving **five** highly non-linear equations is computationally expensive and therefore a more efficient approach is required. This is achieved by introducing two new parameters θ_1 and θ_2 into the **Stormram 2**'s kinematic relations and solving 2 equations instead of the 5 derived above. The following figure (2.17) illustrates the location of both parameters. In addition, two intermediate parameters q_1 and q_2 and two intermediate coordinate systems Ψ^{B2} and Ψ^{B4} are defined in order to obtain the final equations, as presented later in this section.



Figure 2.17: Labeled sketch of **Stormram 2**'s back view: (a) showing all the critical points and the two newly added parameters θ_1 and θ_2 with a side view projection, (b) similar to (a) but with smaller θ_1 and θ_2 , (c) illustrating the two intermediate parameters q_1 and q_2 and the two intermediate coordinate systems Ψ^{B2} and Ψ^{B4} , (d)-(e) showing the distances (L_1, L_2, L_3, L_4 and L_5) between the needle holder points (A, BandC) and their corresponding joints ($B_1, B_2, B_3, B_4, and B_5$)

For further clarification:

- θ_1 is the angle triangle (B1 C B2) makes with the positive z-axis
- θ_2 is the angle triangle (B3 A B4) makes with the positive z-axis
- q_2 is the angle between lines B2 C and B2 B1
- q_4 is the angle between lines B4 A and B4 B3

To begin with, using the cosine rule q_2 and q_4 are obtained as follows:

$$(L_1)^2 = (L_2)^2 + 90.748^2 - 2 \cdot L_2 \cdot 90.748 \cdot \cos(q_2) \Rightarrow q_2 = \arccos\left(\frac{(L_2)^2 + 90.748^2 - (L_1)^2}{2 \cdot L_2 \cdot 90.748}\right) \quad (2.19)$$

$$(L_3)^2 = (L_4)^2 + 90.748^2 - 2 \cdot L_4 \cdot 90.748 \cdot \cos(q_4) \Rightarrow q_4 = \arccos\left(\frac{(L_4)^2 + 90.748^2 - (L_3)^2}{2 \cdot L_4 \cdot 90.748}\right) \quad (2.20)$$

Next, the position of critical point *A* is obtained with respect to the coordinate system Ψ^{B4} and the position of critical point *C* is obtained with respect to the coordinate system Ψ^{B2} :

$$x_A^{B4} = L_4 \cos(q_4)$$
(2.21)
$$x_C^{B2} = L_2 \cos(q_2)$$
(2.24)
$$y_A^{B4} = L_4 \sin(q_4) \sin(\theta_2)$$
(2.22)
$$y_C^{B2} = L_2 \sin(q_2) \sin(\theta_1)$$
(2.25)
$$x_C^{B4} = L_4 \sin(q_4) \sin(\theta_2)$$
(2.26)

$$z_A^{B4} = L_4 \sin(q_4) \cos(\theta_2)$$
 (2.23) $z_C^{B2} = L_2 \sin(q_2) \cos(\theta_1)$ (2.26)

The figure below illustrates clearly how Equations (2.21)–(2.26) are obtained:



Figure 2.18: Labeled sketch of **Stormram 2**'s back view (a) showing the projection of the distance L_2 onto the x - t plane (where t is a temporary axis), (b) showing the projection of the projected distance $L_2 \sin(q_2)$ on the t-axis onto the y - z plane, (c) showing the offsets between the coordinate systems $\Psi^R, \Psi^{B2}, \Psi^{B4}$ and their defined axes orientations.

As illustrated above, the positions of both critical points A and C are determined with respect to the coordinate systems Ψ^{B4} and Ψ^{B2} respectively. However, it is essential to determine the position of both points with respect to the Robot Coordinate System (Ψ^R). To obtain the position vectors with respect to Ψ^R , **Homogeneous matrices** are utilised. The homogeneous matrix is a convenient representation of combined transformations between coordinate systems, i.e. **rotation** of coordinate systems describing the projection of one coordinate system onto another and **translation** which is the offset between the origins of the coordinate systems. The general form of the 4 × 4 homogeneous matrix is given by:

$$H_{i}^{j} = \begin{bmatrix} R_{i}^{j} & o_{i}^{j} \\ \mathbf{0}^{T} & 1 \end{bmatrix} \quad o_{i}^{j} = \begin{bmatrix} o_{x_{i}}^{j} & o_{y_{i}}^{j} & o_{z_{i}}^{j} \end{bmatrix}^{T}$$
(2.27)

where R_i^j represents the 3 × 3 rotation matrix from coordinate system *i* to coordinate system *j*, o_i^j represents the 3 × 1 offset between the origins of the coordinate systems *i* and *j*.

-`@ - Notations:

- $o_i^j \Rightarrow$ origin of coordinate system *i* with respect to coordinate system *j*
- $p_x^j \Rightarrow$ arbitrary point x with respect to coordinate system j
- $R_i^j \Rightarrow$ Rotation matrix from coordinate system *i* to coordinate system *j*
- $H_i^j \Rightarrow$ Homogeneous matrix transformation from coordinate system *i* to coordinate system *j*

Part (c) of the figure above 2.18, illustrates the position and orientation of all three coordinate systems (Ψ^R , Ψ^{B2} , Ψ^{B4}) with respect to each other. Furthermore, the homogeneous matrix transformation from coordinate system Ψ^{B2} to Ψ^R is given by:

$$R_{B2}^{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad o_{B2}^{R} = \begin{bmatrix} -45.374 \\ 0 \\ -22.5 \end{bmatrix} \Rightarrow H_{B2}^{R} = \begin{bmatrix} 1 & 0 & 0 & -45.374 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -22.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
(2.28)

Similarly, the homogeneous matrix transformation from coordinate system Ψ^{B4} to Ψ^{R} is given by:

$$R_{B4}^{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad o_{B4}^{R} = \begin{bmatrix} -45.374 \\ 0 \\ 22.5 \end{bmatrix} \Rightarrow H_{B4}^{R} = \begin{bmatrix} 1 & 0 & 0 & -45.374 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 22.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
(2.29)

In both cases, the rotation matrices R_{B2}^R and R_{B4}^R are identity matrices. This is because, the x, y, z axes of all three coordinate systems are oriented in the same direction as illustrated in 2.18. The offset o_{B2}^R is the origin's position of coordinate system Ψ^{B2} with respect to the coordinate system Ψ^R (the same applies for o_{B4}^R). Furthermore, to satisfy the dimensions of the homogeneous matrix, the position vectors of both critical points A and C are re-written in the following manner (Equations (2.21)–(2.26)):

$$p_A^{B4} = \begin{bmatrix} x_A^{B4} & y_A^{B4} & z_A^{B4} & 1 \end{bmatrix}^T \qquad p_C^{B2} = \begin{bmatrix} x_C^{B2} & y_C^{B2} & z_C^{B2} & 1 \end{bmatrix}^T$$
(2.30)

The position of points A and C with respect to the Robot coordinate system Ψ^R is given by:

$$p_{A}^{R} = H_{B4}^{R} \cdot p_{A}^{B4} \Rightarrow \begin{bmatrix} x_{A}^{R} \\ y_{A}^{R} \\ z_{A}^{R} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -45.374 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 22.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{A}^{B4} \\ y_{A}^{B4} \\ z_{A}^{B4} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{A}^{B4} - 45.374 \\ y_{A}^{B4} \\ z_{A}^{B4} + 22.5 \\ 1 \end{bmatrix}$$

$$p_{C}^{R} = H_{B2}^{R} \cdot p_{C}^{B2} \Rightarrow \begin{bmatrix} x_{C}^{R} \\ y_{C}^{R} \\ z_{C}^{R} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -45.374 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -22.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{C}^{B2} \\ y_{C}^{B2} \\ z_{C}^{B2} \\ 1 \end{bmatrix} = \begin{bmatrix} x_{C}^{B2} - 45.374 \\ y_{C}^{B2} \\ z_{C}^{B2} \\ 1 \end{bmatrix}$$
(2.32)

With the positions of points *A* and *C* defined with respect to the same coordinate system Ψ^R , the position of point *B* is obtained in a similar manner to equation 2.14:

$$x_B^R = \lambda x_C^R + (1 - \lambda) x_A^R$$

$$y_B^R = \lambda y_C^R + (1 - \lambda) y_A^R$$

$$(2.33)$$

$$p_B^R = \begin{bmatrix} x_B & y_B & z_B & 1 \end{bmatrix}^T$$

$$(2.36)$$

$$z_B^R = \lambda z_C^R + (1 - \lambda) z_A^R$$

$$(2.35)$$

In order to evaluate the forward kinematics, two equations are formulated, namely: (1) The distance between points p_A^R and p_C^R and (2) The distance between points p_B^R and p_{B5}^R , where $p_{B5}^R = \begin{bmatrix} x_{B5} & y_{B5} & z_{B5} & 1 \end{bmatrix}^T = \begin{bmatrix} 0 & -9.08 & -63.95 & 1 \end{bmatrix}^T$. For (1): $|p_A^R - p_C^R| = L_{AC} \Rightarrow \sqrt{(x_A^R - x_C^R)^2 + (y_A^R - y_C^R)^2 + (z_A^R - z_C^R)^2 + (1 - 1)^2} = L_{AC}$ $(x_A^R - x_C^R)^2 + (y_A^R - y_C^R)^2 + (z_A^R - z_C^R)^2 = (L_{AC})^2$ (2.37)

For (2):

$$|p_B^R - p_{B5}^R| = L_5 \Rightarrow \sqrt{(x_B^R - x_{B5}^R)^2 + (y_B^R - y_{B5}^R)^2 + (z_B^R - z_{B5}^R)^2 + (1-1)^2} = L_5$$
$$(x_B^R - x_{B5}^R)^2 + (y_B^R - y_{B5}^R)^2 + (z_B^R - z_{B5}^R)^2 = (L_5)^2$$

where, x_A^{B4} , y_A^{B4} , z_A^{B4} , x_C^{B2} , y_C^{B2} and z_C^{B2} are obtained earlier in (Equations (2.21)–(2.26)), $L_AC = 45.833 \text{ mm}$ and L_5 is one of the five provided lengths L_1, L_2, L_3, L_4 and L_5 . Next, using the fsolve.m MATLAB script, the two simultaneous non-linear equations are solved for θ_1 and θ_2 given the five lengths and initial estimates of $\theta_{1,0}$ and $\theta_{2,0}$. Once θ_1 and θ_2 are evaluated, the positions of critical points *C*, *A* and *B* are calculated. Next, since the end-effector point *E* lies along the same line connecting points *C*, *B* and *A*, and the length between points *A* and *E* is known ($L_{AE} = 15.07 \text{ mm}$), the position of *E* can be evaluated in the same manner as point *B*.



Figure 2.19: Point *E* computed in terms of point *A* and *C*



Figure 2.20: Flowchart summarizing the forward kinematics algorithm. (MATLAB implementation A.2 and A.3), $\theta_{1,0}$ and $\theta_{2,0}$ are in rad

In this section, Stormram 2's constraints are thoroughly investigated.

2.2.4.1 Lengths constraints

The first constraint to be considered, is the range of values of the lengths L_1, L_2, L_3, L_4, L_5 . Part (a) of figure 2.21, illustrates the minimum values of the lengths, whereas part(b) of the same figures, illustrates the maximum values of the lengths (occurs when the end-stop comes in contact with the ball joint). Based on these two configurations, L_1, L_2, L_3, L_4 range from 50mm up to 106mm and L_5 ranges from 78mm up to 134mm.



Figure 2.21: (a) Stormram 2's configuration at the minimum lengths of L_1, L_2, L_3, L_4 and L_5 , (b)Stormram 2's configuration at the maximum lengths (racks' end stops hit the ball).

Prior to proceeding with the remaining constraints, as mentioned in section 2.2.1.3, the needle holder consists of six parts, a central shaft (two parts connected by bayonet mount) and 4 sockets. In the figure below (2.22), the needle holder is disassembled and divided into main parts, namely: **Section A** and **Section C**.



Figure 2.22: Disassembled Needle Holder

2.2.4.2 Central shaft and sockets constraint

The first constraint to be considered is the constraint in the motion between the ball joint part of the central shaft and the pair of identical sockets (**Sockets 1 & 2**) enclosing it. Figure 2.23 illustrate the idea of the constraint more clearly. At certain configurations, the ball joint part of the central shaft can no longer move relative to the sockets and they eventually collide. In figure 2.23, the colliding regions of both parts are indicated with two different colors (green -Colliding # 1) for the sockets and (red - Colliding # 2) for the central shaft.



Figure 2.23: Constraint in motion between central shaft and sockets

To ensure that the desired configuration of Stormram 2 doesn't lead to an unwanted collision between parts, the constraint is investigated further. Note that even though the constraint analysis is based on **Section A** of the needle holder, the same applies to **Section C**. To investigate this constraint further, Stormram 2 is moved to 4 extreme configurations whereby **Central Shaft A** collides with it's corresponding **Sockets 1 & 2** at 4 different points.



Figure 2.24: The 4 colliding points of Central Shaft A and it's corresponding socket Sockets 1 & 2

The following figures 2.25, 2.26, 2.27 and 2.28 illustrate the 4 colliding positions of Stormram 2. Each figure consists of a labeled isometric sectional view and a labeled sectional side view of

Stormram 2. In addition, a top (or) side sectional view and a sectional isometric view of Section A are also presented (section AA').



Figure 2.25: Collision Point # 1



Figure 2.26: Collision Point # 2





Figure 2.28: Collision Point # 4

The following equations define the labeled vectors in all 4 preceding figures. The reason behind drawing the vectors is to be explained later. To begin with, the position of the needle holder critical points *A* and *E* and the centre of ball joints *B*1 and *B*2 (all points with respect to the robot coordinate system) are given by:

$$p_{A}^{R} = \begin{bmatrix} x_{A}^{R} \\ y_{A}^{R} \\ z_{A}^{R} \end{bmatrix} \qquad p_{E}^{R} = \begin{bmatrix} x_{E}^{R} \\ y_{E}^{R} \\ z_{E}^{R} \end{bmatrix} \qquad p_{B3}^{R} = \begin{bmatrix} x_{B3}^{R} \\ y_{B3}^{R} \\ z_{B3}^{R} \end{bmatrix} = \begin{bmatrix} 45.37 \\ 0 \\ 22.5 \end{bmatrix} \qquad p_{B4}^{R} = \begin{bmatrix} x_{B4}^{R} \\ y_{B4}^{R} \\ z_{B4}^{R} \end{bmatrix} = \begin{bmatrix} -45.37 \\ 0 \\ 22.5 \end{bmatrix} \qquad (2.40)$$

Next, the vector between points *A* and *B*3, denoted by $\overline{AB3}$ and the vector between points *A* and *B*4 denoted by $\overline{AB4}$ are obtained in the following manner (subtracting the initial point p_A^R from the final point p_{B3}^R/p_{B4}^R):

$$\overrightarrow{A B3} = p_{B3}^{R} - p_{A}^{R} = \begin{bmatrix} x_{B3}^{R} \\ y_{B3}^{R} \\ z_{B3}^{R} \end{bmatrix} - \begin{bmatrix} x_{A}^{R} \\ y_{A}^{R} \\ z_{A}^{R} \end{bmatrix} = \begin{bmatrix} 45.37 - x_{A}^{R} \\ 0 - y_{a}^{R} \\ 22.5 - z_{A}^{R} \end{bmatrix}$$
(2.41)
$$\overrightarrow{A B4} = p_{B4}^{R} - p_{A}^{R} = \begin{bmatrix} x_{B4}^{R} \\ y_{B4}^{R} \\ z_{B4}^{R} \end{bmatrix} - \begin{bmatrix} x_{A}^{R} \\ y_{A}^{R} \\ z_{A}^{R} \end{bmatrix} = \begin{bmatrix} -45.37 - x_{A}^{R} \\ 0 - y_{A}^{R} \\ 22.5 - z_{A}^{R} \end{bmatrix}$$
(2.42)

As shown in all 4 figures, a (purple) plane is drawn. This plane consists of the set of points A, B3 and B4. Not to mention, the vectors defined above $\overrightarrow{A} \overrightarrow{B3}$ and $\overrightarrow{A} \overrightarrow{B4}$ also lie on the plane. Next, a vector normal (perpendicular) to the plane is computed. This is simply achieved by computing the cross-product of both vectors $\overrightarrow{A} \overrightarrow{B4}$ and $\overrightarrow{A} \overrightarrow{B3}$. The cross product is given by:

$$\overrightarrow{p_{43}^R} = \overrightarrow{A \ B4} \times \overrightarrow{A \ B3}$$
(2.43)





The figure above (2.29), illustrates different configurations of Stormram 2, i.e. different plane (*A*, *B*3, *B*4) configurations. In configurations 1 and 2, the vector connecting the needle critical points (*A*, *C* and *E*) is the same. On the other hand, the vector normal to the plane is different. Therefore, the plane's configuration varies according to the points' (*A*, *B*3, *B*4) location. Furthermore, in order to obtain a point on the normal vector $\overrightarrow{p_{43}^R}$, the vector equation can be rearranged in the following manner:

$$\overrightarrow{p_{43}^R} = p_{AN}^R - p_A^R \tag{2.44}$$

$$p_{AN}^R = \overrightarrow{p_{43}^R} + p_A^R \tag{2.45}$$

where p_{AN}^R is a point lying on the normal vector p_{43}^R and p_A^R is the critical needle holder point and the initial point of the vector $\overrightarrow{p_{43}^R}$. In order to check and avoid the collision between section A parts, a condition is required.



