Development of a soft self-healing sensitive hand exoskeleton

Stijn Hamelryckx

Master thesis submitted under the supervision of
Prof. dr. ir. Bram Vanderborght

The co-supervision of
Dr. ir. Julie Legrand and Ir. Ellen Roels

In order to be awarded the Master's Degree in
Electromechanical Engineering – Robotics and Mechanical Construction

Academic year
2022-2023

Contents

Abstract

Stijn Jacques J Hamelryckx

~

Master of Science in Electromechanical Engineering Robotics and Mechanical Construction

~

Academic year 2022-2023

~

Development of a soft self-healing sensitive hand exoskeleton

In this thesis, a soft self-healing multi-material tendon driven actuator for soft hand exoskeleton applications has been developed. Many studies have shown exoskeleton hands can help with rehabilitation as well as help with daily life activities for people with fully paralyzed hands, however these exoskeleton hands are prone to damage due to their extensive daily use. This can be solved using hand exoskeletons made out of self-healing materials which also adds more design freedom (e.g. it is easier to build a multi-material actuator) thanks to the possibility of fusing during manufacturing. First, the design requirements were determined using reference data obtained via motion capture of the grasping motion of a finger. Based on these requirements, designs of the actuator have been proposed. Next, the material properties of the materials were determined for simulation purposes. For each design, a SOFA simulation was performed which was validated using experimental data. In these simulations, it was found that a multi-material design gave advantages such as maintaining the shape and length of the links as well as a better controllability of the actuator. Furthermore, a multi-material design increases the overall force output of the actuator. For the outer actuator, an actuator of two materials was deemed acceptable due to the fact that this actuator requires a lower force output. However, for the inner actuator, a design using four different materials was used to get even closer results to the reference data. In this design, each joint was given a different stiffness to as such change the rate of the joint angles accordingly to the reference data. For validation, the actuators were manufactured using casting. Different casting methods were tested and compared to establish the best way for manufacturing the actuator. It was found that two single stage casts fused together works well. For the final design of the inner actuator, six single casts were used and fused together. Apart from validation, the manufactured actuators were also characterized by determining the bending profile of the actuator in case the bottom side is constrained. This gave different results compared to when the actuator is free on all sides and gives an interesting view on how the actuator would behave when attached to the finger as is the case in the actual application. The force required to pull the tendon was also determined and does not exceed 1N for all developed actuators.

Keywords: Self-Healing Materials, Soft Robot Material and Design, Multi-material, FEM Simulations, Soft Exoskeleton Hand

Motivation

The human hand is a vital part for many activities in daily life. It is used to perform many tasks such as eating, sports or activities and other important tasks. One can imagine that partial or complete loss of hand motion can have severe consequences resulting in a reduced quality of life. In some cases it is possible to regain or at least improve this hand motion by doing rehabilitation. However, this rehabilitation can be very costly and time consuming. Furthermore, the result and progress of rehabilitation heavily depends on the performance of the exercises. If it is not performed correctly, more time is needed for rehabilitation and possibly even bad habits can arise. These issues can be resolved using a rehabilitation glove which provides continuous monitoring of the exercises and assists the movement. These gloves can also be used in case no regain of the hand motion can be achieved, not to assist with exercises but to help with daily activities.

Due to all this, there is growing interest in the development of exoskeleton gloves in particular on soft exoskeleton gloves. In this thesis, the development of a soft self-healing sensitive hand exoskeleton is proposed. The self-healing material allows the exoskeleton hand to heal or repair possible damage caused by daily activities and surroundings. Another advantage of self-healing material includes the possibility of achieving multi-material gloves without weak interfaces. This multi-material design could for example help increase the force output of the exoskeleton, increase the closed loop control of the system and provide a bending profile which accurately approaches the profile of the finger.

Acknowledgements

I’d like to thank my co-supervisors, Dr. ir. Julie Legrand and Ir. Ellen Roels for their extensive help throughout the project. Their insights and assistance have contributed greatly to the final results of the thesis. Furthermore, I’d like to thank Ir. Pasquale Ferrentino for providing me code for the simulations as well as helping me debugging my own simulations. I’d also like to thank my promotor, Prof. dr. ir. Bram Vanderborght, for making the thesis possible and proofreading. I’d also like to thank my family as well as my friends for their support and keeping me motivated when needed.

List of figures

List of tables

Chapter 1: Introduction

1.1 Soft robotics

Based on biological systems, a relatively novel study domain emerged in the form of soft robotics [1]. One example of this inspiration from biological systems (such as animals and plants) is the development of an octopus shaped robot for flexible manipulation [2]. Thanks to soft and deformable materials, advantages can be found in dexterity and environmental adaptivity which creates the ability to do things not possible for rigid robots [3].

In traditional robots the goal is to minimize deflections by design of stiffness, making it easier to create precise movement. A disadvantage coming from this, is that these robots are not suitable to handle delicate objects such as fruits and humans. However, a recent trend in traditional robotics is to focus on compliance to safely interact with humans [4]. In soft robotics, the design is more focused on being compliant and thus soft. This makes that these kinds of robots rarely cause damage to objects or injuries on humans [5]. Applications for this domain can be found in grippers for fruits [6] or other delicate objects [7], robots designed to interact with humans (cobots) [8], social robots [9] but also in many biomedical applications to get through flexible and small places within the human body [10].

Apart from being flexible, soft robotics also brings the advantage of facilitating the use of multifunctional components which are components that can fulfil multiple functions simultaneously [11]. This could for example be achieved with reconfigurable robots that use modular units which can be assembled in different arrangements for different tasks [12]. Another way soft robots can be multifunctional comes from a sensor that can measure multiple signals (e.g. a tactile sensor that can simultaneously measure temperature and pressure) [13].

An important drawback of soft robotics is that they are very susceptible to damage, as they can be damaged more easily than traditional robots. This damage can be inflicted in many forms such as sharp objects and overpressures in case of pneumatic actuation. Some possible ways damage can be inflicted can be seen in Figure 1 [14]. This drawback makes soft robotics less interesting commercially. Furthermore, due to the fact that these robots are quite compliant, another disadvantage is that precise movements are rather difficult to achieve due to out-of-plane deformations occurring [15]. One important research domain in soft robotics is the study of the soft actuation types.

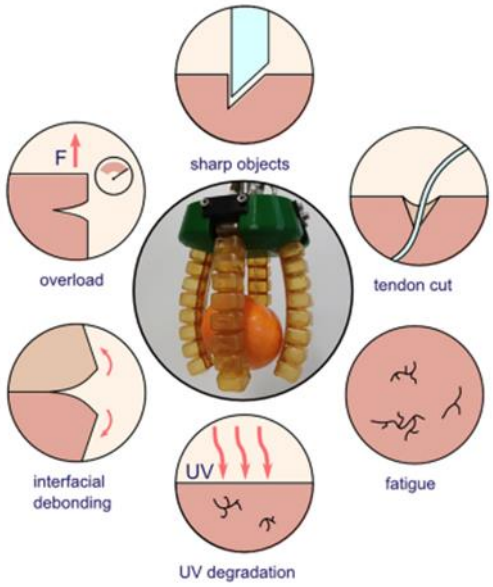


Figure 1 Possible sources of damage for soft robots making them very susceptible to damage [14]

1.2 Soft actuation types

Using well-designed bodies, soft actuation converts an energy source into either force or displacement. Soft actuators can withstand large strains thanks to their flexible or stretchable material.

- Dielectric elastomer actuators (DEA) are made out of one stretchable dielectric layer that is squished in between two electrodes that are also stretchable. The dielectric layer will contract the same way as the electrical field and expand in the other directions. This actuator however, is rather difficult to make and dangerous in case the dielectric layer tears and short circuit happens [3] [5] [16].
- Another type of soft actuators are the hydrogel actuators. Actuation happens due to the absorption of water in the material matrix resulting in an increase in volume of the material [3] [5] [16].
- When the actuation occurs via compressed air, the actuator is called a pneumatic actuator. This type of actuator is made using a membrane and thanks to filling and expanding chambers using air, the material will deform and actuation takes place thanks to the geometry. This type of actuation is frequently applied in soft robotic fingers and grippers and can be used in harsh environments [17] [18]. However, one drawback involves the susceptibility to damage inflicted by overpressure or sharp objects which can reduce the performance of the actuator. An example of this type of actuator are soft pneumatic grippers which consist of multiple bendable fingers and can also be called bending soft pneumatic actuator (BSPA) (Figure 2a). Each finger is made out of multiple chambers connected with a tube to a pressure source. Thanks to the asymmetry of the geometry, when inflated, the finger will curl resulting in a gripping motion [18].
- Another example of a soft actuator is the Fin Ray® gripper (Figure 2b) created by Festo [19]. This gripper consists of a bendable structure which allows it to curl around objects and as such grip the object. The gripper can be moved using a rigid external pneumatic

system that moves the holder of the fingers. This actuator has a rather simple geometry, making it quite easy to fabricate [16].

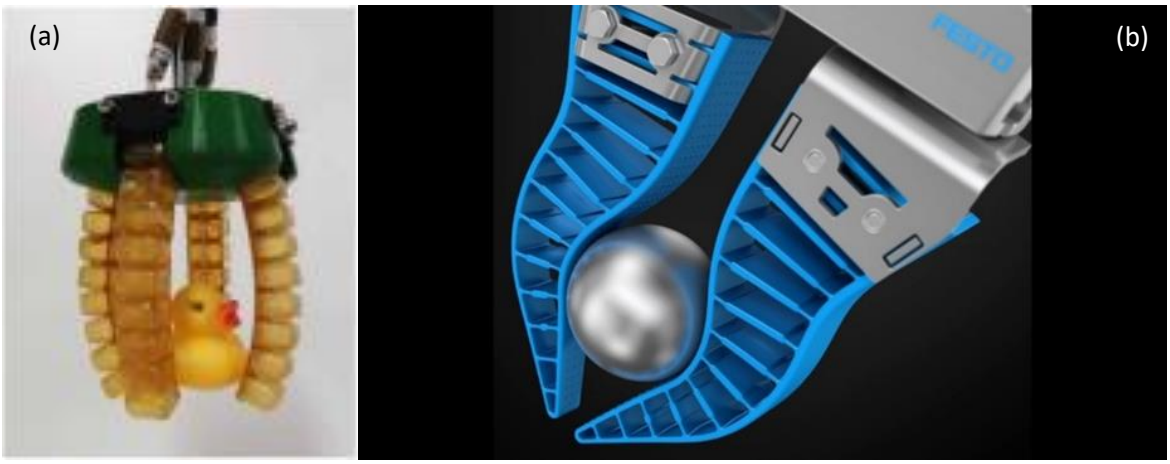


Figure 2 Applications for pneumatic actuators: (a) An example of a BSPA grasping a rubber duck [18] and (b) a FinRay® gripper grasping a ball [19]

- Based on human fingers, tendons can be used to actuate a soft gripper or finger in soft robotics. Notice that for these actuators the gripper and finger are soft and not the tendon. Similar to the human muscles, a tendon can be pulled to actuate the actuator. One tendon is required for each degree of freedom (DOF). Since the actuator can be separated from the joint and one actuator can also actuate multiple joints by using multiple wires, the inertia of the total system can be reduced [20] [21]. An example of a tendon-driven actuator can be found in Figure 3.

Figure 3 An example of a tendon driven actuator used for a gripper grasping multiple objects [22]

1.3 Manufacturing methods

Since these actuators are made out of elastomers, subtractive manufacturing such as milling, turning and drilling are rather difficult [23]. Due to this, other methods usually formative manufacturing such as moulding, casting and additive manufacturing that allows to print layer by layer can be used for these types of materials.

Casting is the process where a liquid monomer is solidified due to cooling or polymerization in a mould. This process makes use of rather easy to obtain tools and has the advantage that the final product will have low internal stresses. A bottleneck of this method is that bubbles can get trapped in the part creating the possibility of unexpected failure. This is certainly a concern when using highly viscous monomers. Certain parts, if the design is simple enough, can be made using single-stage casting. An example of this is the Fin Ray® gripper. Even though multi-stage casting is more time consuming, it can be utilized for more complex shapes or multi-material applications. A disadvantage of this however is the possibility of weak interfaces. Moulding is similar to casting, but an extra stimulus is introduced such as a high pressure (compression moulding) or injection of the liquid monomer (injection moulding) [23].

Alternatively, elastomers can be manufactured using additive manufacturing (AM), defined as a process in which a 3D part is created using a computer-aided design (CAD) by stacking material layer by layer on top of each other to create a 3D object [24]. Additive manufacturing can handle complex geometries. A noticeable additive manufacturing method is fused filament fabrication (FFF), the process where a thermoplastic filament gets extruded through a nozzle on a platform via melting [25] [26].

1.4 Self-healing materials

The disadvantage that soft robots are very susceptible to damage can be solved using self-healing materials. This means that soft robotic components have the ability to keep their function after material damage occurs thanks to healing if they are constructed from self-healing material. In other words, soft robots are rather damage resilient while traditional robots are rather damage resistant meaning they do not suffer damage as easily [3]. Self-healing materials can also resolve the issue of weak interfaces created by multi-stage casting [23]. Another advantage of using self-healing materials lies in multi-material design which can lead to improvement of the performance of the soft robot and complex deformation modes. The issue of weak interfaces between the self-healing materials in soft robots is reduced via a chemical bonding process at the contact surface [11]. On the contrary, materials applied in traditional robotics include steel and hard plastics which are rather difficult to merge without welding or using some connecting components such as bolts [27].

Self-healing systems can be divided into two categories: the autonomous and non-autonomous self-healing systems. For non-autonomous self-healing systems, a trigger is first required before the healing is initiated. This trigger is usually created by reaching a certain temperature. For the autonomous systems, this trigger is already built-in and thus healing starts as soon as damage is inflicted.

Self-healing materials exist based on different principles such as the use of microcapsules containing monomers or formation of hydrates in the cracks thanks to water, but the ones

used in this work are based on Diels-Alder (DA) polymer networks with the crosslinks being DA bonds. These bonds are reversible which means they can either break or reform. There is an equilibrium reaction where an equilibrium is reached between this breaking and reforming of bonds. This equilibrium, among other factors, depends on temperature meaning that a change in temperature will induce a shift in the equilibrium. While a temperature increase breaks bonds, a decrease in temperature leads to bond reformation. The possibility to shift the equilibrium by temperature, thus the reformation and breaking of bonds, allows the healing of damaged material. During the whole healing process, the material stays a cohesive whole, in other words, the gelation temperature is not exceeded (see Figure 4). As the temperature increases, the equilibrium will shift to breaking more bonds until eventually, a point will be reached for which the macroscopic network will be split up into smaller macromolecules. The temperature at this point is called the gelation temperature and can be defined as the temperature below which the polymer is solid as it is a macroscopic network. Above this temperature, the material will become a viscous liquid. Heating above the gelation temperature should be avoided since side reactions will occur which will decrease the healing properties of the material. This means that a high enough temperature to heal the damage is needed while this temperature does not exceed the gelation temperature.

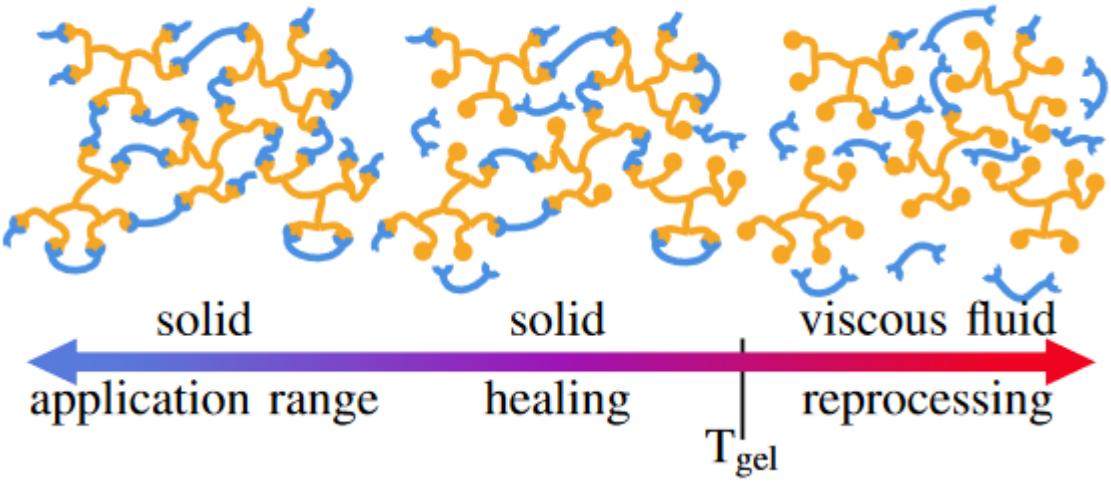


Figure 4 Evolution of the polymer chains for an increased temperature. The gelation temperature, where the material becomes a viscous liquid is not crossed during healing [22].

During synthesis of the material, the mechanical properties can be tuned by for example choosing the length of the chains but also changing the maleide-to-furan ratio (r) (see Figure 5). This means that it is possible to create both flexible and stiff materials. This is very useful for multi-material applications since it is still easy to fuse the materials with different mechanical properties. An example in nature of a multi-material application can be found in the human body where both stiff and flexible materials work together [22].

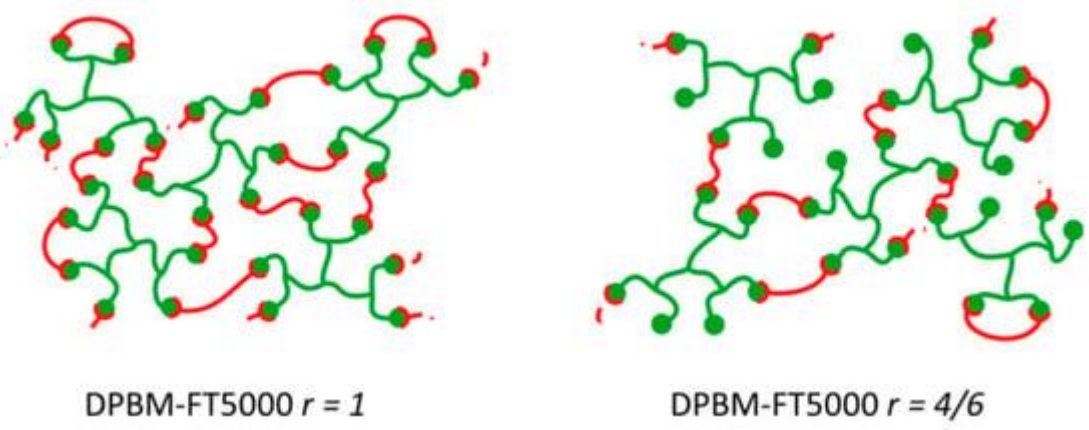


Figure 5 The structure of a self-healing material made out of DA bonds where the properties of the material can be tuned during synthesis. The breaking and reforming of the red bonds lead to the healing ability of the material [28].

While self-healing materials have been developed for some applications such as coating for cars and phone cases [29], others are in a late stage of production such as self-healing asphalt and concrete even though they work based on a different principle [30] [31]. In many applications, research is being done in self-healing materials of which aeronautics, artificial stretchable skin and films are some examples [32] [33] [34].

The whole healing process can be described in four steps. Evidently, before the healing process can start, damage has to be inflicted on the object. Usually, this damage will have the shape of an elongated fracture, as commonly applied during tests.

First, the temperature of the object is increased without exceeding the gelation temperature for the breaking of the bonds, thus increasing the mobility of the polymer chains (the smaller the chains, the higher the mobility). Secondly, the object is kept at this temperature to further increase the mobility by breaking more cross-links. Then, the temperature is gradually decreased to initiate the bond reformation and recovery of the mechanical properties. This decrease occurs slowly since the kinetics of the DA bonds are higher at high temperatures. The slow decrease of temperature increases the bond reformation process thus reducing the total recovery time. Finally, the object is kept at room temperature for the recreation of the final bonds, completely recovering the properties of the object. This final step approximately lasts one day due to the low kinetics of the bonds. The complete healing cycle is shown in Figure 6. A parallel between this healing of the materials and the healing of the human skin, for example a human hand, can be drawn.

