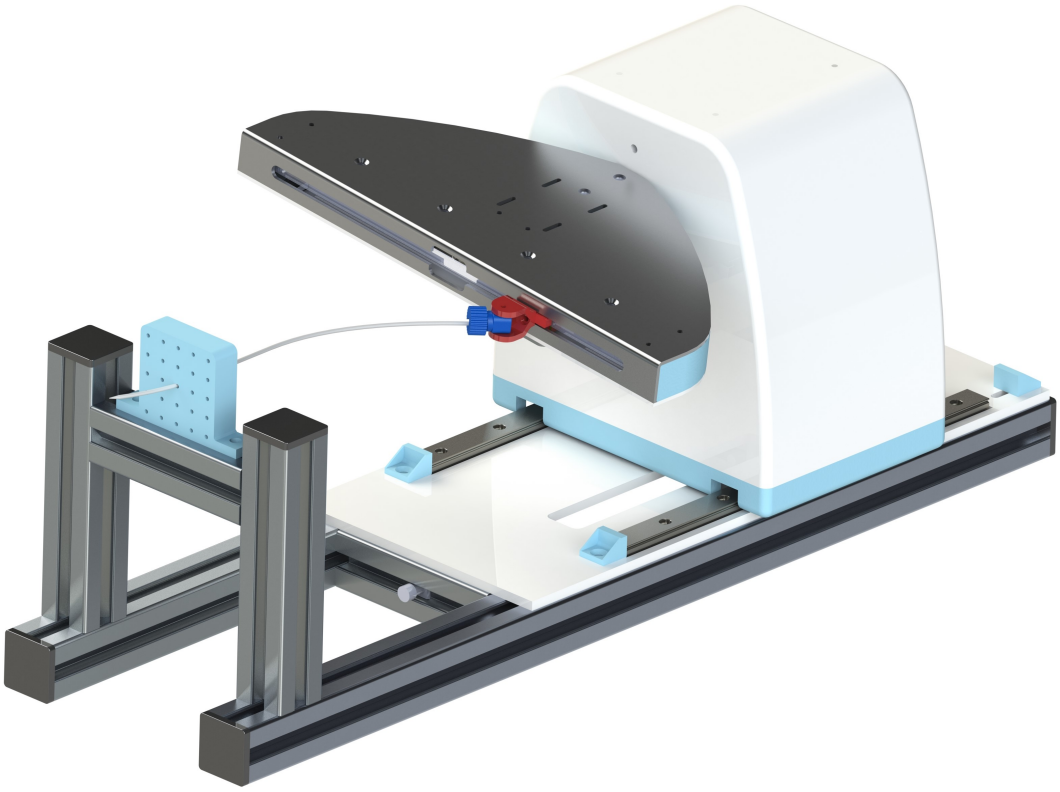
Design Report

by

Tetsuo Martynowicz
Pepijn van Kampen
Rolf Bavelaar
Niels Dee



Steedle

TU Delft

Date : 4-6-2021
Supervisor: Martijn de vries

***Abstract*— To improve the accuracy of the needle insertion during High Dose Rate prostate Brachytherapy, a manually controlled compliant mechanism steerable needle was developed [1]. Operating this steerable needle by hand offers some disadvantages such as limited control and visualization possibilities. Higher needle placement accuracy and visualization by means of MRI could greatly improve the effectiveness of the treatment and could be achieved using a robotically actuated system.**

This paper explores the possibilities of a robotically actuated system for the control of this steerable needle. The developed system uses two linear axes and one rotating axis, which combined with the developed software allows the operator to control the needle with high precision and in an intuitive way.

After careful evaluation and testing it was found that the system can position the needle tip in 3D-space with sub 0,3 mm precision. A drawback of the system is the buckling caused by a lack of support on the distal side of the needle. Fortunately, further testing proved that this issue can be resolved by, for example, using a thicker template or a stiffer needle.

These solutions combined with the high accuracy and intuitive control make the first results look very promising and shows that robotic actuation of this steerable needle has the potential to become a viable option and a feasible substitute for manual control.

I. Background

A. Current procedure

Prostate cancer is the most common form of cancer among men, one in ten males gets diagnosed with prostate cancer at sometime in their life [2]. High Dose Rate Brachytherapy (HDR BT) is a medical procedure to treat prostate cancer by temporarily placing a radioactive source via needles in or close to the prostate gland. HDR BT offers many advantages like precise dose delivery, reduction of tissue damage and quick recovery over other treatment alternatives like Low Dose Rate (LDR) Brachytherapy and External Beam Radiotherapy (ERBT) [3].

There are several different methods for needle placement and treatment planning. All involve a surgeon inserting needles transperineally by using a template that is placed against the perineum and image-guidance under general or spinal anesthetic. Transrectal Ultrasound (TRUS) is often used to acquire images for localizing the needle tips and treatment planning but Magnetic Resonance Imaging (MRI) and computed tomography (CT) imaging are also options.

Typically, 14-18 straight needles are placed in the periphery of the prostate gland with the tips in or beyond the prostate base. After a treatment plan is made and the needles are satisfactorily positioned, the inner-needle (stylet) is removed from the outer-needle (sleeve). Then, the radioactive source is loaded into the needles for several minutes and removed after the treatment is completed. With an experienced team, ultrasound based procedures can be completed within two hours and CT based procedures can take up to six hours [4].

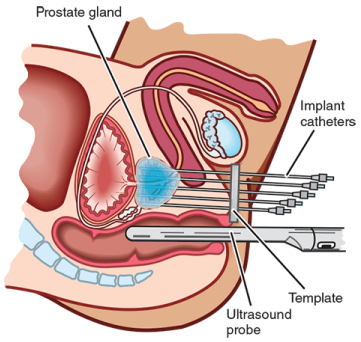


Fig. 1: HDR Brachytherapy [5]

B. Obstacle identification

During HDR BT needles need to go around or through various obstacles. The first set of obstacles are the tissues the needle needs to penetrate to get to the prostate. In order to go from skin surface to the prostate, the following tissues need to be passed:

1) Skin
2) Perlineal fascia
3) Perineal membrane
4) corpus spongiosum
5) Transverse perineal muscles
6) Fibromuscular band of the prostate

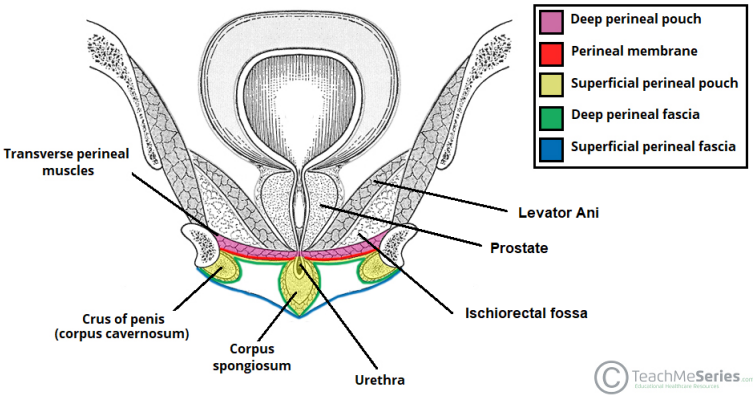


Fig. 2: Male perineum [6]

It is important to minimize the damage to the tissue and to avoid the most critical areas. The following tissues have shown to be linked to toxicities after being damaged by needle insertion [7]:

1) The urethra: a channel that goes through the prostate and the corpus spongiosum
2) Internal pudendal arteries: arteries that supply blood to the penis
3) Cavernous nerves: nerves in the region around the prostate
4) Bulbourethral gland: gland responsible for producing pre-ejaculate
5) Prostatic plexus: innervation of the prostate

Another obstacle that needs to be taken into account is the pubic arch. Typically a prostate is about 30 mm high, 40 mm wide and 20 mm deep [8]. In some cases however, the prostate is much larger which can cause the prostate to overlap with the pubic arch. Pubic arch interference or (PAI) can make it really difficult for the needles to reach the prostate.

Due to these obstacles the current HDR BT procedures are limited in their effectiveness. Currently sensitive tissue, like the corpus spongiosum, has to be penetrated by the needles to reach certain spots in the prostate. Even then it is unfeasible to reach all spots in the prostate due to obstacles which, like the urethra, due to toxicities, should be avoided to be damaged at all. The current procedure also relies on multiple retractions per insertion of a needle to alter its direction slightly, which results in unnecessary trauma.

C. Steerable needle by M. de Vries

HDR Brachytherapy has become essential to effectively treat prostate cancer and to reduce operating time, treatment costs and healthy tissue receiving unnecessary radiation. However, there is still room for improvement to increase the effectiveness of HDR BT. According to clinicians of the University Medical Centre Utrecht and Erasmus Medical Centre, the current procedure could be improved by enhancing the needle placement accuracy and ease of needle placement [1]. This could improve the effectiveness of the treatment and reduce tissue trauma by decreasing needle insertions and retractions. Recently, the use of steerable needles has been researched to improve the potency of the treatment. The robotic actuation system discussed in this report will make use of a steerable needle designed by M. de Vries. This needle is specifically designed for HDR BT interventions. In this section the performance of this needle will be discussed.

The needle, shown in figure 3 consists of a nitinol rod with two perpendicular cuts along its length, splitting the needle in 4 quarters. The quarters are connected to eachother at the front and back end of the needle. This needle works as a compliant mechanism, which means the needle gains it's mobility from the elastic flexibility of the material rather than traditional joints. [9] When the back end of the needle is displaced in a certain the direction, the tip of the needle deflects in the opposite direction.

The sharp side of the needle is called the distal side and the side at which the surgeon manipulates the needle is called the proximal side. A hollow polymer needle, also referred to as the sleeve, is slid over the metal rod. After inserting the stylet and sleeve into the prostate, the stylet is removed, leaving the sleeve in place. This sleeve is used with an afterloader to direct the radioisotope.

For the needle to be steerable two constrained points along the needle are required. This is done by guiding the needle through a template. The needle was tested to a maximum deflection of 70 degrees at the proximal side, which resulted in a deflection of 20 degrees on the distal side. Greater deflections were not tested to sustain the prototype, although they might be possible. The diameter of the stylet is $1.44 + \frac{0.00}{-0.04}$ mm and the sleeve has a diameter of approximately 2 mm.

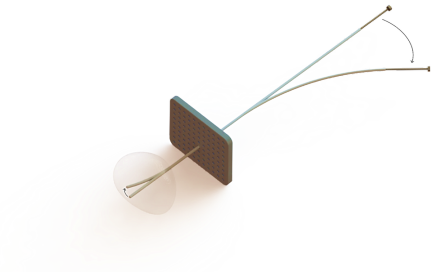


Fig. 3: Steerable needle by M. de Vries [1]

After experimenting with the needle it was found that the 'natural' paths of the back end of the needle can be accurately approximated by three sets of circles, each set with a different center, **A**, **B** and **C**, as can be seen in figure 4. To acquire the natural paths of the needle, the needle was first locked inside a template and a force applied perpendicular to the back end of the needle, thereby bending the needle. The position of the back of the needle is recorded by drawing multiple strokes, shown in dark red in the figure. Next the needle is inserted 10mm further, the back end displaced and its position recorded again. This is done until the needle was inserted for 180 mm into the template. The resulting dashed lines, are be approximated by circles where the further the needle is inserted the closer the center of the circles lie towards the template. Thus center **A** can be used for low, **B** for medium and **C** for high insertion. This test was done with the standard nitinol needle and a needle made from spring steel, both test yielded the same results.

The circular approximations of the back end of the needle was an important step for the designing process, as it allowed for a computer aided compensation program to be built, as described in section II-C.2, instead of a mechanical solution for the problem described in section II-A.