Using the cosine rule:

$$\cos(\theta) = \frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c} \Rightarrow \theta = \arccos \frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c}$$
(2.49)

The SOLIDWORKS model of Stormram 2 is investigated to obtain the maximum angle θ_{max} above which the parts collide. The angles $\theta_1, \theta_2, \theta_3$ and θ_4 for the 4 extreme conditions presented above are evaluated in the SOLIDWORKS model and are approximately 33°. Therefore the maximum value for θ in order to avoid collision in Section A is: $\theta_{max} = 33^{\circ}$

In short, based on the desired configuration, the angle between the vector normal to the plane and the vector along the needle (θ) is computed and compared with θ_{max} . If θ is greater than θ_{max} , then the Section A parts collide. As mentioned earlier, a similar analysis applies to Section C part as shown in the figure below (2.31) and the following equations (2.50 to 2.58).



Figure 2.31: Section C constraint

Equations:

$$p_{A}^{R} = \begin{bmatrix} x_{A}^{R} \\ y_{A}^{R} \\ z_{A}^{R} \end{bmatrix} \qquad p_{C}^{R} = \begin{bmatrix} x_{C}^{R} \\ y_{C}^{R} \\ z_{C}^{R} \end{bmatrix} \qquad p_{B3}^{R} = \begin{bmatrix} x_{B3}^{R} \\ y_{B3}^{R} \\ z_{B3}^{R} \end{bmatrix} = \begin{bmatrix} 45.37 \\ 0 \\ -22.5 \end{bmatrix} \qquad p_{B4}^{R} = \begin{bmatrix} x_{B4}^{R} \\ y_{B4}^{R} \\ z_{B4}^{R} \end{bmatrix} = \begin{bmatrix} -45.37 \\ 0 \\ -22.5 \end{bmatrix} \qquad (2.50)$$

$$\overrightarrow{C B1} = p_{B1}^{R} - p_{C}^{R} = \begin{bmatrix} x_{B1}^{R} \\ y_{B1}^{R} \\ z_{B1}^{R} \end{bmatrix} - \begin{bmatrix} x_{C}^{R} \\ y_{C}^{R} \\ z_{C}^{R} \end{bmatrix} = \begin{bmatrix} 45.37 - x_{C}^{R} \\ 0 - y_{C}^{R} \\ -22.5 - z_{C}^{R} \end{bmatrix}$$
(2.51)

$$\overrightarrow{C B2} = p_{B2}^{R} - p_{C}^{R} = \begin{bmatrix} x_{B2}^{R} \\ y_{B2}^{R} \\ z_{B2}^{R} \end{bmatrix} - \begin{bmatrix} x_{C}^{R} \\ y_{C}^{R} \\ z_{C}^{R} \end{bmatrix} = \begin{bmatrix} -45.37 - x_{C}^{R} \\ 0 - y_{C}^{R} \\ -22.5 - z_{C}^{R} \end{bmatrix}$$
(2.52)

$$\overrightarrow{p_{21}^R} = \overrightarrow{C \ B2} \times \overrightarrow{C \ B1}$$
(2.53)

$$p_{21}^R = p_{CN}^R - p_C^R \qquad p_{CN}^R = \overrightarrow{p_{21}^R} + p_C^R$$
(2.54)

$$a = \left| p_E^R - p_{AN}^R \right| = \sqrt{\left(x_E^R - x_{AN}^R \right)^2 + \left(y_E^R - y_{AN}^R \right)^2 + \left(z_E^R - z_{AN}^R \right)^2}$$
(2.55)

$$b = \left| p_A^R - p_C^R \right| = L_{AC} = 45.833 \,\mathrm{mm} \tag{2.56}$$

$$c = |p_A^R - p_{AN}^R| = \sqrt{(x_A^R - x_{AN}^R)^2 + (y_A^R - y_{AN}^R)^2 + (z_A^R - z_{AN}^R)^2}$$
(2.57)

Using the cosine rule:

$$\cos(\theta) = \frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c} \Rightarrow \theta = \arccos \frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c}$$
(2.58)

Similarly, the SOLIDWORKS model of Stormram 2 is investigated to obtain the maximum angle θ_{max} above which the parts collide. The maximum value for θ in this case is approximately 41.5°. Therefore the maximum value for θ in order to avoid collision in Section C is: $\theta_{max} = 41.5^{\circ}$

2.2.4.3 Sockets constraint

The second constraint to be considered is the constraint in the relative motion between the sockets of **Section A** and **Section C** (see figure 2.22). For this constraint, Section C's sockets 1 & 2 are considered. Note that the same analysis applies to sockets 1 & 2 of Section A.



Figure 2.32: Constraint in the relative motion between the sockets of **Section A** and **Section C**, (a) sockets are labeled, (b) socket 2 moves relative to a fixed socket 1, (c) socket 2 moves again relative to a fixed socket 1 with a zoom in of the two regions were both sockets can collide

In order to ensure that the two sockets do not collide, the following illustration is needed:



Figure 2.33: (a) Labeled schematic representation of the sockets of **Section C**, (b) Labeled schematic representation of the sockets of **Section A** (refer to figure 2.22 for Section A and C Parts)
The angles defined in the figure above (2.33): ϕ_A and ϕ_C are simply computed using the cosine rule, as shown below:

$$\cos(\phi_A) = \frac{(L_3)^2 + (L_4)^2 - 90.74^2}{2 \cdot L_3 \cdot L_4} \Rightarrow \phi_A = \arccos\left(\frac{(L_3)^2 + (L_4)^2 - 90.74^2}{2 \cdot L_3 \cdot L_4}\right)$$
(2.59)

$$\cos(\phi_C) = \frac{(L_1)^2 + (L_2)^2 - 90.74^2}{2 \cdot L_1 \cdot L_2} \Rightarrow \phi_C = \arccos\left(\frac{(L_1)^2 + (L_2)^2 - 90.74^2}{2 \cdot L_1 \cdot L_2}\right)$$
(2.60)

The SOLIDWORKS model of Stormram 2 is investigated to obtain the minimum angles ϕ_{Amin} and ϕ_{Cmin} below which the sockets collide. The minimum value for both ϕ_A and ϕ_C are 50°, i.e. $\phi_{Amin} = \phi_{Cmin} = 50^{\circ}$.

2.2.4.4 Ball joint (Rack) and base constraint

The third constraint to be considered is the constraint in the motion between the rack passing through the ball joints (*B*1, *B*2, *B*3, *B*4) and the base. For a better visualization of the constraint refer to the figure (2.34) below:



Figure 2.34: Different situations where the rack collides with the base (The red circles represent the points of collision)

To ensure that the desired configuration of Stormram 2 doesn't lead to an unwanted collision between the rack and the base, the constraint is investigated further. Note: A similar approach to 2.2.4.2 is undertaken to investigate this constraint.



Figure 2.35: Base planes and their corresponding normal vectors

shaft and sockets constraint (2.2.4.2). To begin with, the 4 ball joints (*B*1, *B*2, *B*3 and *B*4) are placed 45° to the horizontal base surface as shown in figure 2.35.

The **first step** to assure that the rack does not collide with the base is to consider a vector normal (perpendicular) to the plane (see figure 2.35 - denoted by $\overrightarrow{p_{B4n}^R}$ and $\overrightarrow{p_{B2n}^R}$)³ and passing through the central axis of the spherical ball joint as shown also in figure 2.35. When compared to the first constraint, this normal vector remains in place because the plane also doesn't change its configuration, i.e. the base and ball joints' centres are fixed. The **second step** is to select a point along this vector. The manually selected points with respect to the robot coordinate system Φ^R are:

$$p_{B1n}^{R} = \begin{bmatrix} x_{B1n}^{R} \\ y_{B1n}^{R} \\ z_{B1n}^{R} \end{bmatrix} = \begin{bmatrix} 10.58 \\ 29.42 \\ -22.5 \end{bmatrix} \qquad p_{B2n}^{R} = \begin{bmatrix} x_{B2n}^{R} \\ y_{B2n}^{R} \\ z_{B2n}^{R} \end{bmatrix} = \begin{bmatrix} -10.61 \\ 29.39 \\ -22.5 \end{bmatrix} \qquad p_{B3n}^{R} = \begin{bmatrix} x_{B3n}^{R} \\ y_{B3n}^{R} \\ z_{B3n}^{R} \end{bmatrix} = \begin{bmatrix} 10.58 \\ 29.42 \\ 22.5 \end{bmatrix}$$

$$p_{B4n}^{R} = \begin{bmatrix} x_{B4n}^{R} \\ y_{B4n}^{R} \\ z_{B4n}^{R} \end{bmatrix} = \begin{bmatrix} -10.61 \\ 29.39 \\ 22.5 \end{bmatrix} \qquad (2.61)$$

Note: $\overrightarrow{p_{B1n}^R} \dots \overrightarrow{p_{B4n}^R}$ are the vectors and $p_{B1n}^R \dots p_{B4n}^R$ are the corresponding points on the vector. Next (**third step**), compute the vector connecting the ball joint centre *B*1, *B*2, *B*3 and *B*4 with their respective needle holder critical points *A* (for *B*3 and *B*4) and *C* (for *B*1 and *B*2). This vector is nothing but the vector along the central axis of the rack itself (see figure 2.36).



Figure 2.36: (a) Solid lines drawn on the solidworks model, (b)Schematic Traingle, (c)3D cone along which the rack can move

The **final** step is to connect the point (p_{B4n}^R) on the vector normal to the plane $(\overrightarrow{p_{B4n}^R})$, forming a triangle. Using the cosine rule the angle τ is evaluated (refer to figure 2.36).

$$\cos(\tau) = \frac{(L_4)^2 + b^2 - a^2}{2 \cdot L_4 \cdot b} \Rightarrow \tau = \arccos\left(\frac{(L_4)^2 + b^2 - a^2}{2 \cdot L_4 \cdot b}\right)$$
(2.62)

³The same applies to ball joints *B*1 and *B*2, i.e. $\overrightarrow{p_{B1n}^R}$ and $\overrightarrow{p_{B3}^R}$.

Furthermore, in order to avoid any collision between the rack and base, the angle between the vector normal to the plane and the vector along the rack, should be less than a maximum value. After several investigations and attempts using the SOLIDWORKS model, the maximum value above which the parts collide ranges from approximately 34° to 45°. This is because of the variations in the design of the 4 base mounts where the ball joints are placed. To ensure any unwanted collisions, the minimum value of the range is considered as the maximum τ ($\tau_{max} = 34^\circ$). Part (c) of figure 2.36 shows a case where the rack avoids collision (case 1) and another case where the rack collides with the base (case 2). In brief, the vector connecting the centre of the ball joint and its corresponding point should lie within the cone (red).⁴

2.2.4.5 Colliding Racks Constraint

For certain configurations of stormam 2, the racks collide with one another, specifically (rack 5 - connected to ball joint *B*5 collides with both racks 1 - connected to ball joint *B*1 and 2 - connected to ball joint *B*2. This is clearly illustrated in the figure 2.37 below.



Figure 2.37: (a) Collision between rack 5 and rack 1 (b) Collision between rack 5 and rack 2

To avoid this type of collision, a minimum distance is to be set between both racks. This is simply done by computing the shortest distance between two vectors and then comparing it with the minimum distance. If the shortest distance is less than the minimum distance allowed, then the racks collide. The shortest distance is computed in the following manner:

The shortest distance between a line, l and a Point, p, is the length of the line that is perpendicular to l and goes from a point on l to the point p. The general idea of perpendicular lines also applies for two lines, except the line should be perpendicular to both. Let:

$$l_1 \Rightarrow \mathbf{r} = \mathbf{a} + s\mathbf{b} \tag{2.63}$$

$$l_2 \Rightarrow \mathbf{r} = \mathbf{c} + t\mathbf{d} \tag{2.64}$$

where **r** is the position vector of any point on a line; **a** and **c** are the position vector of a point that lies on l_1 and l_2 respectively; *s* and *t* are scalars; and **b** and **d** are vectors

⁴The angle between the central axis of the cone and any of it's edges is equivalent to 34° , the maximum value for τ as shown in part (c) of figure 2.36.

parallel to l_1 and l_2 respectively. A vector perpendicular to both l_1 and l_2 is the cross product of the two directional vectors:

$$\mathbf{n} = \mathbf{b} \times \mathbf{d} \tag{2.65}$$

and the unit vector, $\hat{\mathbf{n}}$ is \mathbf{n} divided by the magnitude \mathbf{n} :

$$\hat{\mathbf{n}} = \frac{\mathbf{b} \times \mathbf{d}}{\|\mathbf{b} \times \mathbf{d}\|} \tag{2.66}$$

If *p* is a point of l_1 and *q* is a point on l_2 , then it possible to find the vector **qp** by calculating the difference between $(\mathbf{a} + s\mathbf{b})$ and $(\mathbf{c} + t\mathbf{d})$ since these give the position vectors on the lines:

$$\mathbf{q}\mathbf{p} = \mathbf{a} - \mathbf{c} + s\mathbf{b} - t\mathbf{d} \tag{2.67}$$

The shortest distance between l_1 and l_2 is the projection of **QP** on *n*, which is independent of the free parameters *s* and *t*. That is, the dot product of **QP** and **n**, all divided by the magnitude of **n**. Since:

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|} \tag{2.68}$$

The shortest distance =
$$\mathbf{QP} \cdot \hat{\mathbf{n}}$$
 (2.69)

$$= (\mathbf{a} - \mathbf{c} + s\mathbf{b} - t\mathbf{d}) \cdot \frac{(\mathbf{b} \times \mathbf{d})}{\|\mathbf{b} \times \mathbf{d}\|}$$
(2.70)

$$=\frac{(\mathbf{a}-\mathbf{c}+s\mathbf{b}-t\mathbf{d})\cdot(\mathbf{b}\times\mathbf{d})}{\|\mathbf{b}\times\mathbf{d}\|}$$
(2.71)

$$=\frac{((\mathbf{a}-\mathbf{c})\cdot(\mathbf{b}\times\mathbf{d})+s\mathbf{b}\cdot(\mathbf{b}\times\mathbf{d})-t\mathbf{d}\cdot(\mathbf{b}\times\mathbf{d})}{\|\mathbf{b}\times\mathbf{d}\|}$$
(2.72)

Since $\mathbf{b} \cdot (\mathbf{b} \times \mathbf{d}) = 0$ (the projection of \mathbf{x} or \mathbf{y} on ($\mathbf{x} \times \mathbf{y}$) is actually always zero), there equation 2.72 can reduced to:

Shortest Distance =
$$\left| \frac{(\mathbf{a} - \mathbf{c}) \cdot (\mathbf{b} \times \mathbf{d})}{\|\mathbf{b} \times \mathbf{d}\|} \right|$$
 (2.73)

For rack 1 and rack 5 case, two lines are taken into consideration l_1 and l_5 . Equations 2.63 and 2.64 are:

$$l_{1} \Rightarrow \mathbf{r} = \mathbf{a} + s\mathbf{b} \Rightarrow \begin{vmatrix} r_{x1}^{R} \\ r_{y1}^{R} \\ r_{z1}^{R} \end{vmatrix} = \begin{vmatrix} x_{B1}^{R} \\ y_{B1}^{R} \\ z_{B1}^{R} \end{vmatrix} + s \begin{vmatrix} x_{C}^{R} - x_{B1}^{R} \\ y_{C}^{R} - y_{B1}^{R} \\ z_{C}^{R} - z_{B1}^{R} \end{vmatrix}$$
(2.74)

$$l_{5} \Rightarrow \mathbf{r} = \mathbf{c} + s\mathbf{d} \Rightarrow \begin{bmatrix} r_{x5}^{R} \\ r_{y5}^{R} \\ r_{z5}^{R} \end{bmatrix} = \begin{bmatrix} x_{B5}^{R} \\ y_{B5}^{R} \\ z_{B5}^{R} \end{bmatrix} + t \begin{bmatrix} x_{B}^{R} - x_{B5}^{R} \\ y_{B}^{R} - 10 - y_{B5}^{Ra} \\ z_{B}^{R} - z_{B5}^{R} \end{bmatrix}$$
(2.75)

For rack 2 and rack 5 case, two lines are taken into consideration l_2 and l_5 . Equations 2.63 and 2.64 are:

$$l_{2} \Rightarrow \mathbf{r} = \mathbf{a} + s\mathbf{b} \Rightarrow \begin{vmatrix} r_{x2}^{R} \\ r_{y2}^{R} \\ r_{z2}^{R} \end{vmatrix} = \begin{vmatrix} x_{B2}^{R} \\ y_{B2}^{R} \\ z_{B2}^{R} \end{vmatrix} + s \begin{vmatrix} x_{C}^{R} - x_{B2}^{R} \\ y_{C}^{R} - y_{B2}^{R} \\ z_{C}^{R} - z_{B2}^{R} \end{vmatrix}$$
(2.76)

$$l_{5} \Rightarrow \mathbf{r} = \mathbf{c} + s\mathbf{d} \Rightarrow \begin{bmatrix} r_{x5}^{R} \\ r_{y5}^{R} \\ r_{z5}^{R} \end{bmatrix} = \begin{bmatrix} x_{B5}^{R} \\ y_{B5}^{R} \\ z_{B5}^{R} \end{bmatrix} + t \begin{bmatrix} x_{B}^{R} - x_{B5}^{R} \\ y_{B}^{R} - 10 - y_{B5}^{R} \\ z_{B}^{R} - z_{B5}^{R} \end{bmatrix}$$
(2.77)

^{*a*}For a better approximation of the distance, the point B' is considered instead of B. B' is approximately 10 mm below point B. It is assumed that point B and B' are in the same x and z positions and $y_{B'}^R = y_B^R - 10$ (see part(b) of figure 2.38 for an illustration).