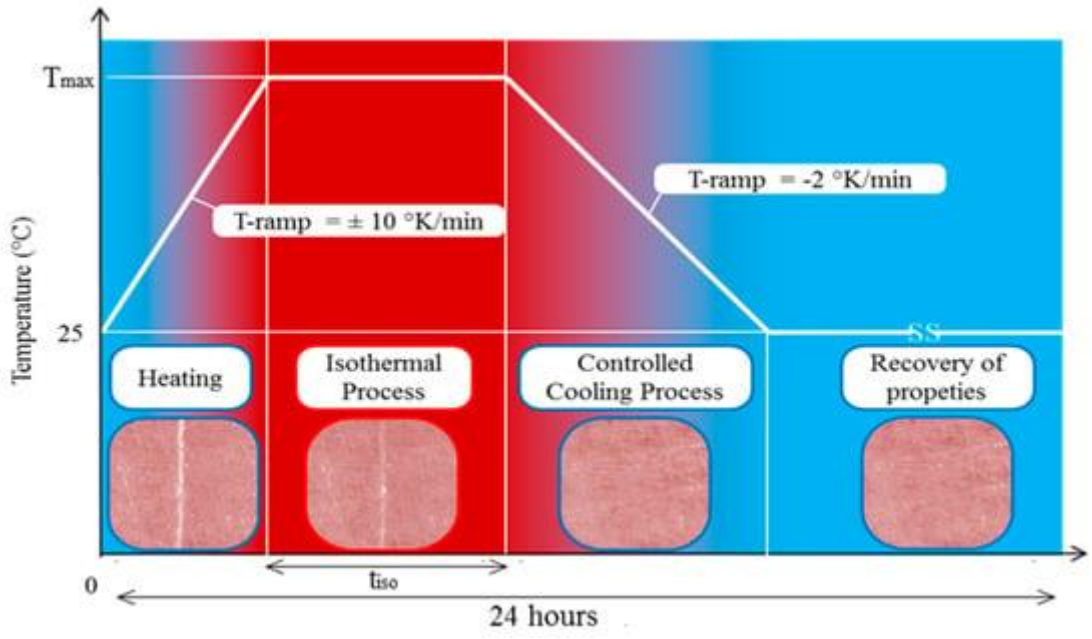


Figure 6 Temperature profile of the healing process where a slow cool down rate is used to reduce the total recovery time [28].

1.5 The human hand

The hand is one of the most complex parts of the human body (see Figure 7). It is primarily made out of bones, joints and soft tissue [35]. The human hand has evolved to perform tasks such as grasping, using tools and gesturing [36]. Hand motions can be divided into two categories: simple and complex hand motions. Simple hand motions are frequently used in real life for tasks such as grasping, lifting, holding, putting and rotation. They can be performed by one or more types of subactions and finger primitives. For a hand motion to be complex, three features must be present: motion of multiple fingers and possibly palm, motion of the wrist working together tightly with in-hand manipulation and there has to be a change in hand posture and location [37].

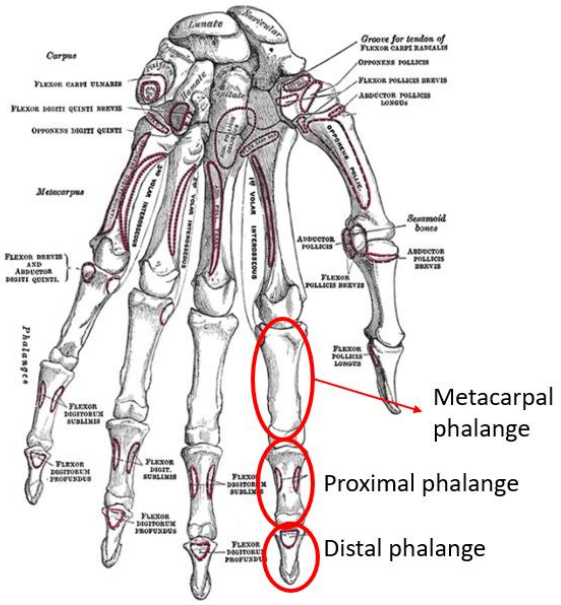


Figure 7 Anatomy of a human hand showing the complexity of the human hand and the phalanges of the finger [38].

The finger is a structure compromising three joints. The first joint, at the base of the finger, is the metacarpal interphalangeal joint (MCP) and has a maximal flexion that ranges from 70° to 85° depending on the orientation of the finger. The second joint is the proximal interphalangeal joint (PIP) which has a maximal flexion of 110°. The third and final joint is the distal interphalangeal joint (DIP) which has a maximum flexion of 90° [39]. Finger movement can be controlled by activating extrinsic and intrinsic muscles. The actuation of the extrinsic muscles comes from the forearm while actuation of the intrinsic muscles comes from the finger itself. The extrinsic muscles control the extensor muscle tendons making it possible to bend the finger while the intrinsic muscles control the lateral motion of each finger [2].

The human hand consists of a total of 27 degrees of freedom (DOF). Each finger has a total of four DOF (except for the thumb which has five): three for the flexion and tension of the finger and one that allows abduction and adduction (sideways movement) of the finger. However, due to biomechanical restrictions coming from the muscle and tendon configuration, there is interdependency between the joints. One example of this can be found in the activation of a muscle coming from the forearm. Activating this muscle will lead to the excursion of multiple tendons. Furthermore, the tendons themselves create constraints due to their configuration or close proximity [40].

Overall, the dynamics of the hand are rather complex which means in general a simplified model is used where some assumptions are made on the constraints of the joints. Multiple models exist each consisting of their own simplifications and assumptions. These assumptions usually originate from one of two natures: the range of motion of the joint is constrained due to the physical structure of the hand/finger or there is a constraint on the movement between the joints and fingers [35]. The constraints can afterwards be used to simulate the movement of a hand.

For wearable robotics or the design of a human-like actuator, data on the human hand is required which can be obtained using contact-based or vision-based sensors. Contact-based sensors exist in the form of a hand glove, surface electromyography (sEMG) and using optical markers. Vision-based sensors on the other hand, mainly work based on normal or more complex cameras [37]. sEMG collects the electrical signal from the muscles to measure possible contraction [41].

1.6 Rehabilitation and traditional robotic gloves

The hand plays a very important part in the activities of daily life. This means that when the ability to move your hand is lost, many of these tasks can no longer be performed without external help. The loss of ability to use a hand can be the consequence of injuries including a stroke or spinal cord injury [42]. Due to complete or partial loss of movement of the hand, the quality of life can be reduced significantly [43]. It has been proven that rehabilitation helps to regain the functionality of the hand. Usually, this rehabilitation happens in the form of repetitive task practice. This involves breaking down a task in individual movements and practicing these fundamental movements to improve hand strength, accuracy and range of motion [44].

A big issue with this however is that these rehabilitation methods are rather time consuming and costly. These issues could be solved by making it possible for the patient to carry out these exercises alone at home or at the hospital to achieve a faster recovery and better results [44]. The application of a robotic (exoskeleton) glove offers a solution to this issue.

Contrary to a prosthetic hand, a robotic glove is designed to fit around the human hand. The gloves are meant to actuate a (partly) paralyzed human hand. This also means that the hand anatomy and motion of the human hand must be considered when developing a robotic glove to minimalize the discomfort of the user [2]. The glove can either be used to help with rehabilitation, but also to help support activities of daily life. The advantage of these gloves lies in the fact that it can provide a continuous motor stimulation and give feedback on the training [35].

The first developed robotic gloves originate from traditional robotics where rigid structures and linkages are used (see Figure 8). These rigid gloves usually make use of an electrical motor as actuator and have a rather complex design making them usually quite heavy, bulky and less compliant [45]. In general, these gloves have many disadvantages such as inconvenient operation, limited freedom, complex structural design and control and difficult to wear due to excessive weight. The actuators used in rigid gloves are commonly less compliant than the joints of the human hand resulting in poor comfort of the wearer [43]. A final issue with these rigid gloves comes from the fact that misalignment between the joint of the hand and the rigid glove can occur resulting in once again discomfort of the user which could lead to improper rehabilitation [42].

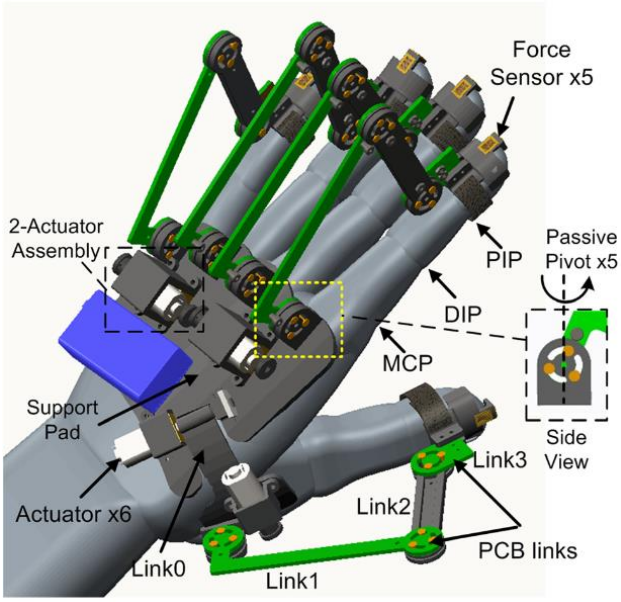


Figure 8 A rather complex design of a traditional robotic glove which leads to many disadvantages [46].

1.7 Soft robotic gloves

Some of these issues found in traditional robotic gloves can be solved by making use of soft robotics, resulting in the soft robotic gloves. Soft robotics is very interesting for biomedical applications including rehabilitation. A soft robotic glove is made out of soft materials which can be actuated like a natural hand. Using soft robotics in robotic gloves creates many

advantages such as a decrease in weight, safer human-robot interaction and simpler design for a high number of DOF. Moreover, the alignment of the joints is less of a concern [2].

These soft robotic gloves are usually made out of fabric, plastic or polymers. Another advantage is that they can easily be designed based on the patient's finger lengths. These gloves can contain actuators, control systems, feedback sensors and sensors to find out which movement is wanted (such as sEMG) [2].

The weight of the glove and actuator needs to be taken into account whereby the weight of the glove must not exceed 500 grams to avoid any discomfort, while the actuator unit should not weigh more than three kilograms so that it is easy to carry [2]. This is a disadvantage for pneumatic actuators since the actuation unit is heavier compared to the one of tendon-driven actuators. Besides the benefit of being capable to apply more force while holding an object, for which the minimum force required contains 10-13N, the tendon driven actuators do not need to be airtight making them easier to manufacture [2].

1.8 multi-material

In the human hand different materials, including bones, ligaments and tissue featuring different properties, all collaborate which emphasizes the great importance of the interface between them. In soft robotics, using the same principle, more complex actuators can be achieved using the strength of multiple materials combined. Hereby, the physical connection between the different materials is crucial in these types of designs as they are not easily achieved for flexible materials with different properties and they tend to lead to failure or damage due to stress concentrations at the interface. Therefore, soft actuators are commonly made out of one flexible material where the properties of the actuator are greatly influenced by the material selection. Research on manufacturing of multi-material actuators usually focusses on additive manufacturing. The main multi-material actuators being manufactured this way contain integrated sensors. As previously mentioned, multi-stage casting is an alternative method of making multi-material components, but has the disadvantage of weak interfaces between the material since there is no chemical bonding and instead makes use of physical adhesion. However, using the self-healing materials described in section 1.4 **Self-healing materials**, a similar approach as multistage casting can be employed to create chemical bonding by first casting the two materials separately followed by fusing them together using the same cycle applied for healing [22].

1.9 Summary state of the art

1.9.1 Rigid glove vs soft glove

In the state of art, both rigid exoskeleton hands and soft exoskeleton hands have been mentioned. This section compares both gloves mentioned above to determine the best suited type of exoskeleton hand for the goal and scope of this project. The benefits of each type of glove are mentioned in Table 1 while their downsides are represented in Table 2.

Table 1 Comparison advantages of rigid and soft exoskeleton gloves [47].

Rigid exoskeleton hand	Soft exoskeleton hand
High output force.	Compliant thus reduced discomfort observed by the wearer and easy to wear.
Precise control is possible due to the high stiffness of the used structures.	Cheap and lightweight design resulting from inexpensive required materials.
Damage resistant.	Easier manufacturing methods, such as casting, are applied.

Table 2 Comparison disadvantages of rigid and soft exoskeleton gloves [47].

Rigid exoskeleton hand	Soft exoskeleton hand
Heavy and bulky which could lead to fatigue after prolonged use.	Easily damaged by external factors.
Compliance originates from the design which is rather complex.	Difficult to control due to many non-linearities arising from the material.
Less compliant than the real hand and misalignment of the joint of the exoskeleton hand and human joint may induce discomfort.	Lower output force.

Based on the observations described above, rigid exoskeletons are preferred for applications requiring a high output force or very precise control. If these requirements are not demanded, usually soft exoskeleton hands are the favoured option. For this project, self-healing materials are available thus eliminating the major drawback that soft exoskeleton hands are easily damaged and shifting the choice in favour of the soft variant over its rigid counterpart.

1.9.2 Tendon vs pneumatic actuator

From the state of the art, two possible actuators applied in soft exoskeleton hands can be distinguished: the pneumatic and tendon-driven actuator. Analogous to the previous section, the advantages and disadvantages of a pneumatic and tendon-driven actuator will be described in Table 3 and Table 4 to identify which type of actuation fits the application better.

Table 3 Advantages of a pneumatic actuator and tendon-driven actuator [2] [47].

Pneumatic actuator	Tendon-driven actuator
More compliant thus increased comfort [47].	Force is directly applied to the digit instead of the joints.
High power to weight ratio.	More intuitive design as the human fingers also use tendons.

Table 4 Disadvantages of a pneumatic actuator and tendon-driven actuator [2] [48].

Pneumatic actuator	Tendon-driven actuator
Heavy actuation system including compressor, storage tank and valves.	Requires separate actuation for flexion and extension.
Only a small moment output is generated [49].	Loss of force and discomfort can occur due to friction caused by the tendons.
Airtightness requirement increases manufacturing complexity.	Certain movements can break the tendons.
maintaining a constant gripping force over time is difficult.	

As the main application in this project involves the grasping of objects, drawbacks such as the change of gripping force over time and the heaviness exclude the pneumatic actuator to be suitable. Furthermore, the simpler manufacturing process resulted in the conclusion to build a tendon-driven actuator.

1.9.3 Aim of the thesis

As mentioned above, pitfalls of soft exoskeleton hands include the low output force. This issue can partly be resolved by guiding the tendon through an actuator instead of attaching it on a glove to increase the distance between the tendon and the finger creating a larger moment. Moreover, the tendon-driven actuator can be equipped with sensors to provide monitoring and feedback during rehabilitation and avoids that the force is directly applied on the finger thus increasing the comfort of the wearer.

On the other hand, the application of this actuator causes a new source of loss in force to occur. Force will be lost if the bending profiles of the actuator and finger are non-identical and induces stresses on the finger provoking discomfort for the wearer. To resolve this issue as much as possible, a study will be performed to ensure an actuator bending profile which closely approaches the one of a human finger. The goal of this thesis is thus to create a tendon-driven actuator that mimics the bending profile of a finger and can be applied for a soft exoskeleton glove.

1.9.4 Outline thesis

In the next chapters, first, reference data of the grasping motion of a finger was obtained. Next, the materials, including their preparation and properties, are discussed followed by the evaluation of the structure of the applied simulations. Afterwards, a section will be devoted to the manufacturing process of the tendon-driven actuator. The created actuators will be characterized and used for validation of the simulations. Chapter 2**: Materials and methods** handles the used materials and methodology, while the results are presented in

Chapter **3: Results**. Finally, the conclusions and future works of this thesis can be found in Chapter 4**: Conclusion and future work**. A general overview of the thesis outline is displayed in Figure 9.

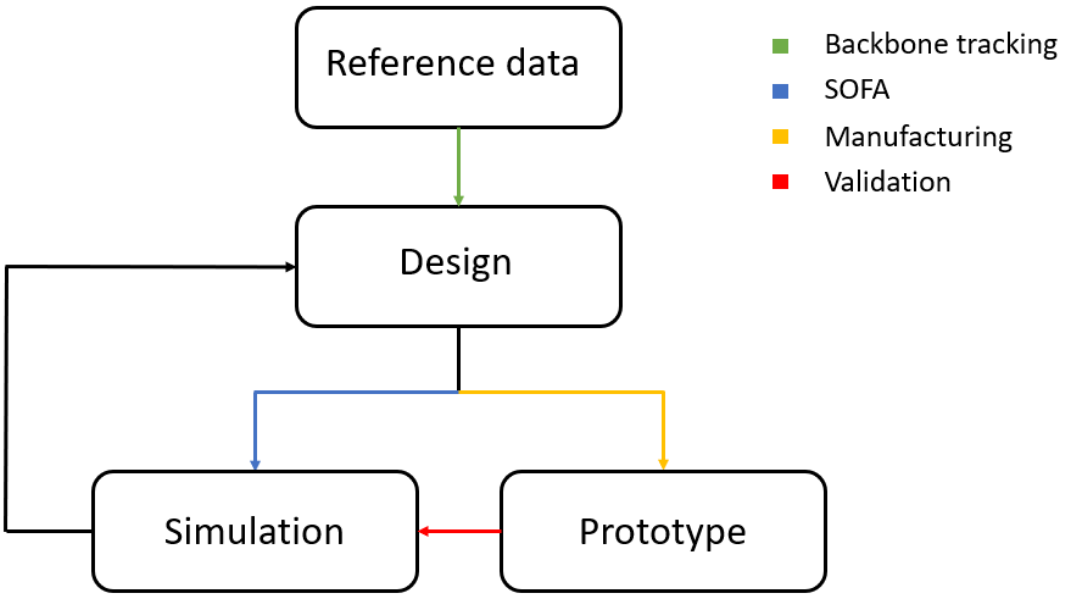


Figure 9 A flowchart explaining the methodology of the thesis in order to develop the actuators.

Chapter 2: Materials and methods

2.1 Backbone tracking algorithm

To develop an actuator that imitates the bending movement of the human finger while grasping, it is important to obtain data as a reference which can either be used for comparison with simulation results or to obtain data from the actuator by employing a similar approach. For this reason, an algorithm was written to retrieve the three main parameters to describe the bending of the finger: the bending angles of the joints. Apart from this, the link lengths can be determined throughout the grasping motion, but will be more accurately measured using a ruler and assumed constant.

The algorithm takes two videos as input: one for calibration and one involving the actual movement of the finger. The calibration video consists of a 360° rotation of the checkerboard pattern, while the video of the finger displays a black background and a green glove equipped with white markers. The next sections will clarify this setup.

2.1.1 Calibration

A calibration video is used to correct the distortion of the camera lens and thus corrects the frames of the video involving the finger motion. The video is divided into pictures of each frame via the cv2 library of python. Approximately twenty frames of the checkerboard in different configurations are required to provide a good calibration. The required twenty frames are obtained via the selection of a uniform timeframe between each frame over the complete video. In theory, one could use all frames but little improvement of the calibration will be achieved at the cost of significantly increased computational time. An example of a frame is given in Figure 10.

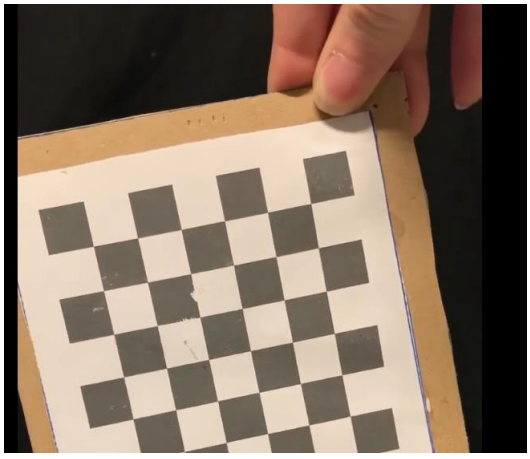


Figure 10 An example of a picture with checker pattern used to calibrate the camera.

The selected frames are first converted to grayscale followed by a built-in function (findChessboardCorners), which requires the number of squares both in length and width, to uncover the corners of the checkerboard. Next, the corners are plotted on the image (Figure 11). This is not necessarily required but makes it possible to confirm the algorithm detects the corners correctly. Next, a function (calibrateCamera) is used to calibrate the camera using as input the previously obtained corner points. The parameters resulting from this function will later be used on the frames of the finger video to compensate for the distortion of the camera.