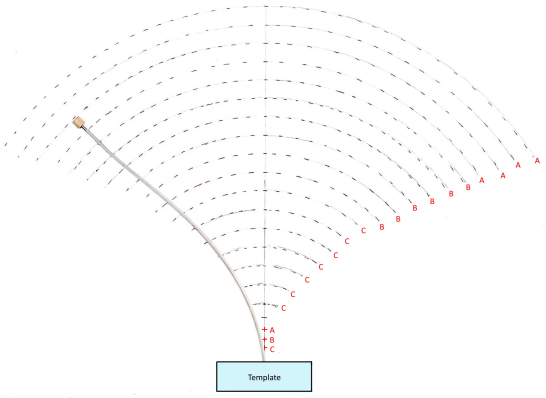


Fig. 4: The back end of the needle follows a path that can be approximated by three sets of circles, each set with a different center: **A**, **B**, and **C**

D. Research objective

Experiments done with a prototype of the needle showed its ability to perform the required steerability to avoid a large part of the sensitive tissues. However, as M.M. Nobel stated in his thesis: 'Like many surgical instruments, the efficiency of the instrument will largely depend on the aptitude of the surgeon. Whether a trained surgeon will be able to steer in two planes with similar accuracy as can be achieved in a single plane, will have to be examined.' Furthermore, the controlling of the steering turned out to be a concern in the prototype, as stated in the thesis. 'The needle tends to buckle during insertion making insertion more difficult. A more robust control mechanism will be necessary to enable the surgeon to effectively steer the needle. Another concern is whether the steering capability enables more accurate needle placement in practice.' [1] These possible weaknesses of the steerable needle can be solved by controlling the steering with a robotic actuation system, which might also improve the accuracy of the needle placement.

Therefore the objective of the research is formulated as:

The development and evaluation of an omnidirectional robotic actuation system for a compliant mechanism steerable needle in high dose rate prostate Brachytherapy.

An accurate relation between the deflection at the proximal side of the needle and the distal side of the needle is still unknown. Eventually a relation can be determined to calculate the displacement at the distal side of the needle with a given displacement at the proximal side of the needle. This relation can directly be used for control of the tip of the needle using a feed forward loop. Another steering solution is to constantly determine the error between the location of the tip of the needle and a desired location which can be used in a feedback loop. However, both are outside the scope of this design project. In this project the main emphasis lies on achieving precise displacement of the proximal and thus distal side of the needle which is imperative for both feed-forward and feedback control. The prototype will undergo three types of precision tests, a practical tests and a functionality test, which will make use of the requirements in appendix C, to verify the prototype's workings and identify its shortcomings.

II. Design

Figure 5 shows the final design together with a life sized human model in a practical environment. This gives a better understanding of the dimensions of the system and its orientation with respect to the patient. A general coordinate system is added to indicate the directions that are referred to further on in this report. Hereby the Z-direction is coincident to the needle in its neutral position and the X direction is perpendicular to the bed as shown in the figure.

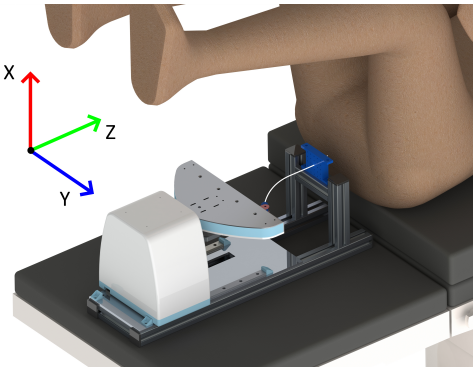


Fig. 5: The design in operation with the general coordinate system

A. Working principle

To impose a displacement of the needle tip, the needle must be displaced in the opposite direction at the extremity of the proximal side (see figure 7). The needle is secured at this point to avoid interference to the other needles as required in requirement MHR-10 in Appendix C. This also means that the insertion force must be applied at the extremity of the proximal side, resulting in multiple possible buckling modes for which some solutions are described in section II-B.1. A design consisting of two linear axes and one rotating axis, as shown in figure 6, was found to be the best option in terms of functionality and compactness. This arrangement allows for the extremity at the proximal side of the needle to move everywhere within a circular plane parallel to the transverse plane. The two linear axes **(A)** and **(C)** combined with the rotating axis **(B)** allow for the distal side of the needle to move everywhere within a cylindrical space. This space fits the natural movement of the needle better than a cubic space, which is related to a system with three linear axes.

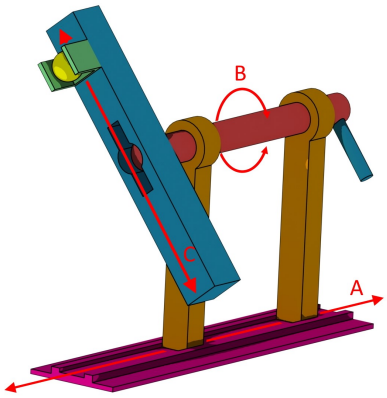


Fig. 6: Fundamental principle: First linear axis **(A)**, Rotating axis **(B)**, Second linear axis **(C)**

When bending the proximal side of the needle it is important to do this in a way that maximizes the displacement at the tip of the needle. Two situations are considered in figure 7: The green line represents a needle of which the proximal

side is held perpendicular to the black line by applying a moment at point **A**. The red line allows point **A** to rotate freely, resulting in a different needle shape. Both needle shapes result in virtually equal displacements of the needle tip, yet due to the external moment applied at point **A** the internal stress of the needle following the green needle path is higher. Therefore the design has a joint that allows for the proximal side of the needle to rotate freely as shown in figure 6. A more detailed description of the needle clamp can be found in section II-B.3. Another problem that should be taken into account is the displacement **(D)** in the Z-direction that occurs when moving the needle in the Y-direction (see figure 7). If this displacement is not prevented or compensated for it would retract the distal side of the needle out of the patient, causing redundant added tissue damage. It was decided to compensate for this displacement **(D)** by means of software which will be further elaborated in section II-D.

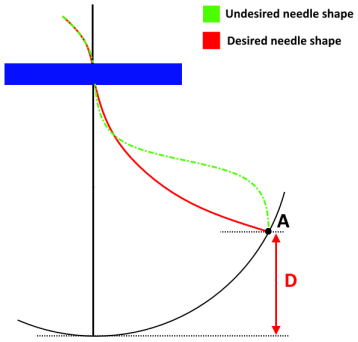


Fig. 7: Schematic of needle displacement: extremity at the proximal side of the needle **(A)**, displacement distance **(D)** that occurs when bending the needle

B. Mechanical Design

1) Frame

The frame of the prototype in figure 8 is a lightweight aluminium structure that consists of several connected extrusion profiles. These profiles guarantee a stiff frame to support the forces applied by the moving cart. As described in section II-A the needle is likely to buckle while inserting because the insertion force is applied to the extremity of the proximal side of the needle, whilst the reaction forces occur at the template. Optionally, to prevent buckling an intermediate support can be attached to the extrusion profiles.

2) The cart

The cart consists of a 3D printed box with a removable cover that is mounted on top of the linear guide bearings (**B**) in figure 8. In figure 9 is shown that all electronic components are integrated within the cart. Furthermore, the cart contains two steppers motors (**A**) which provide the rotation of the main axis (**D**) and the translation of the needle in the direction of the patient (Z-direction). To make sure the first motor can deliver enough torque to rotate the main axis (**D**) a gear ratio of 16/100 between the motor pulley and main axis pulley (**B**) is chosen. The belt between these two pulleys can be tightened by moving the stepper motor down or by moving the motor mount to the left.

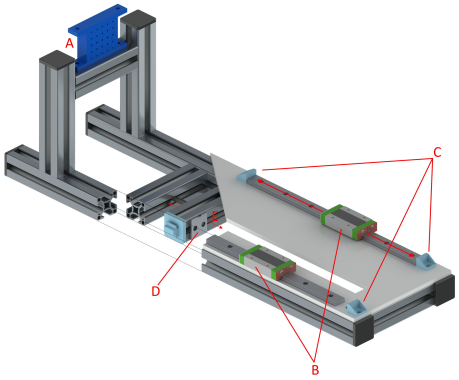


Fig. 8: Sub-assembly of frame: Template (**A**), Linear guide bearings (**B**), Mechanical stops (**C**), Belt tension mechanism (**D**)

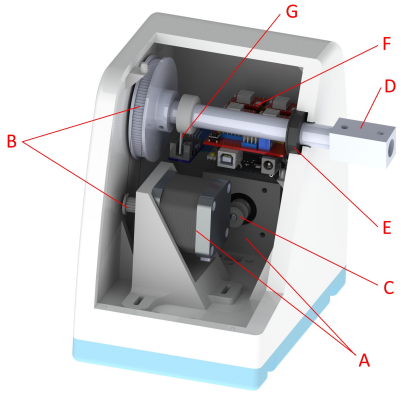


Fig. 9: Sub-assembly of base (a): Stepper motors (**A**), Pulley's main axis (**B**), Pulley (**C**), Main axis (**D**), Bearing (**E**), Arduino (**F**), Sensor (**G**)

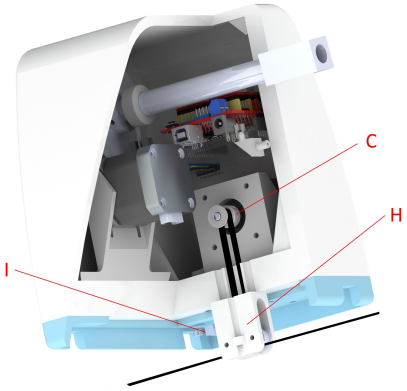


Fig. 10: Sub-assembly of base (b): Pulley (**C**), Belt guide (**H**), Limit switch (**I**)

3) The arm

The arm in figure 11 is designed to be compact, stiff and low on backlash to make the movement of the proximal side of the needle as accurate as possible. The arm consists of two aluminium plates joined together by an extrusion profile (**A**) on the front side and the main axis (**B**) on the back side. Part **F** together with part **G** functions as a rotation joint which is necessary due to the effect described in section II-A. The needle is clamped by inserting its back end in part **G** and afterwards tightening the cap (**H**) onto it. This cap has a slot cut out of it with roughly the width of the needle to allow for the needle to be tightened even when it is already inserted in the template.

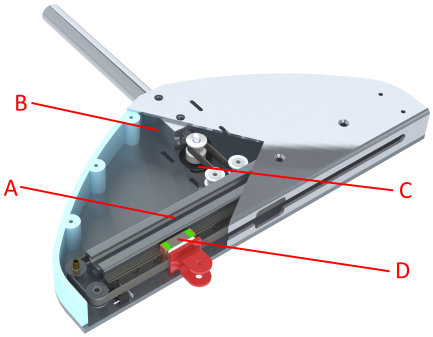


Fig. 11: Sub-assembly of the arm: Extrusion profile (**A**), Main axis (**B**), Stepper motor (**C**), Sliding block (**D**)

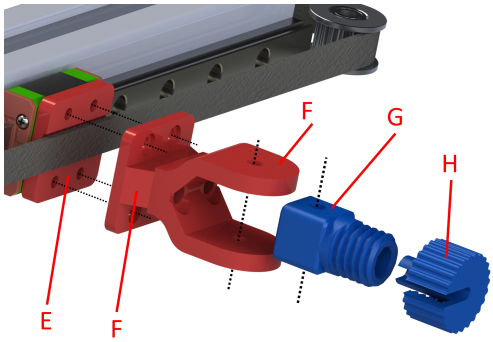


Fig. 12: Close-up of needle clamp: Timing belt clamp (**E**,**F**), Needle clamp (**G**), Cap (**H**)

C. Software Design

1) User interface

To correctly insert the needle in an intuitive way a user interface was made. The user interface, shown in figure 13, functions as follows. First of all, the user needs to connect the system to their computer via Bluetooth. If the pairing is successful this will be indicated in the first line under settings. After connecting the user needs to home the system by pressing the Home button. Homing is crucial as it gives the motor controller a clear reference point.

Once the system is homed the needle tip can be moved to the desired position. This can be done in two ways. The first option is to simply click on the target. The target is a 2D projection of the 3D movement space of the needle tip. When the user clicks on the target, the motion compensation program calculates the corresponding position in the three dimensional space. The positions can also be entered manually via the settings panel. Once the needle tip is deflected to the desired position the user can click on the insert button to insert the needle a set distance.