Figure 2.38: (a) The shortest distance between two racks is approximately 8mm, (b) Labeled SOLID-WORKS model of Stormram 2 showing all the critical points and the newly defined point B', along with a new length of L'_5

As shown in the figure (2.38), the shortest/minimum distance under which the links collide which the is: 8mm. However, based on several investigations using the SOLIDWORKS model, a factor of 1mm is added i.e. 9mm to ensure that racks 1,2 and 5 do not collide.

2.2.4.6 Rack and NeedleHolder Constraint

The final constraint to be considered, is caused by the collision of rack 5 and the needle holder at different configurations. This is clearly illustrated in figure 2.39. In order to avoid such collision, the angle defined in figure 2.39, β should always be greater than a minimum value. If the angle β is less than this minimum value, rack 5 and the needle holder collide as shown in the figure. Based on several attempts, the minimum angle is approximately 10°. Therefore, $\beta_{min} = 10^\circ$. Note: β° is not the same as the angle between rack 5 and central shaft. This constraint is a (good) approximation only.

$$\beta = \frac{(L_{AC})^2 + (L_5)^2 - b^2}{2 \cdot L_{AC} \cdot L_5}$$
(2.78)



Figure 2.39: (a) Labeled SOLIDWORKS model of Stormram 2 showing the critical points *C* and *A* along with the newly defined angle β , (b) Schematic representation of the triangle formed using points *A*, *C* and *B*5

2.2.5 Robot Workspace

In this final section of the chapter, the reachable space of the needle tip (point E) is investigated. To obtain the workspace, the forward kinematics investigated in section 2.2.3 and the constraint analysis in section 2.2.4 are both used. The idea in short can be summarized into the following steps:

1. Create an array with all the possible combinations of lengths $(L_1, L_2, L_3, L_4 \text{ and } L_5)$. To reduce the computation time, an interval of 10 mm is applied.

```
counter = 1;
for L1 = 50:10:100
    for L2 = 50:10:100
        for L3 = 50:10:100
            for L4 = 50:10:100
                for L5 = 80:10:130
                    result(1, counter) = L1;
                    result(2, counter) = L_{2};
                    result(3, counter) = L3;
                     result(4, counter) = L4;
                     result(5, counter) = L5;
                     counter = counter + 1;
                end
            end
        end
    end
end
```

- 2. Next, for each combination of lengths, the forward position kinematics is computed
- 3. Based on the constraints analysis in 2.2.4, a MATLAB function is checkPossibility.m is created. This function checks whether the lengths combination is safe from all constraints.

- 4. If the lengths combination, doesn't lead to the collision of any parts, then this length combination is considered and the end-effector position (x_E, y_E, z_E) is plotted (in a 3-dimensional point cloud). Otherwise, it is simply ignored.
- 5. Once all the successful points are plotted, an interpolation method is used namely: delaunayTriangulation.m to construct new data points within the range of the discrete set of known data points.
- 6. Lastly, using the results of the interpolation technique, the MATLAB function Tetramesh.m is used to visualize the workspace as shown in below:





Figure 2.40: Top and side view of Stormram 2's workspace

The workspace lies within a box sized $126 \times 104 \times 160$ mm. In the following chapter, pneumatic distributors are presented.

3 Pneumatic Distributors

In this chapter, two pneumatic distributors are described, namely: the DC-motors pneumatic distributor and the solenoid valves pneumatic distributor (also referred to as electronic valve manifold). A pneumatic distributor is a device that comprises of an array of controllable ports that can open and close pneumatic connections.

3.1 General Overview

As described earlier, 7 pneumatic linear stepper actuators drive the biopsy system **Stormram 1** and 5 pneumatic linear stepper actuators drive the biopsy system **Stormram 2**. Each stepper motor has three pistons, each having two pressure chambers. So a total of $7 \times 3 \times 2 = 42$ chambers in the case of Stormram 1 and $5 \times 3 \times 2 = 30$ chambers in the case of Stormram 2, need to be pressurized and decompressed in the right sequence, in order to drive the pneumatic linear stepper motors correctly. These pneumatic distributors are developed to control any of the current line of MRI-compatible robotic setups.

Controllable ports are needed to pressurize and decompress the 42 pneumatic lines. The ports can be controlled in different ways: manually, electronically with solenoid valves, or driven by DC motors.



Figure 3.4: Different ways of controlling the pneumatic ports

3.2 Pressure Distribution Modules

This section presents the development and construction of two fully-functional pneumatic distributors.

3.2.1 DC motor controlled pneumatic distributor (double pole double throw (DPDT) switches controlled) (V.Groenhuis and S.Stramigioli, 2016)

The first pneumatic distributor to be discussed is the DC motor controlled pneumatic distributor. DC motors are chosen because one DC motor can control all six pneumatic lines of one pneumatic linear stepper motor at once, reducing the total cost of the system (when compared to the pneumatic valves choices - one valve controls two pneumatic lines, therefore, 3 valves are required per pneumatic linear stepper motor, refer to 3.2.2 for a detailed explanation).



Figure 3.5: (a) Top view of the DC motor controlled pneumatic distributor (b) 8-shaped rubber ring

Each DC motor drives an 8-shaped rubber ring (figure 3.5). Air is delivered to the central hole (figure 3.6). The ring is symmetric, so that it does not tilt towards one side (which would be the case with an eccentric circle), and the pressure on the ring is equal everywhere (otherwise it would easily leak). The consequence is that two identical cycles are performed in one full rotation.



Figure 3.6: Schematic layout of rotating rubber ring (red) and orifices (blue). S: pressurized air supply; A, B, C, A', B', C':outputs to stepper motor. Dimensions not to scale.

The position of the ring controls which of the six other holes are pressurized, or being decompressed(connected with ambient pressure). See Figure 3.6 for the hole configuration. Pressure duty cycle is controlled by the distance from the center, and pressure phase by the angle from the vertical. A single piston is always pressurized from one side, so the total duty cycle for both chambers of one cylinder is 180°, of which the duty cycle of the chamber pressing on the toothed rack is 80° and the other one 100°.

The reason for this asymmetry is that it is not advantageous to have two pistons pushing on the rack at the same time, as this would cause unnecessary stress on the teeth. As the three piston phases are offset by 60° (one-third of one cycle), the overlap is now $\frac{20}{60} = 33\%$ which means that in 33% of the time, two pistons are pushing against the rack and in the remaining 67%, one piston pushes at the rack. In practice, the pistons

take some finite amount of time to slide from one side to the other, so the actual percentages are a bit lower. The DC motors are controlled with double pole double throw (DPDT) switches with spring return and a special center state, all mounted at the control panel (bottom part of figure 3.5(a)). The spring return ensures that the switch goes back to the center state when the switch is released. In the center state, the DC motor is short-circuited to brake it, so that it immediately stops. So the system is manually controlled, which makes testing relatively easy. A

master switch at the right side of the control panel enables or disables all eight motor switches. Next to it is a manual pneumatic ball valve to enable or shut off air pressure for the system.

3.2.2 Solenoid valves pneumatic distributor (Computer-controlled)

This section describes the development and construction of the solenoid valves pneumatic distributor (also refered to as the manifold). The main advantages of using the pneumatic directional control valves over the previously discussed DC motor approach are: (1) The pneumatic directional control valves have a higher airflow and (2) Easier to electrically control the solenoid pneumatic directional control valves when compared to the DC motor (to given positions).

The main function of the manifold is to control the pneumatic linear stepper motors of the robot based on the controller output. The manifold consists of the following main components (see figures 3.7 and 3.8 for the simplified layout of the manifold and the developed manifold respectively):



PNEUMATIC DIRECTIONAL CONTROL VALVES

Figure 3.7: Simplified layout of the electronic valve manifold



Figure 3.8: Electronic valve manifold

Pneumatic Directional Control Valves

- Arduino board with a LCD shield
- Transistor Circuit (Printed Circuit Board)
- Pressure Sensor

In section 3.2.3, a brief overview of Pneumatic Directional Control Valves is presented. The overview will basically summarise how the valves are presented symbolically and how do they operate. In section 3.2.4, the electrical (electronic) side of the manifold is thoroughly discussed, namely: the **Arduino** board and its main task of reading input signals form the computer and controlling the valves accordingly, the transistor circuit (Printed Circuit Board) and its ability to enable the **Arduino** to control the valves (loads with higher electrical requirements) and last but not least, the pressure monitoring of the pressure sensor and its integration with the LCD **Arduino** shield. Lastly, in section 3.2.5, a final overview of the electrical valve manifold is illustrated clearly alongside several design considerations.

3.2.3 Pneumatic Directional Control Valves

Pneumatic directional control valves ¹ start, stop or change the direction of flow in compressed air applications. The valves used in this manifold are chosen based on the following requirements:

- 1. The ability to computer-control the **up** and **down** motion of the pneumatic linear stepper motor's toothed pistons (pressurize and decompress the chamber). Each motor consists of 3 toothed pistons (3 chambers).
- 2. The ability to computer-control the **main supply** of compressed air.

Based on the above requirements, the valve chosen to satisfy the first requirement is the Single Solenoid 5/2 way valve (Valve type: PV5211-24VDC-1/8). In the case of the second requirement, the Double Solenoid 5/3 way valve (Valve type: PV5211-24VDC-1/8) is considered. To begin with, the manifold consists of 25 valves namely: 24 Single Solenoid 5/2 way valves and a Double Solenoid 5/3 way valve as illustrated in figure (3.7). Each pneumatic linear stepper motor in the robot is controlled using 3, 5/2 way valves. Hence, the 24 valves can control upto 8 pneumatic linear stepper motors. Note: As mentioned in the previous section, the manifold is created to not only control the MRI-compatible robot (**Stormram 2 - 5 linear stepper motors**) but also future robots (Stormram 3 and beyond) driven by up to 8 motors/24 valves. Therefore, the minimum number of valves required to control any of the above-mentioned robots is $7 \times 3 = 21$ valves. Not to mention, to make the manifold even and balanced an additional set of 3 valves is added ($\therefore 24$ valves).

As discussed earlier, the pneumatic linear stepper motors work with three toothed pistons which can move up and down individually, acting on a rack. As illustrated in figure 3.9, the rack is locked in its current position by the middle piston (T2). When the right piston (T3) is moved down and the middle one retracted, then the rack moves one step to the left. By moving the three pistons (T1, T2 and T3) in 3-phase fashion, the position of the rack or gear can be controlled. As shown in figure 3.9, each toothed piston is moved up and down using a single solenoid 5/2 way valve. The naming convention (nomenclature) of each single solenoid 5/2 way valve is the following: the number of the pneumatic linear stepper motor that the valve controls (i.e. a number from 1 to 8), followed by A,B or C depending on which toothed piston (T1, T2 or T3) the valve controls.

Concerning the double solenoid 5/3 way valve, the main tasks of this valve are to supply compressed air to the 24 valves, block/shut off compressed air from the 24 valves and the exhaust

¹From this point onward, Pneumatic Directional Control Valves are to be simply referred to as valves

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 42 biopsy system



Figure 3.10: Summary of the symbolic/graphical representation of pneumatic directional control valves

of any compressed air left out in the valves. This is also understood from its unique position when compared to the rest of the valves. The valve is denoted by "**MAIN AIR SUPPLY VALVE**". More details about the pneumatic connections of the valves are to be presented in section 3.2.5. For the remaining part of this section, an illustrative summary is provided explaining how the valves are graphically/symbolically presented and how do they operate.

The main advantage of having standard symbols for pneumatic components is that they allow pneumatic schematic diagrams to be read and understood by persons in many different countries, even when they don't speak the same language. The figure above (3.10) illustrates clearly what is meant by 5/2 and 5/3 way values and the valve actuation method being utilised solenoid). Furthemore, the here (i.e. symbolic representations of both valves: single solenoid 5/2 way valves and the double solenoid 5/3 way valve are illustrated clearly in the figure below 3.11.



Figure 3.9: Three-piston stepper motor mechanism controlled using 3 valves 1A, 1B and 1C



Figure 3.11: Symbolic representations of both the 5/2-way pneumatic solenoid valve and the 5/3-way pneumatic solenoid valve

The 5/2 way valve has 5 ports and 2 positions and can therefore switch between two circuits. On the other hand, the 5/3 way valve has 5 ports and 3 positions and can therefore switch

between three circuits. Furthemore, a solenoid valve is an electromechanically operated valve. The valve is controlled by an electric current through a solenoid. The solenoid opens/closes a small control valve which in turn drives the actual valve pneumatically. That is why the valve do not work if the system pressure drops below a certain threshold. Therefore, when the solenoid is actuated (i.e. current is flowing through the solenoid), the valve switches from one position to the other. In the case of the 5/2 way valve, when the solenoid is actuated (Energized State), the valve switches positions as illustrated clearly in figure 3.12. So depending on the position of the valve, the toothed piston moves up or down. In the current manifold setting, when the valve is in the De-energized state, the toothed piston is pushed down and when the valve is in the energized state, the toothed piston is pulled up.

For the 5/3 way valve, depending on the actuated solenoid (Energized State 1 or Energized State 2) the valves switches from the centre position to one of the other two positions. In the deenergized state, the main supply of compressed air is blocked/shut off. For the first energized state, compressed air is supplied to the 24 valves. Lastly, for the second energized state, any compressed air left out in the 24 valves is exhausted out.



Figure 3.12: Operation of both the 5/2-way pneumatic solenoid valve and the 5/3-way pneumatic solenoid valve

With this final illustration (3.12), the section concludes. In the following section 3.2.4, the electronic side of the manifold is presented and thoroughly discussed.

3.2.4 Manifold Electronics

In order to computer control the valves, an **Arduino MEGA 2560** board is used. An **Arduino** board is nothing but a kit which is comprised of an **Atmel 8–**, **16– or 32–bit AVR microcontroller** with complementary components that facilitate programming and incorporation into other circuits (i.e. pneumatic circuit). The reason why **Arduino MEGA**

2560 is used instead of its small counterparts (e.g.: **Arduino Uno** board) is because, the number of digital input/output pins in the **Arduino Uno** are insufficient to control all 26 valve solenoids $(24 (5/2 \text{ way valves}) + 2 (5/3 \text{ way valve}))^2$.

Furthermore, the **Arduino** can only provide 40mA at 5V on its digital pins. Most valves require more current and/or voltage to operate. In this case, the valves require 24 V (DC) and they have a power rating of 2.4 W. Consequently, the current drawn by the valves at an operating voltage of 24 V (DC) is:

$$P = VI \Rightarrow 2.8 \text{ W} = 24 \text{ V} \cdot I \Rightarrow I = 0.11667 \text{ A} = 1116.667 \text{ mA} \approx 120 \text{ mA}$$

In order to overcome this problem, a transistor can be used as a digital switch, enabling the **Arduino** to control loads with higher electrical requirements (valves). For the first trial, the NPN power Darlington transistor **TIP120**, which can switch up to 60V at 5A is used. A Darlington pair is two transistors that act as a single transistor but with a much higher current gain. Transistors have a characteristic called current gain. This is referred to as its hFE. The amount of current that can pass through the load in the circuit above when the transistor is turned on is:

Load current = input current × transistor gain (hFE)

In the case of a Darlington transistor, the current gain equals:

Load current = input current × transistor gain 1 (hFE t1) × transistor gain 2 (hFE t2)



BREADBOARD VIEW



²Arduino Uno board consists of 14 digital input/output pins, whereas Arduino MEGA 2560 consists of 54 digital input/output pins. Currently, outside the USA the Arduino MEGA 2560 is referred to as Genuino MEGA 2560

The current gain varies for different transistors and for a normal transistor this would typically be about 100. This would mean that the current available to drive the load would be 100 times larger than the input to the transistor, i.e. a tiny amount of current from a sensor, microcontroller (**Arduino**) or similar can be used to drive a larger load. An example circuit is shown in figure 3.15.

Due to the large number of valves used in the manifold (i.e. 25), 26 transistors are needed. In such a case, the control circuit would be cumbersome and harder to debug for faults, etc. To avoid this issue and enable a compact design, a high current transistor array IC, **ULN2803**, is used. Thic IC consists of a eight NPN Darlington connected transistors with common Clamp diodes for switching the loads connected to the output. The rating of each Darlington pair is 50V/500mA, which fits the requirements.





An example circuit is shown below:



Figure 3.15: VavlesConnection

The example circuit 3.15 above, is scaled up to 4 **ULN2803** chips $(4 \times 8 = 32 \text{ transistors})$ to control all 26 valve solenoids. The circuit is designed using **Eagle PCB Design**[®] software.



Figure 3.16: (a) Board Design on **Eagle PCB Design**[®] software, refer to the appendix (A.4 for the detailed PCB schematic view) (b)Transistor Circuit PCB

Lastly, a pressure sensor module is integrated with the **Arduino** LCD shield ³. The pressure sensor consists of 2 ports (i.e. two absolute pressure sensors (which could be used as a signle differential sensor). The the main supply pressure is monitored and the pressure magnitude is simply displayed on the LCD screen.

3.2.5 Detailed Overview

In this final section, a detailed overview of the manifold is presented in the form of a figure as shown in 3.18. The overview consists of:

- Components such as the valves, male fitting, silencers, blanking plugs
- Acrylic and rubber layers and their respective dimensions



PORTS 1 & 2



• Design Considerations

In the next chapter, the computer control of the MRI-compatible robotic system is developed.