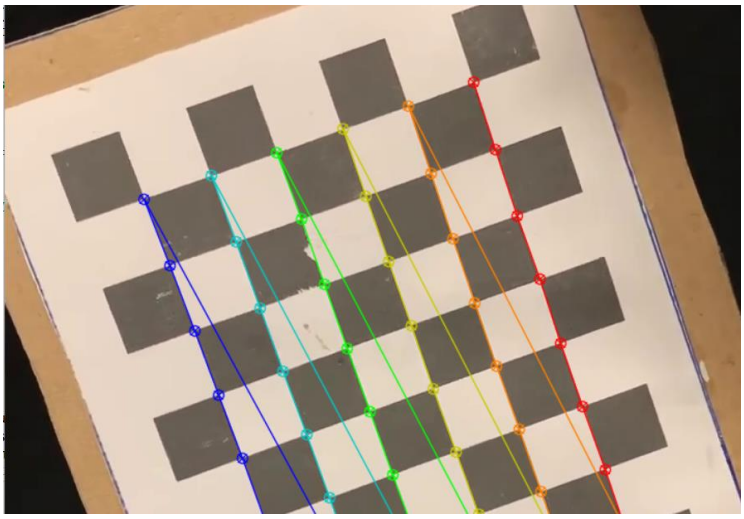


Figure 11 The found corner points drawn on the picture to verify the calibration was done correctly.

2.1.2 Image processing

The finger video is split up in separate frames analogous to the calibration video. Here again, in case the video is long there is the option to not save every frame but only one frame out of two or three frames. However, for the videos used in this project, all frames were processed to maintain a fluent video and avoid interrupted data.

The saved frames are undistorted using the previously obtained parameters from the calibration and the size of the image by applying a series of functions which detects a new optimal camera matrix (internal parameters of the camera) and implements this to remap the image. Afterwards, the frames are re-saved and can be converted into a video via the python cv2 library to verify the correctness of the calibration (Figure 12).

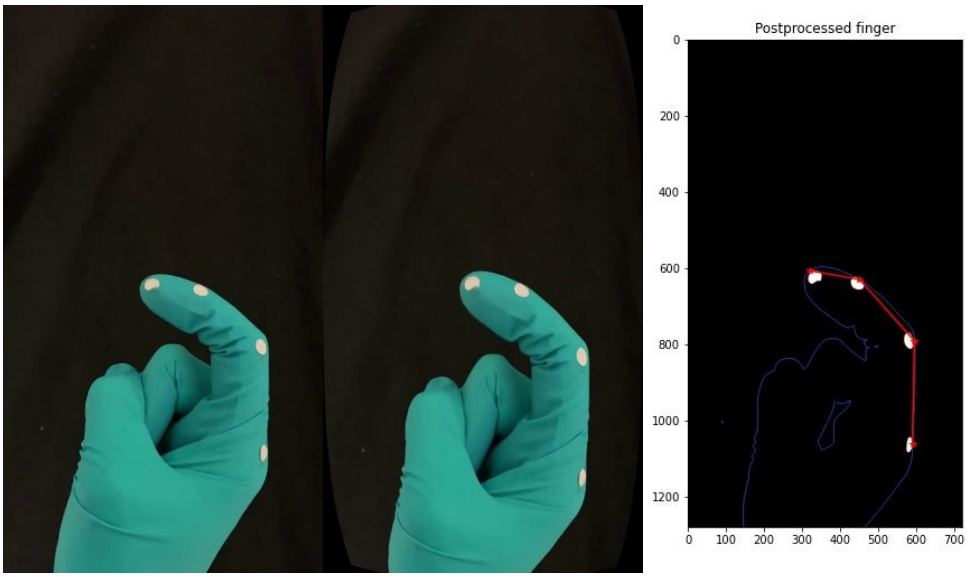


Figure 12 Comparison inputted image with a calibrated image. It can be seen that the calibration makes a slight change on the shape of the finger. The post-processed result of the picture is also shown.

The frames are now ready for the actual algorithm to obtain the bending shape of the finger. As mentioned, the video consists of markers that are placed on four specific points: one on each joint and one on the fingertip. The goal of the code is to have an accurate tracking of the

backbone of the finger. Evidently, correctly placing the markers on the designated locations is rather difficult and tedious, thus the markers were put in the middle of the width of the finger instead of the backbone. Consequently, connecting these markers through their centres will introduce a great error. To resolve this issue, the connection points were selected at the edge of the finger closest to the marker. The way this is handled in the code will be explained in the next sections.

The algorithm which provides the bending shape of the finger consists of fundamental functions required for the code to operate correctly. The *BoundaryCheck* function deals with the issue related to neighbouring pixels. When a pixel located on the frame border is given as input, some neighbouring pixels will be located outside the frame, resulting in an error. It has the list of neighbours obtained in the next function as input, loops over the pixels in this list and checks if they are in the frame. If not, this pixel is removed and the function is called again. It returns thus an updated list of neighbours that ensures all the neighbours are in the frame. A flowchart of this function can be found in Figure 13.

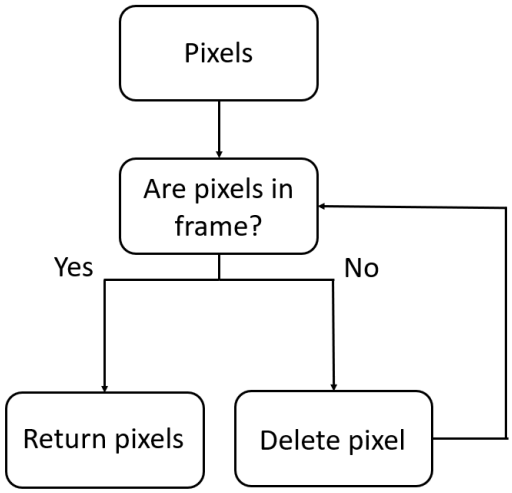


Figure 13 A flowchart of the BoundaryCheck function.

The second function is *findneighbours* which takes a certain pixel, the number of neighbours (either four or eight) and a threshold value as input to provide a Boolean as output. Whether or not the input pixel is located on the border of the finger is determined by first obtaining the neighbours of the given pixel. Here, two options can be distinguished: either four neighbours are used (using only the pixels above, below, right and left of the input pixel) or eight (all the surrounding pixels of the given pixel are used). Next, it is checked if all neighbours are in the frame followed by a loop to verify if at least one neighbour has the same colour as the background (in this case black). If so, the output of this *findneighbours* function will be true as this proves that the pixel is part of the finger's border. Important to notice is that this function is only used for pixels that are part of the finger. A flowchart of this function is shown in Figure 14.

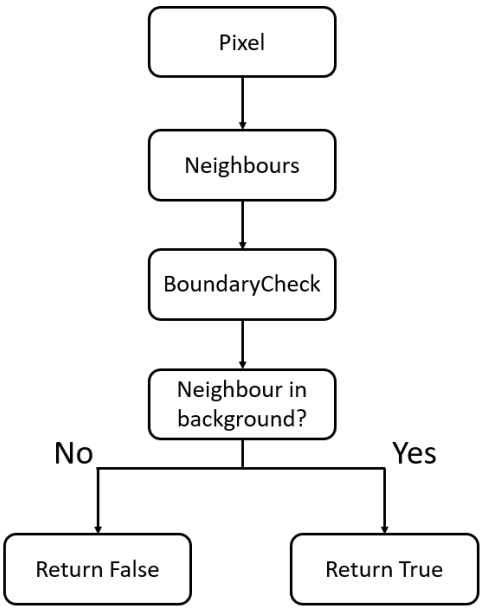


Figure 14 A flowchart explaining the workings of the findneighbours function.

The next function called *DrawLine* is rather simple and takes as input two points and plots a straight line between those points. It thus returns a list of X and Y values of the line as well as the slope. This slope will later be used to determine the joint angles.

The *MarkerOrder* function determines, as the name implies, the order of the markers. This is important to know which points should be connected with each other using the *DrawLine* function but also for the determination of the joint angles. It requires a list of four points as input: the points on the edge of the finger closest to each of the four markers. Additionally, it takes a Boolean *Initialize* that is set to True for the first frame and False for the others. It also takes the already ordered four points from the previous frame.

For the first frame (*Initialize* is set to True), the order is determined using the x-values of the points. The x-values are ordered and the highest value is considered as the base marker, while the lowest value is considered as the marker on the tip. The opposite would work as well as long as the markers are in the correct order looking at the width of the image. This means some configurations of the finger cannot be used as a first frame. However, in case one of these specific configurations is the necessary starting position, one can start recording earlier and cut the beginning of the data to still be able to perform this specific test.

The marker order in the next frames is determined differently compared to the initial frame whereby the correctly ordered points from the previous frame are used to order the markers in the new frame by looping over the four points of the new frame and calculating their distance to the four points of the previous frame. The point from the previous frame that gives the smallest distance will then correspond to the correct marker of the point in the new frame. Here, there is a check built in: in case the frame wrongly determined the edge of the finger or the markers, the distance between the newly obtained points and the previous points will be too large, meaning something is wrong. So, by providing a threshold on the minimal distance, it can be detected when there is a faulty frame which will be destroyed. This function thus

returns an ordered list of the markers and a Boolean that is set to True if the frame needs to be destroyed. A flowchart is shown describing the structure of this function in Figure 15.

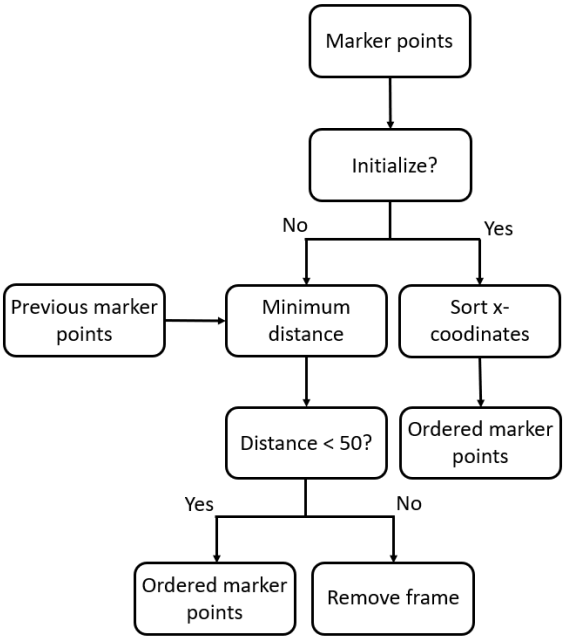


Figure 15 A flowchart explaining the working of the function MarkerOrder.

The final function, *segmentation*, is used to perform the actual segmentation, find the points closest to the markers and return an image where the finger outline and markers are shown in different colours. The function takes as input the thresholds values for the rgb values to distantiate the different colours, the Boolean *Initialization* and the previous obtained points. First, all pixels of the image are looped over and categorized by their rgb value into finger, marker or background. Since the marker is white, all three rgb values are high while the finger is green, so the g value is high while the r value is low. Finally, the background is black, so all three values are low. From the pixels belonging to the finger, the *findneighbours* function is used to save solely the edge pixels. Once each pixel has been put into the correct category, a kmeans algorithm is used to separate the four markers. In other words, using kmeans the set of pixels belonging to markers will be divided into four sets each corresponding to one marker. Next, for each marker the middle point of the marker is determined by taking the median of the pixels' x- and y-values. The median is used rather than the average to reduce the effect of outliers. Once the middle point is determined, a loop over the pixels on the edge of the finger is performed determining the distance between that pixel and the middle point of the marker. The point with the smallest obtained distance is saved and will be used in the function *MarkerOrder* to determine the order of the markers. The function thus returns the processed image, the four points correctly ordered and whether the image should be removed. A flowchart of this function is shown in Figure 16.

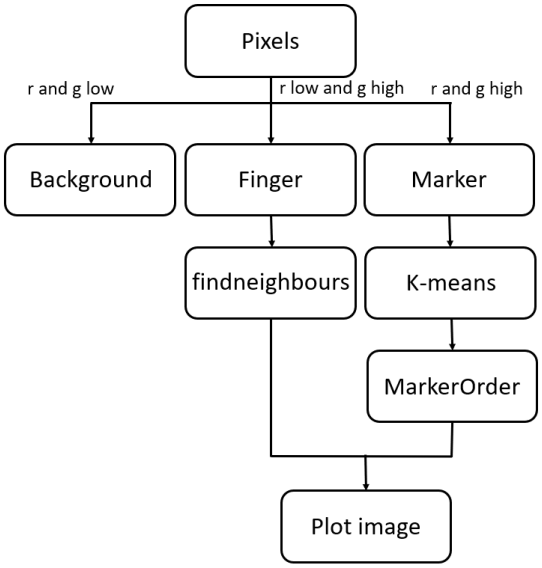


Figure 16 A flowchart explaining the workings of the segmentation function.

The main loop of the algorithm loops over the saved frames obtained after the undistortion of the images. It starts by opening this image and defining some parameters such as thresholds and number of markers. Next, a list of pixel values is created containing the rgb values for each pixel. Then, it calls the *segmentation* function with as input, the points from the previous frame coming from a list saving the obtained data that will later be used to plot the results. If the picture does not need to be removed, it plots on the obtained image received by the function *segmentation,* the four points on the edge of the finger and a line between those points. It also adds the slope and four points to a list that will later be used for getting the joint angles. The obtained image is also put in an image array that will be used to create the postprocessed video. It also gives an output to let the user know which frame is being processed and if the frame was destroyed or not. The overall structure of the algorithm is shown in Figure 17.

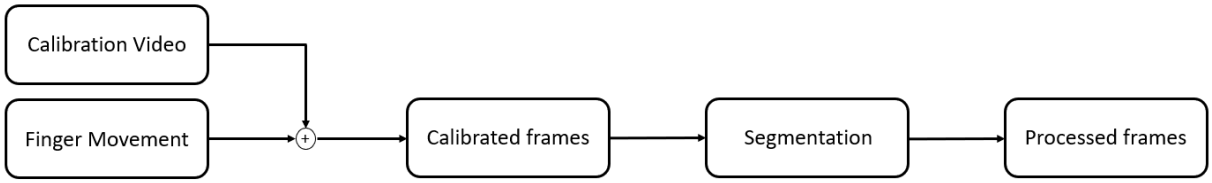


Figure 17 Flowchart explaining the overall structure of the algorithm.

2.1.3 bending angle and link lengths

The final step is to calculate the joint angles and link lengths for each frame and plot them over time. The size of one pixel is a parameter given by measurement of the length of the first link using a ruler. The link length could be obtained by determining the distance between two points. The angle of the first joint can be obtained by simply looking at the angle made

between the y-axis and the line drawn between first and second joint ($\tan\theta = \frac{x_2 - x_1}{y_2 - y_1}$). For the other joint angles, the change in slope can be used with this formula: $\tan\theta = \frac{m_1 - m_2}{1 + m_1 m_2}$.

2.1.4 Actuator processing

The previously obtained code can be adapted to post-process the bending profile of a manufactured actuator. In this case, the code can be simplified by no longer trying to track the complete actuator but instead only tracking the markers. The background will remain black while the markers will be white. Since only a distinction between black and white needs to be made, the rgb values will be replaced by looking at the intensity of the pixels. Once the markers have been found, the median is taken after using a kmeans algorithm just as explained in section 2.1.2 **Image processing**. This algorithm will be used to be able to compare the experimental data with the simulations as well as to characterize the actuator. The simplified segmentation function can be found in the flowchart given in Figure 18.

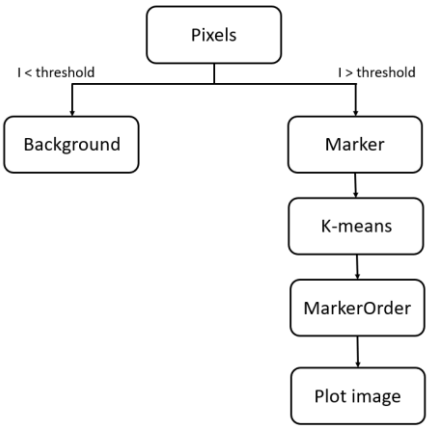


Figure 18 Simplified flowchart of the segmentation function in case of post-processing an actuator.

2.2 Material preparation and properties

In this thesis, four different materials will be used in the final inner actuator. One of these materials is PLA with a Young's modulus of 3.5GPa and Poisson coefficient of 0.35. The other three materials are different self-healing materials. The properties of two of these three will be acquired and compared to reference data found in [50]. The three self-healing materials are BMI1400-FT5000-r0.7, BMI689-FT5000-r1 and BMI689-FT3000-r0.6. The final's material properties will not be determined but are taken from [50]. The Young's modulus of this material is 0.92 MPa with a Poisson coefficient of 0.45. The gelation temperature is 101.5 °C.

2.2.1 Material preparation

The self-healing material is made based on a chemical reaction between maleimide and furan. A third reagent, 4-tert-Butyl-catechol (minimum purity of 99 %), is used to inhibit possible side reactions at high temperature. To have the correct quantities of the three reagents, an excel is used. Important to notice is that among others the ratio depends on the required material properties. Once the ratio of each material is determined, the material is made following a couple steps. The way explained here is not the only way to make this self-healing material but was the method used for this thesis.

The first step is weighing the correct amount of maleimide (BMI) and 4-tert-Butyl-catechol and combining these two reagents. Since the 4-tert-Butyl-catechol is a powder and the BMI is a viscous liquid (much like honey), the powder needs to be melted to have a homogeneous mixture. This is done by heating them up to around 70°C in an oven. Once the powder has completely melted, the final reagent, furan can be added. Possibly, pigment can be added to give the material a specific colour. Next, the materials need to be well mixed together to have a homogeneous mixture. Once this homogeneous mixture is reached, the mixture is placed in a centrifuge to avoid air bubbles appearing during casting.

2.2.2 Tensile testing

Two materials will be tested that are used in the casting and thus manufacturing of the actuator. Both differ greatly in material properties. The material will be characterised using two tests: a tensile test and a rheological test to characterize the ultimate strain, ultimate stress, Young's modulus and gelation temperature. The found values will be compared to the ones found in [50]. For the tensile test, the Q800 Dynamic Mechanical Analyzer of T.A. Instruments (Waters Corporation, Massachusetts, USA) was used. The samples used were rectangular with a length of 20 mm, a width of 6 mm and a depth of 2 mm. The pull speed applied was 60% strain per minute.

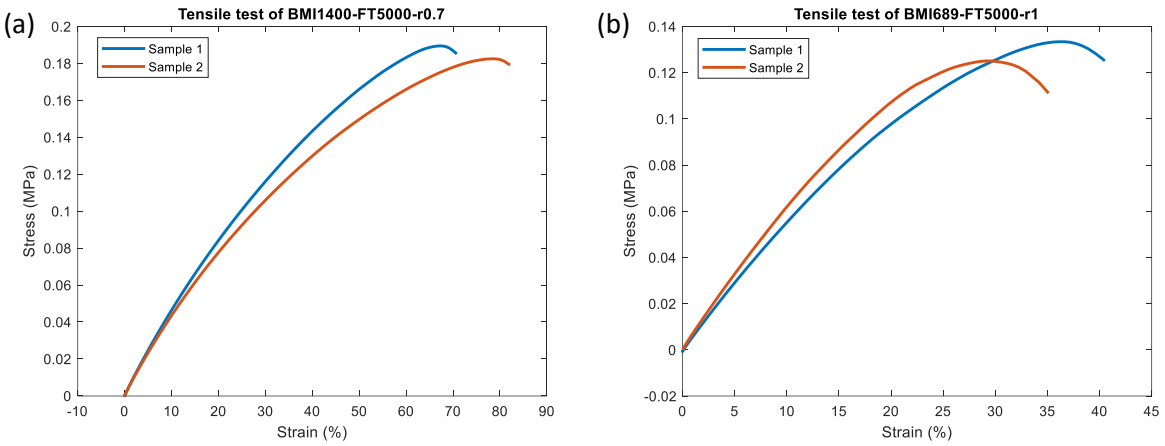


Figure 19 Tensile results of (a) BMI1400-FT5000-R0.7 and (b) BMI689-FT5000-R1 which can be used to determine the ultimate stress, ultimate strain and Young's modulus.