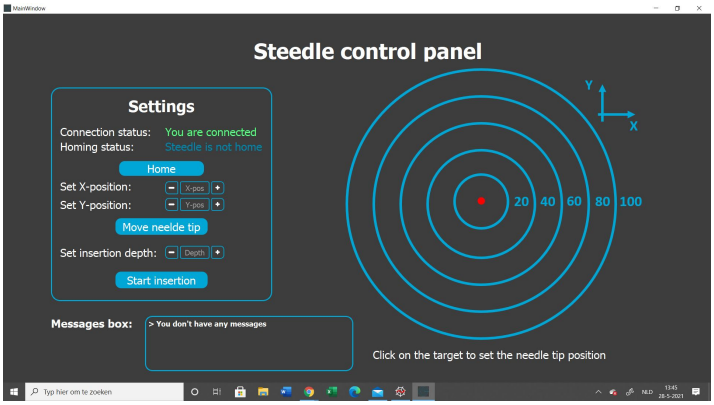


Fig. 13: User interface Steedle

2) Motor compensation program

When the needle is deflected its extremity does not only move to the side, but also slightly in the Z-direction (see figure 14). This undesired movement, which as described in section I-C follows a circular path, must be compensated to prevent the needle from being retracted out of the patient. Furthermore, it was found that the extremity of the needle must be pushed tangent to the needles shape whilst inserting the needle, as this is the direction in which the needle is least likely to buckle. This means that the path of the insertion force needs to follow the shape of the needle.

It was decided to approximate the shape of the needle by a parabola, because it is relatively easy to calculate and has a shape fairly similar to that of the needle. Which, as explained above, limits the tendency of the needle to buckle. Based on these assumptions the calculations for the motor compensation program are performed as follows:

Every time a user enters an X and Y percentage in the user interface the values are converted to a deflection on the proximal side of the needle. This is done by calculating the maximum possible deflection, which depends on the insertion depth of the needle, and multiplying this number with the entered percentage. This multiplication combined with the angle corresponding to the entered X and Y percentages gives the actual position of the extremity of the needle in the XY-plane.

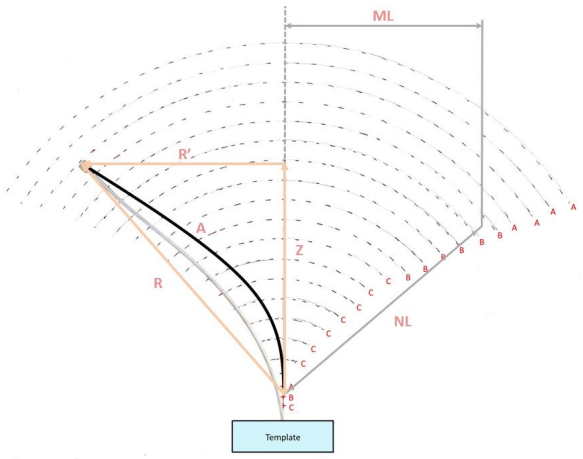


Fig. 14: Needle tip path with math illustration where **R** represents the radius of the path of the needle's extremity, **R'** the amount of bending of the needle's extremity, **A** the approximation of the needle shape, **Z** the vertical distance between the needle's extremity and the midpoint of the circle, **ML** Mechanical deflection limit and **NL** Needle deflection limit.

The maximum deflection of the needle highly depends on the insertion depth and can be divided into two parts. For the first part of the insertion the maximum deflection is determined by the mechanical limit (**ML**) of the system, which is 100 mm. It was found that after the needle is inserted 60 mm the maximum deflection is no longer determined by the system, but instead determined by the deflection limit of the needle (**NL**). The deflection limit at every insertion depth is visualized by the grey lines in figure 14.

After the position of the extremity of the needle is determined in the XY-plane, the corresponding Z-position can be calculated using the Pythagoras theorem. Equation 1 applies and can be used to calculate the position of the needles extremity in 3D-space:

$$Z = \sqrt{R^2 - R'^2} \tag{1}$$

$$X_{2d} = \frac{R_{start}}{Z^2_{start}} Z^2 \tag{2}$$

$$Y_{3d} = \begin{bmatrix} \frac{R_{start}}{Z^2_{start}} Z^2 \\ 0 \\ Z \end{bmatrix} \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The needles shape is approximated using the standard form of a parabola. Given the two known points, the needles extremity and the origin, and the fact that the slope of the line is zero at the origin, means that three equations can be compiled and the equation 2 for the needle shape in the XZ-plane can be derived. In the formula start refers to the position at the start of the insertion. To get the equation 3 for the needle shape in 3D-space the matrix is multiplied with a rotation matrix using the angle derived from the X and Y position of the needles extremity.

The aforementioned findings are visualized in figure 15. The gray lines show one half of the movement space of the needles extremity. When the needle tip moves from one point to another it follows a path over the gray dome. An example path is visualized by the blue line. The orange and green line represent the shape of the needle in start and end position of an arbitrary movement of the needles extremity respectively. The shown needle lines do not represent the actual needle shapes, but the ones approximated by a parabola.

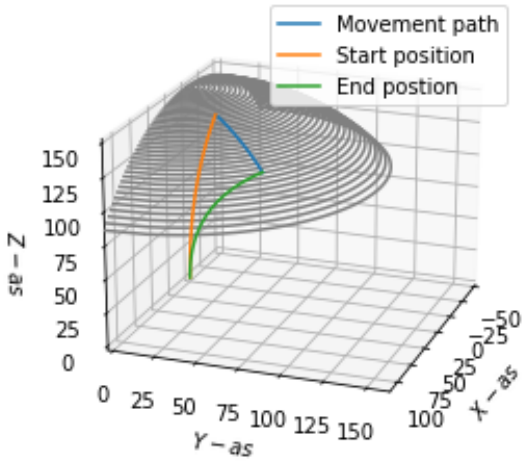


Fig. 15: Visualization of the movement path of the needle

D. *Mathematics to movement*

The arrays used to make the visualization shown in figure 15 can be seen as discrete versions of the above mentioned equations. Because of the limited memory capacity of the Arduino the arrays currently only consist of 30 elements. The arrays are eventually converted into motor positions. The difference between two consecutive points in the array are then put into a separate array converted to steps and send to the Arduino. The Arduino will wait until all the elements of the array are received and thereafter move the stepper motors. For this it uses the Multi-stepper library. This library automatically adjusts the speed of the steppers so that all three of them arrive at the desired points in the array at the same time. The full source code can be found in appendix D.

III. Method

A. *Functionality tests*

Prior to the design process a list of requirements, as shown in Appendix C, was established. Multiple functionality tests are carried out to test the capabilities of the system to satisfy the requirements. These tests are relatively short and the results are mostly binary: the requirement is satisfied or it is not. The results of the functionality tests are discussed in chapter IV.

B. Precision tests

The prototype will undergo three different precision tests in which the needle tip is steered to the same location ten times in sequence. These tests are performed in air to minimize the factors that influence the final position of the needle tip. This test shows the consistency of the prototype in steering the needle to a predetermined location.

The test setup used to execute the precision tests is shown in figure 16. The template used has standard dimensions, a thickness of 15 mm and hole diameter of 2,0 mm.

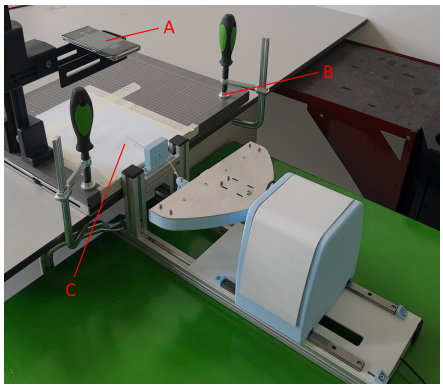


Fig. 16: Test setup: Camera (**A**), which records the needle tip positions, is fixed on the frame (**B**) to which a graph paper (**C**) is attached.

1) 2D precision test with constant insertion depth

With the first test only the displacement of the proximal side of the needle is varied in the Y-direction, while the insertion depth remains constant. To realize this the stepper motor that provides rotation of the arm is off. The goal of this test is to measure the precision when the system repeats a specific deflection. The test is performed as follows and repeated ten times:

1) Starting in the home position, the needle is inserted 60 mm in the Z-direction.
2) The proximal side of the needle is deflected to + 80%: 80 mm of its maximum deflection in the positive Y-direction.
3) A photo is taken of the deflection on the distal side of the needle.
4) The proximal side of the needle is deflected to - 80%: 80 mm of its maximum deflection in the negative Y-direction.
5) The needle is returned to + 80% and the sequence repeats.

2) 2D precision test with varying insertion depth

With the second precision test the displacement of the proximal side of the needle varies within a horizontal plane while the insertion depth also varies. To realize this two of the three stepper motors move, whilst the third stepper motor, which provides rotation of the arm, is off. The goal of this test is to measure the precision when the system repeats a specific deflection and insertion simultaneously multiple times. The test is performed as follows and repeated ten times:

1) Starting in the home position, the needle is inserted 30 mm in the Z-direction.
2) The proximal side of the needle is deflected to + 90%: 90 mm of its maximum deflection in the positive Y-direction.
3) A photo is taken of the deflection on the distal side of the needle.
4) The proximal side of the needle is deflected to + 70%: 70 mm of its maximum deflection in the positive Y-direction.
5) The needle is inserted again for another 40 mm.
6) Another photo is taken of the deflection on the distal side of the needle at the second position.
7) The needle is returned to its neutral (home) position and the sequence repeats.

3) 3D precision test

The final precision test engages all three stepper motors by which the precision of the entire system is tested. To measure the precision in a 3D space a second camera is added to visualize the precision in the X- and Z-direction. The test is performed as follows and repeated ten times:

1) Starting in the home position, the needle is inserted 70 mm in the Z-direction.
2) The proximal side of the needle is deflected +60%: 54 mm in the X-direction and +60%: 54 mm in the Y-direction.
3) A photo is taken with both cameras of the deflection on the distal side of the needle.
4) The needle is returned to its neutral (home) position and the sequence repeats.

C. Practical Test

To test the practical potential of the prototype the needle is steered in a 12% wt. gelatin phantom. To optimally mimic human tissue the phantom is kept between 6-10 °C. If the temperature of the phantom rises above this temperature, it becomes significantly softer which can affect the tests. The testing is done by moving the needle in small steps and adjusting the deflection of the needle with every step until a certain target is reached as accurately as possible. The goal of these tests is to prove the practical potential of the design.

For the practical test the same setup is used as for the precision tests, with the same standard template dimensions, a thickness of 15 mm and hole diameter of 2,0 mm. The gelatin phantom is placed in front of the template on top of the graph paper. The phantom is put on a thin sheet of transparent polycarbonate so the graph paper is clearly visible from above through the gelatin and the polycarbonate. The prototype is tested for its practical potential by letting a human interact with the prototype, using the human for control, and testing if a certain position within a phantom can

be reached as accurately as possible. This is done by drawing six points on a piece of graph paper and continuously inserting and changing the displacement on the proximal side of the needle to reach one of these points. The test is performed as follows:

1) Starting in the home position, the needle is inserted 10 mm in the positive Z-direction.
2) The proximal side of the needle is displaced to a desired displacement.
3) The needle is inserted a small step of approximately 10mm.
4) The course of the needle is interpreted, and the proximal displacement is changed if necessary.
5) The needle is inserted for another small step and the sequence repeats until the desired final position is reached.

D. Buckling test

Buckling of the needle is observed to be a reoccurring problem that must be solved for the system to have practical value, as stated in requirement MHR-3. Three variables are separately tested for decreasing the tendency to buckle: 1. Template hole diameter 2. Template thickness 3. Needle stiffness. These variables are relatively simple to adjust in the current procedure.

The same test setup is used for the buckling tests as for the precision tests and practical test, but without camera since buckling can be identified by eye. Note that to optimally mimic human tissue the phantom is kept between 6-10 °C, if the temperature of the phantom rises above this temperature, it becomes significantly softer which can affect the buckling tests. For this test three template hole diameters ranging from 1,9 - 2,1 mm are tested at first. Then the template thickness is varied from 10 mm to 35 mm with 5 mm increments. At last, a stiffer needle made from spring steel is tested. This test makes use of a standard test setup where one of the variables is changed at a time. The standard setup is: a nitinol needle and a 15mm thick template with a 2,0 mm hole. For each variable the following test is conducted:

1) The needle is inserted 40 mm into the phantom.
2) The needle is inserted for another 40 mm with a certain deflection: potential buckling is observed.
3) The needle is fully retracted and again inserted 40 mm at a different location.
4) The sequence is repeated for 0%, 30%, 60% and 90% deflection at the proximal side.