 $^{^3}$ ADAFRUIT INDUSTRIES 772 LCD SHIELD KIT, 16 \times 2 BLUE/WHITE DISPLAY, ARDUINO



47

Figure 3.18: Layout Explained

4 Stormram 2 Computer Control

In this chapter, a computer controller for the MRI-Compatible breast biopsy robot (Stormram 2) is developed. Firstly, a general overview of the proposed computer-controller is presented. The overview goes through the important stages of the computer-controlled biopsy procedure.

This is followed by a thorough explanation of the actuators' control strategy (actuators - pneumatic linear stepper motors, control strategy - Pulse Width Modulation (PWM)). Added to that, the inverse position kinematics (discussed earlier in detail - 2.2.2) is integrated with the aforementioned control strategy. This is illustrated with the help of a detailed, easy to follow example.

Next, the software implementation of a graphical user interface control panel for Stormram 2 is presented. Lastly, the integration between the software part and the hardware part (electronic valve manifold and the robot) of the overall system is also shown. Note: The softwares utilised to develop the computer controller are: MATLAB/SIMULINK, **Arduino I/O** SIMULINK library and MATLAB GUI to design and implement the graphical user interface.

4.1 Control Approach

This section describes the control approach undertaken to control the MRI-compatible robot, Stormram 2.

4.1.1 General Overview

The main aim of the project is to localize the lesion in the breast (using MRI) and guide the biopsy needle of the MRI-compatible robot (**Stormram 2**) towards it. In order guide the biopsy needle (end-effector) of Stormram 2 towards the target lesion, the pneumatic linear stepper motors are to be controlled.

Controlling the pneumatic linear stepper motors using the solenoid valves pneumatic distributor (electronic valve manifold) in turn changes the lengths between the needle holder and the centre of the ball joints (denoted by L_1, L_2, L_3, L_4 and L_5). The values of these lengths are evaluated based on a given desired end-effector position and orientation. This is achieved by applying the Inverse Position Kinematics algorithm explained in detail in chapter 2, section 2.2.2. The following flowchart presents an overview of the process described above:



Figure 4.1: Overview of the computer-controlled MRI breast biopsy process



Figure 4.2: (a) Correct Sequence vs. (b) Incorrect Sequence

The following section describes how the pneumatic linear stepper motors are actuated and controlled (hence the red color of the block in the flowchart above (4.1)). The technique used to control the motors is nothing but the Pulse Width Modulation (PWM) approach.

4.1.2 Pneumatic Linear Stepper Motor Control

The control approach used to control the 25 solenoid valves (24 solenoid 5/2 way valves and the double solenoid 5/3 way valve) which in turn control the pneumatic linear stepper motors, is the PWM technique. PWM is a DC supply voltage that is switched on and off at a given frequency for a modulated period of time (duty cycle). The duty cycle is the "on" time of the voltage and is expressed as a percentage of the time period. At 50% duty cycle the voltage is "on" for 50% of the time period and "off" for the remaining 50%. Therefore the time averaged voltage is only 50% of the maximum supply voltage and the current to the solenoid is only 50% of maximum current as well. It is this time averaging that allows PWM signals to be used for proportionally controlling solenoids.

Furthermore, in order to achieve a desired (computed) length L_1 , i.e. moving the rack left (or) right, the teeth of actuator 1 should be pressurized and decompressed in a particular sequence. This sequence is achieved using the PWM technique. An example is illustrated below to move the rack passing through ball joint B1 to the right. For a better understanding of the whole sequence, a state diagram is also presented. A state diagram describes the flow of control from one state to another state.

Part (a) of the figure above (4.2) illustrates an example of how the PWM technique is applied to pressurize and decompress the 3 chambers (i.e. T1, T2 and T3) to move the rack to the right. Pressurization is denoted by the state (0) and decompressing is denoted by the state (1). **One step** is defined as the transition from one tooth to another. It is also observed that the transition

needs to take place in two steps to ensure that the rack is in never lose. For example: in order to move one step to the right, first, tooth T1 is pressurized (state - 011), and then tooth T2 is pressurized (001) and finally tooth T1 is decompressed (state - 101). See figure 4.2 for the difference between the correct and incorrect sequence. As illustrated in the aforementioned figure, in the second state where the all three teeth are decompressed, the rack is loose and it can be moved left and right while interacting with the breast during the biopsy. This is definitely unsafe and undesirable, hence, the transition in two steps to ensure safety during the biopsy.

After a single step, the linear displacement of the rack is computed using the equation below:

$$l = P \times \frac{1}{3} \tag{4.1}$$

where l is the linear displacement and P is the rack pitch. In this case, the rack pitch in Stormram 2 is 3mm. Therefore, each step is equivalent to a linear displacement (l) of:

$$l = 3 \times \frac{1}{3} = 1 \text{ mm}$$
 (4.2)

After *N* steps:

$$l = N \times P \times \frac{1}{3} = 3 \times \frac{N}{3} = N \text{ mm}$$
(4.3)

In the next section, the integration between the inverse position kinematics algorithm and the pneumatic linear stepper motor PWM control (explained here) is explained in more detail with examples.

4.1.3 Inverse Position Kinematics and PWM Control Integration

As stated earlier, the integration between the inverse kinematics and PWM control are explained further with the help of an example.

Example:

The example to be considered here for this detailed explanation, is **Case 1** from section 2.2.2 (used earlier to verify the inverse position kinematics algorithm) - all lengths are in mm and angles in degrees.

• **Step 1**: The desired end-effector position and orientation:

$$-x_E = 15$$
 $y_E = 80$ $z_E = 90$ $\gamma = \alpha = 0^\circ$.

- Step 2: Apply the inverse kinematics algorithm
- **Step 3**: The desired lengths (*L*₁, *L*₂, *L*₃, *L*₄ and *L*₅) between the needle holder and the centres of the ball joints (*B*1, *B*2, *B*3, *B*4, *B*5) are computed:

 $- L_1 = 85.7 L_2 = 100.3 L_3 = 85.7 L_4 = 100.4 L_5 = 118.3$

These values are rounded to the nearest integer:

 $-L_{1d} = 86 \qquad L_{2d} = 100 \qquad L_{3d} = 86 \qquad L_{4d} = 100 \qquad L_{5d} = 118$

where d stands for desired.

• **Step 4**: Prior to proceeding with any computations, the current lengths L_{1c} , L_{2c} , L_{3c} , L_{4c} and L_{5c} are noted down, where *c* stands for the current length. For simplicity, Stormram 2 is assumed to be in the homing position. The homing position is when the end-stops

come in contact with the ball joint, as explained and illustrated in section 2.2.4.1. The lengths in the homing position are given by:

 $-L_{1c} = 106 \qquad L_{2c} = 106 \qquad L_{3c} = 106 \qquad L_{4c} = 106 \qquad L_{5c} = 134$

where *c* stands for current.

• **Step 5**: Next, the difference between the desired and current lengths (in this case the homing lengths) are calculated:

$$- d_1 = L_{1d} - L_{1c} = 86 - 106 = -21 \qquad d_2 = L_{2d} - L_{2c} = 100 - 106 = -6$$

$$d_3 = L_{3d} - L_{3c} = 86 - 106 = -21 \qquad \qquad d_4 = L_{4d} - L_{4c} = 100 - 106 = -6$$

$$- d_5 = L_{5d} - L_{5c} = 118 - 134 = -16$$

• **Step 6**: As stated earlier in the previous section (see equation 4.3), 1 mm is equivalent to 1 step. Therefore, the difference obtained above are in fact the number of steps to be moved by the actuators, i.e. $s_1 = d_1$, $s_2 = d_2$, $s_3 = d_3$, $s_4 = d_4$ and $s_5 = d_5$.

Note: the sign of the value obtained determines the direction of motion of the rack. If the value is negative, this indicates that the length should decrease and the rack should move in direction x, else if the value is positive, then the length should increase and the rack should move in the opposite direction -x.

• **Step 7:** There are two important factors to be considered before moving each actuator the required number of steps: (1) The current pressurized tooth in the actuator (2) The desired direction of motion. Since there are 3 teeth in total and 2 directions for each rack, the total number of scenarios are 6 (3 teeth × 2 directions = 6 scenarios). To avoid confusion, the two opposite directions are denoted by direction 1 and direction 2.

The 6 scenarios (for each actuator) are:

- **Scenario 1:** The current pressurized tooth is "T1(A)" and the rack should move in direction 1 to achieve the desired length (L_i) .
- Scenario 2: The current pressurized tooth is "T2(B)" and the rack should move in direction 1 to achieve the desired length (L_i) .
- Scenario 3: The current pressurized tooth is "T3(C)" and the rack should move in direction 1 to achieve the desired length (L_i) .
- Scenario 4: The current pressurized tooth is "T1(A)" and the rack should move in direction 2 to achieve the desired length (L_i) .
- Scenario 5: The current pressurized tooth is "T2(B)" and the rack should move in direction 2 to achieve the desired length (L_i) .
- Scenario 6: The current pressurized tooth is "T3(C)" and the rack should move in direction 2 to achieve the desired length (L_i) .

where *i* is the actuator number (i = 1...5).

For each scenario, a state diagram is presented and a special PWM is generated (as an output from the **Arduino**) to actuate the pneumatic linear stepper motor accordingly. As shown in the figures, depending on the scenario (i.e. the current pressurized tooth and desired direction), a different set of PWM signals is generated for the three teeth. In fact, the PWM signals for the three teeth (T1(A), T2(B) and T3(C)) are the same in terms of duty cycle (50%) but are shifted with respect to each other depending on the scenario being dealt with. With these illustrations, the integration steps between the inverse kinematics and PWM control conclude.





Figure 4.3: Scenarios 1 to 3





Figure 4.4: Scenarios 4 to 6

The control approach described above emulates a **feedforward controller**, whereby a predefined trajectory is defined (PWM signal) and this control signal is fed into the system to actuate the motors accordingly. This method is effective in this case because, pneumatic linear stepper motors are used. Therefore, it is possible to count the number of steps moved in both directions and to keep track of the current lengths at all times. Moreover, in the next section, the implementation of **Stormram 2**'s control interface is presented.

4.2 Stormram 2's Control Interface

This section focuses on the platform used to design and develop a control GUI panel for Stormram 2. Added to that, the interface between the software entities and hardware components is also presented.

4.2.1 Software & Hardware Interface

To begin with, as stated earlier, MATLAB/SIMULINK is the platform used to create a GUI Control Panel in order to control the MRI-compatible robotic system Stormram 2. In addition to the default MATLAB functions and SIMULINK libraries, a complementary SIMULINK library (**Arduino I/O**) is also utilised to communicate with the **Arduino MEGA 2560 Board**. This additional library consists of several SIMULINK blocks, including: **Digital Read** and **Digital Write**. The figure below illustrates some of the **Arduino I/O** blocks.



Figure 4.5: Arduino I/O blocks

- Arduino IO Setup: This block allows the user to identify the Serial (COM) Port to which the Arduino board is connected. In this case the Arduino board is connected to serial (COM) Port 6.
- Real-Time Pacer: This block ensures that the SIMULINK model runs at the same pace as real time, i.e., 1 second of simulation time corresponds to 1 second of physical time.
- Arduino Digital Write: Write a HIGH or a LOW value to one of arduino's 54 digital pins.

These blocks are used extensively throughout the GUI Control Panel (to be presented later).

This line of MRI-compatible robotics (Stormram 1 and Stormram 2) is in the prototyping stage, and therefore, it is advisable to implement such an interface at such a preliminary stage, using a very powerful tool for scientific computing like MATLAB (Refer to the future improvements for more information regarding this point). The main reasons behind choosing (MATLAB/SIMULINK) as a platform are:

- ★ It is possible to integrate MATLAB Scripts, SIMULINK Models (with the help of the additional Arduino I/O library) and GUI MATLAB with each other to control the hardware setup. To make it clear, the term hardware refers to the following components:
 - Input Devices: Keyboard, Mouse and Joystick
 - Electronic Valve Manifold (Pneumatic Distributor)

- MRI-compatible robotic system (Stormram 2)
- ★ Easier to debug for errors with the help of various debugging techniques in SIMULINK and MATLAB. One commonly used SIMULINK block is Scope, which allows the user to observe the time domain signals with respect to simulation time at any point of the model. An example of such is illustrated in figure 4.6. These techniques come in handy because of the large number of valves being controlled (25 solenoid valves) and various scenarios being dealt with.



Figure 4.6: Example where the SIMULINK block Scope is used for debugging

Moreover, based on the aforementioned introduction and brief discussion, the utilised software entities in the MATLAB/SIMULINK software are: (1) MATLAB Graphical User Interface



Figure 4.7: Software and hardware link

(GUI) - also refer to as Guide, (2) SIMULINK libraries (including **Arduino I/O**), (3) MAT-LAB Functions (Scripts) and MATLAB Workspace. These components are all linked together and they communicate with the manifold's **Arduino MEGA 2560 Board** to control the solenoid valves (with the help of the transistor circuit) and in turn actuate Stormram 2's pneumatic linear stepper motors (Stormram 2). Figure 4.7 illustrates the link between software and hardware components. Similarly, the figure below 4.8, presents an abstract layout of 4.7.



Figure 4.8: Abstract layout of the link between the Operator, GUI, electronic valve manifold and the MRI-compatible robotic system (Stormram2)

Moreover, as mentioned earlier, the GUI is designed and implemented in MATLAB, using the Guide tool. In the following section, the GUI Control Panel is presented and explained in more detail.

4.2.2 Graphical User Interface Control Panel

This section focuses mainly on the implementation ¹ of Stormram 2's Graphical User Interface Control Panel. The GUI is designed to be simple, user friendly and intuitive to use by the Radiologists ² during the localization and targeting of the lesion during the computer controlled breast biopsy procedure. Figure 4.9 shows the GUI Control Panel designed and implemented using MATLAB Guide. The GUI control panel consists of 9 sub-panels, namely: (1) Direct Control Panel, (2) Arduino Panel, (3) Control Panel (Single Actuation), (4) Control Panel (Multiple Actuation), (5) Inverse Kinematics Panel, (6) Joystick Panel, (7) Debug Panel, (8) Demo Panel and (9) Instructions Panel. Next, the purpose of each of the 9 sub-panels is summarised briefly. Note: This GUI Control Panel is mainly concerned with the control of Stormram 2, i.e. it can only control 5 pneumatic linear stepper motors (5 motors × 3 valves for each motor + 1 Main Supply valve (double solenoid) = 16 valves and 17 solenoids).

Arduino panel: This panel is used to connect and disconnect the Arduino board to and from MATLAB/SIMULINK. In order to establish a connection between the Arduino board and MAT-LAB/SIMULINK, edit the COM Port³ and then click on **Setup**. To terminate this connection and erase the Arduino object from MATLAB's workspace, click on **Delete**. Note: In order to achieve a successful connection, upload the file adioes.pde to the **Arduino Mega board** before establishing the aforementioned connection. This is achieved by navigating through the **Arduino I/O** library folder (..ArduinoIO \pde \adioes) and opening the file in the open-source **Arduino** Software (IDE) and uploading it to the **Arduino** board. This step should be executed before establishing the connection through the GUI.

¹The different panels of the GUI are presented and how are the panels linked to the MATLAB Functions, MATLAB Workspace and SIMULINK Models

²Without having to worry much about the technicalities of the actuators of the robot, etc.

³In order to obtain the COM Port number, check the Device Manager (Windows)





Debug panel: In simple terms, the debug panel is used to check whether the 16 solenoid valves are functioning properly or not. To start debugging, click on the **Start Debug** push button. The checking can be done *manually* by clicking on the toggle buttons ⁴ representing each of the 16 solenoid valves (i.e. Main Switch 1, Main Switch 2, 1A, 1B, 1C ... 5A, 5B, 5C) or *automatically* by simply clicking on the **Automatic Debug** push button. The Automatic Debug turns **on** all 17 solenoids in the sequence (Main Switch 1, Main Switch 2, 1A, 1B, 1C, 2A, 2B, 2C ... 5A, 5B, 5C) and then switches **off** all 17 solenoids in the reverse order (5C, 5B, 5A, ... 2C, 2B, 2A, 1C, 1B, 1A, Main Switch 2, Main Switch 1). The debugging/checking can be interrupted and stopped by simply clicking on the **Stop Debug** push button. In case of any errors or any other reason, the valves can be all turned off (reset) by clicking on the **Reset Valves** button. Note: This sub-panel basically controls the previously described SIMULINK Model: Check_Valves.mdl. This is achieved by simply switching "on" and "off" the switches (MSW1, MSW2, SW1A, SW1B, SW1C, ... SW5A, SW5B, SW5C) in the GUI MATLAB script (GUI_Controller.m). The following piece of code illustrates an example of how this is done:

```
function togglebutton5_Callback(hObject, eventdata, handles)
% hObject handle to togglebutton5
% Hint: get(hObject,'Value') returns toggle state of togglebutton5
.value = get(hObject,'Value');
if value == 1
    set_param('Manifold_Check/SW2B','sw','0');
else
    set_param('Manifold_Check/SW2B','sw','1');
    end
```

The code above is the Callback function of the toggle button 5. Depending on the state of toggle button 5, the solenoid coil of valve (2B) is switched on or off. ⁵ Toggle button 5 is labeled in the figure below (4.10), along side the Manual Switch SIMULINK block (SW2B) which it controls. Moreover, for each SIMULINK block, there is a parameter or a list of parameters that describe this block. These parameters can be accessed and modified/changed through Matlab code. For the Manual Switch block, the parameter of concern is sw. So, the basic idea here is to control the SIMULINK block using Matlab code. This is achieved by using the MATLAB function, set_param as shown above. The code above is described below:

- 1. Retrieve the state of the toggle button using the get function and save it in a variable named value.
- 2. Check the state of the button. If the button is off, i.e. value == 0, then the Manual Switch simulink block parameter sw is modified to 1, else if the button is on, i.e. value = 1, then the Manual Switch simulink parameter sw is modified to 1.
- 3. Note: If sw = 1, the switch points upwards and passes the first input (in this case the value of 0 and it is positioned on the top). If sw = 0, the switch points downwards and passes the second input (in this case the value of 1 and it is positioned on the bottom).