A tensile test was performed on two samples for each material (Figure 19). From this tensile test, the ultimate stress and strain can be found. The Youngs moduli can be calculated at a certain deformation which was chosen to be 15%. Comparing the results found in Table 5 and Table 6 with the ones from [50], it can be found that the ultimate stress and Young's moduli are the expected values while the ultimate strain seems to be lower than expected. This can be explained due to the fact that the tensile test was performed at a different time after the curing of the material. The material is known to have a slight change in properties in the first few days after curing since the equilibrium is not yet completely reached.

Table 5 The calculated Young's modulus and obtained ultimate stress and strain for BMI1400-FT5000-R0.7 for 2 samples as well as the average and standard error.

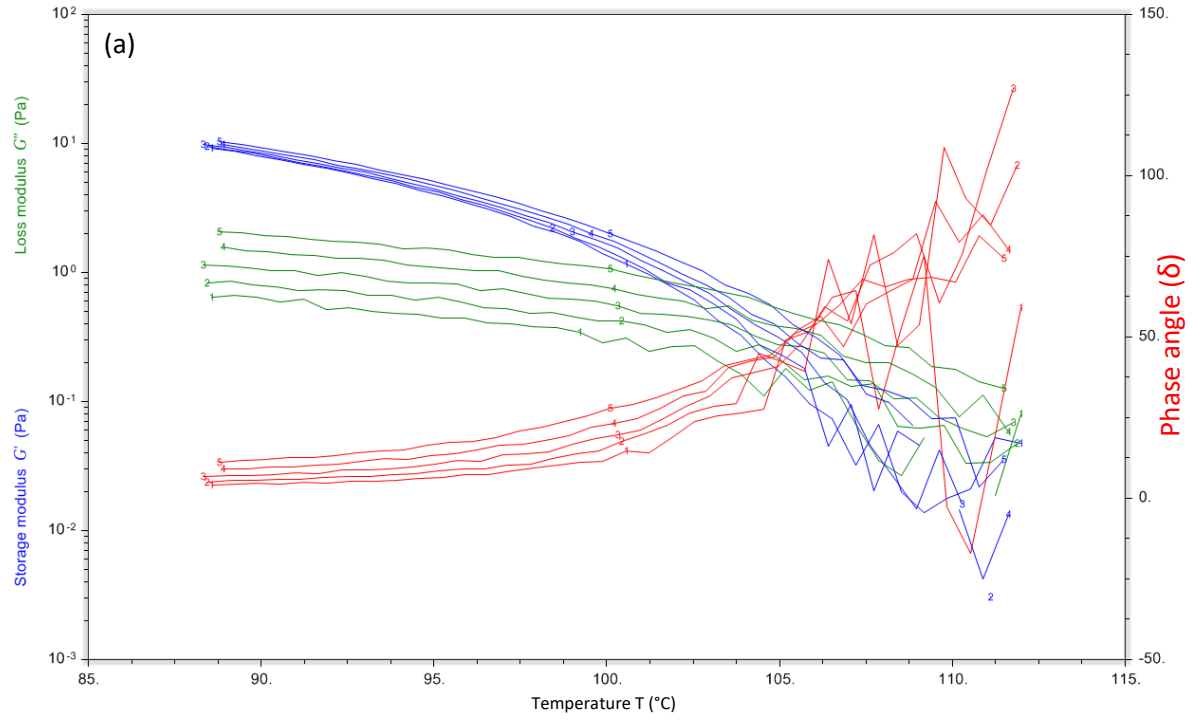
BMI1400-FT5000-r0.7	Young's modulus (MPa)	Ultimate stress (MPa)	Ultimate strain (%)
Sample 1	0.3504	0.1894	70.8
Sample 2	0.3279	0.1825	82.2
Average	0.3392	0.1860	76.5
Standard error	0.011	0.004	/

Table 6 The calculated Young's modulus and obtained ultimate stress and strain for BMI689-FT5000-R1 for 2 samples as well as the average and standard error.

BMI689-FT5000-r1	Young's modulus (MPa)	Ultimate stress (MPa)	Ultimate strain (%)
Sample 1	0.4020	0.1334	40.5
Sample 2	0.4067	0.1251	35.1
Average	0.4044	0.1293	37.8
Standard error	0.016	0.01	/

2.2.3 Rheological test

By using dynamic rheometry to measure the viscoelastic properties of the material, the gelation temperature can be measured. This is done with the Discovery Hybrid Rheometer-3 of T.A. Instruments using 16 mm disposable aluminium parallel plates. To determine the gelation temperature, the change in loss and storage modulus were observed for a temperature ramp (both heating up and cooling down) for multiple frequencies ranging from 0.3 Hz to 3 Hz. The test was performed on a circular sample with a diameter of 16 mm. When the phase angle δ, becomes frequency independent, the gelation temperature is reached.



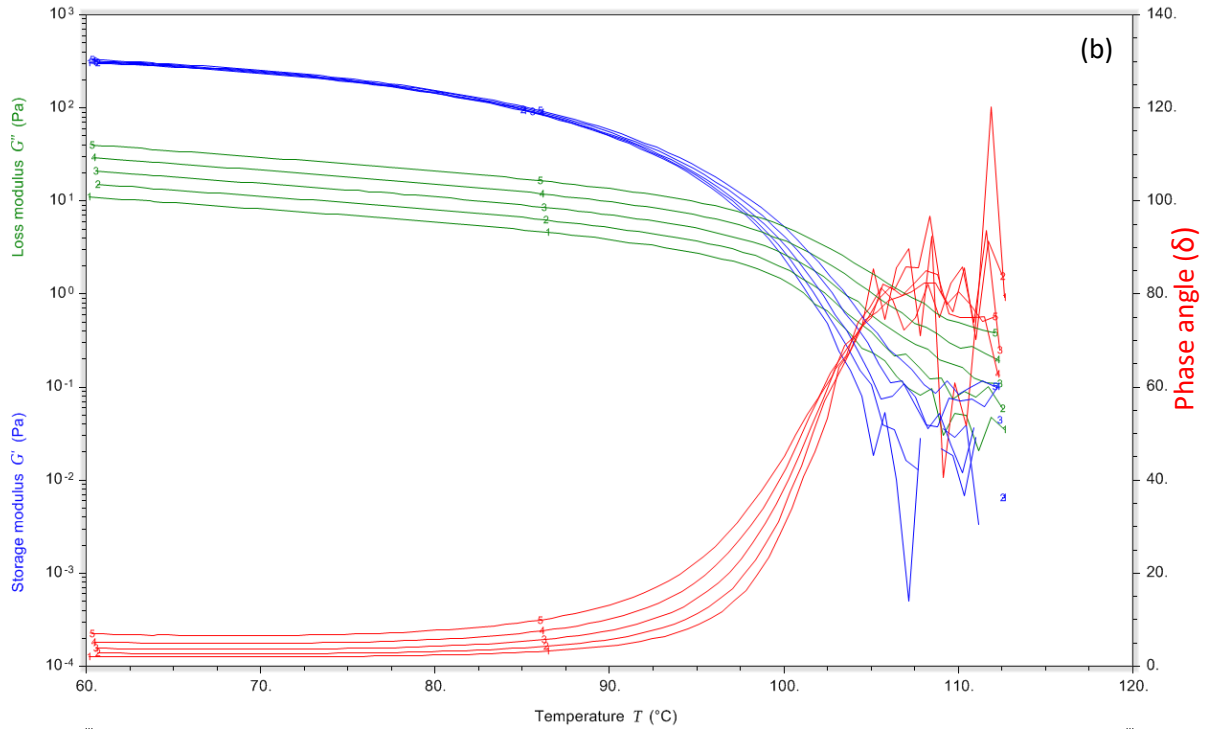


Figure 20 results from the rheological measurement for (a) BMI1400-FT5000-r0.7 and (b) BMI689-FT5000-r1. The gelation temperature of both materials lays around 105°C.

As can be observed in Figure 20, the gelation temperature of the two materials lay close to each other. The gelation temperature of BMI1400-FT5000-r0.7 is 106°C while the gelation temperature of the BMI689-FT5000-r1 is 103°C. As mentioned earlier, the third self-healing material used has a gelation temperature of 101.5 °C. All three materials have thus a gelation temperature quite close to each other. This is important for the fusing of the materials during manufacturing but also in case damage occurs and the actuator needs to be healed. If the gelation temperatures differ too much, some damage would not be healable without losing the shape of one of the other materials. Overall, these values match the ones found in [50].

2.2.4 Self-healing properties

Apart from determining the material properties, the healing properties have also been studied for BMI1400-FT5000-r0.7. This is done by performing tensile tests on healed samples and comparing them to reference samples (Figure 21). These samples come from the same batch and are tested at the same time. To have an idea on how well the healing took place, the average ultimate stress of the healed ones is compared to the reference ones. The relative decrease in ultimate stress is 0.8%. This is a rather small value which means that the material returns to its properties after healing.

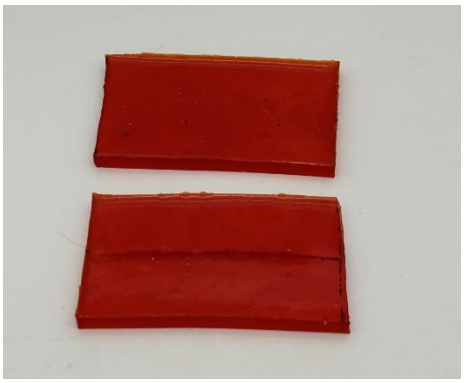


Figure 21 Samples used to determine properties after healing.

2.3 Mechanical design

With the conclusions made from examining the bending motion of a human finger, a design can be made and simulated to look at the bending motion of the actuator. As found in the data for the grasping motion, the second and third joint should have approximately the same joint movement. Furthermore, the length of the links must remain approximately the same throughout the bending motion. Apart from these requirements which are necessary to maintain a natural behaviour throughout bending motion, there is also a requirement on the force output. The actuator should deliver a sufficient force to move the finger and grasp certain objects. The link lengths are based on measurements of a human finger while the maximal joint angles have been found in [39].

2.3.1 A first design

A tendon-driven actuator has been chosen, which adds a complexity for maintaining the length of the finger due to the hinges. A first solution could be to minimize the thickness of the actuator to reduce the amount of change in length of the actuator coming from these hinges (Figure 22). However, this approach is rather naïve since some issues arise of which one is the force output of this actuator that will be rather small and may be insufficient to move the human finger. However, this is a rather simple design and thus a good starting point for simulations.

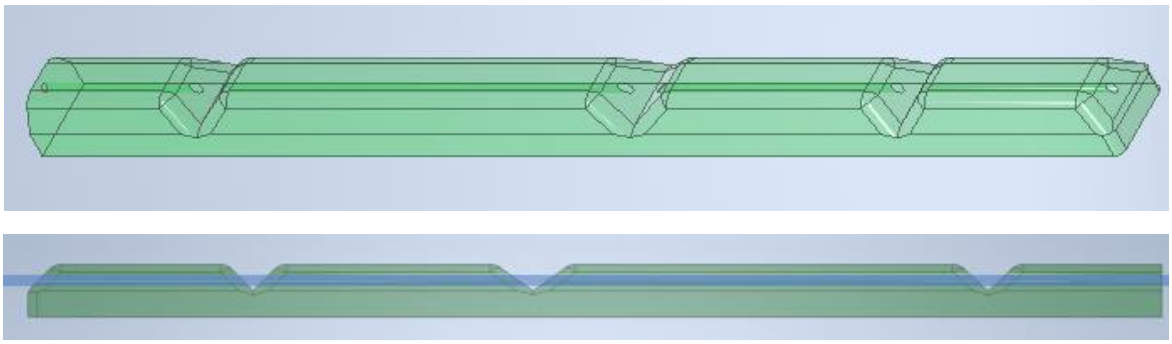


Figure 22 A single material design for simulation purposes.

2.3.2 A multi-material solution – Outer actuator

One way to ensure a high force output, is to use a rigid material in the links. This also solves the issue of maintaining a constant length and shape for the links. The hinges will be made out of very soft and deformable material. An advantage of using such soft material at the hinges is that the actuator can be very compliant at the joints. In case the actuator would not be actuated and the finger is used without assistance of the actuator, the effect of the actuator will be very minimal. Another advantage is that the tendon pull force to move the actuator (not considering the finger itself) can be rather low. It is thus clear that having a design giving both advantages from a rigid design and a soft design can increase performances. Due to this, it was opted to make a multi-material design where rigid parts at the links are placed to increase force output (Figure 23). The rigid material is surrounded and connected using a very soft material that allows the bending at the hinges and creates a good compliance with the finger.

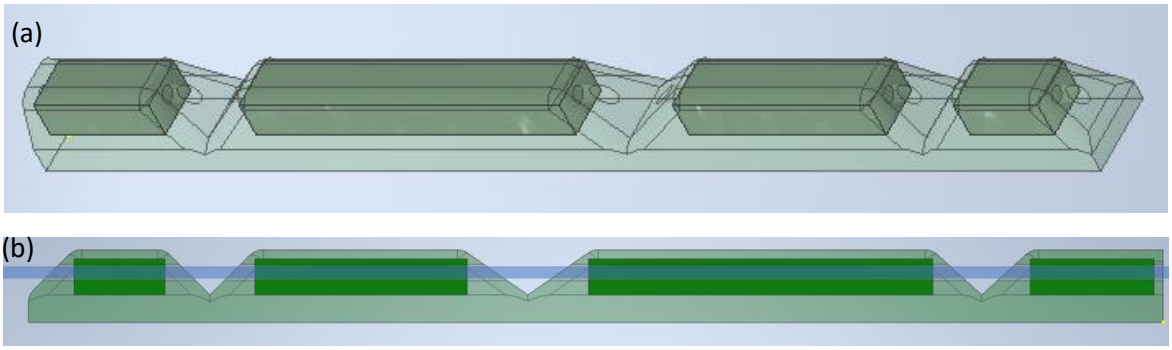


Figure 23 A multi-material design for the outer actuator with in (b) a cut to show the inside of the actuator with the tendon indicated in blue.

2.3.3 A multi-material solution – Inner actuator

Similar to the outer actuator, a multi-material design will be used for the inner actuator. The link lengths have changed to the ones measured using a ruler. The remainder of the actuator is similar to the one made for the outer actuator. The design can be seen in Figure 24.

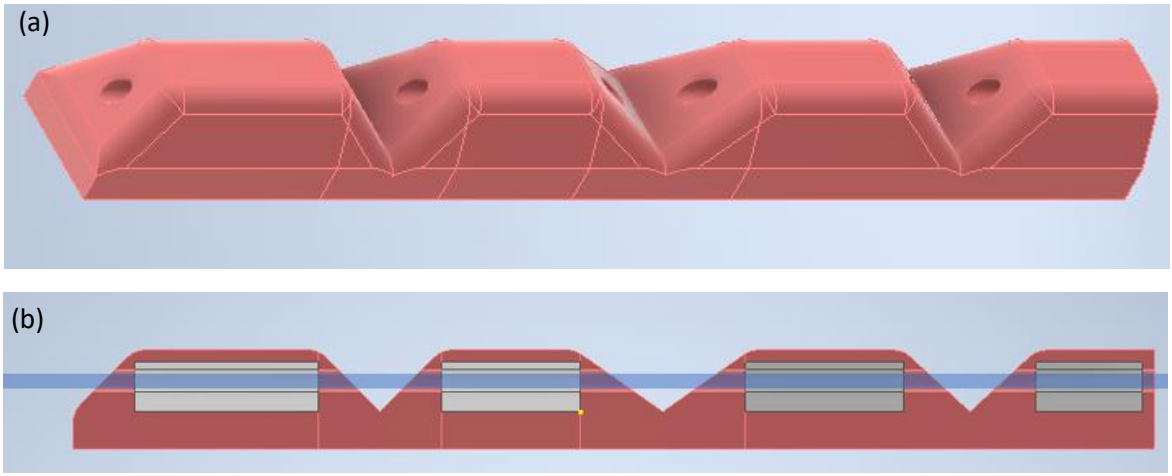


Figure 24 A multi-material design for the inner actuator with in (b) a cut to show the inside of the actuator with the tendon indicated in blue.

The bending profile of this actuator is even more crucial since this actuator will need a higher force output for the grasping. Having this increased force, if the bending profile is not correct,

the force will be applied incorrectly which would also lead to an increased discomfort. For this reason, it can be opted to give a different stiffness for the different joints to ensure a correct bending profile. This would lead to an increased difficulty for the casting as will be discussed in section 2.5.2 **Manufacturing of an actuator with multiple self-healing materials**.

As can be observed in section 3.2.3 **Inner actuator simulations**, the second joint moves faster than the third. The first joint should preferably move as little as possible. For this reason, multiple self-healing materials having different stiffnesses can be used as a solution to match the bending profile of the actuator to the profile of the finger. Two materials have been proposed for which the material properties are obtained and/or given in section 2.2 **Material preparation and properties**. Using BMI689-FT5000-r1 (yellow) for the second joint and BMI689-FT3000-r0.6 (black) for the first joint (Figure 25), the obtained results from simulation are compared to the ones where only one self-healing material (BMI1400-FT5000-r0.7) was used.

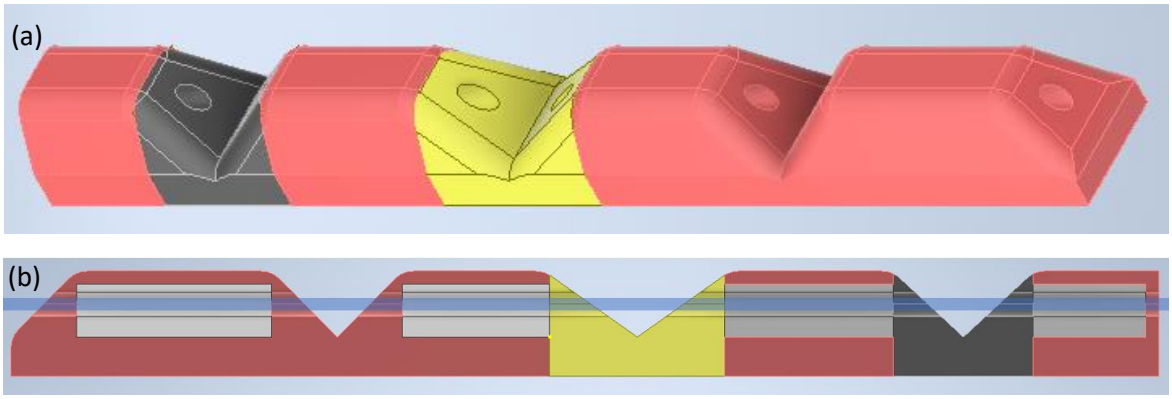


Figure 25 A multi-material design for the inner actuator where each joint has a different stiffness with in (b) a cut to show the inside of the actuator with the tendon indicated in blue.

2.4 SOFA Simulations

2.4.1 SOFA

SOFA is an open-source FEM software that was first released in 2007. SOFA, short for Simulation Open Framework Architecture, makes use of the concept of a scenegraph-based multi-model representation. The objects and algorithm used during the simulation (scene) are represented using a hierarchical data structure (similar to scenegraph). Each object consists of multiple models that can be optimized for one specific computation. As an example a liver can be split up in three models: one for computation of internal forces (internal model), one for collision and one for visualization. Usually one of the models (the internal model) functions as master while the other models function as slaves using mapping [51] [52].

SOFA makes use of different types of solvers. The first type are ODE solvers which have as purpose to apply animation algorithms for each time step by computing velocity and position of the next time step. Next, there are the linear solvers and direct solvers. Due to the FEM, usually the matrices are rather sparse making it possible to have fast computations using sparse factorization. To solve constraints, constraint solvers are used. Those solvers first solve the free motion and constraints separately after which a correction is done on the free motion to consider the constraints [51].

SOFA has been developed for and greatly used in the medical sector [53] [54]. Apart from the medical field, SOFA can also be used for simulations, control and even design optimization in soft robotics [55] [56] [57]. More specifically, SOFA can be used to simulate soft actuators and help to understand the bending profile of the actuator before having built a prototype.