For the template thickness test, the length of the needle at the proximal side is kept equal. This means that for these tests the needles insertion into the phantom is less for a higher template thickness.

IV. Results

A. Results functionality tests

Several simple functionality tests proved that the systems satisfies almost all must have requirements. Requirement MHR-11, stating that the tip must be able to displace 10 mm at an insertion depth of 90 mm, was comfortably achieved as the maximum deflection is 30 mm at this insertion depth, see appendix: B. Unfortunately, requirement MHR-3: The system must be able to prevent buckling to the needle, is not satisfied by the prototype. However, multiple variables can be changed to decrease the chance of buckling as discussed in section III-D and proven in section IV-F. More solutions to prevent the needle from buckling are proposed in the discussion (V). Should have requirement SHR-1, stating the system should be integrated with the current HDR BT procedure with minimal changes required, is satisfied due to the system grabbing the, already existing, needle at the very back end, being relatively small in size and allowing for manual control.

B. Results 2D precision test with constant insertion depth

Figure 17 shows the two needle tip locations (**A** and **B**). A detailed position of the needle tip for both locations is visualized in the red boxes. Each red dot represents the location of the needle tip from one single measurement. Combining the ten measurements results in the scatter plots for both locations. Note that in some boxes less then 10 dots are visible, this is caused by coinciding locations. As seen in the figure, a very high precision is achieved, both measurements at location **A** and **B** show a scatter of less then 0,3×0,3 mm.

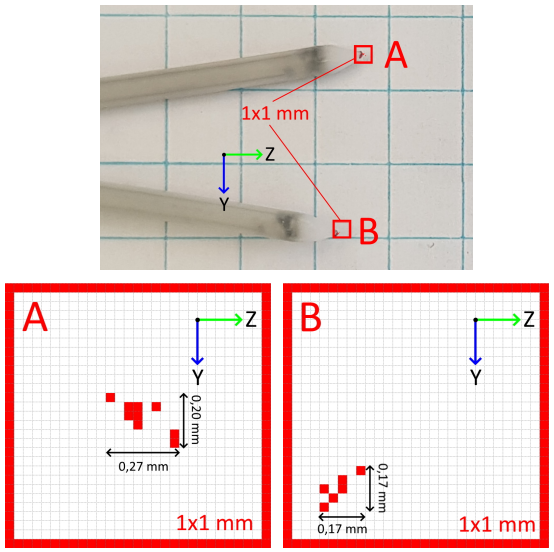


Fig. 17: Needle tip locations of 2D precision test with constant insertion depth: **A**, with a scatter of 0,27×0,20 mm, and **B**, with a scatter of 0,17×0,17 mm.

C. Results 2D precision test with varying insertion depth

Figure 18 shows the two needle tip locations (**A** and **B**). A detailed position of the needle tip for both locations is visualized in the red boxes. Each red dot represents the location of the needle tip in one single measurement. Combining the ten measurements results in the scatter plots

for both locations. Note that in some boxes less then 10 dots are visible, this is caused by coinciding locations. As seen in the figure, a very high precision is achieved, both measurements at location **A** and **B** show a scatter of less then 0,2×0,2 millimeters.

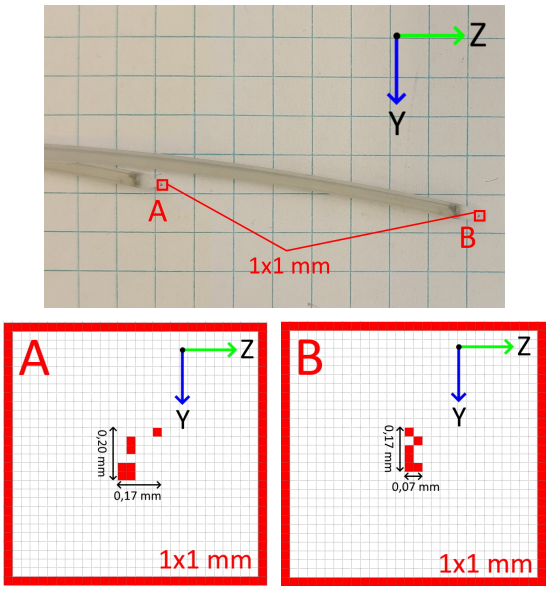


Fig. 18: 2D precision depth with varying insertion depth, needle tip locations **A**, with a scatter of 0,17×0,20 mm, and **B**, with a scatter of 0,07×0,17 mm.

D. Results 3D Precision test

Figure 19 and figure 20 show the detailed position of the needle tip locations in the YZ-plane (**A**) and XZ-plane (**B**) respectively. Each red dot represents the location of the needle tip in one single measurement. Combining the ten measurements results in the scatter plots for both locations. Note that in some boxes less then 10 dots are visible, this is caused by coinciding dots. As seen in the figure, a very high precision is achieved, both measurements at location **A** and **B** show a scatter of less then 0,3×0,3 millimeters.

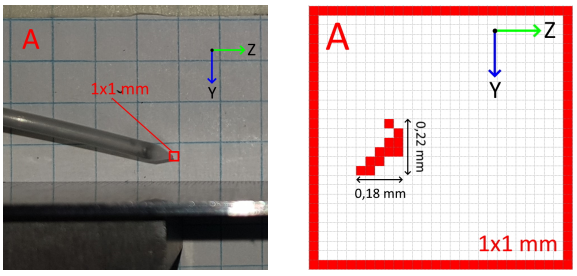


Fig. 19: 3D precision test, needle tip location in YZ-plane, with a scatter of 0,18×0,22 mm

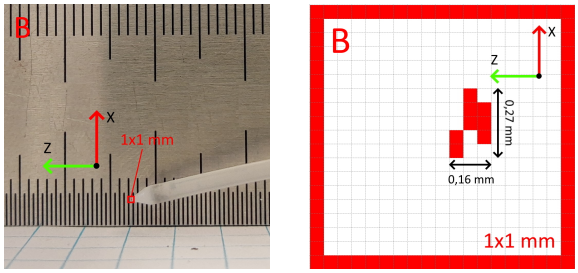


Fig. 20: 3D precision test, needle tip location in XZ-plane, with a scatter of 0,16×0,27 mm

E. Results practical test

The test results of the practical test are shown in figure 21. The figure shows that the six desired targets were almost all achieved with high accuracy. Target **D** and **E** were achieved just barely, it can therefore be concluded that these target are just within the reach of the system. Reaching target **B**, having a deflection of 10 mm with an insertion depth of 90 mm, immediately satisfied requirement MHR-11. Note that during the test the needle was supported by hand to prevent buckling. This test proves that the system has practical potential, since untrained users were able to steer the needle accurately towards multiple targets within a tissue phantom.

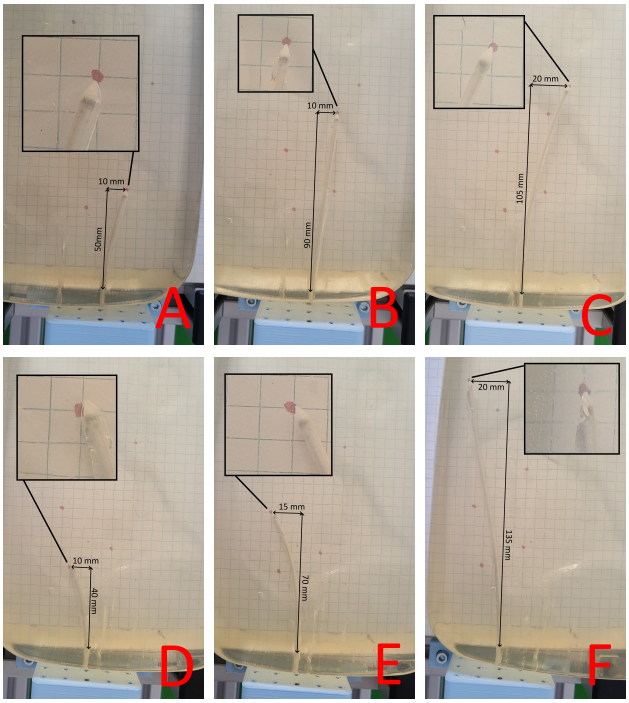


Fig. 21: Practical test, showing six different targets being reached inside a phantom by the needle

F. Results buckling Test

From the results of the buckling test (table I and table II) it can be concluded that a larger hole diameter in the template and a larger template thickness decreases the tendency of the needle to buckle. A stiffer, spring steel needle, was also tested for buckling. This needle, contrary to the nitinol needle, did not buckle with deflections of 60% and 90%, both needles did not buckle for a 30% deflection.

Diameter \ Deflection	0%	30%	60%	90%
2,1 mm	No	No	No	No
2,0 mm	No	No	Yes	Yes
1,9 mm	No	Yes	Yes	Yes

TABLE I: Buckling test with varying template hole diameters (template thickness of 15 mm and nitinol needle)

Thickness \ Deflection	0%	30%	60%	90%
10 mm	No	No	Yes	Yes
15 mm	No	No	Yes	Yes
20 mm	No	No	Yes	Yes
25 mm	No	No	Yes	Yes
30 mm	No	No	No	No
35 mm	No	No	No	No

TABLE II: Buckling test with varying template thickness and deflection (template hole diameter of 2,0 mm and nitinol needle)

V. Discussion

The results of the functionality, precision and practical tests show that the prototype is able to perform basic maneuvers and is able to displace the needle tip in three dimensions with high precision. The lowest precision is found in the 2D precision test with constant insertion depth, where a scatter within a space of 0,27 mm$\times$0,20 mm is recorded. This measurement indicates that there is minimal play in the system. The other precision tests, showing even less scatter, indicate that the system can move the needle tip precisely within a three dimensional space.

The results of the functionality tests show that the prototype meets most of the must have and should have requirements. Because the needle occasionally buckles requirement MHR-3 is not fully met. However, test results from the buckling test in section III-D show that several variables can easily be changed to decrease buckling significantly. Three solutions for buckling of the needle have been tested with positive results. The simplest solution would be using a stiffer needle, but this also affects its steerability and thus functionality. Another possible solution could be the use of an extruder at the template that pushes the needle forward by means of two rotating driver wheels. A major disadvantage of this solution is that it could easily interfere with the already inserted needles, conflicting with requirement MHR-10. Finally, using a better estimation of the shape of the needle that is closer to reality than the currently used parabola, mounting a thicker template, increasing the template hole size or mounting an extra intermediate support are satisfactory solutions as well.

These first results look very promising and we suggest that more tests are conducted to gain full insight in the capabilities and potential of this design. This would require more testing in other phantoms like porcine abdomen, testing with ultrasound visual feedback, and performing more difficult maneuvers in which obstacles are to be avoided.

Furthermore, implementing some form of feedback control and replacing any ferromagnetic parts by non-magnetic parts would make this design suitable for use in combination with Magnetic Resonance Imaging (MRI), which can provide far better imaging than Trans Rectal Ultrasound (TRUS). We believe that proper visualisation is of tremendous importance for the realisation of robotically actuated HDR Brachytherapy steerable needles.

This paper establishes that the robotic actuation of a compliant mechanism steerable needle used for HDR Brachytherapy has practical potential. The amount of displacement with the more extensively researched asymmetric bevel tip needle is achieved by varying the rotational speed of the needle, resulting in high dependency on the tissue it is passing through [10]. In contrary, the compliant mechanism steerable needle by M. de Vries has a straight input-output relation between the displacement at the proximal and distal side. This needle can supply a displacement which is more independent of the tissue it is passing through. We believe that it is, due to aforementioned points, more predictable and therefore better suited for robotic actuation combined with feedback control than a bevel tip needle.

Additionally, the needle and system could be used for many other applications as well. Some examples are: Low Dose Rate Brachytherapy, where accurate placement of seeds is required. It could also be applied in other glands or organs such as the bladder, where Brachytherapy can spare the bladder which improves the quality of life of the patient. Other possible applications are: brain surgery, accurate biopts and drug delivery.