The aforementioned strategy of linking the GUI control panel and its corresponding MATLAB script GUI_Controller.m with the SIMULINK model Check_Valves.mdl is applied on the remaining remaining 16 buttons. All in all, with a simple click of a button, the solenoid coils of the pneumatic valves are switched on and off.

⁴A toggle button is a button that has an on and off state.

⁵A Callback function is a function that is provided to another piece of code, allowing it to be called by that code. In this case, togglebutton5_Callback is the callback function, it is added to the GUI MATLAB script GUI_Controller.m and it is allowed to be called by it

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 58 biopsy system



Figure 4.10: Link between GUI and SIMULINK model - Check_Valves

Control panel (single actuation): The main purpose of this sub-panel is to control one pneumatic linear stepper motor at a time by increasing and decreasing the lengths associated to it. That is, for **pneumatic linear stepper motor 1**, length L_1 is increased or decreased accordingly. The same principle is applied to motors 2, 3, 4 and 5 and their corresponding lengths L_2, L_3, L_4, L_5 . To illustrate how this panel works, a simple example is illustrated in figure A.2.

Prior to proceeding with the example, the MRI-compatible robot Stormram 2, is assumed to be in the homing position where the 5 lengths are maximum in value. ⁶ In this example, from steps 1 to 4, length L_3 is decreased from the initial value of 106 to a value of 99. Note: In step 4, once the **Make Move** button is clicked, the main supply is switched on. From steps 5 to 9, another move is performed by increasing back L_3 to a value of 104 mm. From steps 10 to 12, the length L_3 is reset. Reset in this case means that length is set to the maximum value (i.e. 106 mm for lengths L_1, L_2, L_3 and L_4 and 134 mm for lengths L_5). Finally, in step 13, the main switch is closed to shut off the compressed air from the valves.

Concerning the implementation of this panel; depending on the selected option:

- 1. length $(L_1, L_2, L_3, L_4, L_5)$
- 2. Action (decrease/increase)
- 3. Value, where the values for lengths L_1, L_2, L_3, L_4 range from a minimum value of 50 mm to a maximum value of 106 mm and L_5 fro a minimum value of 78mm to a maximum value of 134mm.

⁶Note: The length L_3 varies from a minimum value of 50 mm to a maximum value of 106 mm, refer to 2.2.4.1. The same applies to L_1, L_2 and L_3 . For L_5 , the length varies from 78mm to 134mm.



Main.mdl





GUI_Controller.m

a particular pulse width is generated to control the pneumatic linear stepper motors. As stated earlier, for each push button, toggle button, pop-up menu placed in the GUI_Controller.fig, there is a corresponding Callback MATLAB function. The callback functions are located in GUI_Controller.m. Furthermore, based on the operator selection, the callback function sets parameters in the SIMULINK model (Main.mdl). For example, as shown in figure 4.11, a snippet of the callback function of the **Make Another Move** button is shown. The idea is as follows: once the push button is clicked, the callback function is executed. In this case, the code mainly focuses on controlling the blocks and retrieving data from MATLAB's workspace ⁷.

- * In order to modify the SIMULINK parameters through MATLAB script, the MATLAB function set_param is used as explained in the Debug Control Panel.
- ★ In order to retrieve data from MATLAB's workspace, the MATLAB function evalin is used.

```
L1x = evalin('base','L1');
%Retrieves the variable L1 from the workspace and assigns it to ...
the newly defined variable L1x
```

* In order to assign a value to variable in MATLAB in the workspace, the MATLAB function assignin is used. Example:

```
assignin('base','L3',L3);
%Assigns the value L3 to the variable L3 in
% Matlab's workspace (denoted by base)
```

These three MATLAB functions are used extensively throughout the interface between MATLAB, SIMULINK models and the workspace. Figure 4.7, illustrates this interface. Furthermore, an example of this link is illustrated in figure 4.11. In this case, once the **Make Another Move** button is clicked, the parameters of the constant SIMULINK blocks: **Amplitude1A1**, **Amplitude1A1** and **Amplitude1A1** are set to the value **A**. On the other hand, **PulseWidth1A1**, **PulseWidth1B1** and **PulseWidth1C1** are set to the value **PW1A**, **PW1B**, **PW1C** respectively. Note: The code in figure 4.11 is just a snippet of the entire callback function.

Lastly, in figure 4.11, the **Counting Steps** group of SIMULINK blocks keeps track of the number of steps by checking the falling edge of the pulses generated. The falling edge is detected using the **Detect Decrease** block. When a falling edge is detected, the SIMULINK block outputs a value 1 of the type boolean. In addition, **Interpreted MATLAB Function** block is used to pass the input to an already existing or newly created MATLAB function. In this case, a MATLAB function is created to count keep track of the falling edges and save them in the workspace, so it is reachable by the remaining components of the software interface. It also keeps track of the current length and save it to the workspace. This comes in handy helps when the robot is reset back to its homing position. Note: This function takes in input of the type double. That is why, the **Data Type Conversion** block is used (to convert the boolean output of the **Detect** increase block to the double input of the **Interpreted MATLAB Function** block).

Control panel (multiple actuation): The main purpose of this sub-panel is to control multiple pneumatic linear stepper motors. During the actual control of the robot it is desired to have all pneumatic linear stepper motors operating at the same time to achieve a desired end-effector position. For example, all 5 lengths L_1, L_2, L_3, L_4 and L_5 are decreased to different val-

⁷From this point on wards, MATLAB's workspace is referred to as workspace only

ues. Once this step is done, the **Make Move** button is clicked (and the main supply is switched on). Next, another move is performed by increasing and decreasing three of the five lengths, namely: L_1, L_2 and L_3 . After selecting the lengths, **Make Move** is clicked and the pneumatic linear stepper motors are actuated depending on the lengths chosen. The robot is put into the reset position by simply clicking on **Initiate Reset**, followed by **Reset Robot**. A similar implementation is applied for this as well.

Inverse Kinematic Panel: This panel enables the user to compute the required lengths L_1, L_2, L_3, L_4 and L_5 to achieve the desired end-effector position and orientation $(x_E, y_E, z_E, \gamma, \alpha)$. This panel also shows the computed lengths and the current lengths L_1, L_2, L_3, L_4 and L_5 . It basically calls the inverse position kinematics **Matlab** function A.1. The inverse position kinematics is described in 2.2.2.

Demo Panel: This robot is basically shows visitors how does the robot moves and operate.

Instructions Panel: This panel provides the operator with step by step instructions to use the GUI.

The remaining two panels: **Direct Control Panel** and **Joystick Panel** are discussed in 5.3 and 6.2 respectively. In the next chapter, the MRI-robotic system, Stormram 2 (Chapter 2), the electronic valve menifold 3 (Chapter 3) and the software control (Chapter 4) are integrated together to conduct experiments in the MRI room.

5 Experiments and Results

This chapter thoroughly describes the experiments conducted and subsequent results. Firstly, the experimental setup is presented. Then, the different experiments and the steps taken in each experiment are given. Lastly, the results of the aforementioned are presented and discussed.

5.1 Experimental Setup

The figure below (5.1) illustrates an overview of the floor plan of the MRI-room in and the experimental setup to test the overall system.



Figure 5.1: Experimental Setup

The experimental setup consists of the following components:

- (a) MRI-compatible robotic breast biopsy system (Stormram 2), presented in 2.2 with markers added to it. The reason behind the addition of the markers are explained further later.
- (b) Breast phantom with markers mimicking the lesion. Section 5.2 describes how the breast phantoms are made.
- (c) Solenoid valves pneumatic distributor (Computer-controlled), which is also referred to as the Electronic Valve Manifold, presented in 3.2.2.
- (d) Pneumatic tubes connecting the electronic valve manifold to the MRI-compatible robot (Stormram 2).
- (e) Portable Air Compressor
- (f) MRI (Magnetic Resonance Imaging) Scanner. In this case, MRI tests were performed with an Esaote G-scan Brio 0.25T (where T stands for Tesla) scanner.



Figure 5.2: Esaote G-scan Brio 0.25T

- (g) 24V/2.5A power supply
- (h) Desktop computer connected to the MRI-scanner and a laptop with MATLAB and 3D SLICER installed
- (i) PACS sever, where PACS is short for picture archiving and communication system. PACS is a medical imaging technology which provides economical storage and convenient access to images from multiple modalities (source machine types)[citation].

5.2 Experiment Preparations

Breast Phantoms

As illustrated in figure 5.1, a breast phantom is immobilized in a frame and placed in the MRI scanner. The following figure 5.3, describes how breast phantoms are made. Note: In this explanation, two different phantoms with two different stiffnesses are made. The reason behind this is presented and discussed in section 5.3.

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 64 biopsy system



Figure 5.3: How the breast phantoms are made



Figure 5.4: One MRI slice (image), visualizing markers in the breast phantom and the contrast between them

As shown in the figure above, plastileurre (liquid mixture of PVC and plasticizer) and assouplissant plastileurre (a softener) are added together in a beaker in different quantities and heated at 220°C until the white liquid becomes transparent. Next, the transparent liquid is poured into the breast mould (red) and 2 to 4 fish oil capsules are added to the mould. The liquid is then left to cool until it solidifies. The fish oil capsules mimic the lesions inside the breast. As explained earlier in section 1.5.2, in essence, MRI measure the water content (or fluid characteristics) of different tissues, which is processed by the computer to create a black and white image. In this case, the fish oil capsules have different water contents when compared to the breast phantom, leading to a contrast between them in the MRI-image as shown in figure 5.4.

Furthermore, as pointed out in figure 5.3, the softener (assouplissant plastileurre - Container **B**), determines the stiffness of the phantom. In short, the larger the quantity of the softener, the less stiff the phantom becomes and vice versa (lesser quantity, more stiff). In the first case, a stiff breast phantom is made by adding 600 grams of **A** and 0 grams

of **B**, i.e. (i.e. 100% of **A** and 0% of **B**). On the other hand, a softer breast phantom, is made using 510 grams of **A** and 90 grams of **B** $\left(i.e.\frac{510}{600} \times 100\% = 85\% \text{ of } \mathbf{A} \text{ and } 15\% \text{ of } \mathbf{B}\right)$. The figure below shows the two resulting breast phantoms.



Figure 5.5: The two breast phantoms and how the breast phantom is placed inside the MRI-scanner

Robot Markers

As illustrated in figure 5.1, the MRI-compatible robot (Stormram 2) is positioned next to the breast phantom. To perform the procedure multiple fish oil capsules are inserted into the breast phantom as explained earlier. The main concern now is to localize the lesion using the MRI image and obtain the desired end-effector (needle) position and orientation (x_E^R, y_E^R, z_E^R) with respect to the Robot Coordinate System (Ψ^R), refer to figure 4.1 for an overview of the process. There are several ways to approach this task, the suggested approach here is to add four markers on the robot as shown in the figure below.



Figure 5.6: (a) Stormram 2 SOLIDWORKS model with markers added, (b) Sketch of Stormram 2 showing the coordinate of the markers with respect to the Robot Coordinate System (Ψ^R)

The four markers are positioned at known positions with respect to Ψ^R as illustrated in the sketch, part (b) of figure 5.6. The algorithm implemented to evaluate the position of the lesion (the desired end-effector position) with respect to Ψ^R , (x_E^R, y_E^R, z_E^R) is explained in further detail in the following section (5.3).

5.3 Experimental Procedure

This section thoroughly describes the computer controlled breast biopsy procedure. For a brief overview refer to figure 4.1.

- 1. The first step is to setup the experiment in a similar manner to the outline illustrated to figure 5.1.
 - (a) Connect the electronic valve manifold to Stormram 2 via the pneumatic tubes
 - (b) Connect the electronic valve manifold to the 24V/2.5A and the portable air compressor
 - (c) Connect the laptop (with MATLAB and 3D SLICER installed) to the Arduino Mega Board via USB cable.
 - (d) Connect the Laptop to the university network wirelessly or via Ethernet cable.
 - (e) Place the phantom breast and Stormram 2 inside the MRI scanner as illustrated clearly in figure 5.1.
- 2. Switch on the desktop computer connected to the MRI-scanner
 - (a) The MRI-scanner is a low field systems (0.25T)
 - (b) The MRI-scanner has 2D and 3D sequences. After several attempts with different sequences, the 3D sequence (3D Hyce) is chosen.
 - (c) The MRI-scanner uses one of the most important model coordinate systems for medical imaging, namely: the anatomical coordinate system (also called patient coordinate system) Φ^A . This coordinate system consists of three planes to describe the standard anatomical position of a human (in this context, the robot/breast phantom setup): Coronal, Sagittal and Transverse planes. The coronal plane divides the robot + breast phantom into front and back (also called posterior (*P*) and anterior (*A*)). The sagittal plane divides the body into left (*L*) and right (*R*) halves. The transverse plane is parallel to the ground and divides the body into top (head / superior (*S*)) and bottom (feet / inferior (I)) parts. The figure below illustrates the planes and the two coordinate systems: Ψ^R , the Robot Coordinate System and Ψ^A , the Anatomical Coordinate System used by the MRI scanner.



Figure 5.7: Coronal, Saggital and Transverse planes with the two coordinate systems Ψ^R and Ψ^M . Note: The two coordinate systems are not located in their respective origins, the main intention of adding them is to illustrate the different axes and their relation with respect to each other

- 3. A scan of Stormram 2 and the breast phantom is made (pre-targeting scan). Some of the settings used: Coronal Plane, Reading FOV = 300, Encoding FOV = 300 and Isotropic Acq is checked.
- 4. The scan obtained is a set of images (slices) and the scanner usually supplies slices in all three planes. The coronal plane slices are illustrated in the figure below alongside an image of a single coronal slice (see figure 5.8).




- 5. Once the pre-targeting scan is obtained, the scan is exported to the server
- 6. Next,the images are downloaded onto the laptop through the network from the server for analysis
- 7. Open 3D SLICER and import the scans
- 8. Select Green Slices viewing mode
- 9. The next task is to locate the centres of the robot markers, i.e. the centres of the fish oil capsules in the MRI-scan (refer to figure 5.6 for a better understanding of the points of interest). This is achieved by locating the bottom tip and the top tip of the capsule and noting down the *A*-coordinate as shown in the figure below.







Figure 5.9: The location of the capsules' bottom and top tips along the *A*-axis, where $A_{top} = 16.881$ mm and $A_{bottom} = -4.706$ mm. Note: The tips of the markers are not very clear in the slices above due to the fact the tips are very small white dots/circles. The whole purpose of this figure to illustrate the basic idea of how to use 3D SLICER to obtain the centre coordinates.

In order to get the approximate location of the centre of the capsules along the *A*-axis, the following equation is evaluated:

$$A_{\text{center}} \approx A_{\text{top}} - \left[\frac{A_{\text{top}} - A_{\text{bottom}}}{2}\right] = 16.881 - \left[\frac{16.881 - (-4.706)}{2}\right] = 6.0875 \text{ mm}$$
 (5.1)

Next, navigate through the slices until *A* is approximately 6.0875 mm. Tip: It is also possible to edit the value of A. This is followed by obtaining the centres of the capsules by adding

fiducial markers (A fiducial marker is an object placed in the field of view of an imaging system which appears in the image produced, to be used as a point of reference or a measure). The option is highlighted in the figure below (5.10).

Image: Second					B 10 Silcer 4.5.0-1 File Edit View Help					- 0	
Image: States Image: States<					Modules: 🔍 🚠 Markupe	• = 0, 0, 1 E		🔲 \$ * 🗑 An An 🕂	- 🖬 🤣		
Image: State Sta					3DSIIcer		O to	bot_markers-2	erobot marker	Crobot markers-1	
Image: State of the state					 Help & Advrowledgement 						
Image: State of the state			tat entry parties entry and entry an								
Image: Second							Grob		Probot marke	Probat markers-3	
Image: Second and Second											
Image: State of the construction of						Transformed Hidel	us				
Image: Construction of the image of the					12 += +- Name Description 1 ✔ 🚰 # robot_markers-1 Robot Marker M1 +	R A S 55.104 6.088 99.830				$\Lambda \cdot 6.088 \text{ mm}$	
Image: Second structure Image: Second structure <th></th> <th></th> <th></th> <th></th> <th>2 V 🛃 🐢 robot_markers-2 Robot Marker M2 3</th> <th>4.075 6.088 103.823</th> <th></th> <th></th> <th></th> <th>A. 0.000 mm</th>					2 V 🛃 🐢 robot_markers-2 Robot Marker M2 3	4.075 6.088 103.823				A. 0.000 mm	
Image: Second secon		_	/		4 🖌 🚊 👁 robot_markers-4 Robot Marker M4 3	6.738 6.688 76.403					
Image: Second	/										
Image:			1	1	1	1			4		
Image: Set in the set i	V	84	*0	Name	Description	R	Α	S			
		٢X									
▼ ● robot_markers-2 Robot Marker M2 34.075 6.088 103.823 ▼ ● robot_markers-3 Robot Marker M3 -58.033 6.088 72.410 ▼ ● ● robot_markers-4 Robot Marker M4 36.738 6.088 76.403	•	•		robot_markers-1	Robot Marker M1	-55.104	6.088	99.830			
Image: Second state	7	XX	***	rebet markers 2	Dobot Markor M2	24.075	6 000	102 922			
✓ ✓ ✓ robot_markers-3 Robot Marker M3 -58.033 6.088 72.410 ✓ ✓ ✓ ✓ robot_markers-4 Robot Marker M4 36.738 6.088 76.403	•	<u></u>		TODOL_INALKEIS*2	RODULIMAI NEI 142	34.075	0.000	105.025			
Image: Second markers -4 Robot Marker M4 36.738 6.088 76.403	~	×	*	robot markers-3	Robot Marker M3	-58.033	6.088	72 410			
✓ Image: Marker S-4 Robot Marker M4 36.738 6.088 76.403	-					551055	0.000				
	~	X	-	robot markers-4	Robot Marker M4	36.738	6.088	76.403			
t international second s			-		L						
					1			Laboratoria 5 cm			

Figure 5.10: The coordinates of the 4 markers with respect to the MRI-coordinate system

As illustrated in the figure above, four fiducial are placed on top of the four centers of the fish oil capsules. It is also possible to see the coordinates of each marker with respect to the MRI-coordinate system Ψ^R in the 3 axes (R,A and S).