2.4.2 A first model

Apart from SOFA being open-source, another advantage is that the simulations can be programmed in Python. SOFA simulations are based on a simulation graph. This graph consists of nodes that have a certain hierarchy. The nodes are linked using a parent-child relationship. The initialization takes place in the rootNode, on which the other nodes will be created. Each node can be subdivided into four subnodes: the mechanical node, actuation node, collision model and visual model (Figure 26) [58]. Not all four subnodes need to be present: for example, in the simulations performed during this work, collision model will not be used. Also, the visual model is not necessary and could be left out. Starting from code that was provided by ir. Pasquale Ferrentino that was also used for the simulations in [58], adaptions have been made to simulate the actuators described in section 2.3 **Mechanical design**.

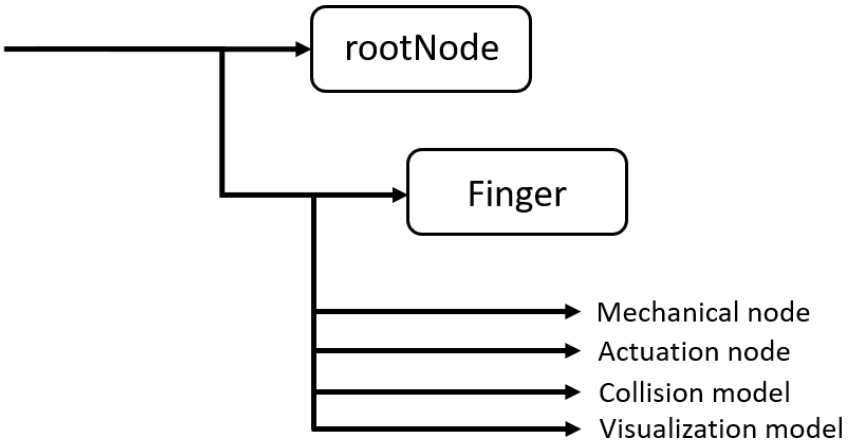


Figure 26 The general hierarchy of a SOFA simulation where everything is built up starting from the rootNode.

The mechanical node is important since all mechanical properties are described here. These consist of for example the mass or density but also geometrical constraints. In this subnode, a model for the material properties needs to be chosen and the parameters need to be set for this chosen model [58].

The actuation node makes use of the SoftRobots plugin to model the deformation due to actuation. Here, the type of actuation is also defined which in this case is tendon-driven. It also animates the motion by building an internal controller. The results for implementing the actuation and mechanical model can be seen in Figure 27.

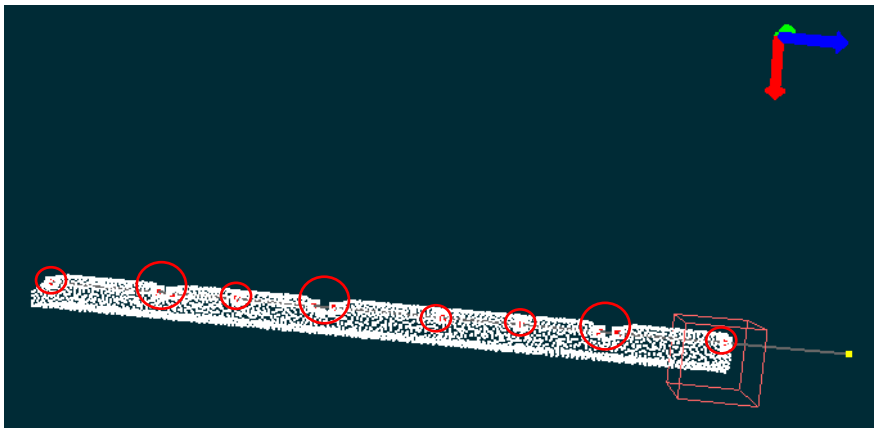


Figure 27 The graphical result after implementation of the mechanical and actuation model where the red circles show the fixed points of the tendon.

The collision model is used in case contact between two objects is simulated. It also requires the location at which the forces will be calculated. Finally, the visual model deals with how the objects are shown (Figure 28).

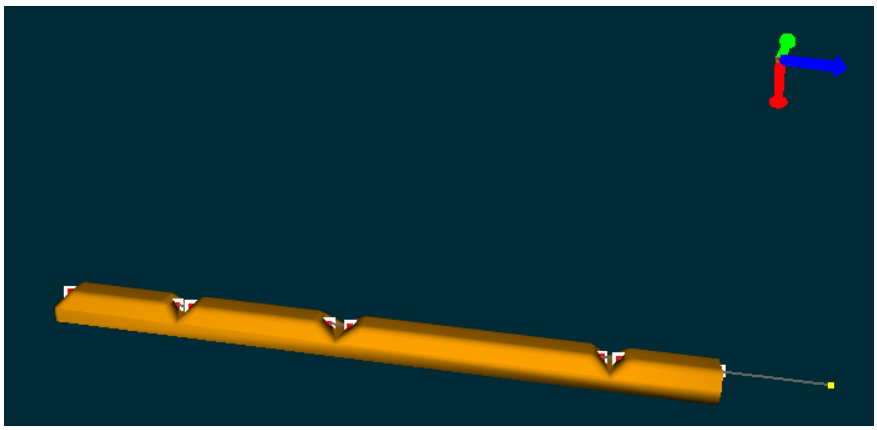


Figure 28 The graphical result when the visualization model is added.

As mentioned, the rootNode is the base of the simulation graph. It defines some global variables such as time step, gravity and the simulation environment. If present, it also describes the type of contact needed for the collision model. The most important function present in the rootNode is the animation loop. This loop updates the scene for each time step. This is done by solving and building the linear equations but also the constraints and collisions. The way this is solved can differ creating different types of animation loops. If the collision and constraints are solved together with the linear equations, the loop is called default animation loop. If first the free solution is determined, i.e., the linear equations are solved without any constraints or collisions and then corrected for the constraints and collisions using a correction factor, the loop is called a free animation loop [58].

Important to understand, is that the rootNode is not the only node containing solvers. For each node, in the mechanical subnode, an ODE solver is used in combination with an integration scheme. In soft robotics, the Euler Implicit Solver and SparseLDL Solver are used that solve the mechanical equations. This solution is then sent to the animation loop which combines the solution of all the nodes to build the linear equations and solve it [58].

Of course, important in the mechanical node, is to define the mass or density of the object. This can be done either distributing the total mass over all the nodes but also by taking the density and integrating over the volume. Another important part is the material characterization for which SOFA offers many options. It was found in [59] that a Neo-Hookean hyperelastic model fits best for the self-healing materials used in this thesis. However, due to an issue in SOFA with the interaction between the SparseLDL solver and the hyperelastic material model, the simulations will be performed using a linear elastic material model. The used materials and their properties are determined and/or given in section 2.2 **Material preparation and properties** and are 0.34 MPa, 0.42 MPa and 0.92 MPa for the Young's moduli for BMI1400-FT5000-r0.7, BMI689-FT5000-r1 and BMI689-FT3000-r0.6 respectively. The Poisson coefficient is assumed the same for all three materials and is estimated as 0.45. For the rigid material, an elastic model containing as parameters the Young's Modulus of 3.5 GPa and Poisson ratio of 0.35 should suffice. A detailed structure of the mechanical model can be found in Figure 29.

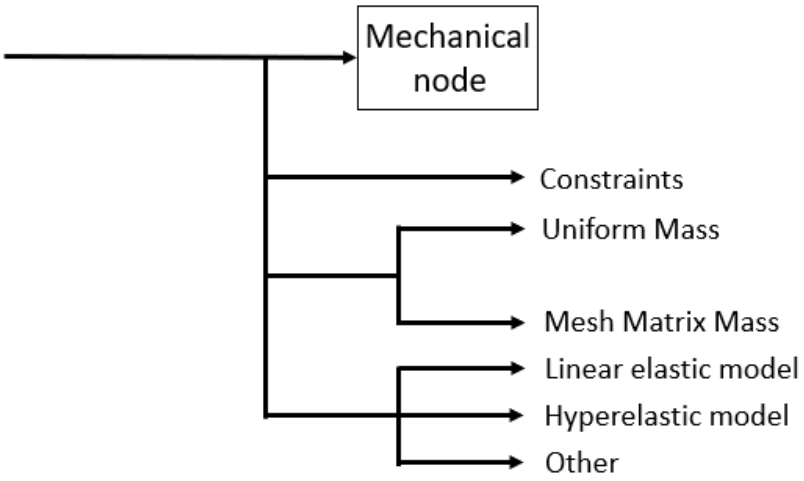


Figure 29 A detailed overview of the mechanical model containing mechanical constraints, material model and mass definition.

As for the actuation model, since a tendon-driven actuator will be simulated, a CableConstraint is used. For this, points need to be given to determine where the cable passes. The cable is assumed inextensible and is controlled using a displacement or pulling force. Also, a pulling point is required. If a pneumatic actuator was opted, the SurfacePressureConstraint could be used.

In order to simulate in Sofa, the CAD file will first be converted to a '.vtu' or '.vtk' file. This is done in Sofa using a separate code in which the CAD file is loaded after which a mesh is generated. Once the mesh is generated, it is saved as a '.vtu' file which can be used in the actual code of the simulation. There exists another way to create and save this mesh which will be explained in section 2.4.3 **A multi-material simulation**.

The structure of the code is as followed: a controller is made to control the finger. This controller is made of an initialization and animated event as well as a create scene. In the initialization, some parameters are defined such as time, nodes and positions. This part also contains the code that is used to track a certain point over time and write the data in a text

file. In these simulations eight points will be tracked: The fingertip, base point and two points for each hinge. In the animated event, the type of motion is coded. In these simulations, the tendon will be pulled using a ramp or in other words, a constant pull rate with a maximum displacement of 7 mm for the single material actuator and 10 mm for the multi-material actuators.

Next, a create scene is made which contains the simulation graph as explained earlier. For the mechanical node, the mass is given by using the density integrated over the volume. The finger is loaded as a '.vtu' file and the model used for the material is a linear elastic model as explained earlier. The finger is fixed using a box that constraints the movement of all the points inside this box. The finger is constrained at its end.

For the actuation model, a CableConstraint is added. The pull point is determined and is located behind the end of the finger at the same height and width as the hole through which the cable is guided. The cable is fixed to the finger at some defined points. These points are the points for which the cable leaves or enters the actuator. A hierarchy of the complete simulation for a single material actuator can be found in Figure 30.

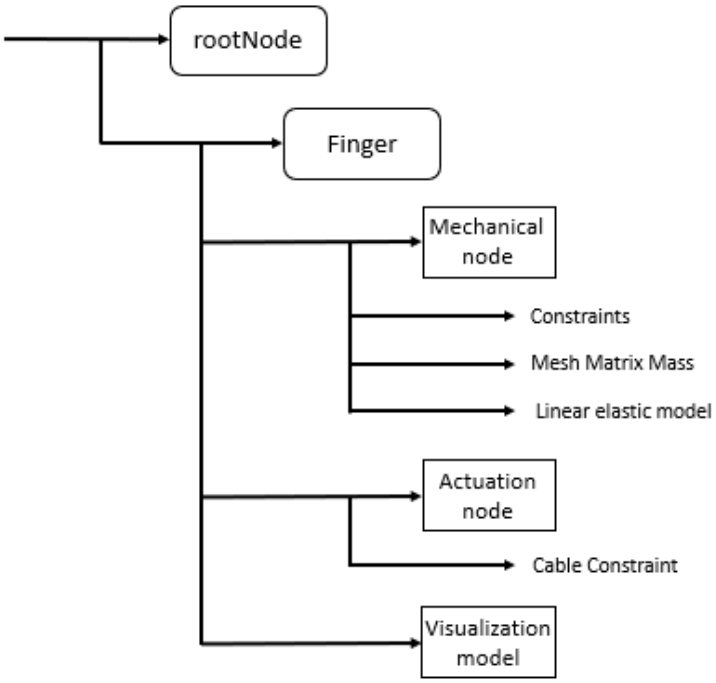


Figure 30 The complete hierarchy of the simulation used for the single material actuator.

2.4.3 A multi-material simulation

A multi-material design was made, and the previously used SOFA code is extended to simulate this multi-material finger. Since the rigid parts are bars, a box can be drawn at the locations of the rigid parts. The tetrahedra in these boxes will be saved under a child node of the actuator in order to be able to give these elements different material parameters. This is done for each bar separately. Since the material used for the rigid parts is PLA, a linear elastic material model can be used. Similarly, the same method can be applied for giving different material properties for a certain joint. This will also be done for the simulation of the final inner actuator. The complete hierarchy for this final inner actuator can be found in Figure 31.

Alternatively, a Matlab script can be used to create a mesh after which a different function can be used to determine the tetrahedra belonging to the rigid material. The advantage of doing it this way, is that rigid parts with more complex shapes than bars can also be separated. In this code, an area can be defined using coordinates of the CAD file and loop over the nodes to select the ones that are in the specified area. For a bar for example this can be done by creating an area by using the centre point of the bar and looking at which nodes are inside the area [$x_c \pm \frac{L}{2}$, $y_c \pm \frac{B}{2}$, $z_c \pm \frac{W}{2}$,] with [x_c, y_c, z_c] the centre point of the bar and L, B and W the length, width and height of the bar respectively. However, it is clear to see that this area can also be defined for a cylinder or other shapes as was done in [60]. Using this script allows for more flexibility in the definition of the rigid area but is more complex. Once these nodes have been determined, the corresponding tetrahedra are determined and saved in a text file that can be used to create a child node in the code for the simulations.

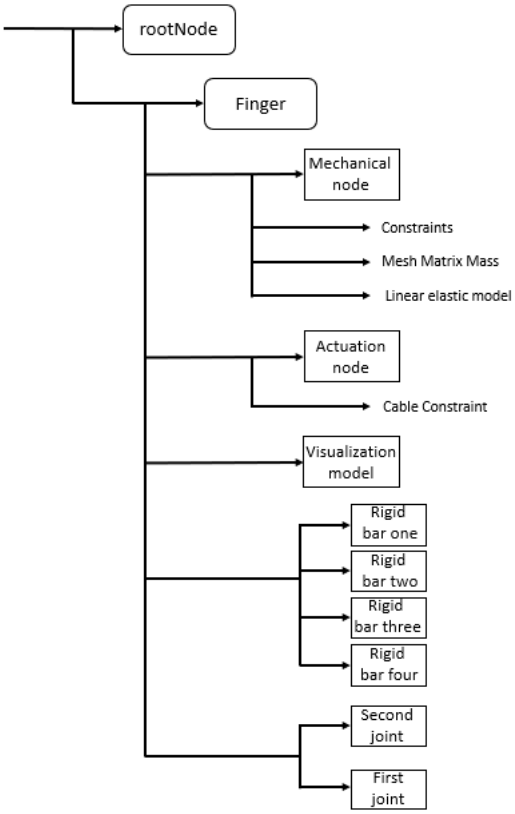


Figure 31 Complete hierarchy used for the simulation of the final inner actuator.

2.5 Manufacturing of the actuator

To manufacture the actuators, it was opted to use casting. Since the finger is multi-material, the finger will have to be casted in multiple parts or stages. This can be done by either multi-stage casting or multiple single-stage castings that can be fused together. In order to make the mould used during casting, first a negative of this mould design is 3D-printed. This 3D-printed part will then be filled with silicone which will create the actual mould. Having this mould built out of silicone rather than a rigid material, allows to remove the actuator easier

after curing of the material. The designs of the negatives of the moulds can be found in Chapter 6**: Appendix**.

2.5.1 Different casting methods

During this thesis, three different casting methods have been used. The first one is a multi-stage casting process. The finger is cast from top to bottom where in the first stage, a small layer of material is cast. On this small layer, the rigid parts can be placed with a cable going completely trough the actuator. After this, the second casting stage will fill up the remainder of the mould with self-healing material. The different steps are shown in Figure 32.

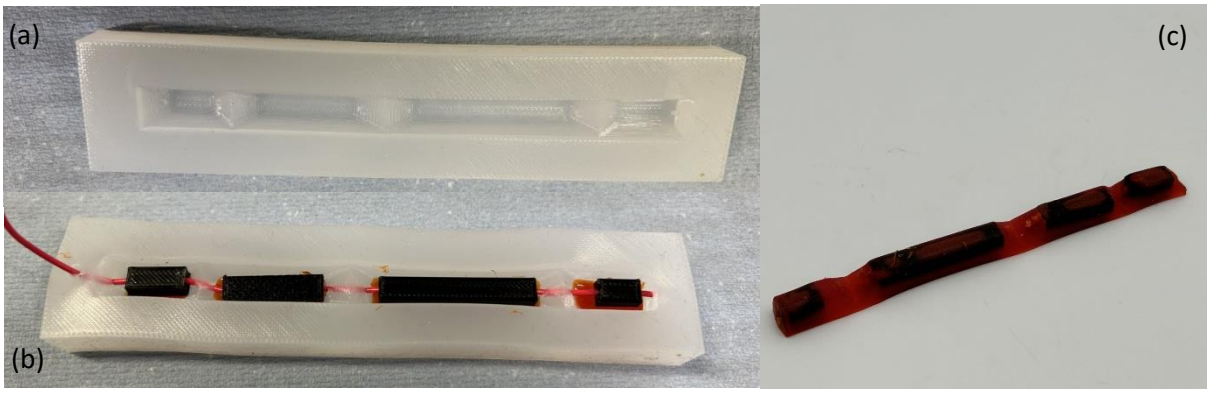


Figure 32 An actuator being manufactures using multi-stage casting. (a) shows the used mould, (b) shows the mould before the second casting step where it can be observed that the rigid parts are placed on the already cast material and (c) shows the final result.

The second method is similar to the first but instead of using multi-stage casting, two single-stage casts are used. Using two moulds, the thin top layer is cast separately from the rest of the actuator. Apart from this, the method of casting for the bulk of the finger remains the same as for the first method. First, the rigid parts are placed in the mould with a cable going through the complete actuator after which the material is cast filling the remainder of the mould. Once the two parts have been casted separately, they are fused together to have a complete actuator. The steps of this method are shown in Figure 33.

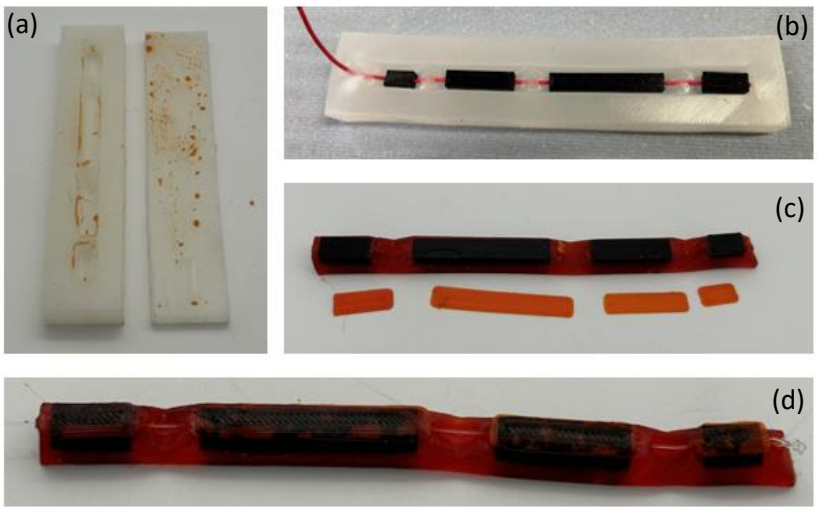


Figure 33 Two single stage casts that are fused together to have the finished actuator. (a) shows the moulds used, (b) the mould preparation, (c) the casted parts before fusing and (d) the finished actuator.

The third casting method consists of two multi-stage casts and is thus more complex. However, it simplifies making a hole through which the tendon can be guided. The finger is split in half and for each half the first method is applied. After this, the two halves can be fused together. The mould and obtained half of the finger can be found in Figure 34. Alternatively, this approach could also be done using four single-stage casts in the same way as explained for the second method.

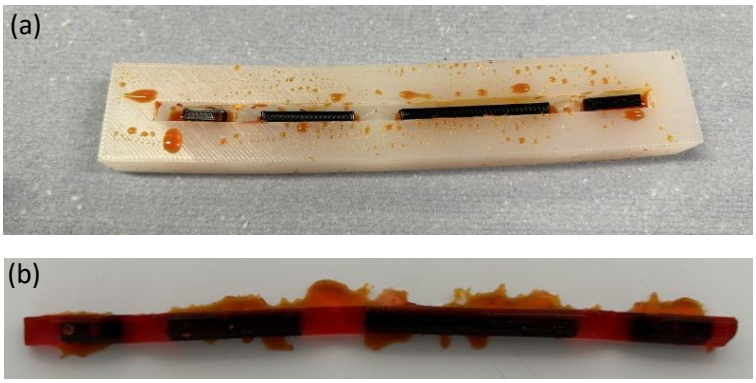


Figure 34 (a) The mould prepared to cast half of the actuator and (b) the result of this casting.