VI. Conclusion

The described system using a combination of two linear axes and one rotating axis has proven to be a suitable way of controlling a steerable needle. The high three dimensional precision and user-friendly control provide it some serious benefits over manual steering and indicate that robotically actuated needle steering is feasible and with extended research can be considered for High Dose Rate Brachytherapy procedures.

APPENDIX

A. Design Methodology

To develop a final design that maximally satisfies the requirements, the following design methodology is applied. First the design space is investigated by drafting a morphological chart. This chart contains all working principles deducted from the requirements combined with a large number of solutions per working principle. Combinations of individual solutions from the morphological chart are used to form initial concepts. To cover more of the design space, the ACCREx method [11] is applied. With these insights three main concepts are developed further and modelled in a 3D CAD software to get a better understanding of their feasibility and limitations. Finally these main concepts are evaluated using the weighted scoring method. A final concept resulting from this method is then further elaborated upon to create a potentially functional design. To examine the functionality of the design a prototype is built and its accordance to the requirements is tested. For testing, gelatin is used as a phantom to represent human-tissue. This process is repeated until a satisfactory design has been established.

B. Requirements proof

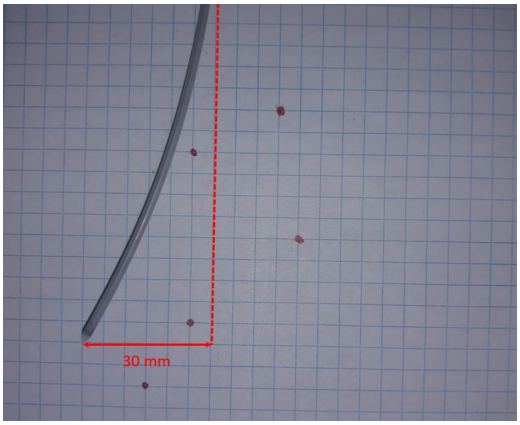


Fig. 22: Maximum deflection at 90 mm insertion

C. Requirements

Table III specifies the requirements that were determined based on the required functionality. The requirements are prioritized using the MoSCoW method[1]. Each requirement is paired with a justification, which can be used to trace the origin of each requirement. Additionally, each requirement received an ID which will be used throughout the report and documentation to refer to a specific requirement.

TABLE III: Requirements and justification

ID	Requirement	Justification
Must haves	This list contains aspects that must be functional in the final product in order to deliver a working product	
MHR-1	The system must utilise the existing stylet, sleeve and template provided	Changing the current procedure might hinder adoption of the system
MHR-2	The system must be able to move the needle in the sagittal and coronal plane simultaneously	To accurately place a needle steering in both the sagittal and coronal plane are required, steering in one plane would increase retractions and increased tissue trauma [1]
MHR-3	The system must be able to prevent buckling to the needle when penetrating human tissue	According to Nobel buckling of the needle can be an issue [1]
MHR-4	The system must ensure the tip of the stylet and the tip of the sleeve are always coincident	If the tip of the stylet is not coincident with the tip of the sleeve the needle will have unpredictable behaviour. For example the sleeve could buckle
MHR-5	The system must be able to move the needle to reach a minimum depth of 90mm	The insertion depth of a standard HDR BT needle is 80 to 90 mm [12]
MHR-6	The system must move the needle without a human as the actuation source	The system should move the needle accuractly in two planes simultaneously, therefore much training is required if this is done by surgeons [1]
MHR-7	The system must limit the movement of the needle to prevent plastic deformation of the needle	If the needle deforms plastically the behaviour of the needle will become unpredictable. The plastic deformation might also not allow for the stylet to retract without damaging the tissue
MHR-8	The system must provide sufficient force for the needle to penetrate gelatin phantom representing soft tissue	The needle must overcome the resistance caused by the tissue of the body to reach the prostate
MHR-9	The system must be able to decrease the amount of insertions needed to effectively reach the desired location	Decreasing the amount of insertions also decreases tissue damage
MHR-10	The system must not limit other, already placed and to be placed, needles in the same template	If the system limits the placement of needles it will decrease the effectiveness and ease of the therapy and therewith decrease the likeliness of adoption
MHR-11	The system must be able to displace the tip of the needle 10mm with an insertion depth of 90mm	The insertion depth of a standard HDR BT needle is 80 to 90 mm [12], from this depth a displacement of 10mm is desired.
Should haves	This list contains aspects that should be functional in the final product in order to further improve upon the product	
SHR-1	The system should be integrated with the current HDR BT procedure with minimal changes required	This will endorse the adoption of the steerable needle actuation system

[1]Prioritization method where the categories: must have, should have, could have and won't have, are used

TABLE III: Requirements and justification

ID	Requirement	Justification
SHR-2	The system should not lengthen the procedure time of the HDR BT prostate needle implantation, which currently is approximately 30 to 45 minutes	Increasing procedure time in HDR BT will decrease cost-effectiveness and patient outcome. This would hinder adoption of the steerable needle [1]
SHR-3	The system should be able to move the needle more accurately than with manual steering	It is important for the system to be more accurate than a surgeon for likeliness of adoption of the system
Could haves	This list contains a series of stretch goals that could be implemented	
CHR-1	The system could be designed so that it can be used inside a MRI machine	Compatibility with a MRI machine adds more functionality to the system, which might help with the adoption of the system
Won't haves	This list contains a couple of aspects that the robot will not have in this iteration	
WHR-1	The system wont use a feedback loop using an observed position of the needle	This project focuses on the actuation of the needle, another project runs parallel to focus on feedback of the needle

D. Python code

User interface source code:

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'Steedle.ui'
#
# Created by: PyQt5 UI code generator 5.11.3
#
# WARNING! All changes made in this file will be lost!

import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QAction
import pyautogui as pg


from Motor_controller import motor_controller

Controller = motor_controller()
const = 365
const2 =372
_translate = QtCore.QCoreApplication.translate


#Set up empty list to display messages:
Messages = ""

button_style = ("QPushButton::pressed"
                "{"
                "border: 2px solid;"
                "border-color: rgb(255, 255, 255)"
                "}"

                "QPushButton::hover"
                "{"
                "background-color: rgb( 0, 134, 172);"
                "}"

                "QPushButton{\n"
                "border-radius: 15px;\n"
                "border: none;"
                "background-color: rgb( 0, 166, 214);\n"
                "color: rgb(255, 255, 255);"
                "}")

button_style2 = ("QPushButton::pressed"
                "{"
                "border: 4px solid;"
                "border-color: rgb( 255, 255, 255)"
                "}"

                "QPushButton::hover"
                "{"
                "border-color: rgb( 0, 134, 172);"
                "}"

                "QPushButton{\n"
                "background-color: rgb(61, 61, 60);\n"
                "border: 2px solid rgb(0,166,214);\n"
                "border-bottom-right-radius: 4px;\n"
                "border-top-right-radius: 4px;\n"
                "border-bottom-left-radius: 10px;\n"
                "border-top-left-radius: 10px;\n"
                "image: url(Minus_arrow.png);"
                "}")

button_style3 = ("QPushButton::pressed"
```

```
                "{"
                "border: 4px solid;"
                "border-color: rgb( 255, 255, 255)"
                "}"

                "QPushButton::hover"
                "{"
                "border-color: rgb( 0, 134, 172);"
                "}"

                "QPushButton{\n"
                "border: 2px solid rgb(0,166,214);\n"
                "border-bottom-right-radius: 10px;\n"
                "border-top-right-radius: 10px;\n"
                "border-bottom-left-radius: 4px;\n"
                "border-top-left-radius: 4px;\n"
                "image: url(Plus_arrow.png);"
                "}")


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        #MainWindow
        app.aboutToQuit.connect(self.closeEvent)
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize( 1900, 1080)
        font = QtGui.QFont()
        font.setPointSize(18)
        MainWindow.setFont(font)
        MainWindow.setWindowIcon(QtGui.QIcon("TargetV2.png"))

        #Centralwidget
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setStyleSheet("background-color: rgb(61, 61, 60);")
        self.centralwidget.setObjectName("centralwidget")

        #Steedle_control_panal_T
        self.steedle_control_panel_T = QtWidgets.QLabel(self.centralwidget)
        self.steedle_control_panel_T.setGeometry(QtCore.QRect(590, 50, 800, 100))
        font = QtGui.QFont()
        font.setPointSize(35)
        font.setBold(True)
        font.setWeight(75)
        self.steedle_control_panel_T.setFont(font)
        self.steedle_control_panel_T.setStyleSheet("color: rgb(255, 255, 255);")
        self.steedle_control_panel_T.setAlignment(QtCore.Qt.AlignCenter)
        self.steedle_control_panel_T.setObjectName("steedle_control_panel_T")

        self.Setting_panel = QtWidgets.QWidget(self.centralwidget)
        self.Setting_panel.setGeometry(QtCore.QRect(130, 200, 600, 580))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.Setting_panel.setFont(font)
        self.Setting_panel.setStyleSheet("color: rgb(255, 255, 255);\n"
                                         "border: 3px solid rgb(0, 166, 214);\n"
                                         "border-radius: 25px;")
        self.Setting_panel.setObjectName("Setting_panel")
        self.Connection_status_T = QtWidgets.QLabel(self.Setting_panel)
        self.Connection_status_T.setGeometry(QtCore.QRect(25, 100, 260, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Connection_status_T.setFont(font)
        self.Connection_status_T.setStyleSheet("color: rgb(255, 255, 255);\n"
                                               "border: None;\n"
                                               "")
        self.Connection_status_T.setObjectName("Connection_status_T")
```

```
        self.Set_X_pos_T = QtWidgets.QLabel(self.Setting_panel)
        self.Set_X_pos_T.setGeometry(QtCore.QRect(25, 250, 260, 35))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Set_X_pos_T.setFont(font)
        self.Set_X_pos_T.setStyleSheet("color: rgb(255, 255, 255);\n"
                                       "border: None;")
        self.Set_X_pos_T.setObjectName("Set_X_pos_T")
        self.Set_Y_pos_T = QtWidgets.QLabel(self.Setting_panel)
        self.Set_Y_pos_T.setGeometry(QtCore.QRect(25, 300, 260, 35))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Set_Y_pos_T.setFont(font)
        self.Set_Y_pos_T.setStyleSheet("color: rgb(255, 255, 255);\n"
                                       "border: None;")

        self.Set_Y_pos_T.setObjectName("Set_Y_pos_T")

        self.Homing_status_T = QtWidgets.QLabel(self.Setting_panel)
        self.Homing_status_T.setGeometry(QtCore.QRect(25, 140, 260, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Homing_status_T.setFont(font)
        self.Homing_status_T.setStyleSheet("border: None;")
        self.Homing_status_T.setObjectName("Homing_status_T")
        self.Settings_T = QtWidgets.QLabel(self.Setting_panel)
        self.Settings_T.setGeometry(QtCore.QRect(200, 10, 200, 80))
        font = QtGui.QFont()
        font.setPointSize(26)
        font.setBold(True)
        font.setWeight(75)
        self.Settings_T.setFont(font)
        self.Settings_T.setStyleSheet("color: rgb(255, 255, 255);\n"
                                      "border: None;")
        self.Settings_T.setAlignment(QtCore.Qt.AlignCenter)
        self.Settings_T.setObjectName("Settings_T")

        self.Connection_status = QtWidgets.QLabel(self.Setting_panel)
        self.Connection_status.setGeometry(QtCore.QRect(310, 100, 260, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Connection_status.setFont(font)
        self.Connection_status.setStyleSheet("color:  rgb(85, 255, 127);\n"
                                             "border: None;")
        self.Connection_status.setObjectName("Connection_status")
        self.Homing_status = QtWidgets.QLabel(self.Setting_panel)
        self.Homing_status.setGeometry(QtCore.QRect(310, 140, 270, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Homing_status.setFont(font)
        self.Homing_status.setStyleSheet("color:  rgb( 0, 134, 172);\n"
                                         "border: None;")
        self.Homing_status.setObjectName("Homing_status")

        #Home button
        self.Home_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Home_BT.setGeometry(QtCore.QRect(185, 200, 230, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Home_BT.setFont(font)
        self.Home_BT.setStyleSheet(button_style)
        self.Home_BT.setObjectName("Home_BT")
        self.Home_BT.clicked.connect(self.Home)