10. Next obtain locate the lesion of interest in a similar manner.



Figure 5.11: The coordinates of the 4 markers with respect to the MRI-coordinate system

11. Note down and copy them into the GUI as shown below. Next, the following steps are taken. (Make sure Arduino is setup properly, refer to a the previous chapter for a step by step procedure)



Figure 5.12: GUI Steps to actuate the robot towards the lesion

12. After actuating the robot, another scan is made (post-targeting scan) in order to verify whether the needle targeted the lesion of interest or not. An example of a successful experiment is illustrated below:



Figure 5.13: (a) Pre-target MRI-scan (b) Post-target MRI-scan (c) Post-target MRI-scan with 2.2 needle added at the measured position with the blue circle as the desired end-effector position (This figure is NOT TO SCALE)

5.4 Experiments, Results and Discussion

In this section, several experiments are attempted to test the overall system. The same procedure described in section 5.3 is followed in all experiments. The experiments are conducted on two breast phantoms with different stiffnesses as illustrated and explained earlier in section 5.2. Different breast phantoms are used to emulate the differences in breast tissues of patients. Normally as women get older, there breast tissue gets less dense (i.e. more fat and less fibrous tissues). Moreover, the two breast phantoms used are the **stiff breast phantom** and the **soft breast phantom** (Refer to figure 5.5). In the stiff breast phantom, 2 fish oil capsules (lesions) are targeted. On the other hand, in the soft breast phantom, 1 fish oil capsule (lesion) is targeted. The lesions of interest are illustrated in the figure below:



Figure 5.14: The three lesions to be targeted and how the size of each lesion is calculated. The lesion sizes are (approximately lesion 1 = 15.625 mm, lesion 2 = 15.5 mm, lesion 3 = 6.9 mm

Note: All experiments are conducted with a desired orientation of zero (i.e. $\gamma = 0^{\circ}$ and $\alpha = 0^{\circ}$). While attempting some of the trial experiments, a problem is observed, namely: the trajectory (path) followed by the needle while inserting it into the breast. This issue is clarified with the help of the figure below (5.15).



Figure 5.15: Trajectory leading to a vertical collision with the breast phantom and unsuccessful targeting of the lesion

Figure 5.15 illustrates the trajectory (path) taken by the needle in order to target the lesion. The main problem caused by this trajectory is that the needle collides with the breast vertically. Since the needle is not sharp enough in this direction, the resistance caused by the breast phantom forces the needle to move up and the needle misses the target lesion. In order to overcome this problem, there are two possible ways: (1) Modifying the orientation of the needle by setting the desired end effector orientations γ and α to values in order to target the lesion at an orientation which allows it to penetrate through the breast. (2) Follow a trajectory with zero orientation ($\gamma = \alpha = 0^\circ$) as shown below:



Figure 5.16: Second Trajectory

In this approach, the trajectory can be categorized into 3 steps:

- ★ Step 1: Needle is retracted back by simply actuating the pneumatic linear stepper motor (integrated inside the ball joint *B*5) and decreasing the length L_5 by 40 steps = 40mm, which is obtained by trial and error. L_5 is the length between critical point *B* and the centre of ball joint *B*5.
- ★ Step 2: Needle moves downwards or upwards until it reaches the y-coordinate of the target lesion.
- ★ Step 3: Move needle towards the lesion

Example:

- 1. The initial lengths of the robot (based on the homing position) are $L_1 = L_2 = L_3 = L_4 = 106 \text{ mm} \text{ and } L_5 = 134 \text{ mm}$. L_5 is decreased by 40 steps = 40 mm \Rightarrow $L_5 = 134-40 = 94 \text{ mm}$. The end-effector position after this move, is equivalent to: $x_{Ec} = 0$, $y_{Ec} = 83$, $z_{Ec} = 36$.
- 2. Based on the analysis from the pre-targeing scan, the desired end-effector position is $x_{Ed} = -2$, $y_{Ed} = 72$, $z_{Ed} = 107$. In order to place the needle on the same level as the lesion (i.e. same y coordinate), let: $x_{Et} = x_{Ec} = 0$, $y_{Et} = y_{Ed} = 72$ and $z_{Et} = z_{Ec} = 107$, where *t* stands for a temporary (parameter). Actuate the robot to target point (x_{Et} , y_{Et} , z_{Et}).
- 3. In the final step, simply actuate the robot to target the robot towards the point (x_{Ed}, y_{Ed}, z_{Ed}) .

The following figures illustrate the results of the three experiments using the rendering tool in SLICER after segmenting (taking into account the second trajectory approach) ¹:

Pre-targeting scan vs Post-targeting scan (Lesion 1, Stiff Breast):



Figure 5.17: Rendering using 3D slicer (stiff breast, lesion 1)

Pre-targeting scan vs Post-targeting scan (Lesion 2, Stiff Breast):



Figure 5.18: Rendering using 3D slicer (stiff breast, lesion 2)

Pre-targeting scan vs Post-targeting scan (Lesion 2, Stiff Breast):



Figure 5.19: Rendering using 3D slicer (soft breast, lesion 3)

¹ The 1st approach is not considered because of time constraint, but it is definitely worth looking into in future experiments

As illustrated in the three figures above, the pre-targeting scans are compared with the posttargeting scans. Different views are shown to illustrate clearly what happens once the needle enters the breast and targets the lesion (marker). Next, the overall system is analysed based on the problem statement of the project, namely: accuracy and efficiency of the system. Firstly, the accuracy of the computer-controlled biopsy system is studied through an in-depth analysis of the scans by comparing the end-effector desired and actual positions. To begin with, when no needle is inserted, the robot has no measurable influence on MRI scans at all. When the off-the-shelf titanium needle is inserted (the only metallic component of the robot), it causes artifacts in the direct vicinity of the needle. This is illustrated clearly in the figure below.



Figure 5.20: (a) The artifacts (yellow) in the MRI scan caused by the titanium (metallic) needle, (b) Artifacts within the vicinity of the needle, (c) Possible locations of the needle within the region defined, i.e. the needle might lie anywhere within the yellow region, (d) Size of the artifact region and it's location with respect to the needle. The size of the artifact is obtained based as a result of investigating several MRI-scans, the diameter of the artifact is approximately 8mm (e) Error computation in all 3 axes of the Robot Coordinate System Ψ^R

The difference between the actual and desired end-effector positions is calculated from the MRI-scans. Part (e) of the figure above 5.20,

shows how the errors in all 3 axes (x, y and z) are computed. Note: The errors are computed with respect to the Robot Coordinate System Φ^R (i.e. x, y and z axes). Moreover, the errors in x and z are straightforward, however, in order to obtain the error in the y axis, the following procedure is followed: (1) It is assumed that the position of the target lesion is already known, note down the position of lesion 3 with respect to A. (3) Select the scan (coronal slice) where the needle starts to appear and note down the value of A. (4) Select the scan (coronal slice) where the needle starts to disappear and record the value of A, (5)Compute the position of the scan in between the above-mentioned 2 scans.(6) The error can be simply calculated using a similar approach in 5.2:

$$A_{\text{center}} \approx A_1 + \left[\frac{A_2 - A_1}{2}\right] = 16 + \left[\frac{24 - 16}{2}\right] = 20 \text{ mm}$$
 (5.2)

(7) Once A_{center} is calculated, the error e_y is computed by subtracting A_{center} from the target lesion A-coordinate (21.7 mm). Taking into account the aforementioned artifact, an error-bar plot is presented to take the uncertainties caused by it. Part (a) of the figure below **??** illustrates the maximum and minimum values of the error taking into consideration the artifact.

Part(a) of the figure above illustrates an example of how the minimum and maximum errors of the differences between the actual and desired end-effector positions are calculated. In addition, part(b) gives an overview of all error-bars for the 3 experiments conducted with the 2 breast phantoms. Furthermore, it would be also safe to assume that the needle is positioned at the centre of the artifact. The reason is because, both the needle and the artifact are symmetric. So it is valid to say that the needle is situated in the centre. Based on this assumption:





Figure 5.21: Error minimum and maximum

$$e_{1} = \sqrt{(e_{x1})^{2} + (e_{y1})^{2} + (e_{z1})^{2}}$$

= $\sqrt{(4)^{2} + (2)^{2} + (1.5)^{2}} = 4.7170 \text{ mm}$
$$e_{2} = \sqrt{(e_{x2})^{2} + (e_{y2})^{2} + (e_{z2})^{2}}$$

= $\sqrt{(6)^{2} + (4)^{2} + (1)^{2}} = 7.2801 \text{ mm}$
$$e_{3} = \sqrt{(e_{x3})^{2} + (e_{y3})^{2} + (e_{z3})^{2}}$$

= $\sqrt{(3)^{2} + (4)^{2} + (1.7)^{2}} = 5.2811 \text{ mm}$

Figure 5.22: Error bars plot, alongside the error calculation based on the assumption that the needle is situated in the center

As illustrated in the figure above (5.22), the error ranges from 4.7 - 7.3 mm.

Next, the efficiency of the system is investigated by estimating the time taken for the entire procedure explained in 5.3. Only one experiment is recorded (i.e. targeting of Lesion 3) and the time taken for each of the main tasks is summarized in the figure below under the caption (Current computer controlled breast biopsy procedure):



Current computer controlled breast biopsy procedure

Future computer controlled breast biopsy procedure

Figure 5.23: On the left hand side is the timeline for the current computer controlled breast biopsy procedure (measured while targeting lesion 3) and on the right hand side is the timeline for expected future timeline for a more efficient biopsy procedure. Note: The **Calculate Path** step is nothing but the trajectory planning done to avoid collisions with the breast (as illustrated clearly in 5.15).

As shown in the figure, the time taken for the complete computer controlled procedure (excluding some interactions with the patient) is approximately 30 to 32 minutes. Since the MRI-scanner is a low magnetic field MRI scanner, the time needed to take a good quality image (i.e. 3D Hyce) is long (approximately 5 minutes and 30 seconds scanning and 2 minutes and 30 seconds post processing). Therefore, the two scans (pre-targetting scan and post-targeting scan) take in the current procedure 16 minutes. This can be reduced by either using a lower quality MRI scanning sequence, which can be obtained in around 2-3 minutes (or) using a high field strength MRI scanner (1.5T or more, similar to the one used in the study (?)).

5.4.1 Further Analysis and Discussion

High field strength MRI scanners: The post-targeting MRI scan is taken with the needle inside the breast. This led to some artifacts around the needle as shown in figures 5.20 and 5.22. As mentioned earlier, the experiments are conducted inside a low field strength (0.25T) MRI scanner. Using a higher field strength MRI-scanner 2 will not suppress the needle artifacts and eliminate them. In fact, the use of high magnetic field strength MRI produces more obtrus-ive/noticeable artifacts than does the use of lower field strength. Moreover, in future experiments it is important to take the aforementioned into account. Another issue to be discussed here for the first time is the fixation of the breast and breast tissue movement caused by the insertion of the needle and chest movement.

Based on (BEKINK, 2014), the accuracy of the system can be affected by two main factors: the way the breast is fixated, and the device and method that is used for taking a biopsy from the target lesion. Other important factors are the time-efficiency, the safety of the device, ease of use for the radiologists and radiologic technologists, and the comfort of the patient. This thesis focuses mainly on tackling accuracy and efficiency as stated earlier. Therefore, it is important to discuss the issue of breast tissue movement.

Concerning breast tissue movement, a lot of research has been carried out in assessing the mechanical properties of the tissue, also known as elastography. Therefore, it would be possible to integrate or use elastography in the development of a more accurate computer-controlled system. On the other hand, an alternative solution would be to use a live stream of MRI scans. Even though this technology is not as developed as the currently existing MRI-scanners but it is definitely a method to consider in the future, because it enables the radiologists to keep track of the tissue/lesion movements.

In (Veltman et al., 2005), the average accuracy for the needle placement in the localization procedure is in the range of (0 - 5mm) and the average time taken for the localization procedures is 33 minutes for an MRI-guided localization. Even though it is difficult and unfair to compare between the experiments conducted using the computer controlled system and the experiments conducted by the radiologists in (Veltman et al., 2005), the results obtained: range (4.7 - 7.3 mm) and a localization procedure time of 31 minutes ³ minutes, are definitely promising.

Having said that, in the next chapter, the project is concluded and additional ideas for future work/development are presented.

²Similar to the one used by the Doctor in (Veltman et al., 2005)

³Can be cut down even further up to approximately 12 minutes

6 Conclusions and Future Work

6.1 Final Discussion and Conclusion

In order to improve targeted breast biopsy and reduce the false negative results obtained after an MRI-guided breast biopsy procedure, an innovative MRI-compatible breast biopsy system is developed. Using this system, the study focused on: how can the efficiency and accuracy of MRI-guided breast biopsy be improved?

To tackle this question, an already existing (proof-of-concept) pneumatically actuated, 5-DOF MRI-compatible robot (Stormram 2) was thoroughly investigated by: **deriving** the position kinematics (inverse and forward), **analysing** the constraints of the robot and **determining** the robot's reachable workspace. In parallel to that, a computer controlled pressure distributor (electronic valve manifold) was assembled and carefully sealed to avoid any air leakage. The manifold's main function is to control the pneumatic linear stepper motors (actuators) of Stormram 2. An intuitive, easy to use, Graphical User Interface (GUI) was designed and implemented using MATLAB/SIMULINK, so that end-user (radiologists/operator) can control the robot. Moreover, to emulate the actual breast biopsy procedure, breast phantoms: (1) with fish capsules (mimicking lesions) and (2) with different stiffnesses were made.

To check the functionality of the overall system, the aforementioned components are integrated with the MRI scanner to build the experimental setup described in 5.1. The main aim of the experiments was to localize the lesion in the breast (using MRI) and guide the biopsy needle of the MRI-compatible robot (Stormram 2) towards it. Several experiments were conducted with different phantoms, targeting different lesions with different sizes (approximately 6.9 mm, 15.5 mm and 15.62 mm).

During the course of the experiments, it was observed that the trajectory followed by the needle is very crucial. Therefore, a basic strategy to estimate a trajectory was proposed to avoid unwanted collisions of the needle with breast phantom. For all experiments, two scans are made: **Post-targeting scan** and **Pre-targeting scan**.

After thoroughly analysing the images (scans), it was observed that the metal needle caused some **metal** artifacts in the image. However, with a symmetric needle and a symmetric artifact, it can be concluded that the needle lies in the centre of the artifact. Based on this assumption the error was estimated (i.e. how accurate is the computer controlled biopsy system) and the results turned out to be promising. The errors ranged form (4.7 - 7.3 mm) and the time taken to perform the MRI-guided localization was approximately 31 minutes (**can be reduced to** 12 **minutes**).

Furthermore, there is definitely room for improvement and development as explained in the following and final section of the thesis.

6.2 Future Work

For a medical robot to co-exist safely with patients and doctors, the doctor should be capable of interfering with the robotic procedure and take control. This is implemented because fully automated robotic systems in clinical environments are banned by law. One method of controlling the robot is using a joystick. A joystick controller interface was developed in this project to control Stormram 2 (refer to A.6 for more information regarding the implementation).

It would also be of great benefit to redesign the homing procedure and implement an **alternat-ive homing/starting mechanism** for future MRI-guided robotic breast biopsy prototypes. Even though, the main function of the end stops described in section 2.2.4.1, is to provide a homing position for Stormram 2, it was difficult to ensure that all 5 end stops are in direct contact with

the ball joints at the same time. The trajectory strategy described in 5.16 should be improved further, since it is preferable for the needle to follow smoother paths. In addition, more number of experiments should be conducted, taking into account lesions with varying **lesion sizes** and breast phantoms with varying **stiffnesses**. Moreover, concerning the software implementation, as described earlier, it is correct to start prototyping with MATLAB, but since the prototyping stage of the control loop is accomplished, the project can now be translated to C++ (to gain 10x to 100x) performance improvement.

Lastly, for a fair comparison between the **current MRI-guided breast biopsy** and the **proposed computer controlled robotic breast biopsy**, it would be preferable if the experiments are conducted under the same conditions of the current procedure, i.e. same MRI-scanner, same scanning sequence, etc.