2.5.2 Manufacturing of an actuator with multiple self-healing materials

The previously mentioned casting methods work well in case the finger is constructed using one self-healing material. In case the stiffness of the hinges have to differ to get a closer bending profile compared to the one of the finger, the casting method should be changed as well. For example, if an actuator is made where each joint has a decreasing stiffness, i.e., the first joint is the stiffest and the third joint the softest, the actuator needs to be split up in multiple pieces. Casting such actuator requires six moulds after which the pieces will be fused together. The first two moulds follow a similar approach as the second method mentioned in section 2.5.1 **Different casting methods** to cast the fingertip, third joint and last link. Next, the second and first joint are casted separately since it is made from a different material. The first and second link will also need to be casted separately but by casting the links vertically, the link can be casted in one mould instead of two. As already mentioned, since the materials are chosen such that the gelation temperature of all materials lie in the same range, the actuator can be fused together in an oven at a temperature of 80°C for approximately 30 minutes. The result can be observed in Figure 35.

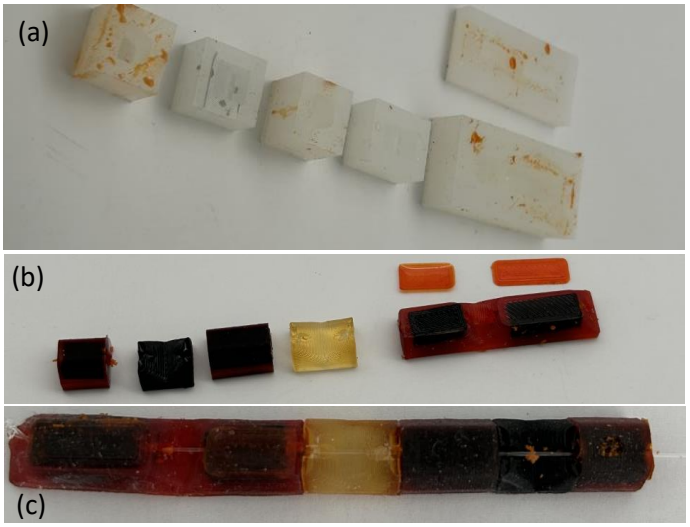


Figure 35 (a) shows the six moulds used while (b) shows the parts after casting before fusing them together and (c) shows the result after fusing.

2.6 Validation and characterization of the actuator

2.6.1 Comparison experimental results and simulations

In order to compare the manufactured actuator with the simulations to verify whether the simulation gives meaningful results, the code developed in section 2.1.4 **Actuator processing** will be used on a video of the bending of the actuator. In this test, the tendon will be pulled manually, which means the speed of the bending is uncontrolled. However, the fact that no set-up is required allows an easy way to have an idea of the bending profile. It can also help to check whether the simulations lie within a range of the experimental results. In section 2.6.2 **Force measurements**, a more thorough set-up will be used to have a more correct bending profile of the actuator. This set-up will give a different bending profile since the actuator is resting on a surface similar to the actuator being attached to the finger. However, the results obtained from using this set-up cannot be compared to the simulations where it is assumed that the actuator is not constrained on any side. For this reason, this experiment is important for the verification of the simulations.

Since no data is available on the tendon displacement during the experiment, the simulation and experimental data will be compared by looking at the bending angle and joint angles for a certain total joint angle. As all joints have the same depth of cut, a change of one degree in any joint, has a similar effect on the tendon displacement. Assuming that all the change of length of the tendon comes from the bending of the joints, the total joint angle can be used, providing similar information as the tendon displacement. This assumption thus means that close to no compression of the links occurs which is acceptable thanks to the rigid parts placed into the links. The bending angle and other joint angles will then be used to determine a relative error between the simulations and experimental data:

$$\frac{|\alpha_{exp} - \alpha_{sim}|}{\alpha_{exp}}$$

However, previous method gives information on the bending angle and joint angles, but does not allow to validate the simulation. Looking at literature, validation can be achieved by looking at the position of the joints as well as the fingertip for both simulation and experiment. By determining the relative RMS error of the points by dividing over the complete actuator length as was done in [60], the simulation can be validated. A simulation is considered valid if this error remains below 5% for any tendon displacement or in this case total joint angle.

2.6.2 Force measurements

Both simulation and previous experiment, assume that the actuator is only constrained at the end of the actuator and free over the remainder of the actuator. However, in reality, the actuator will be constrained to the finger which means the actuator will not be free to move at its bottom side. In order to take this into account, a new set-up is built (Figure 36). This set-up fixes the finger at the back by having a screw push on the actuator. Over the complete length of the actuator, it is supported using a wooden platform. The cable is pulled by attaching the tendon on a part that is fixed on a rail. Using a millimetre, the tendon can be pulled while the displacement of the tendon can be measured. This set-up thus allows to measure the bending profile of the actuator (supported on one side) in function of the tendon displacement. Attaching a force sensor on the part that pulls the tendon, allows to measure the pull force as well.

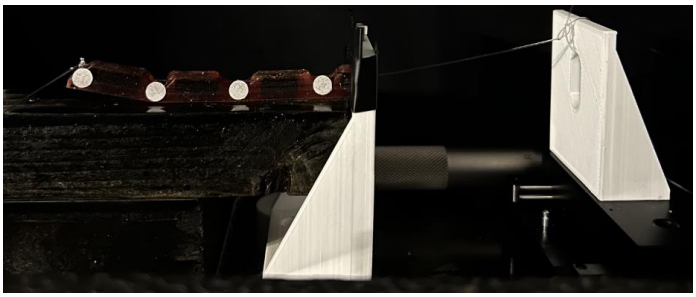


Figure 36 The set-up used where the finger is constrained at the bottom and the tendon is pulled using a railing system.

Chapter 3: Results

3.1 Bending profile during grasping

Using the algorithm described throughout section 2.1.2 **Image processing**, for a video containing two bending cycles, the pictures found in Figure 37 are obtained. Here, the outline of the finger is plotted as well as the four marker points. A line is drawn connecting all the points creating a bending profile of the finger.

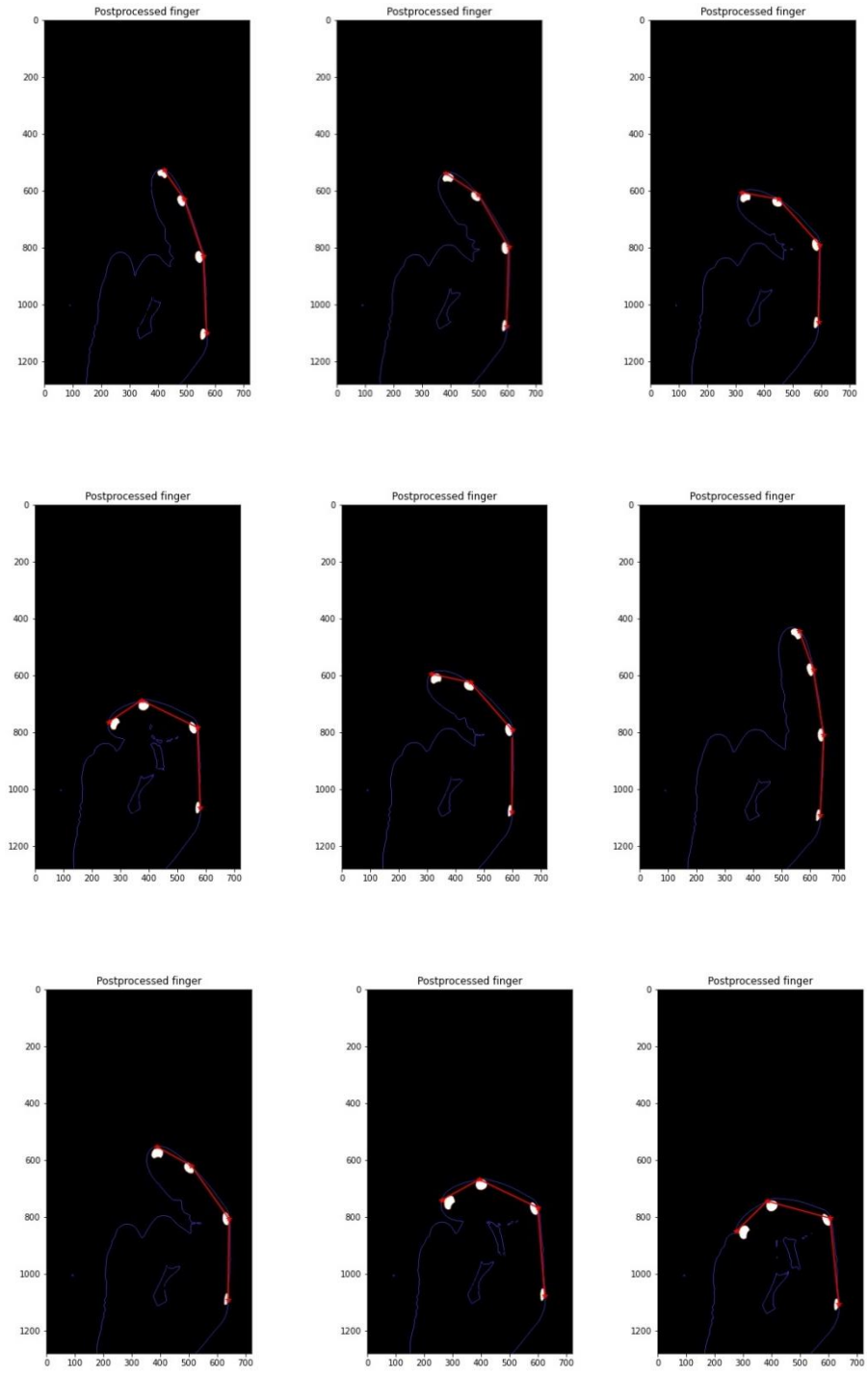


Figure 37 The evolution of the processed finger where the joint angles are marked and connected with a straight line.

The video of the finger that was post-processed and thus used for obtaining the data found in Figure 38, contains two bending motions of the finger. The bending is mainly focused on the second and third joints while the first joint remains constant. Bending was performed in a rather natural way, mimicing the expected joint angles troughout a grasping motion of a human hand. From this code (explained in section 2.1.2 **Image processing**), It can be concluded that the joint angles of the second and third joint change similarly, while the first joint displays different behaviour. This indicates that the comfort of the wearer of the exoskeleton hand can increase by providing a different stiffness for the first joint. Since most of the motion during grasping originates from the second and third joint, increasing the stiffness of the first joint creates more motion in the other joints.

The raw results from the length of the finger were noisy and at first glance did not really show any correlation with the movement. To remove some of that noise, a filter was added. However, the starting length of the three links were verified with a ruler and an error of ± 2 mm was found, as expected taking into account that the picture was treated as a 2D frame throughout the code while in reality there is depth as third dimension. However, A bigger error arose from the assumption that the link between the two joints is straight. While bending, the backbone of the finger starts arcing, thus the length of the bent finger is inaccurate and no real correlation between the movement and length can be made. As a consequence, the code can only be used to determine the length in rest but not the change in length during the bending motion. However, one can assume that the finger length does not significantly change throughout the motion and can remain constant.

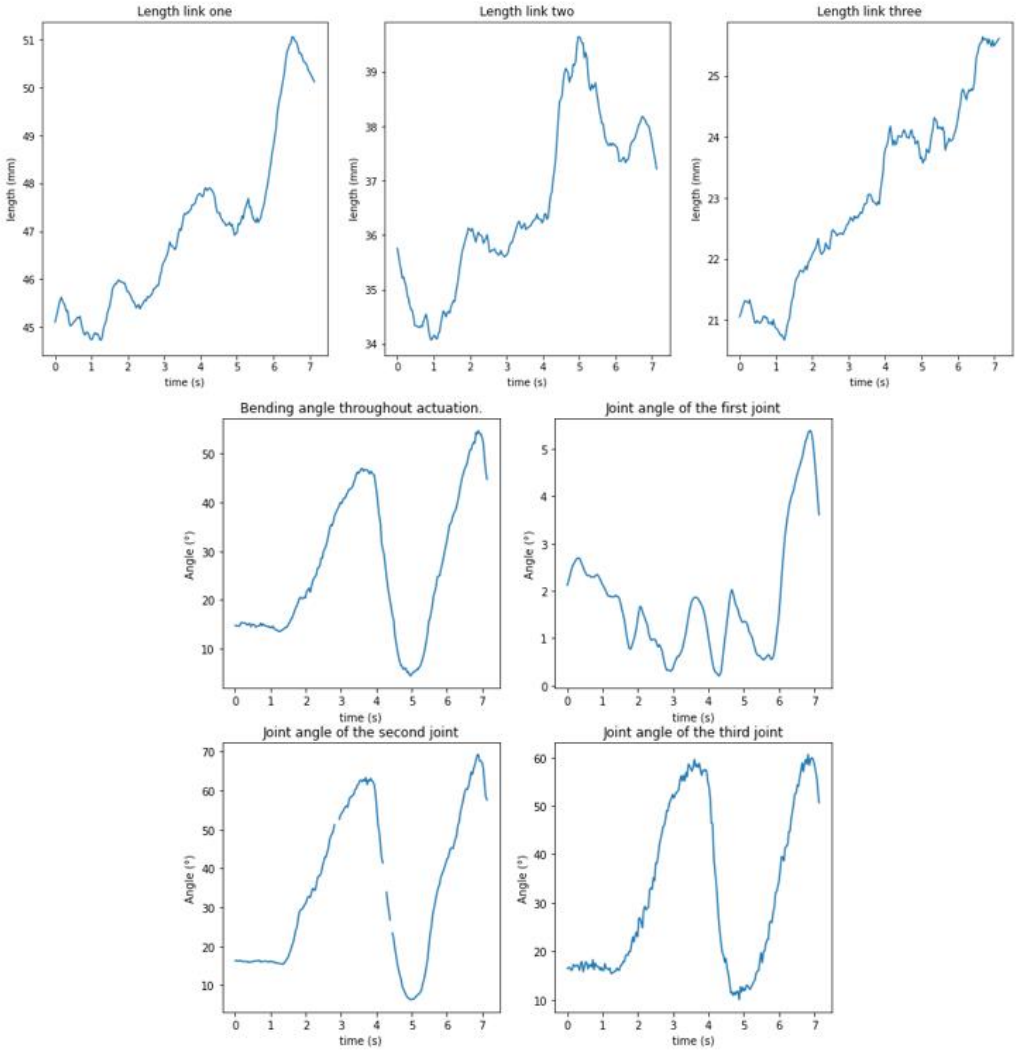


Figure 38 The link lengths, bending angle and joint angles of the finger throughout two grasping motions. The rate of the second and third joint are near constant while the first joint does not move.

3.2 SOFA simulations

3.2.1 a single material simulation

For the single material actuator, two simulations have been performed. The difference between the two is the amount of fixed points used for the tendon. In the first simulation, eight points were used while the second simulation used eleven points. Important to understand is that the cable is only fixed to the finger in these points. In case the distance between two points is too large, the cable will go outside the finger and the model gives unrealistic results. In that case, some extra points inside the finger need to be added as is done in the second simulation. To get a more correct model, the cable should be fixed in all the points of contact with the finger. However, this is not realistic and thus some error can be expected between the model and reality due to this simplification. To demonstrate this importance of selecting enough cable points, a simulation is done where not enough points were used resulting in the cable leaving the finger and one where the cable stays inside the finger (Figure 39).

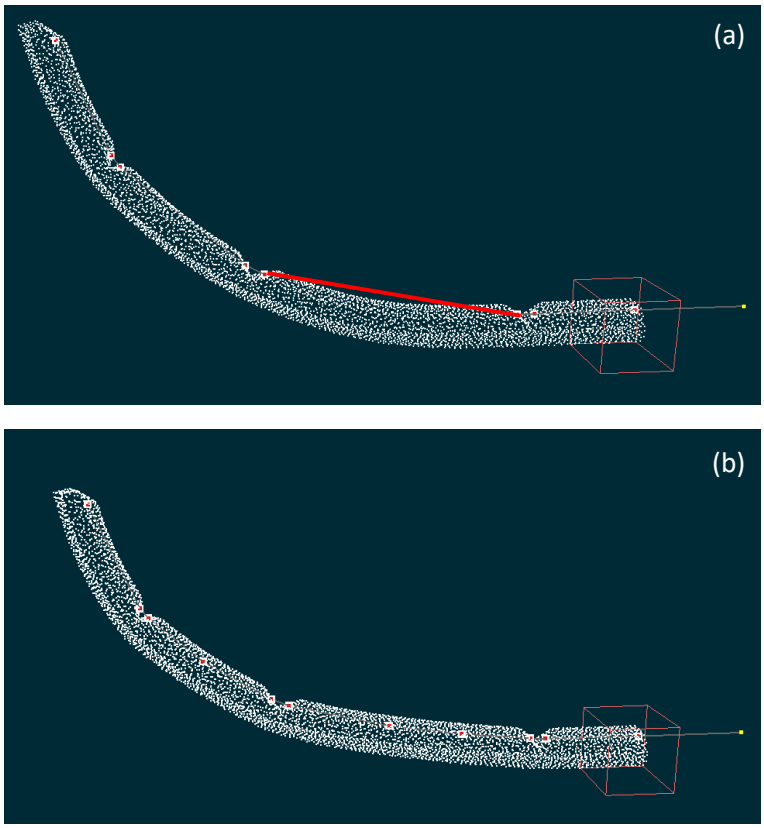
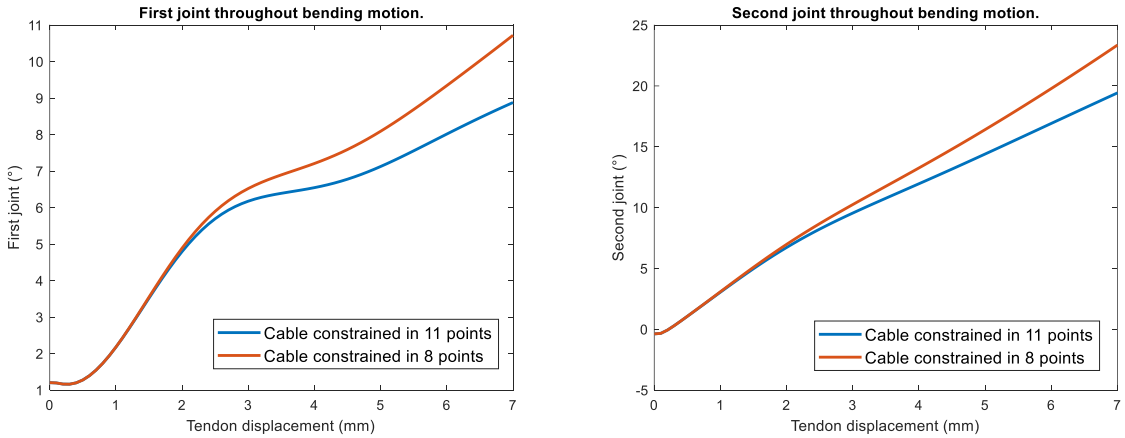


Figure 39 (a) Simulation containing 8 fixed points for the tendon where the tendon leaves the actuator (indicated in red) and (b) simulation containing 11 fixed points for the tendon such that the tendon does not leave the actuator

The results where the cable leaves the finger (i.e., with little points) is shown in Figure 39.a while the simulation where the cable remains in the finger is shown in Figure 39.b. In the first picture, the actuator will act like a beam fixed on one side and a load applied on the other side for the link at which the cable leaves the actuator. The link behaving as a beam is not unexpected as the actuator is very thin and, if the cable is not constrained enough on the link, the load can be assumed as applied in one point at the tip of the link. However, this simulation is unrealistic as the cable goes entirely through the finger. Increasing the amount of constraint points will make the simulation tend to a more realistic bending profile.