        #Move needle tip button
        self.Move_needle_tip_B = QtWidgets.QPushButton(self.Setting_panel)
        self.Move_needle_tip_B.setGeometry(QtCore.QRect(175, 355, 250, 45))
        font = QtGui.QFont()
```

```
        font.setPointSize(18)
        self.Move_needle_tip_B.setFont(font)
        self.Move_needle_tip_B.setStyleSheet(button_style)
        self.Move_needle_tip_B.setObjectName("Move_needle_tip_B")
        self.Move_needle_tip_B.clicked.connect(self.Move_needle_tip)

        #Start insertion button
        self.Start_insertion_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Start_insertion_BT.setGeometry(QtCore.QRect(175, 500, 250, 45))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Start_insertion_BT.setFont(font)
        self.Start_insertion_BT.setStyleSheet(button_style)
        self.Start_insertion_BT.setObjectName("Start_insertion_BT")
        self.Start_insertion_BT.clicked.connect(self.Insert)

        #X minus one
        self.Min_X_pos_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Min_X_pos_BT.setGeometry(QtCore.QRect(310, 255, 35, 35))
        self.Min_X_pos_BT.setStyleSheet(button_style2)
        self.Min_X_pos_BT.setText("")
        self.Min_X_pos_BT.setObjectName("Min_X_pos_BT")
        self.Min_X_pos_BT.clicked.connect(lambda: self.Minus_one(0))

        #X plus one line edit
        self.X_pos_LE = QtWidgets.QLineEdit(self.Setting_panel)
        self.X_pos_LE.setGeometry(QtCore.QRect(350, 255, 80, 35))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(False)
        font.setWeight(50)
        self.X_pos_LE.setFont(font)
        self.X_pos_LE.setStyleSheet("border: 2px solid;\n"
                                    "border-radius: 5px;\n"
                                    "border-color: rgb(0,166,214);\n"
                                    "")
        self.X_pos_LE.setAlignment(QtCore.Qt.AlignCenter)
        self.X_pos_LE.setObjectName("X_pos_LE")
        self.X_pos_LE.setPlaceholderText("X-pos")

        #X plus one button
        self.Plus_X_pos_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Plus_X_pos_BT.setGeometry(QtCore.QRect(435, 255, 35, 35))
        self.Plus_X_pos_BT.setStyleSheet(button_style3)
        self.Plus_X_pos_BT.setText("")
        self.Plus_X_pos_BT.setObjectName("Plus_X_pos_BT")
        self.Plus_X_pos_BT.clicked.connect(lambda: self.Plus_one(0))

        #Y min one button
        self.Min_Y_pos_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Min_Y_pos_BT.setGeometry(QtCore.QRect(310, 300, 35, 35))
        self.Min_Y_pos_BT.setStyleSheet(button_style2)
        self.Min_Y_pos_BT.setText("")
        self.Min_Y_pos_BT.setObjectName("Min_Y_pos_BT")
        self.Y_pos_LE = QtWidgets.QLineEdit(self.Setting_panel)
        self.Y_pos_LE.setGeometry(QtCore.QRect(350, 300, 80, 35))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(False)
        font.setWeight(50)
        self.Min_Y_pos_BT.clicked.connect(lambda: self.Minus_one(1))

        #Y position line edit
        self.Y_pos_LE.setFont(font)
        self.Y_pos_LE.setStyleSheet("border: 2px solid;\n"
                                    "border-radius: 5px;\n"
                                    "border-color: rgb(0,166,214);\n"
                                    "")
```

```
        self.Y_pos_LE.setObjectName("Y_pos_LE")
        self.Y_pos_LE.setAlignment(QtCore.Qt.AlignCenter)
        self.Y_pos_LE.setPlaceholderText("Y-pos")

        #Y plus one button
        self.Plus_Y_pos_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Plus_Y_pos_BT.setGeometry(QtCore.QRect(435, 300, 35, 35))
        self.Plus_Y_pos_BT.setStyleSheet(button_style3)
        self.Plus_Y_pos_BT.setText("")
        self.Plus_Y_pos_BT.setObjectName("Plus_Y_pos_BT")
        self.Plus_Y_pos_BT.clicked.connect(lambda: self.Plus_one(1))

        #Min_insertion_depth_BT
        self.Min_insertion_depth_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Min_insertion_depth_BT.setGeometry(QtCore.QRect(310, 430, 35, 35))
        self.Min_insertion_depth_BT.setStyleSheet(button_style2)
        self.Min_insertion_depth_BT.setText("")
        self.Min_insertion_depth_BT.setObjectName("Min_insertion_depth_BT")
        self.Min_insertion_depth_BT.clicked.connect(lambda: self.Minus_one(2))

        #Insertion depth line edit
        self.Insertion_depth_LE = QtWidgets.QLineEdit(self.Setting_panel)
        self.Insertion_depth_LE.setGeometry(QtCore.QRect(350, 430, 80, 35))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(False)
        font.setWeight(50)

        #Insertion_depth_LE
        self.Insertion_depth_LE.setFont(font)
        self.Insertion_depth_LE.setStyleSheet("border: 2px solid;\n"
                                              "border-radius: 5px;\n"
                                              "border-color: rgb(0,166,214);\n"
                                              "")
        self.Insertion_depth_LE.setObjectName("Insertion_depth_LE")
        self.Insertion_depth_LE.setAlignment(QtCore.Qt.AlignCenter)
        self.Insertion_depth_LE.setPlaceholderText("Depth")
        #self.Insertion_depth.returnPressed.connect(change_text)

        #Insertion depth plus one
        self.Plus_insertion_depth_BT = QtWidgets.QPushButton(self.Setting_panel)
        self.Plus_insertion_depth_BT.setGeometry(QtCore.QRect(435, 430, 35, 35))
        self.Plus_insertion_depth_BT.setStyleSheet(button_style3)
        self.Plus_insertion_depth_BT.setText("")
        self.Plus_insertion_depth_BT.setObjectName("Plus_insertion_depth_BT")
        self.Plus_insertion_depth_BT.clicked.connect(lambda: self.Plus_one(2))

        #Insertion depth title
        self.Set_intersion_depth_T = QtWidgets.QLabel(self.Setting_panel)
        self.Set_intersion_depth_T.setGeometry(QtCore.QRect(25, 430, 265, 35))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Set_intersion_depth_T.setFont(font)
        self.Set_intersion_depth_T.setStyleSheet("color: rgb(255, 255, 255);\n"
                                                 "border: None;")
        self.Set_intersion_depth_T.setObjectName("Set_intersion_depth_T")

        #Click on target widget
        self.widget_2 = QtWidgets.QWidget(self.centralwidget)
        self.widget_2.setGeometry(QtCore.QRect(900, 130, 838, 763))
        self.widget_2.setObjectName("widget_2")

        #Click on target button
        self.Click_on_target = QtWidgets.QPushButton(self.widget_2)
        self.Click_on_target.setGeometry(QtCore.QRect(0, 0, 838, 763))
        self.Click_on_target.setStyleSheet("image: url(TargetV2.png);"
                                           "border: None;")
        self.Click_on_target.setText("")
```

```
        self.Click_on_target.setObjectName("Click_on_target")
        self.Click_on_target.pressed.connect(self.click_target)

        #Target
        self.Needle_pos = QtWidgets.QLabel(self.widget_2)
        self.Needle_pos.setStyleSheet("background-color: rgb(255, 0, 0);\n"
                                      "border-radius: 10px;")
        self.Needle_pos.setText("")
        self.Needle_pos.setObjectName("Needle_pos")
        self.x_tag = 387
        self.y_tag = 368
        self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))

        #Click on target title
        self.Clic_on_target_T = QtWidgets.QLabel(self.centralwidget)
        self.Clic_on_target_T.setGeometry(QtCore.QRect(900, 900, 838, 50))
        font = QtGui.QFont()
        font.setPointSize(18)
        self.Clic_on_target_T.setFont(font)
        self.Clic_on_target_T.setStyleSheet("color: rgb(255, 255, 255);")
        self.Clic_on_target_T.setAlignment(QtCore.Qt.AlignCenter)
        #self.Clic_on_target_T.setObjectName("Clic_on_target_T")

        #Message box
        self.Message_box = QtWidgets.QLabel(self.centralwidget)
        self.Message_box.setGeometry(QtCore.QRect( 385, 820, 565, 150))
        font = QtGui.QFont()
        font.setPointSize(12)
        font.setBold(True)
        font.setWeight(75)
        self.Message_box.setFont(font)
        self.Message_box.setStyleSheet("color: rgb(255, 255 ,255);\n"
                                       "border: 3px solid rgb(0, 166, 214);\n"
                                       "border-radius: 20px;")
        self.Message_box.setAlignment(QtCore.Qt.AlignLeading
                                      |QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)

        #Message box title
        self.Message_box.setObjectName("Message_box")
        self.Message_box_T = QtWidgets.QLabel(self.centralwidget)
        self.Message_box_T.setGeometry(QtCore.QRect( 130, 820, 230, 40))
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(75)
        self.Message_box_T.setFont(font)
        self.Message_box_T.setStyleSheet("color: rgb(255,255,255);\n"
                                         "")
        self.Message_box_T.setObjectName("Message_box_T")
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.steedle_control_panel_T.setText(_translate("MainWindow", "Steedle control panel"))
        self.Connection_status_T.setText(_translate("MainWindow", "Connection status:"))
        self.Set_X_pos_T.setText(_translate("MainWindow", "Set X-position:"))
        self.Set_Y_pos_T.setText(_translate("MainWindow", "Set Y-position:"))
        self.Homing_status_T.setText(_translate("MainWindow", "Homing status:"))
        self.Settings_T.setText(_translate("MainWindow", "Settings"))
        self.Connection_status.setText(_translate("MainWindow", "You are connected"))
        self.Homing_status.setText(_translate("MainWindow", "Steedle is not home"))
        self.Home_BT.setText(_translate("MainWindow", "Home"))
        self.Move_needle_tip_B.setText(_translate("MainWindow", "Move neelde tip"))
        self.Start_insertion_BT.setText(_translate("MainWindow", "Start insertion"))
```

```
406         self.Set_intersion_depth_T.setText(_translate("MainWindow", "Set insertion depth:"))
407         self.Clic_on_target_T.setText(_translate("MainWindow",
408                                                  "Click on the target to set the needle tip position"))
409         self.Message_box.setText(_translate("MainWindow", "> You don't have any messages"))
410         self.Message_box_T.setText(_translate("MainWindow", "Messages box:"))
411 
412 ##########################################################################################
413                                     #Button actions#
414 ##########################################################################################
415 
416     def Move_needle_tip(self):
417         X = int(self.X_pos_LE.text())
418         Y = int(self.Y_pos_LE.text())
419 
420         X_coord, Y_coord = Controller.Move_tip( X, Y)
421 
422         global Messages
423         Messages = "> X_pos = {}, Y_pos = {} \n".format(int(X_coord), int(Y_coord)) + Messages
424         self.Message_box.setText(_translate("MainWindow", Messages))
425 
426     def Insert(self):
427         depth = int(self.Insertion_depth_LE.text())
428         Current_X, Current_Y, X_coord, Y_coord = Controller.Insert(depth)
429 
430         global Messages
431 
432         if Current_X != 'To much':
433             self.X_pos_LE.setText('{}'.format( Current_X))
434             self.Y_pos_LE.setText('{}'.format( Current_Y))
435 
436             Messages = ("> X_pos = {}, Y_pos = {} \n".format(int(X_coord), int(Y_coord))
437                         + Messages)
438             self.Message_box.setText(_translate("MainWindow", Messages))
439 
440         else:
441             Messages = ">You can't take out the needle tip this far\n" + Messages
442             self.Message_box.setText(_translate("MainWindow", Messages))
443 
444     def closeEvent(self):
445         Controller.close_connection()
446 
447     def Home(self):
448         Controller.Home()
449         self.Homing_status.setStyleSheet("color:  rgb( 85, 255, 127);\n"
450                                          "border: None;")
451         self.Homing_status.setText(_translate("MainWindow", "Steedle is home"))
452 
453     def click_target(self):
454         x, y = pg.position()
455         R_targ = 362                #Radius of circle in pixels
456         x_widg = x - 1297
457         y_widg = 543 - y
458         x_Le = int(100 -((R_targ - x_widg)/R_targ)*100)
459         y_Le = int(100 -((R_targ - y_widg)/R_targ)*100)
460 
461         self.Needle_pos.setObjectName("Needle_pos")
462         self.X_pos_LE.setText('{}'.format(x_Le))
463         self.Y_pos_LE.setText('{}'.format(y_Le))
464         self.x_tag = x - 905
465         self.y_tag = y - 165
466 
467         self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))
468         QtCore.QCoreApplication.processEvents()
469 
470     def Minus_one(self, x):
471         Arr_1 = [self.X_pos_LE.text(),
472                  self.Y_pos_LE.text(),
473                  self.Insertion_depth_LE.text()]
```