A Appendix 1

Tip: Make a copy of this document, since this is a manual of the template. This will prevent losing the document while modifying it. (And probably breaking it.)

A.1 InverseKinematics.m

```
function [L1,L2,L3,L4,L5] = InverseKinematics(xe,ye,ze,gamma,alpha)
% Computes the inverse position kinematics of the MRI-compatible breast
% biopsy system (stormram 2.0). The kinematic model describes the static
% relationships between the linear stepper motors and the Cartesian
% position and orienation of the end-effector.
%
00
   Inputs:
8
            xe,ye,ze:
                        The position of the end-effector with ...
   respect of
00
            the robot coordinate system - input positions in mm
8
            gamma: Rotation about the y-axis (orientation of the
            end-effector about the y-axis) - input gamma in degrees
8
00
            alpha: Rotation about the x-axis (orientation of the
            end-effector about the x-axis) - input alpha in degrees
%
8
    Outputs (in mm):
8
00
            L1: The distance between the critical point of the needle
            holder (point C) and the centre of the ball joint (B1)
÷
           L2: The distance between the critical point of the needle
÷
           holder (point C) and the centre of the ball joint (B2)
8
8
            L3: The distance between the critical point of the needle
00
           holder (point A) and the centre of the ball joint (B3)
00
            L4: The distance between the critical point of the needle
%
            holder (point A) and the centre of the ball joint (B4)
00
            L5: The distance between the critical point of the needle
8
            holder (point B) and the centre of the ball joint (B5)
%
%
   Different test cases to verify the inverse kinematics model
8
            % Case 1
00
            xe = 15;
00
            ye = 80;
8
            ze = 90;
            gamma = 0;
8
                         % rotation about y
                        % rotation about x
8
            alpha = 0;
00
            Call Matlab Function:
00
            [L1,L2,L3,L4,L5] = InverseKinematics(15,80,90,0,0);
%
%
           % Case 2
           xe = 10;
÷
            ye = 65;
00
00
            ze = 75;
90
            qamma = 5;
                        % (pi/36 radians) rotation about y
8
            alpha = 0;
                        % rotation about x
8
           Call Matlab Function:
%
            [L1,L2,L3,L4,L5] = InverseKinematics(10,65,75,5,0);
00
8
            % Case 3
8
            xe = 9;
%
            ye = 62;
            ze = 73;
%
00
            gamma = 10;
                         % (pi/18 radians) rotation about y
8
            alpha = -8;
                         %((-45*pi)/(4) radians) rotation about x
```

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 80 biopsy system

```
Call Matlab Function:
8
%
            [L1, L2, L3, L4, L5] = InverseKinematics (9, 62, 73, 10, -8);
% Angle Conversion - Conversion from degrees to radians
gamma = (gamma*pi)/180; % Rotation about y
alpha = (alpha*pi)/180; % Rotation about x
% Measured (Known/Given) Lengths (in mm)
L_AC = 45.83; % Distance between needle holder critical points A ...
   and C
L_AE = 62.43; % Distance between needle holder critical points A ...
   and E
L_AB = 15.07; % Distance between needle holder critical points A ...
   and B
% Position of the centre of Ball Joint B1 (in mm)
% (With Respect to Robot Coordinate System)
xb1 = 45.37401154;
yb1 = 0;
zb1 = -22.5;
% Position of the centre of Ball Joint B2 (in mm)
% (With Respect to Robot Coordinate System)
xb2 = -45.37401154;
yb2 = 0;
zb2 = -22.5;
% Position of the centre of Ball Joint B3 (in mm)
% (With Respect to Robot Coordinate System)
xb3 = 45.37401154;
vb3 = 0;
zb3 = 22.5;
% Position of the centre of Ball Joint B4 (in mm)
% (With Respect to Robot Coordinate System)
xb4 = -45.37401154;
yb4 = 0;
zb4 = 22.5;
% Position of the centre of Ball Joint B5 (in mm)
% (With Respect to Robot Coordinate System)
xb5 = 0;
yb5 = -9.08076118;
zb5 = -63.95477272;
% Position of critical point A (needle holder) with respect to the Robot
% Coordinate System
xa = xe - (L_AE*sin(gamma));
ya = ye + (L_AE*cos(gamma)*sin(alpha));
za = ze - (L_AE*cos(gamma)*cos(alpha));
% Position of critical point C (needle holder) with respect to the Robot
% Coordinate System
xc = xa - (L_AC \star sin(gamma));
yc = ya + (L_AC*cos(gamma)*sin(alpha));
zc = za - (L_AC*cos(gamma)*cos(alpha));
% Position of critical point B (needle holder) with respect to the Robot
% Coordinate System
lambda = 0.3288;
xb = lambda*xc + (1-lambda)*xa;
yb = lambda*yc + (1-lambda)*ya;
```

```
zb = lambda*zc + (1-lambda)*za;
% Equations to be solved for the 5 knowns L1, L2, L3, L4 and L5 are ...
   aiven
% by:
2
% Equation 1
%(xa - xb3)^2 + (ya - yb3)^2 + (za - zb3)^2 == L3^2;
% Equation 2
(xa - xb4)^2 + (ya - yb4)^2 + (za - zb4)^2 = L4^2;
% Equation 3
%(xc - xb1)^2 + (yc - yb1)^2 + (zc - zb1)^2 == L1^2;
% Equation 4
(xc - xb2)^2 + (yc - yb2)^2 + (zc - zb2)^2 = L2^2;
2
% Equation 5
%(xb - xb5)^2 + (yb - yb5)^2 + (zb - zb5)^2 == L5^2;
syms L1 L2 L3 L4 L5
% Solving the 5 equations for the 5 unknowns L1, L2, L3, L4 and L5
Sol1 = solve((xc - xb1)^2 + (yc - yb1)^2 + (zc - zb1)^2 == L1^2,L1);
Sol2 = solve((xc - xb2)^2 + (yc - yb2)^2 + (zc - zb2)^2 == L2^2, L2);
Sol3 = solve((xa - xb3)^2 + (ya - yb3)^2 + (za - zb3)^2 = L3^2,L3);
Sol4 = solve((xa - xb4)^2 + (ya - yb4)^2 + (za - zb4)^2 = L4^2, L4);
Sol5 = solve((xb - xb5)^2 + (yb - yb5)^2 + (zb - zb5)^2 = L5^2, L5);
% Notel: The solve matlab function outputs symbolic objects. In ...
   order to utilise
% the values for further purposes, it is converted to a numeric object
% using the function double
% Note2: Soll consists of two values -L1 (first element of the ...
   vector) and
% L1 (second element of the vector). The same applies for Sol2, ...
   Sol3, Sol4
% and Sol5. Since only positive lengths are utilised, the absolute value
% of the first element is obtained. It is also possible to take the ...
   second
% element directly.
                               % Second Alternative
                              % L1 = double(Sol1(2));
L1 = abs(double(Sol1(1)));
L2 = abs(double(Sol2(1)));
                              % L2 = double(Sol2(2));
L3 = abs(double(Sol3(1)));
                              % L3 = double(Sol3(2));
                              % L4 = double(Sol4(2));
L4 = abs(double(Sol4(1)));
                             % L5 = double(Sol5(2));
L5 = abs(double(Sol5(1)));
end
```

A.2 ForwardKinematics.m

```
function endEffector = ForwardKinematics(L1,L2,L3,L4,L5)
% Computes the forward position kinematics of the MRI-compatible breast
% biopsy system (stormram 2.0). The kinematic model describes the static
% relationships between the linear stepper motors and the Cartesian
% position and orienation of the end-effector. The function ...
    determines the
% end-effector position (xe,ye, ze) based on given lengths (L1, L2, L3,
```

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 82 biopsy system

```
% L4 and L5).
   Inputs (in mm):
8
           L1: The distance between the critical point of the needle
8
           holder (point C) and the centre of the ball joint (B1)
8
           L2: The distance between the critical point of the needle
8
           holder (point C) and the centre of the ball joint (B2)
8
           L3: The distance between the critical point of the needle
00
8
           holder (point A) and the centre of the ball joint (B3)
8
           L4: The distance between the critical point of the needle
8
           holder (point A) and the centre of the ball joint (B4)
           L5: The distance between the critical point of the needle
8
00
           holder (point B) and the centre of the ball joint (B5)
%
% Output (in mm):
00
          endEffector: The position of the end-effector (xe, ye, ze) ...
   with
%
          respect to the robot coordinate system
00
  Different test cases to verify the forward kinematics model
8
           (All dimensions are in mm)
8
2
            % Case 1
%
           L1 = 85.6771;
8
           L2 = 100.3145;
8
           L3 = 85.7221;
%
           L4 = 100.3530;
           L5 = 118.3464;
8
           Call Matlab Function:
8
           endEffector = ForwardKinematics(L1, L2, L3, L4, L5);
8
8
          Case 2
8
          L1 = 79.6239;
2
           L2 = 80.2648;
8
          L3 = 77.3616;
00
          L4 = 82.5363;
00
8
          L5 = 96.4969;
8
          Call Matlab Function:
8
           endEffector = ForwardKinematics(L1, L2, L3, L4, L5);
2
\ 'assignin' function assigns value to variable in specified workspace.
% This is used for debugging purposes, when this function is called, the
% values will be visible in Matlab's workspace after function ...
   exectution.
% If assingin is not used as shown below, only the outputs of the ...
   function
% will be visible
% The lengths L1, L2, L3, L4 and L5 are assigned so that they can be ...
   used
% to solve for the thetas (check the next part of this code)
assignin('base','L1',L1); % Assigns the value L1 to the variable L1 in
                          % Matlab's workspace (denoted by base)
assignin('base','L2',L2); % Assigns the value L2 to the variable L2 in
                          % Matlab's workspace (denoted by base)
assignin('base','L3',L3); % Assigns the value L3 to the variable L3 in
                          % Matlab's workspace (denoted by base)
assignin('base','L4',L4); % Assigns the value L4 to the variable L4 in
                          % Matlab's workspace (denoted by base)
assignin('base','L5',L5); % Assigns the value L5 to the variable L5 in
                          % Matlab's workspace (denoted by base)
```

```
% To obtain the end-effector position, the two new parameters (angles)
% theta 1 and theta 2 should be solved for. Since both equations (2.37)
% and (2.38) are nonlinear, fsolve Matlab function is utilised. The way
% fsolve works is:
% (1) Convert equations (2.37) and (2.38) into functions by simply ...
   shifting
\% (LAC)^2 and (L5)^2 to the left hand side and equating the \ldots
   equations to
\frac{1}{2} zero (f(x) = 0).
% (2) Write down the functions in a separate Matlab Function ...
   (Script). In
% this case the matlab function is named: ThetaSolver
% (3) Select an initial guess for (theta 1) and (theta 2)
% (4) Call the Matlab function (ThetaSolver) where the 2 functions are
% defined
\% (5) The output should be the 2 angles (theta 1) and (theta 2), ...
   which are
% then used to compute the end-effector position
% - refer to Mohamed Essam's MSc Thesis for equations (2.37) and (2.38)
% Initial Guesses for the newly added parameters (theta 1) and ...
   (theta 2)
theta0 = [pi/2 pi/2];
% Call the matlab function (ThetaSolver) where the 2 functions are ...
   defined
\% and solve for the angles (theta 1 and theta 2), theta consists of ...
   (theta
% 1) and (theta 2).
% Output = fsolve(@Script_where_functions_are_defined, initial ...
   conditions)
theta = fsolve(@ThetaSolver,theta0);
% Angle between lines B2-C and B2-B1 (Intermediate Parameter)
q2 = acos(((L2^2) + (90.7480^2) - (L1^2))/(2*L2*90.7480));
% Angle between lines B4-A and B4-B3 (Intermediate Parameter)
q4 = acos(((L4^2) + (90.7480^2) - (L3^2))/(2*L4*90.7480));
% Position of critical point C (needle holder) with respect to the
% Intermediate Coordinate System (B2)
xCB2 = L2 * cos(q2);
                                   % x-position of critical point C ...
    (needle
                                    % holder with respect to Intermediate
                                   % Coordinate System (B2)
yCB2 = L2 * sin(q2) * sin(theta(1));
                                 % y-position of critical point C ...
   (needle
                                   % holder with respect to Intermediate
                                    % Coordinate System (B2)
zCB2 = L2*sin(q2)*cos(theta(1)); % z-position of critical point C ...
   (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B2)
% Position vector of critical point C (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B2)
PCB2 = [xCB2;yCB2;zCB2;1];
```

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 84 biopsy system

```
% Homogeneous matrix transformation from the intermediate coordinate ...
   system
% (B2) to Robot Coordinate System (R)
HB2toR = [1 \ 0 \ 0 \ -45.374; \ 0 \ 1 \ 0 \ 0; \ 0 \ 0 \ 1 \ -22.5; \ 0 \ 0 \ 1];
% Position of critical point C (needle holder) with respect to the
% Robot Coordinate System (R)
PCR = HB2toR*PCB2;
% Indexing is used to access the elements of the vector PCR
xCR = PCR(1); % x-position of critical point C (needle holder) with ...
    respect
              % to the Robot Coordinate System (R)
yCR = PCR(2); % y-position of critical point C (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
zCR = PCR(3); % z-position of critical point C (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
assignin('base', 'xCR', xCR); % Assigns the value xCR to the variable ...
    xCR in
                             % Matlab's workspace (denoted by base)
assignin('base','yCR',yCR); % Assigns the value yCR to the variable ...
    yCR in
                             % Matlab's workspace (denoted by base)
assignin('base','zCR',zCR); % Assigns the value zCR to the variable ...
    zCR in
                             % Matlab's workspace (denoted by base)
% Position vector of critical point A (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B4)
xAB4 = L4 \star cos(q4);
                                    % x-position of critical point A ...
   (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B4)
yAB4 = L4*sin(q4)*sin(theta(2)); % y-position of critical point A ...
    (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B4)
zAB4 = L4*sin(q4)*cos(theta(2)); % z-position of critical point A ...
   (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B4)
% Position vector of critical point A (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B4)
PAB4 = [xAB4; yAB4; zAB4; 1];
% Homogeneous matrix transformation from the intermediate coordinate ...
    svstem
% (B4) to Robot Coordinate System (R)
HB4toR = [1 0 0 -45.374; 0 1 0 0; 0 0 1 22.5;0 0 0 1];
```

```
% Position of critical point A (needle holder) with respect to the
% Robot Coordinate System (R)
PAR= HB4toR*PAB4;
% Indexing is used to access the elements of the vector PAR
xAR = PAR(1); % x-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
yAR = PAR(2); % y-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
zAR = PAR(3); % z-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
assignin('base','xAR',xAR); % Assigns the value xAR to the variable ...
   xAR in
                            % Matlab's workspace (denoted by base)
assignin('base','yAR', yAR); % Assigns the value yAR to the variable ...
   yAR in
                            % Matlab's workspace (denoted by base)
assignin('base','zAR',zAR); % Assigns the value zAR to the variable ...
   zAR in
                            % Matlab's workspace (denoted by base)
% Measured/Known Lengths
LAB = 15.07; % Distance between critical points A and B (needle holder)
LAC = 45.833; % Distance between critical points A and C (needle holder)
% Ratio of the known lengths (LAB) and (LAC)
lambda1 = LAB/LAC;
% Position of critical point B (needle holder) with respect to the
% Robot Coordinate System (R)
xBR = lambda1*xCR + (1-lambda1)*xAR; % x-position of critical point A
                                     % (needle holder) with respect ...
                                        to the
                                     % Robot Coordinate System (R)
yBR = lambda1*yCR + (1-lambda1)*yAR; % y-position of critical point A
                                     % (needle holder) with respect ...
                                         to the
                                     % Robot Coordinate System (R)
zBR = lambda1*zCR + (1-lambda1)*zAR; % z-position of critical point A
                                     % (needle holder) with respect ...
                                        to the
                                     % Robot Coordinate System (R)
assignin('base', 'xBR', xBR); % Assigns the value xBR to the variable ...
   xBR in
                            % Matlab's workspace (denoted by base)
assignin('base','yBR',yBR); % Assigns the value yBR to the variable ...
   yBR in
                            % Matlab's workspace (denoted by base)
assignin('base','zBR',zBR); % Assigns the value zBR to the variable ...
   zBR in
```

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 86 biopsy system

```
% Matlab's workspace (denoted by base)
% Measured/Known Lengths
LAE = 62.430852; % Distance between critical points A and E (needle ...
   holder)
% Ratio of the known lengths (LAB) and (LAC)
lambda2 = LAE/(LAE + LAC);
% Position of end-effector (E) with respect to the Robot Coordinate ...
   System
% (R)
xER = (xAR - (lambda2*xCR))/(1-lambda2); % x-position of ...
   end-effector (E)
                                         % with respect to the Robot
                                         % Coordinate System (R)
yER = (yAR - (lambda2*yCR))/(1-lambda2); % y-position of ...
   end-effector (E)
                                         % with respect to the Robot
                                         % Coordinate System (R)
zER = (zAR - (lambda2*zCR))/(1-lambda2); % z-position of ...
   end-effector (E)
                                         % with respect to the Robot
                                         % Coordinate System (R)
assignin('base','xER', xER); % Assigns the value xBR to the variable ...
   xBR in
                            % Matlab's workspace (denoted by base)
assignin('base','yER',yER); % Assigns the value yBR to the variable ...
   yBR in
                            % Matlab's workspace (denoted by base)
assignin('base','zER', zER); % Assigns the value zBR to the variable ...
  zBR in
                            % Matlab's workspace (denoted by base)
% Output of the function
endEffector = [xEWCS;yEWCS;zEWCS]; %End-effector position x,y and z
end
```