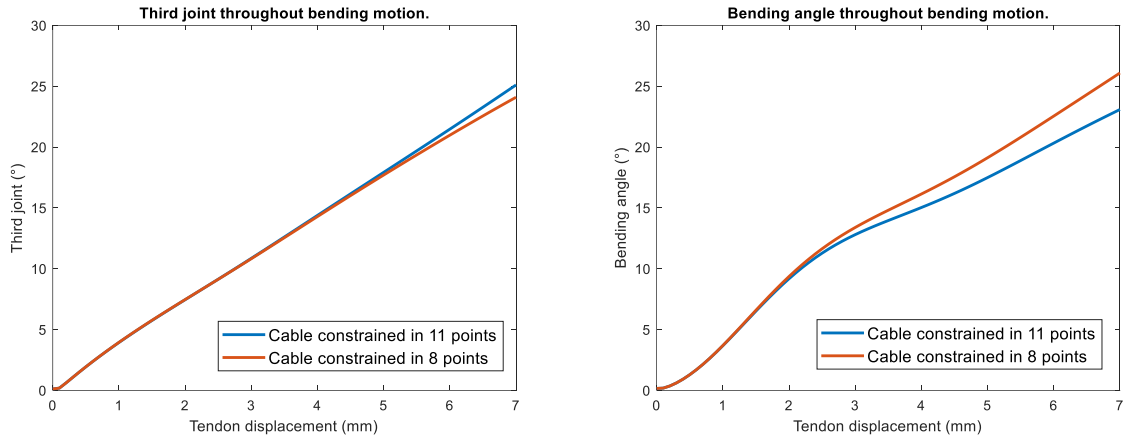


Figure 40 The evolution of the first joint, the second joint, the third joint and the bending angle. It can be observed that the simulation with more fixed points, gives a stiffer result.

Looking at the post-processed data from the simulation (Figure 40), it is clear that the way the cable is constrained has an effect on the overall results of the simulations. Adding more points of constraint increases the stiffness of the actuator. This is expected since added constraints restrict more the movement of the actuator. However, in the third joint, since the last link is already short, no extra point has been fixed with the cable. Due to this and the fact that the remainder of the actuator has a higher stiffness, the third joint actually has more movement compared to the under-constrained simulation. The effect of this constraint is mainly visible at higher deformations since then the assumption that the cable is only fixed in a selected amount of points has a bigger impact. Another thing to notice is that even though the material model is linear, the first and second joint do not evolve at constant rate. This means geometrical non-linearities are occurring. One possible non-linearity could be buckling since the actuator is very slender. These non-linearities increase the difficulty of the control of the actuator.

Moreover, the material is very soft thus there is no guarantee of the links maintaining their shape and length. A multi-material approach can be applied to resolve this issue whereby a rigid part in the links will fix the length, while the hinges can be manufactured from soft material. This way, the change of length in the hinges can be compensated by the strain of the soft material and the force output remains sufficiently high due to the rigid parts in the links. Furthermore, this offers a solution to the occurring non-linearities since these rigid parts will increase the overall stiffness of the structure.

3.2.2 A multi-material simulation

The multi-material simulation (Figure 41) shows that most motion takes place in the second joint while the least movement occurs from the third joint. This aligns with the higher maximum joint angle of the second joint compared to the third. A higher maximum joint angles means less material in the hinges thus resulting into softer joints. Even though the used material is equivalent for both joints, the second hinge is the softest and the third hinge the stiffest.

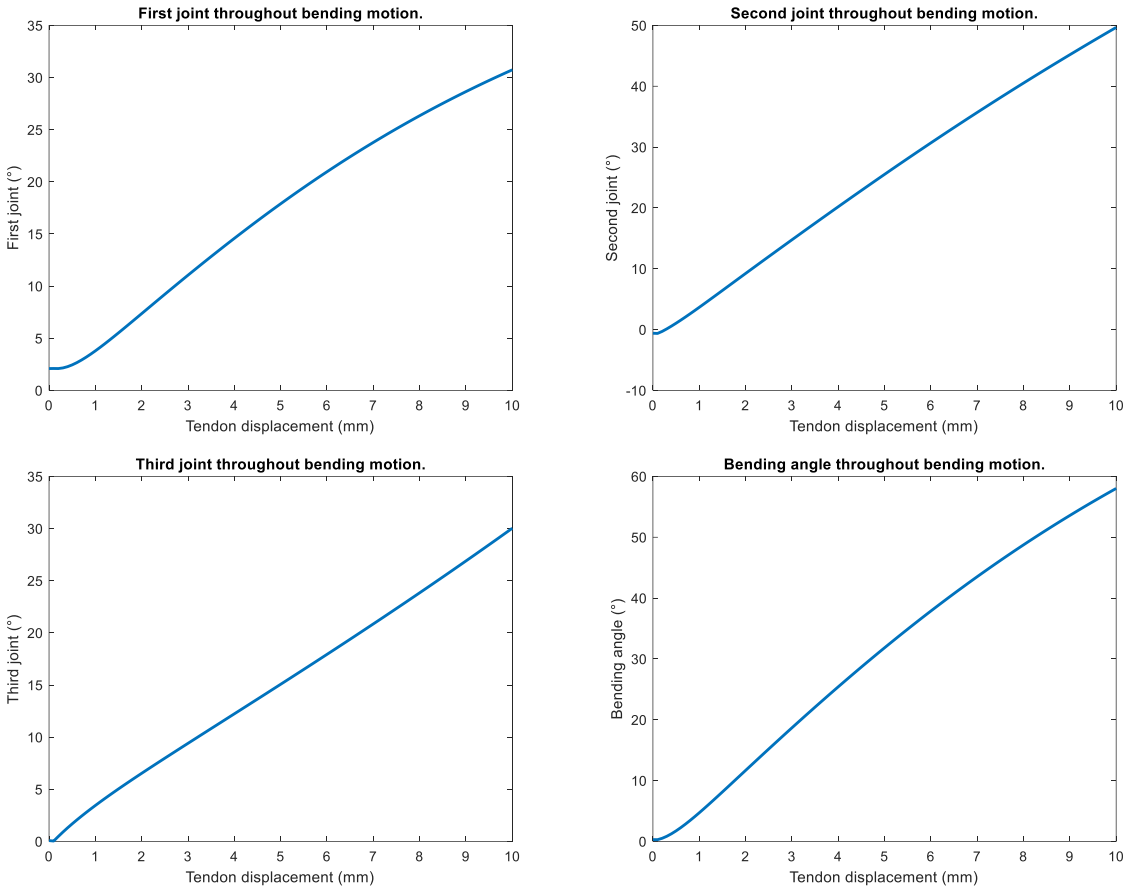


Figure 41 The evolution of the first joint, the second joint, the third joint and the bending angle for the multi-material outer actuator. Most movement is observed in the second joint while the least movement is seen in the third joint.

Additionally, a linear increase in cable displacement (constant pull velocity) results in a linear increase of the joint angles up to a bending angle of 58° at a tendon displacement of 10 mm. This is in agreement with the fact that the material has been modelled using a linear elastic model and means the geometric non-linearities no longer occur.

3.2.3 Inner actuator simulations

Looking at the simulations of the inner actuator made out of one self-healing material (Figure 42), similar results to the outer actuator can be observed. This is expected since the only thing that changed in the design are the link lengths. The length of the final link (fingertip) remains the same and thus it is found the third joint has the same behaviour as for the outer actuator. The other link lengths however, were shortened resulting in a lower joint angle.

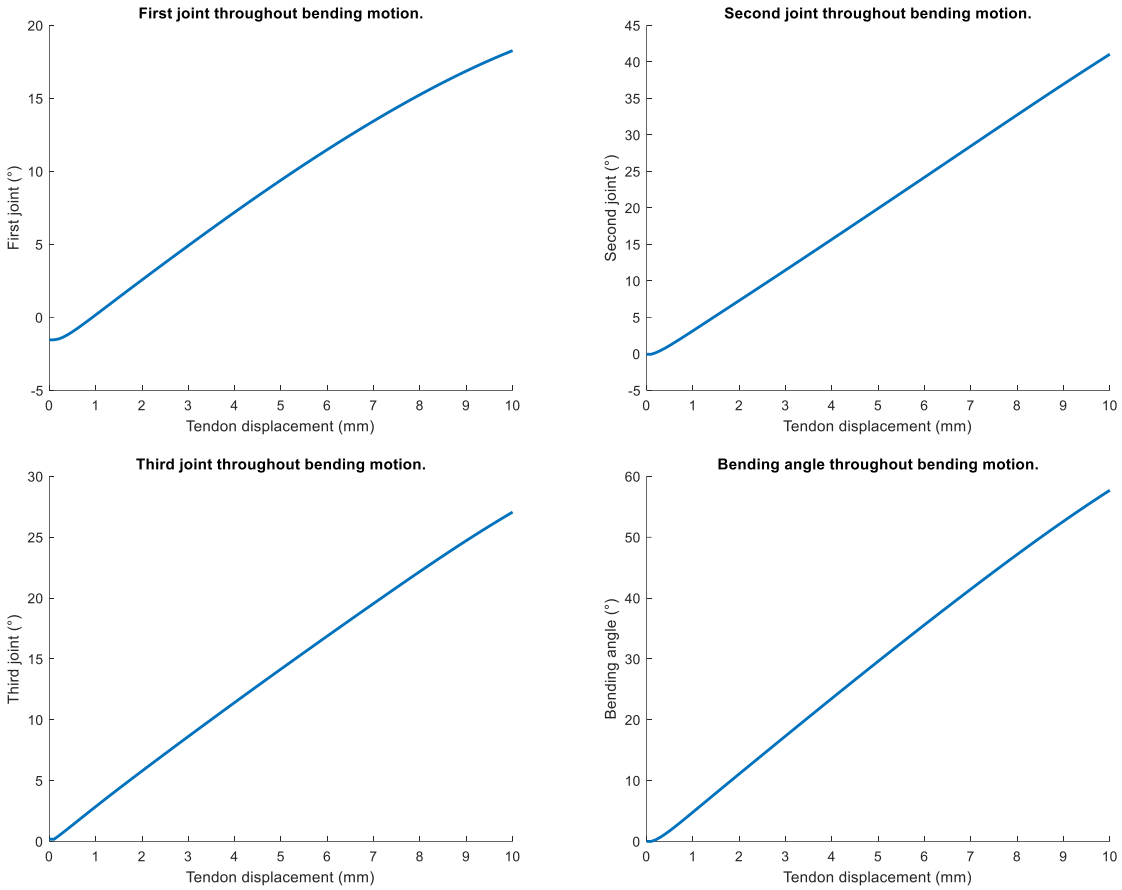


Figure 42 The evolution of the first joint, the second joint, the third joint and the bending angle for the multi-material inner actuator made out of one self-healing material. The third joint moves the least while the third joint moves most.

Looking at the simulations for the inner actuator made out of three self-healing materials (Figure 43), it can be seen that the first joint has little bending compared to the second and third joint and the third joint has a slightly higher joint angle than the second joint. Preferably, both would have the same joint angle which could be obtained by decreasing the stiffness of the material used for the second joint. However, other factors need to be considered including the gelation temperature of all the materials for fusing. Also, limited materials are available ready to make of which none have the required stiffness to obtain a perfect match of the joint angles.

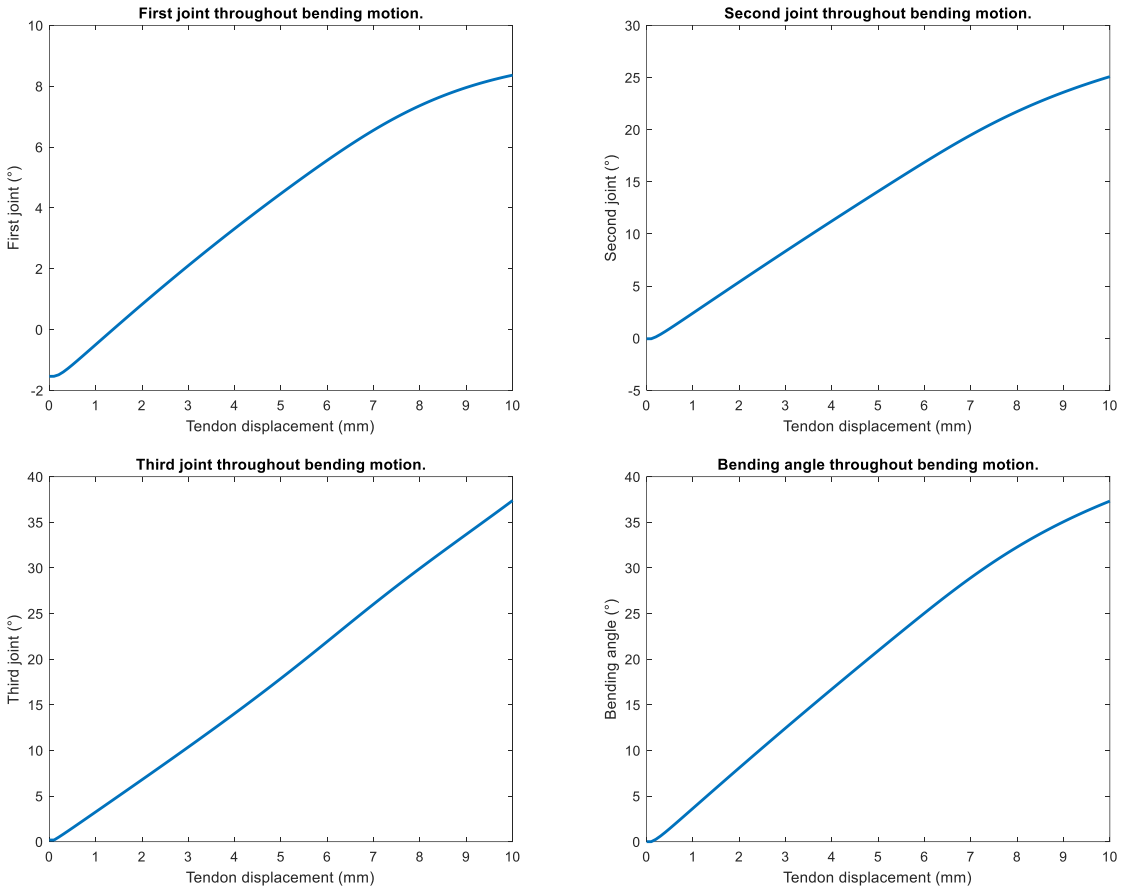


Figure 43 The evolution of the first joint, the second joint, the third joint and the bending angle for the multi-material outer actuator. The first joint bends very little while the third joint moves most.

Comparing the two actuators, it can be seen that since the joints are now stiffer, the overall bending of the finger is reduced compared to the previous design. As the gap between the second and third joint is smaller for the inner actuator made out of three materials, it is worth building this actuator and characterizing it.

3.3 Comparison of the manufacturing methods

The three methods described in section 2.5.1 **Different casting methods** each feature various advantages. While method one only requires a mould and no fusing, the material only needs to be produced once instead of twice for method two. Considering the steps required to make the material explained in section 2.2.1 **Material preparation**, one can understand that this is very beneficial. The third method ensures no cable is necessary during the casting for the creation of the hole for the tendon. As will be explained in the next paragraph, using this cable can induce extra complexity to the manufacturing process. In the remainder of this section, the three methods will be compared via the time consumption and complexity involved with the manufacturing of the actuator.

Table 7 Advantages and disadvantages of the casting methods used during the thesis. The second casting method gave the best results and was easiest to perform.

	Advantages	Disadvantages
One multi-stage cast (method one)	- Only one mould required. - No fusing needed.	- Weak interfaces between castings. - Cable required to ensure tendon hole. - Possible damage from removing cable. - Layer of the first casting is hard to get right thickness.
Two single-stage casts (method two)	- Only one material preparation required. - Most time efficient. - No weak interfaces possible.	- Quality fusing depends on placement parts - Cable required to ensure tendon hole.
Two multi-stage casts (method three)	- No cable required during casting.	- Least time efficient.

As mentioned, the first method is the only method that does not require fusing. However, from literature it is found that multi-stage casting can lead to weak interfaces. Another issue with this method includes the removal of the cable used during casting to create the hole for the tendon which can cause damage to the actuator. The additional healing increases both manufacturing time and complexity, but weaker interfaces originating from multi-stage casting are no longer a concern as the interface will fuse together as well.

For the second method, damage inflicted due to pulling out of the cable is not an issue as fusing is required anyway. The main concern here is that the top parts need to be placed manually on the remainder of the finger to fuse them together. The strength of this fusing depends on the surface of the interface which may vary due to poor placement of the top part but, if performed carefully this is a rather small issue. This was done multiple times throughout the thesis and overall a good adhesion was achieved.

The third method does not have issues coming from the cable since no cable is required during casting. However, the same issue as mentioned in the second method can occur. Also, the fact that this method is more complex and time consuming than the others can certainly not be neglected.

The first and second method both require an equal amount of manufacturing steps. Differentiation occurs when the active time, time where the human needs to execute actions, is taken into account. As previously mentioned, the second method requires the material to be prepared only once resulting in a reduced active time compared to the first method. On the other hand, fusing is not applied in the first method but, as this is performed passively via an oven, the amount of avoided active time is rather limited. Based on time consumption, the second method scores best.

The first and second method show a higher complexity compared to the third due to the cable movement as both the insertion and removal request time and care to avoid breaking of the mould and actuator. The cable is inserted by poking a hole through the mould at the hinges which is facilitated by the fact that the mould is made of silicone. Removal of the cable involves cutting it into smaller fragments to remove the actuator from the mould as the cable can be easier pulled out of silicone than the self-healing material, followed by individual removal of each cable part from the actuator to inflict as little damage to the actuator as possible. The (dis)advantages of the methods are summarized in Table 7.

3.4 Validation simulation

3.4.1 Validation outer actuator

Looking at the experimental data obtained from the outer actuator (Figure 44), during the first 0.5 seconds no bending took place and after 1.5 seconds the actuator remains approximately at rest. So, the data best used to compare the results lies in between these two time stamps. The rates of change in the joint angles are close to constant and are 33.3 °/s, 28,42 °/s and 20 °/s respectively. The rate for the first and second joint are thus similar while the rate of the third joint is lower. The rate of the bending angle is 51.63 °/s.

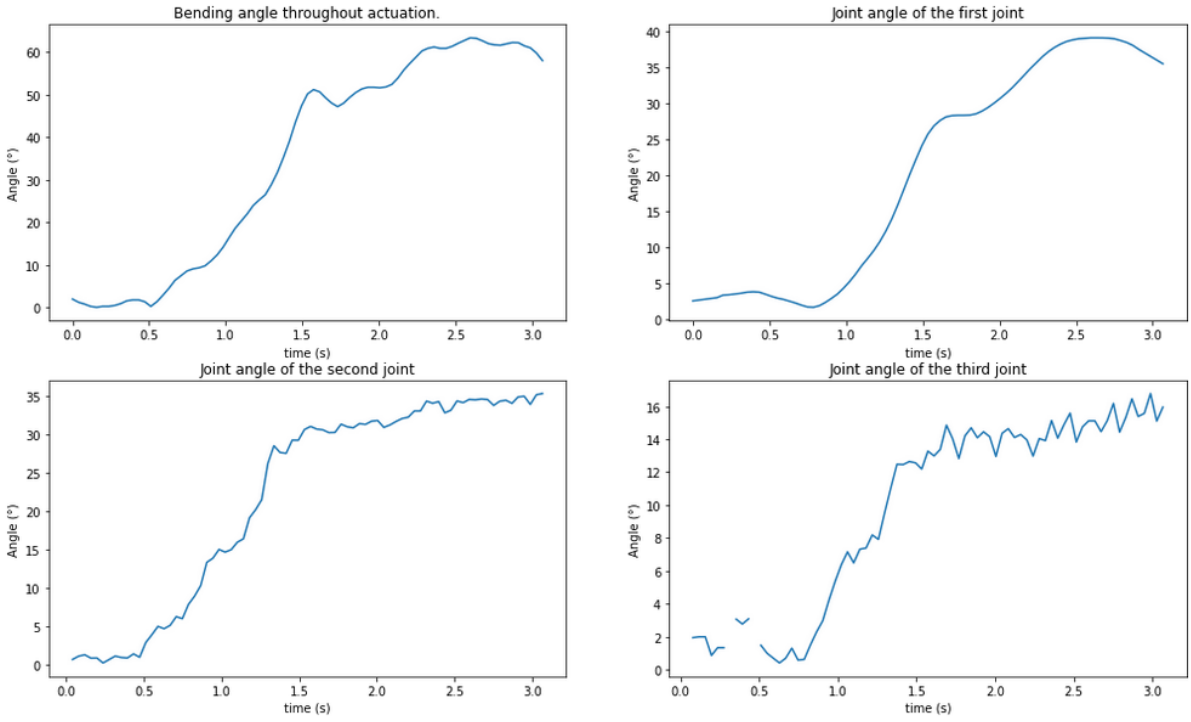


Figure 44 The post-processed results of the outer actuator that can be compared to the simulations for validation.