```
474
475         if Arr_1[x] != '':
476             Minus_one = int(Arr_1[x]) - 1
477         else:                                       #Check if line edit box is empty
478             Minus_one = -1
479
480         if x == 0:
481             self.x_tag = int((const * Minus_one)/ 100) + const
482             self.X_pos_LE.setText('{}'.format(Minus_one))
483             self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))
484         elif x == 1:
485             print('y_tag=', self.y_tag)
486             self.y_tag = int((const2 * Minus_one)/ 100)*-1 + const2 + 10
487             self.Y_pos_LE.setText('{}'.format(Minus_one))
488             print(self.y_tag, const2, Minus_one)
489             self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))
490         else:
491             self.Insertion_depth_LE.setText('{}'.format(Minus_one))
492
493     def Plus_one(self, x):
494         Arr_1 = [self.X_pos_LE.text(),
495                  self.Y_pos_LE.text(),
496                  self.Insertion_depth_LE.text()]
497
498         if Arr_1[x] != '':
499             Plus_one = int(Arr_1[x]) + 1
500         else:                                       #Check if line edit box is empty
501             Plus_one = +1
502
503         if x == 0:
504             self.x_tag = int((const * Plus_one)/ 100) + const
505             self.X_pos_LE.setText('{}'.format(Plus_one))
506             self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))
507         elif x == 1:
508             self.y_tag = int((const2 * Plus_one)/ 100)*-1 + const2 + 10
509             print(self.y_tag)
510             self.Y_pos_LE.setText('{}'.format(Plus_one))
511             self.Needle_pos.setGeometry(QtCore.QRect( self.x_tag, self.y_tag, 20, 20))
512         else:
513             self.Insertion_depth_LE.setText('{}'.format(Plus_one))
514
515 if __name__ == "__main__":
516     app = QtWidgets.QApplication(sys.argv)
517
518     MainWindow = QtWidgets.QMainWindow()
519     ui = Ui_MainWindow()
520     ui.setupUi(MainWindow)
521     MainWindow.showMaximized()
522     sys.exit(app.exec_())
```

Motor controller source code:

```
# -*- coding: utf-8 -*-
"""
Created on Fri May  7 09:40:35 2021

@author: Rolf
"""

import time
from pySerialTransfer import pySerialTransfer as txfer
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

import numpy as np

def open_ser_com():
        try:
            link = txfer.SerialTransfer('COM6')

            link.open()
            time.sleep(2) # allow some time for the Arduino to completely reset
            print('link = open')

            return link

        except:
            import traceback
            traceback.print_exc()

            try:
                link.close()

            except:
                pass

def send_arr( link, step_count_M1, step_count_M2, step_count_M3):

    list_of_lists = [step_count_M1, step_count_M2, step_count_M3]


    length = len(list_of_lists)

    for i in range(length):
        send_size = 0

        ###################################################################
        # Send lists
        ###################################################################

        list_ = list_of_lists[i]

        list_size = link.tx_obj(list_)
        send_size += list_size

        ###################################################################
        # Transmit all the data to send in a single packet
        ###################################################################
        link.send(send_size)

        ###################################################################
        # Wait for a response and report any errors while receiving packets
        ###################################################################
        while not link.available():
            if link.status < 0:
                if link.status == txfer.CRC_ERROR:
                    print('ERROR: CRC_ERROR')
                elif link.status == txfer.PAYLOAD_ERROR:
```

```
                    print('ERROR: PAYLOAD_ERROR')
                elif link.status == txfer.STOP_BYTE_ERROR:
                    print('ERROR: STOP_BYTE_ERROR')
                else:
                    print('ERROR: {}'.format(link.status))

        ######################################################################
        # Parse response list
        ######################################################################
        rec_list  = link.rx_obj(obj_type=type(list_),
                                        obj_byte_size=list_size,
                                        list_format='i')

link = open_ser_com()

#Variable set up
I = 20          #The centre of the circle measured from the template
K_0 = 171       #Radius of the needle curve before insertion
D_max = 100     #Maximum deflection

C1 = 3200        #1/16 step mode
C2 = 3200        #1/16 step mode
C3 = 3200        #1/16 step mode

x2 = []
y2 = []
z2 = []

steps = 30     #Determines the precision of the movement
straight = 50  #Point where maximum deflection is no longer
               #determined by mechanical limit

corner = np.arctan(D_max/straight)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

class motor_controller(object):
    def __init__(self):
        self.K_pos = K_0    #Radius of circle
        self.Z_pos = K_0
        self.X_pos = 0
        self.Y_pos = 0
        self.theta = 0
        self.round_counter = 0

    def close_connection(self):
        link.close()

    def Home(self):
        step_count_M1 = [ 28282, -2600]
        step_count_M2 = [ 28282, -2195]
        step_count_M3 = [ 28282, 9570]

        self.K_pos = K_0    #Radius of circle
        self.Z_pos = K_0
        self.X_pos = 0
        self.Y_pos = 0
        self.theta = 0
        self.round_counter = 0

        send_arr( link, step_count_M1, step_count_M2, step_count_M3)

    def Move_tip( self, X, Y):
        if self.K_pos > np.sqrt(D_max**2 + straight**2):
            R_max = D_max
        else:
            R_max = np.sin(corner)*self.K_pos
```

```
        X_coord = 0.01*X*R_max
        Y_coord = 0.01*Y*R_max

        if X_coord != 0:                          #Check if x is 0
            theta_end = np.arctan(Y_coord/X_coord)
            if X_coord < 0:                       #Check if tip is in second or third quadrant
                theta_end = np.pi + theta_end
            elif Y_coord < 0 and theta_end < 0: #Check if tip is in fourth quadrant
                theta_end = 2*np.pi + theta_end
            else:
                theta_end = self.theta

        else:
            if Y_coord > 0:
                theta_end = 0.5*np.pi
            elif Y_coord < 0:
                theta_end = 1.5*np.pi
            else:
                theta_end = self.theta

        #Projected on the x-y plane the tip of the needle follows a path
        #characterized by the equation y= Ax + B the co fficient A and B can be found
        #through the following equations.

        if round(X_coord, 6) != round(self.X_pos, 6):         #Checking for a vertical line
            A = (Y_coord - self.Y_pos)/(X_coord - self.X_pos) #Directional coefficient of path
            B = self.Y_pos - A*self.X_pos                     #Intersection point with y-axis

            x_arr, y_arr, z_arr, R_arr, theta_arr = self.Non_vertical_line( A, B,
                                                                    X_coord, Y_coord)

        else:
            x_arr, y_arr, z_arr, R_arr, theta_arr = self.vertical_line(X_coord, Y_coord)

        step_count_M1 = []
        step_count_M2 = []
        step_count_M3 = []

        Z_steps = []
        theta_steps = []
        R_steps = []

        print('x_arr=', x_arr, '\n')
        print('y_arr=', y_arr, '\n')
        print('R_arr=', R_arr, '\n')
        print('Theta_arr=', theta_arr, '\n')
        print('z_arr=', z_arr, '\n')

        #Convert positions of needle tip path to Motor positions
        for i in range(steps):
            Z_steps.append(int(round((z_arr[i] - z_arr[0])/(32/C1))))
            theta_steps.append(int(round(-1*(theta_arr[i] - theta_arr[0])/
                                         ((2*np.pi/C2)*(16/100)))))
            R_steps.append(int(round(-1*(R_arr[i] - R_arr[0])/(32/C3))))

            if i > 0:
                step_count_M1.append(Z_steps[i] - Z_steps[i - 1])
                step_count_M2.append(theta_steps[i] - theta_steps[i - 1])
                step_count_M3.append(R_steps[i] - R_steps[i - 1])

        send_arr( link, step_count_M1, step_count_M2, step_count_M3)

        #Update position
        self.X_pos = X_coord
        self.Y_pos = Y_coord
        self.Z_pos = z_arr[steps - 1]
```

```
        return X_coord, Y_coord

    def Insert(self, depth):
        R_start = np.sqrt(self.X_pos**2 + self.Y_pos**2)
        Z_start  = np.sqrt(self.K_pos**2 - R_start**2)
        Q = Z_start

        if self.X_pos != 0:                    #Check if x is 0
            theta = np.arctan(self.Y_pos/self.X_pos)
            if self.X_pos < 0:                 #Check if tip is in second or third quadrant
                theta = np.pi + theta
            elif self.Y_pos < 0 and theta < 0: #Check if tip is in fourth quadrant
                theta = 2*np.pi + theta
            else:
                theta = self.theta

        else:
            if self.Y_pos > 0:
                theta = 0.5*np.pi
            elif self.Y_pos < 0:
                theta = 1.5*np.pi
            else:
                theta = self.theta

        zN = np.linspace( Z_start - depth, Z_start, steps)
        yN = R_start/(Q**2)*zN**2                          # Re_arr
        xN = np.zeros(steps)

        #Rotating the needle curve in the right configuration in 3 dimensional space
        zR = zN
        xR = yN*np.cos(theta) - xN*np.sin(theta)
        yR = yN*np.sin(theta) + xN*np.cos(theta)

        z_arr_ins = []              #Array with all the z positions of the needle tip
        R_arr_ins = []              #Array of all radi
        theta_arr_ins = []          #Array of all the theta angles

        #Calculating the motor positions
        #zR is given from the end of the insertion to the begining
        #Therefore this array needs to be flipped so that it start at the beginning

        for i in range(len(zR)):
            z_arr_ins.append(zR[len(zR) - i-1])
            R_arr_ins.append(np.sqrt(xR[len(zR) - i-1]**2 + yR[len(zR) - i-1]**2))

            if xR[len(zR) - i-1] == 0:
                theta_arr_ins.append(0)
            else:
                theta_arr_ins.append(np.arctan(yR[len(zR) - i-1]/xR[len(zR) - i-1]))

        step_count_M1 = []
        step_count_M2 = []
        step_count_M3 = []

        Z_steps = []
        theta_steps = []
        R_steps = []

        #Check if the insertion doesn't cause to much bending:
        Check_K = np.sqrt(zR[0]**2 + (xR[0]**2 + yR[0]**2))
        if Check_K > np.sqrt(D_max**2 + straight**2):
            Current_Rmax = D_max
        else:
            Current_Rmax = np.sin(corner)*Check_K

        X_perc = int(round(xR[0]/Current_Rmax * 100))
        Y_perc = int(round(yR[0]/Current_Rmax * 100))
```