A.3 ThetaSolver.m

```
function F = ThetaSolver(theta)
% Solves the 2 non-linear equations (2.37) and (2.38) [Refer to Mohamed
% Essam's MSc Thesis] for the two unknowns (theta 1) and (theta 2). The
8
8
  Inputs:
8
           theta: Initial guesses of (theta 1) and (theta 2) in radians
8
   Output:
8
           F: Equations of m
% 'evalin' Matlab function accesses the Matlab workspace (denoted by ...
   base)
L1 = evalin('base','L1'); % Assign Matlab's workspace (base) ...
   variable L1's
                          % value to L1
```

```
L2 = evalin('base','L2'); % Assign Matlab's workspace (base) ...
    variable L2's
                          % value to L2
L3 = evalin('base','L3'); % Assign Matlab's workspace (base) ...
    variable L3's
                          % value to L3
L4 = evalin('base','L4'); % Assign Matlab's workspace (base) ...
    variable L4's
                          % value to L4
L5 = evalin('base','L5'); % Assign Matlab's workspace (base) ...
   variable L5's
                          % value to L5
\ Angle between lines B2-C and B2-B1 (Intermediate Parameter)
q2 = acos(((L2^2) + (90.7480^2) - (L1^2))/(2*L2*90.7480));
% Angle between lines B4-A and B4-B3 (Intermediate Parameter)
q4 = a\cos(((L4^2) + (90.7480^2) - (L3^2))/(2*L4*90.7480));
% Position of the centre of Ball Joint B5 with Respect to Robot ...
   Coordinate
% System (R)
xb5R = 0;
yb5R = -9.08076118;
zb5R = -63.95477272;
% Position of critical point C (needle holder) with respect to the
% Intermediate Coordinate System (B2)
xCB2 = L2 \star cos(q2);
                                    % x-position of critical point C ...
    (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B2)
yCB2 = L2*sin(q2)*sin(theta(1)); % y-position of critical point C ...
   (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B2)
zCB2 = L2*sin(q2)*cos(theta(1)); % z-position of critical point C ...
    (needle
                                    % holder with respect to Intermediate
                                    % Coordinate System (B2)
% Position vector of critical point C (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B2)
PCB2 = [xCB2;yCB2;zCB2;1];
% Homogeneous matrix transformation from the intermediate coordinate ...
    system
% (B2) to Robot Coordinate System (R)
HB2toR = [1 \ 0 \ 0 \ -45.374; \ 0 \ 1 \ 0 \ 0; \ 0 \ 0 \ 1 \ -22.5; \ 0 \ 0 \ 1];
% Position of critical point C (needle holder) with respect to the
% Robot Coordinate System (R)
PCR = HB2toR*PCB2;
% Indexing is used to access the elements of the vector PCR
```

Designing, implementing and integrating a controller for the MRI-compatible robotic breast 88 biopsy system

```
xCR = PCR(1); % x-position of critical point C (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
yCR = PCR(2); % y-position of critical point C (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
zCR = PCR(3); % z-position of critical point C (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
% Position vector of critical point A (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B4)
xAB4 = L4 \star cos(q4);
                                   % x-position of critical point A ...
   (needle
                                   % holder with respect to Intermediate
                                   % Coordinate System (B4)
yAB4 = L4*sin(q4)*sin(theta(2)); % y-position of critical point A ...
   (needle
                                   % holder with respect to Intermediate
                                   % Coordinate System (B4)
zAB4 = L4 * sin(q4) * cos(theta(2));
                                  % z-position of critical point A ...
   (needle
                                   % holder with respect to Intermediate
                                   % Coordinate System (B4)
% Position vector of critical point A (needle holder) with respect ...
   to the
% Intermediate Coordinate System (B4)
PAB4 = [xAB4; yAB4; zAB4; 1];
% Homogeneous matrix transformation from the intermediate coordinate ...
   system
% (B4) to Robot Coordinate System (R)
HB4toR = [1 0 0 -45.374; 0 1 0 0; 0 0 1 22.5;0 0 0 1];
% Position of critical point A (needle holder) with respect to the
% Robot Coordinate System (R)
PAR= HB4toR*PAB4;
% Indexing is used to access the elements of the vector PAR
xAR = PAR(1); % x-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
yAR = PAR(2); % y-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
zAR = PAR(3); % z-position of critical point A (needle holder) with ...
   respect
              % to the Robot Coordinate System (R)
% Measured/Known Lengths
LAB = 15.07; % Distance between critical points A and B (needle holder)
LAC = 45.833; % Distance between critical points A and C (needle holder)
% Ratio of the known lengths (LAB) and (LAC)
```

```
lambda1 = LAB/LAC;
% Position of critical point B (needle holder) with respect to the
% Robot Coordinate System (R)
xBR = lambda1*xCR + (1-lambda1)*xAR; % x-position of critical point A
                                     % (needle holder) with respect ...
                                         to the
                                     % Robot Coordinate System (R)
yBR = lambda1*yCR + (1-lambda1)*yAR; % y-position of critical point A
                                     % (needle holder) with respect ...
                                        to the
                                     % Robot Coordinate System (R)
zBR = lambda1*zCR + (1-lambda1)*zAR; % z-position of critical point A
                                     % (needle holder) with respect ...
                                        to the
                                     % Robot Coordinate System (R)
% Function 1
%(xAWCS - xCWCS)^2 + (yAWCS - yCWCS)^2 + (zAWCS - zCWCS)^2 - LAC^2
% Function 2
%(xBWCS - xb5)^2 + (yBWCS - yb5)^2 + (zBWCS - zb5)^2 - L5^2
 Define the non-linear equations (functions (f(x)=0) to be solved using
% the matlab function fsolve.m
F = [(xAR - xCR)^2 + (yAR - yCR)^2 + (zAR - zCR)^2 - LAC^2;
    (xBR - xb5R)^2 + (yBR - yb5R)^2 + (zBR - zb5R)^2 - L5^2];
end
```

A.4 PCB Schematic



Figure A.1: PCB Schematic

GUI - step by step

A.5

1 Setup Delet Start Stop MANUAL Start Debug Reset Valves Stop Debug CONTROL PANEL (SINGLE ACTUATION) Main Switch 1 A c 2 A c Direction Length L1 3 A 6 4 B 6 5 B 6 Length ake Another Me Stop & Reset L1 Increase/De AUTOMATIC Debug L2 Value L3 STORMRAM 2.0 1 14 -ц Г12 Г13 Е14 Г15 Make Move L5 0 Delta L1 Delta L2 Delta L3 Delta L4 Make Another Move Length Decrease u Stop & Reset и и и и и в L1 ... Close Main Switch Reset Actuator Close Main Switch Start 0 Demo 2 3 4 CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) L3 L3 Length Length Length L3 Increase/Decrease Decrease Increase/Decrease Decrease Increase/Decr ~ Dec rease Decrea Value Value 105 Value 99 ~ Increase RESET PANEL 105 RESET PANEL Make Move Make Move 104 Make Move Initiate Reset Initiate Reset 103 Make Another Move Make Another Mov Make Another M Length 102 Length 101 Stop & Reset Stop & Reset L1 Stop & Reset 100 Close Main Switch **Close Main Switch Reset Actuator** 99 Close Main Switch **Reset Actuator** 98 ß 6 7 CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) Length 13 Length L1 L3 Length Increase/D Decrease Increase/Decrease L1 Decrease Increase/I L2 Value Value Decrease 99 Value Increase RESET PANEL L4 RESET PANEL Make Move Make Move Make Move L5 Initiate Reset Initiate Reset Make Another Mo Make Another Move Make Another Move Length Length Length Stop & Reset 11 Stop & Reset L1 Stop & Reset L1 **Close Main Switch Reset Actuator** Close Main Switch Reset Actuator Close Main Switch Reset Actuator 8 9 10 CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) Length L2 Length L3 L3 Length ~ Increase/D Inci Increase/Decrease Increase Increase/Decrease Increase Value Value 104 100 ~ Value 104 ~ 100 RESET PANEL Make Move RESET PANEL 101 102 Make Move Make Move Initiate Res Initiate Rese Make Another Move Make Another Move 103 Length Make Another Move Length Stop & Reset Stop & Reset L1 Stop & Reset L1 105 **Close Main Switch** Close Main Switch Reset Actuator 106 Reset Actuator Close Main Switch ന Ð Æ CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) CONTROL PANEL (SINGLE ACTUATION) L3 Length L3 Length Length L3 Increase/Decrease Increase Increase/Decrease Increase/Decrease Increase Increase Value Value 104 \sim Value 104 104 RESET PANEL RESET PANEL RESET PANEL Make Move Make Move Make Move Initiate Reset Initiate Reset Initiate Reset Make Another Move Make Another Move Make Another Move Length Length Length Stop & Reset Stop & Reset L3 Stop & Reset L3 L1 R L1 L2 Close Main Switch Close Main Switch Reset Actuato Close Main Switch Reset Actuator

Figure A.2: Control Panel (Single Actuation) Steps



Figure A.3: Three-piston stepper motor mechanism controlled using 3 valves 1A, 1B and 1C

A.6 Differential Kinematics - Joystick Control

From equations Equations (2.3)–(2.7):

$$L_1 = \sqrt{(x_C - x_{B1})^2 + (y_C - y_{B1})^2 + (z_C - z_{B1})^2}$$
(A.1)

$$L_2 = \sqrt{(x_C - x_{B2})^2 + (y_C - y_{B2})^2 + (z_C - z_{B2})^2}$$
(A.2)

$$L_3 = \sqrt{(x_A - x_{B3})^2 + (y_A - y_{B3})^2 + (z_A - z_{B3})^2}$$
(A.3)

$$L_4 = \sqrt{(x_A - x_{B4})^2 + (y_A - y_{B4})^2 + (z_A - z_{B4})^2}$$
(A.4)

$$L_5 = \sqrt{(x_B - x_{B5})^2 + (y_B - y_{B5})^2 + (z_B - z_{B5})^2}$$
(A.5)

The five equations can be written in the following form:

$$\mathbf{L} = f(\mathbf{q}) \tag{A.6}$$

where $\mathbf{L} = [L_1 L_2 L_3 L_4 L_5]^T$ and $\mathbf{q} = [x_e y_e z_e \gamma \alpha]^T$. Furthermore, the

ſ

$$\frac{\partial \mathbf{L}}{\partial \mathbf{q}} = \mathbf{J}(\mathbf{q}) \tag{A.7}$$

where J(q) (Jacobian) is equivalent to:

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \frac{\partial L_1}{\partial x_e} & \frac{\partial L_1}{\partial y_e} & \frac{\partial L_1}{\partial z_e} & \frac{\partial L_1}{\partial \gamma} & \frac{\partial L_1}{\partial \alpha} \\ \frac{\partial L_2}{\partial x_e} & \frac{\partial L_2}{\partial y_e} & \frac{\partial L_2}{\partial z_e} & \frac{\partial L_2}{\partial \gamma} & \frac{\partial L_2}{\partial \alpha} \\ \frac{\partial L_3}{\partial x_e} & \frac{\partial L_3}{\partial y_e} & \frac{\partial L_3}{\partial z_e} & \frac{\partial L_3}{\partial \gamma} & \frac{\partial L_3}{\partial \alpha} \\ \frac{\partial L_4}{\partial x_e} & \frac{\partial L_4}{\partial y_e} & \frac{\partial L_4}{\partial z_e} & \frac{\partial L_4}{\partial \gamma} & \frac{\partial L_4}{\partial \alpha} \\ \frac{\partial L_5}{\partial x_e} & \frac{\partial L_5}{\partial y_e} & \frac{\partial L_5}{\partial z_e} & \frac{\partial L_5}{\partial \gamma} & \frac{\partial L_5}{\partial \alpha} \end{bmatrix}$$
(A.8)

Equation A.7 is reformulated in the following manner:

This final result can be explained with a simple example, if the operator wants to move the needle only 1 mm in the positive z-direction, the following computation is evaluated:

$$\partial \mathbf{L} = \mathbf{J}(\mathbf{q})\partial \mathbf{q} \tag{A.9}$$

$$= \mathbf{J}(\mathbf{q}) \begin{bmatrix} \partial x_e & \partial y_e & \partial z_e & \partial \gamma & \partial \alpha \end{bmatrix}$$
(A.10)

$$\begin{bmatrix} \partial L_1 & \partial L_2 & \partial L_3 & \partial L_4 & \partial L_5 \end{bmatrix}^T = \mathbf{J}(\mathbf{q}) \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$$
(A.11)

Evaluating the equation above, results in the changes in lengths to be made in order to achieve a change of 1 mm in the positive z-direction. The aforementioned is linked to the joystick in such a way: when the operator moves the joystick forward (+ve *z*), the Jacobian matrix is multiplied by the change in end-effector position and orientation vector $(\partial \mathbf{q} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T)$, to obtain the change in lengths required to achieve it. Another example: rotate joystick clockwise (-ve *y*), then the vector is equivalent to $\partial \mathbf{q} = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \end{bmatrix}^T$, refer to figure A.3. Note: A joystick panel was added to the GUI, 4 to enable also control through the joystick. The change in orientations γ and α were not handled at this stage. However, it can be easily extended to include both of them.

Bibliography

An, Y. Y., S. H. Kim, B. J. Kang and J. H. Lee (2013), Usefulness of magnetic resonance imaging-guided vacuum-assisted breast biopsy in Korean women: a pilot study, *World Journal of Surgical Oncology*, vol. 11, pp. 1–7, ISSN 1477-7819, doi:10.1186/1477-7819-11-200.

http://dx.doi.org/10.1186/1477-7819-11-200

BEKINK, L. (2014), *THE MARKET VALUE OF A NOVEL HIGH-TECH PROCEDURE TO DIAGNOSE BREAST CANCER L.*, Master's thesis, University of Twente, the Netherlands.

cijfersoverkanker (2015), Nederlandse Kankerregistratie, http://www.cijfersoverkanker.nl/, [Online; accessed 25-August-2016].

Fracheboud, J., H. de Koning, R. Boer, J. Groenewoud, A. Verbeek, M. Broeders, B. van Ineveld, J. Hendriks, A. de Bruyn, R. Holland and P. van der Maas (2001), Nationwide breast cancer screening programme fully implemented in the Netherlands, *The Breast*, vol. 10, pp. 6–11, ISSN 0960-9776, doi:http://dx.doi.org/10.1054/brst.2000.0212. http:

//www.sciencedirect.com/science/article/pii/S0960977600902121

- Hauth, E., H. Jaeger, J. Lubnau, S.Maderwald, F. . Otterbach and R.Kimmig (2008), MR-guided vacuumassisted breast biopsy with a handheld biopsy system: clinical experience and results in postinterventional MR mammography after 24h., *Eur Radiol*, vol. 18, pp. 168–76.
- Industry, D. (2016), Spool valve / manually-controlled / pneumatic / regulating, http://www.directindustry.com/prod/az-pneumatica/ product-19702-475856.html, [Online; accessed 25-August-2016].
- Lee, J., E. Halpern, E. Rafferty and G. Gazelle (2009), Evaluating the Correlation between Film Mammography and MRI for Screening Women with Increased Breast Cancer Risk, *Academic Radiology*, **vol. 16**, pp. 1323–1328, ISSN 1076-6332, doi:10.1016/j.acra.2009.05.011.
- Lu, W., L. Jansen, W. Post, J. Bonnema, J. van de Velde and G. de Bock (2009), Impact on survival of early detection of isolated breast recurrences after the primary treatment for breast cancer: a meta-analysis, *Breast Cancer Research and Treatment*, **vol. 114**, pp. 403–412, ISSN 0167-6806, doi:10.1007/s10549-008-0023-4.
- StcValve (2016), Solenoid Valves & Air-Pilot Activated Valves, https://stcvalve.com/ Pneumatic_Solenoid_valve_specifiaction-4V300.htm, [Online; accessed 25-August-2016].
- Su, H., G. A. Cole and G. S. Fischer (2012), *High-Field MRI-Compatible Needle Placement Robots for Prostate Interventions: Pneumatic and Piezoelectric Approaches*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 3–32, ISBN 978-3-642-23363-0, doi:10.1007/978-3-642-23363-0_1.

http://dx.doi.org/10.1007/978-3-642-23363-0_1

- T., L. B., E. A. G. and T. N. V. (2004), Design of an MRI-Compatible Robotic Stereotactic Device for Minimally Invasive Interventions in the Breast, *Biomech Eng 126*, pp. 458–465.
- Veltman, J., C. Boetes, T. Wobbes, J. G. Blickman and J. O. Barentsz (2005), Magnetic Resonance-Guided Biopsies and Localizations of the Breast: Initial Experiences Using an Open Breast Coil and Compatible Intervention Device, *Investigative Radiology*, pp. 379–384.
- V.Groenhuis and S.Stramigioli (2016), Laster-Cutting Pneumatics, *IEEE/ASME Transactions on Mechatronics*, **vol. 21**, pp. 1604–1611.
- Yang, B., S. Roys, U.-X. Tan, M. Philip, H. Richard, R. P. Gullapalli and J. P. Desai (2014), Design, Development, and Evaluation of a Master-slave Surgical System for Breast Biopsy Under

Continuous MRI, *Int. J. Rob. Res.*, vol. 33, pp. 616–630, ISSN 0278-3649, doi:10.1177/0278364913500365. http://dx.doi.org/10.1177/0278364913500365