```
271         if X_perc > 100 or Y_perc > 100:
272             return 'To much', 'To much', 'To much', 'To much'
273 
274         else:
275             #Convert positions of needle tip path to Motor positions
276             for i in range(steps):
277                 Z_steps.append(int(round((z_arr_ins[i] - z_arr_ins[0])/(32/C1))))
278                 theta_steps.append(int(round(-1*(theta_arr_ins[i] - theta_arr_ins[0])
279                                              /((2*np.pi/C2)*(16/100)))))
280                 R_steps.append(int(round(-1*(R_arr_ins[i] - R_arr_ins[0])/(32/C3))))
281 
282                 if i > 0:
283                     step_count_M1.append(Z_steps[i] - Z_steps[i - 1])
284                     step_count_M2.append(theta_steps[i] - theta_steps[i - 1])
285                     step_count_M3.append(R_steps[i] - R_steps[i - 1])
286 
287             send_arr( link, step_count_M1, step_count_M2, step_count_M3)
288 
289             #Update position
290             self.X_pos = xR[0]
291             self.Y_pos = yR[0]
292             self.Z_pos = zR[0]
293             self.K_pos = np.sqrt(self.Z_pos**2 + (self.X_pos**2 + self.Y_pos**2))
294 
295             #Update bending percentages
296             if self.K_pos > np.sqrt(D_max**2 + straight**2):
297                 Current_Rmax = D_max
298             else:
299                 Current_Rmax = np.sin(corner)*self.K_pos
300 
301             X_perc = int(round(self.X_pos/Current_Rmax * 100))
302             Y_perc = int(round(self.Y_pos/Current_Rmax * 100))
303 
304             return X_perc, Y_perc, self.X_pos, self.Y_pos
305 
306     def Non_vertical_line(self, A, B, X_coord, Y_coord):
307         x_arr = np.linspace( self.X_pos, X_coord, steps)#Array of all x-pos of needle exteremity
308         y_arr = []                                      #Array of all y-pos of needle exteremity
309         z_arr = []                                      #Array of all z-pos of needle exteremity
310         R_arr = []                                      #Array of all radi
311         theta_arr = []                                  #Array of all the theta angles
312 
313         #Get positions of needle tip path
314         for i in range(len(x_arr)):
315             y_arr.append(A*x_arr[i] + B)
316             R_arr.append(np.sqrt(x_arr[i]**2 + y_arr[i]**2))
317             z_arr.append(np.sqrt(self.K_pos**2 - R_arr[i]**2))
318 
319             if (y_arr[i] * y_arr[i-1]) <= 0 and x_arr[i] > 0 and i > 0 and y_arr[0] != 0:
320                 if y_arr[i] <= y_arr[i-1]:
321                     self.round_counter -= 1
322                 else:
323                     self.round_counter += 1
324 
325             if x_arr[i] != 0:                           #Check if x is 0
326                 self.theta = np.arctan(y_arr[i]/x_arr[i]) + 2*np.pi*self.round_counter
327                 if x_arr[i] < 0:                        #Check if tip is in second or third quadrant
328                     self.theta = np.pi + self.theta
329                     theta_arr.append(self.theta)
330                 elif y_arr[i] < 0 and self.theta < 0:  #Check if tip is in fourth quadrant
331                     self.theta = 2*np.pi + self.theta
332                     theta_arr.append(self.theta)
333                 else:                                   #Check if tip is in first quadrant
334                     theta_arr.append(self.theta)
335             else:
336                 if round(y_arr[i], 6) > 0:
337                     self.theta = 0.5*np.pi + 2*np.pi*self.round_counter
338                     theta_arr.append(self.theta)
```

```
339                 elif round(y_arr[i], 6) < 0:
340                     self.theta = 1.5*np.pi + 2*np.pi*self.round_counter
341                     theta_arr.append(self.theta)
342                 else:
343                     self.theta = 0
344                     theta_arr.append(self.theta)
345 
346         return x_arr, y_arr, z_arr, R_arr, theta_arr
347 
348     def vertical_line(self, X_coord, Y_coord):
349         x_arr = []                                          #Array of all x-pos of needle exteremity
350         y_arr = np.linspace( self.Y_pos, Y_coord, steps)#Array of all y-pos of needle exteremity
351         z_arr = []                                          #Array of all z-pos of needle exteremity
352         R_arr = []                                          #Array of all radi
353         theta_arr = []                                      #Array of all the theta angles
354 
355         #Get positions of needle tip path
356         for i in range(len(y_arr)):
357             x_arr.append(self.X_pos)
358             R_arr.append(np.sqrt(x_arr[i]**2 + y_arr[i]**2))
359             z_arr.append(np.sqrt(self.K_pos**2 - R_arr[i]**2))
360             if x_arr[i] != 0:
361                 theta_arr.append(np.arctan(y_arr[i]/x_arr[i]))
362             else:
363                 if y_arr[i] == 0:
364                     self.theta = 0
365                     theta_arr.append(self.theta)
366                 elif y_arr[i] > 0:
367                     self.theta = 0.5*np.pi
368                     theta_arr.append(self.theta)
369                 else:
370                     self.theta = -0.5*np.pi
371                     theta_arr.append(self.theta)
372 
373         return x_arr, y_arr, z_arr, R_arr, theta_arr
```

Arduino source code:

```
#include <AccelStepper.h>
#include <MultiStepper.h>

#include "SerialTransfer.h"
#include <SoftwareSerial.h>

SoftwareSerial HC05( A3, 13); //HC05-TX Pin 10, HC05-RX to Arduino Pin 11
SerialTransfer myTransfer;

#define M1_STEP_PIN         2
#define M1_DIR_PIN          5

#define M2_STEP_PIN         3
#define M2_DIR_PIN          6

#define M3_STEP_PIN         4
#define M3_DIR_PIN          7

#define ENABLE_PIN         8

AccelStepper stepperM1(1, M1_STEP_PIN, M1_DIR_PIN);
AccelStepper stepperM2(1, M2_STEP_PIN, M2_DIR_PIN);
AccelStepper stepperM3(1, M3_STEP_PIN, M3_DIR_PIN);

//Set empty arrays for the incoming steps
int step_count_M1[30];
int step_count_M2[30];
int step_count_M3[30];

//Assign stop_switches
int M1_switch = 9;       //Z-axis limit switch
int M2_switch = 10;      //Theta-axis limit switch
int M3_switch = 12;      //X-axis limit switch

char LED = LED_BUILTIN;
int counter = 0;
boolean M2H = false;
boolean Home = false;

MultiStepper steppers;

void setup()
{
  HC05.begin(9600);
  myTransfer.begin(HC05);           //Start bluetooth communication

  //Enable the steppers
  pinMode(ENABLE_PIN, OUTPUT);
  digitalWrite(ENABLE_PIN, LOW);

  //Set max speed settings
  stepperM1.setMaxSpeed(500);
  stepperM2.setMaxSpeed(500);
  stepperM3.setMaxSpeed(500);

  //Set acc setting
  stepperM1.setAcceleration(3000);
  stepperM2.setAcceleration(3000);
  stepperM3.setAcceleration(3000);

  //stepperM1.setSpeed(500);

  //Add steppers to multi stepper
  steppers.addStepper(stepperM1);
  steppers.addStepper(stepperM2);
  steppers.addStepper(stepperM3);
```

```
67
68    //Set up stop switches
69    pinMode(M1_switch, INPUT_PULLUP);
70    pinMode(M2_switch, INPUT_PULLUP);
71    pinMode(M3_switch, INPUT_PULLUP);
72
73    //stepperM3.move(3200);
74    //stepperM3.runToPosition();
75
76  }
77
78  void loop(){
79     if(myTransfer.available())
80     {
81        uint16_t recSize = 0;
82
83        //Put received data in assigened arrays
84       for(int i=0; i < myTransfer.bytesRead/4; i++){
85          if(counter == 0){
86            myTransfer.rxObj( step_count_M1[i], recSize);
87          }
88          if(counter == 1){
89            myTransfer.rxObj( step_count_M2[i], recSize);
90          }
91          if(counter == 2){
92            myTransfer.rxObj( step_count_M3[i], recSize);
93          }
94          recSize += 4;
95       }
96
97       //Prepare buffer to be send back
98       for(uint16_t i=0; i < myTransfer.bytesRead; i++){
99          myTransfer.packet.txBuff[i] = myTransfer.packet.rxBuff[i];
100      }
101     //Send received array back to computer
102     myTransfer.sendData(myTransfer.bytesRead);
103
104     //Move the steppers if all arrays are filled
105     if( counter == 2){
106         counter = 0;
107
108         //Check for go home command
109         if( step_count_M1[0] == 28282 and step_count_M2[0] == 28282){
110           GoHome();
111         }
112         else{
113           MoveEngines();
114         }
115     }
116     else{
117       counter +=1;
118     }
119    }
120  }
121
122  void MoveEngines(){
123     for(int i=0; i < myTransfer.bytesRead/4; i++){
124      stepperM1.setCurrentPosition(0);
125      stepperM2.setCurrentPosition(0);
126      stepperM3.setCurrentPosition(0);
127
128      stepperM1.setSpeed(1000);
129
130      long positions[3] = {step_count_M1[i], step_count_M2[i], step_count_M3[i]};
131
132      steppers.moveTo(positions);
133      steppers.runSpeedToPosition();
134      //delay(1000);
```

```
135     }
136 }
137
138 void GoHome(){
139   if(Home == false){
140    while(true)
141      {
142          int counter = 0;
143          stepperM1.setSpeed(500);
144          stepperM2.setSpeed(500);
145          stepperM3.setSpeed(-800);
146
147          if(digitalRead(M1_switch) == HIGH){
148           stepperM1.stop();
149
150           counter += 1;
151          }
152          else{
153           stepperM1.runSpeed();
154          }
155
156          if(digitalRead(M2_switch) == HIGH or M2H == true){
157           stepperM2.stop();
158           M2H = true;
159           counter += 1;
160          }
161          else{
162           stepperM2.runSpeed();
163          }
164
165          if(digitalRead(M3_switch) == HIGH){
166           stepperM3.stop();
167           counter += 1;
168          }
169          else{
170           stepperM3.runSpeed();
171          }
172
173          if(counter == 3){
174           Home = true;
175           break;
176          }
177      }
178
179    //Move to starting position
180    Home = false;
181    delay(1000);
182    stepperM1.setCurrentPosition(0);
183    stepperM2.setCurrentPosition(0);
184    stepperM3.setCurrentPosition(0);
185
186    //Steps to home position from stop switch
187    int M1_dis = step_count_M1[1];
188    int M2_dis = step_count_M2[1];
189    int M3_dis = step_count_M3[1];
190
191    stepperM1.move(M1_dis);
192    stepperM2.move(M2_dis);
193    stepperM3.move(M3_dis);
194    M2H = false;
195
196    while(true)
197    {
198       stepperM1.setSpeed(-2500);
199       stepperM2.setSpeed(-500);
200       stepperM3.setSpeed(800);
201       int counter = 0;
202       if (stepperM1.distanceToGo()){
```

```
203         stepperM1.run();
204         counter +=1;
205        }
206        if (stepperM2.distanceToGo()){
207         stepperM2.run();
208         counter +=1;
209        }
210        if (stepperM3.distanceToGo()){
211         stepperM3.run();
212         counter +=1;
213        }
214        if (counter == 0){
215          Serial.print(counter);
216          break;
217        }
218     }
219    }
220 }
```

REFERENCES

[1] M. Nobel, "The design of a manually operated compliant mechanism steerable needle for high dose-rate brachytherapy of the prostate," 2020.

[2] Prostaatkankerstichting, "Wat is prostaatkanker?."

[3] D. Georg, "Dosimetric considerations to determine the optimal technique for localized prostate cancer among external photon, proton, or carbon-ion therapy and high-dose-rate or low-dose-rate brachytherapy," *International Journal of Radiation Oncology* Biology* Physics*, vol. 88, 2014.

[4] J. Crook, M. Marbán, and D. Batchelar, "Hdr prostate brachytherapy," *Seminars in radiation Oncology*, vol. 30, 2020.

[5] O. M. Physics, "Prostate interstitial needle placement guided by ultrasound."

[6] S. Fidoe, "The perineum."

[7] F. P. J. Waschke, *Sabotta Atlas of Human Anatomy Association*. Urban Fischer, 15 ed., 2013.

[8] L. E. K H Leissner, Tisell, "The weight of the dorsal, lateral and medial prostatic lobes in man," 1979.

[9] T. Delft, "Compliant mechanisms."

[10] N. Van De Berg, *Needle Steering Mechanics and Design Cases*. PhD thesis, Delft University of Technology, 2016.

[11] P. Breedveld, J. L. Herder, and T. Tomiyama, "Teaching creativity in mechanical design," in *4th World Conference on Design Research (IASDR2011), Delft, The Netherlands, Oct*, 2011.

[12] T. Podder, J. Sherman, D. Fuller, E. Messing, D. Rubens, J. Strang, R. Brasacchio, and Y. Yu, "In-vivo measurement of surgical needle intervention parameters: A pilot study," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 1, 2006.