From Table 8, it can be observed that the error on any joint does not exceed 30% and for the bending angle, except for the first value, the maximum error is 11%. As the actuator is assumed to be held perfectly vertical, which in reality is not the case, the offset of the actuator causes the smaller angles to display a significant error on the experimental data (mainly the first joint). On the other hand, the assumption of a linear elastic material increases the error of the larger angles as the deformation increases. Other error factors include imperfect manufacturing of the actuator introducing discrepancies between the actual and simulated

actuator, the experimental setup involves cable pulling that deviates from the assumed perfectly straight motion in the simulations and slight variation in material properties resulting from the preparation process. Furthermore, fixing the tendon in a specified amount of points in the simulation will also contribute to errors. As already found in the comparison of simulations with a different amount of fixed points, the more points, the stiffer the actuator acts. Even though Table 8 cannot be applied for validation purposes, it provides insight on the errors of the joint angles and bending angle.

Table 8 Comparison of the experimental results with the simulations for the outer actuator at fixed second joint angles.

α_{tot}		Bending angle (°)	First joint (°)	Second joint (°)	Third joint (°)
10	Experimental	6,05	1,32	4,18	5,19
	Simulation	8,32	3,79	3,66	3,45
	Relative error	0,38	1,87	0,12	0,34
20	Experimental	10,96	4,89	9,02	5,83
	Simulation	9,75	6,25	7,55	5,63
	Relative error	0,11	0,28	0,16	0,03
30	Experimental	17,98	10,34	13,09	8,19
	Simulation	15,98	9,96	13,1	8,55
	Relative error	0,11	0,04	0,00	0,04
40	Experimental	24,09	11,4	19,29	11,96
	Simulation	23,94	13,19	18,03	11,11
	Relative error	0,01	0,16	0,07	0,07
50	Experimental	28,66	15,32	23,12	12,27
	Simulation	28,94	15,6	21,8	13,08
	Relative error	0,01	0,02	0,06	0,07
60	Experimental	33,68	23,08	23,99	12,8
	Simulation	34,99	18,22	26,04	15,35
	Relative error	0,04	0,21	0,09	0,20

As explained in section 2.6.1 **Comparison experimental results and simulations**, a different way of calculating the error can be used. In literature, the RMS error (RMSE) on the position is determined and if the error is smaller than 5%, the simulation can be considered valid. The RMS error of fixed points on the actuator can thus help validate the simulation.

Table 9 shows for all sums of joint angles, except 10 °, errors below the 5% threshold value thus it can be concluded that the simulation is valid and does not significantly differ from experimental data.

Table 9 The RMS error of the outer actuator, proving the simulation is valid.

α_{tot} (°)	RMSE (%)
10	5.58
20	3.92
30	3.85
40	4.59
50	2.87
60	2.92

3.4.2: Validation inner actuator – one self-healing material

In order to compare the simulation with experimental results, the bending profile of the inner actuator, consisting of a single self-healing material, is obtained in (Figure 45). The initial 0.5 seconds of the measurements can be neglected as there is no actual bending occurring during this period. During the first 0.5 seconds of the bending, the first joint remains inactive. Subsequently, all three angles are actuated at approximately constant rates, although they differ from each other. The rate of change in bending angle is 25.75 °/s, while the first, third, and second joints exhibit rates of 13.57 °/s, 11.57 °/s, and 14.15 °/s, respectively.

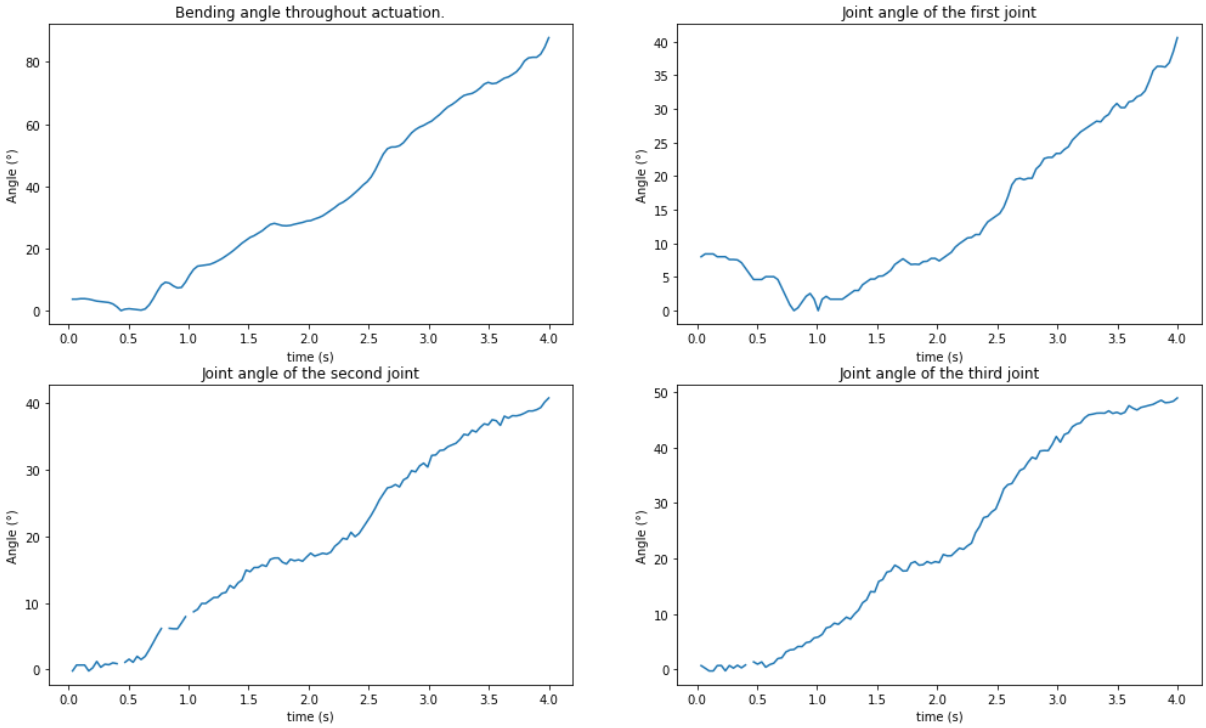


Figure 45 The post-processed results of the inner actuator made out of one self-healing material that can be compared to the simulations for validation.

Comparing the experimental data and the simulation, the results can be found in Table 10. Just as done for the outer actuator, the total joint angle will be used as reference and the error of the joints as well as the bending angle can be used to compare the results. It can be observed that in general the error tends to increase for higher joint angles for the second and third joint as is expected. The error on the first joint is large for small angles due to the small offset coming from the experiments. The error on the second joint is slightly higher compared to the

others for a total joint angle of 20° and 30°. Once again, it can be observed that the error on the bending angle remains below 10% except for one outlier.

Table 10 Comparison of the experimental results with the simulations for the inner actuator made out of one self-healing material at fixed second joint angles.

α_{tot}		Bending angle (°)	First joint (°)	Second joint (°)	Third joint (°)
10	Experimental	6,31	2,11	4,17	3,15
	Simulation	6,08	2,15	4	3,46
	Relative error	0,04	0,02	0,04	0,10
20	Experimental	15,45	2,12	11,95	6,1
	Simulation	12,35	4,78	8,56	6,65
	Relative error	0,20	1,25	0,28	0,09
30	Experimental	20,69	4,27	16,84	8,01
	Simulation	19,81	7,11	11,84	9,1
	Relative error	0,04	0,67	0,30	0,14
40	Experimental	26,78	6,86	19,92	13,1
	Simulation	25,97	9,6	17,39	12,52
	Relative error	0,03	0,40	0,13	0,04
50	Experimental	33,28	10,38	22,52	17,23
	Simulation	32,63	12,17	22,49	15,8
	Relative error	0,02	0,17	0,00	0,08
60	Experimental	39,13	12,77	25,23	22,48
	Simulation	39,79	14,38	22,48	18,76
	Relative error	0,02	0,13	0,11	0,17
70	Experimental	45,12	15,41	28,24	27,87
	Simulation	47,18	16,75	32,75	22,19
	Relative error	0,05	0,09	0,16	0,20
80	Experimental	53,13	19,95	31,02	30,03
	Simulation	52,09	18,53	37,37	24,96
	Relative error	0,02	0,07	0,20	0,17
90	Experimental	57,33	22,45	33,19	34,31
	Simulation	59,03	20,22	42,07	27,32
	Relative error	0,03	0,10	0,27	0,20

The RMS errors can be found in Table 11. The error increases with increasing bending of the actuator, probably as a consequence of the selected material law which features an increasing error for a higher deformation. In order to verify the simulation, it should be redone with a Neo-Hookean hyperelastic model which cannot be accomplished during this thesis due to the error mentioned in section 2.4.2 **A first model**.

Table 11 The RMS error in function of the sum of the joint angles. Overall, the error increases with an increase of bending. This leads to the hypothesis that the main contribution to the error comes from the material law used.

α_{tot} (°)	RMSE (%)
10	2.19
20	2.4
30	4.2
40	5.78
50	6.34
60	7.99
70	9.46
80	11.46
90	11.62

3.4.3: Validation inner actuator – three self-healing materials

Examining the experimental data obtained for the actuator constructed of three distinct self-healing materials (Figure 46), two bending cycles were conducted. During the intervals between bending and unbending, the actuator remained at rest. While different behaviour is observed for the first joint, the remaining joints closely resemble the results displayed in Figure 45.

The second peak observed in the first joint should be disregarded as it does not correspond to the actuator's bending, but originates from the actuator not being perfectly vertical between cycles. The rates of change in angle for the joints are respectively 10°/s, 32.69 °/s, and 28.57 °/s. The first joint exhibits a lower rate compared to the other two joints, which display a similar rate, according to the expected behaviour. The rate of change in the bending angle is approximately 34 °/s.

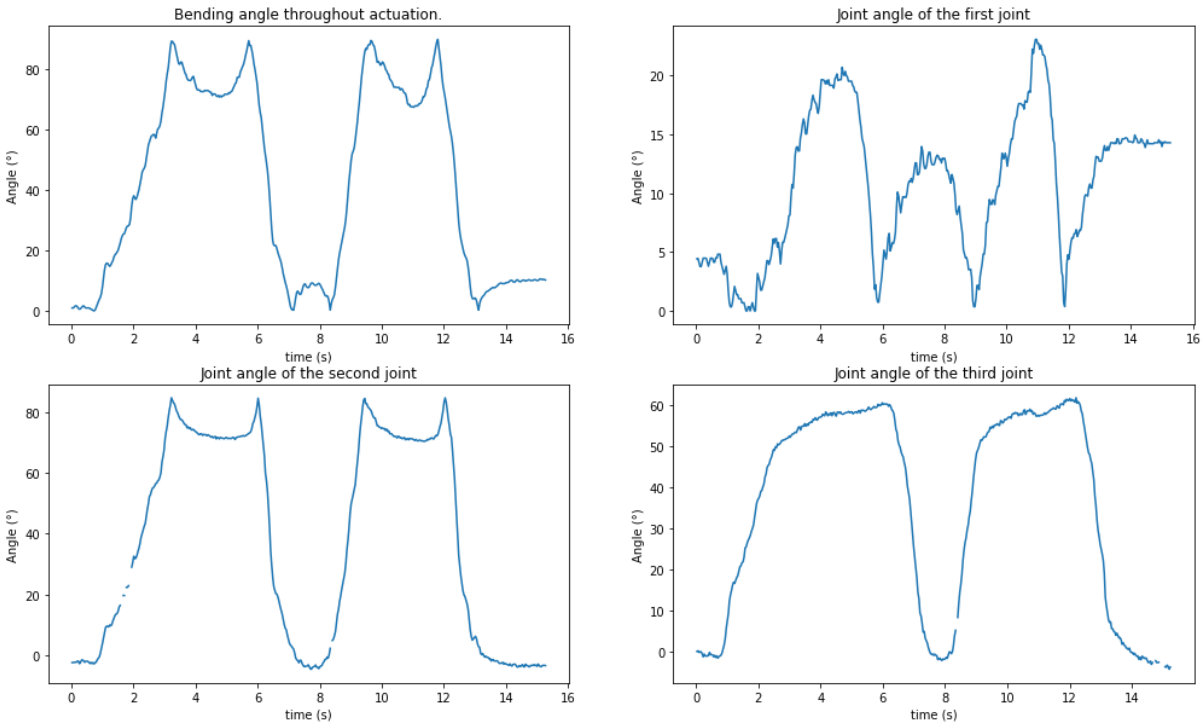


Figure 46 The post-processed results of the inner actuator made out of three self-healing materials that can be compared to the simulations for validation

Finally, looking at Table 12, similar results as the two previous actuators can be found. A big error arises on the first joint which can be explained due to the material model used. Due to the fact that there is more deformation in the second and third joint, the stiffness of these joints will decrease, resulting in even more deformation and less deformation in the first joint. However, for a linear model this stiffness decrease for an increased deformation does not exist, leading to a mismatch between simulation and experiments. Due to this mismatch, the error on the other angles will also be higher due to the fact that the sum of the angles is taken equal for simulation and experiment. The error on the bending angle, however, remains under 10%.

Table 12 Comparison of the experimental results with the simulations for the inner actuator made out of three self-healing materials at fixed second joint angles.

α_{tot}		Bending angle (°)	First joint (°)	Second joint (°)	Third joint (°)
10	Experimental	4,51	3,45	2,44	3,09
	Simulation	4,1	1,73	2,71	3,61
	Relative error	0,09	0,50	0,11	0,17
20	Experimental	10,8	1,73	7,53	9,5
	Simulation	9,85	1,53	6,59	8,22
	Relative error	0,09	0,12	0,12	0,13
30	Experimental	15,2	1,38	11,26	16,9
	Simulation	15,46	2,96	10,38	12,94
	Relative error	0,02	1,14	0,08	0,23
40	Experimental	20,05	0,69	16,38	21,62
	Simulation	20,51	4,36	13,81	17,5
	Relative error	0,02	5,32	0,16	0,19
50	Experimental	25,45	0,35	21,59	27,15
	Simulation	25,86	6,18	18,48	24,41
	Relative error	0,02	16,66	0,14	0,10
60	Experimental	30,94	0,35	29,12	34,37
	Simulation	33,18	7,86	23,25	32,94
	Relative error	0,07	21,46	0,20	0,04
70	Experimental	34,72	1,77	30,96	36,13
	Simulation	35,55	8,25	24,67	36,26
	Relative error	0,02	3,66	0,20	0,00

To finalize the validation, the RMS error is also determined for this actuator. Looking at the results shown in Table 13, a similar conclusion can be drawn as for the actuator made out of one self-healing material. The error appears to be slightly higher and increases with an increasing bending angle. Once again, to verify the simulation, it should be reran using the Neo-Hookean hyperelastic model.

Table 13 The RMS error in function of the sum of the joint angles. Overall, the error increases with an increase of bending. This leads to the hypothesis that the main contribution to the error comes from the material law used.

α_{tot} (°)	RMSE (%)
10	3.33
20	6.55
30	5.01
40	7.24
50	6.15
60	8.35
70	11.35

3.5: Characterization of the actuator

3.5.1: Characterization outer actuator

Using the set-up mentioned in section 2.6.2 **Force measurements** for the outer actuator, the results expressed in Figure 47 are achieved. As expected, changing the boundary conditions influences the evolution of the joint angles throughout bending. It can be observed that first the third joint is actuated after which shortly also the second joint is actuated. The rate of the third joint angle is highest at the beginning of the bending and decreases afterwards, while for the second joint the rate remains constant until the joint is no longer actuated. At this point, the third joint also nears a rate of zero while the first joint starts being actuated indicating that movement solely is provided by the first joint with a constant rate. This evolution is also observed in the bending angle. First, there is only movement in the third joint resulting in a low slope for the bending angle. Once the second joint joins the movement, the bending angle raises in a constant rate that is higher than the original one. Finally, once the first joint starts actuating, the bending angle raises with an even higher but constant rate. To summarize, first the second and third joint will be actuated up to a certain angle after which the movement solely comes from the first joint. This is not particularly a bad behaviour for the intended function of the actuator since for grasping usually the second and third joint move simultaneously while the first joint moves at a different instance. The fact that the rate of change in the third joint angle is not constant, differs from what can be expected from a natural grasping motion. However, since this is the outer actuator, this small difference can be accepted. The force required from the actuator and thus the discomfort from this small difference in behaviour, is rather small since this actuator is only used to move the finger back from a bent position to a straight one.

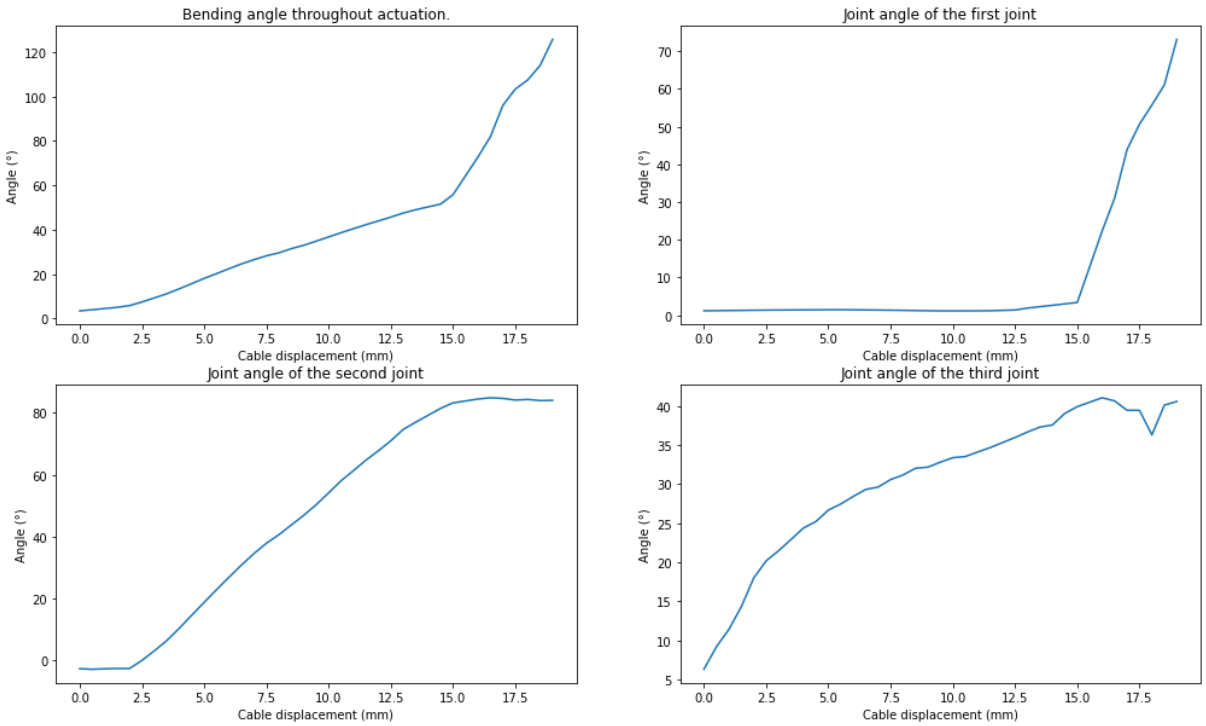


Figure 47 The evolution of (a) the bending profile, (b) the first joint, (c) the second joint and (d) the third joint for the outer actuator.

The set-up explained in section 2.6.2 **Force measurements** can also be enriched with a force sensor such that the pull force on the cable can be recorded throughout bending of the actuator. The results can be found in Figure 48. Since a rather soft and flexible material is used, the pull force does not exceed more than 1N. It can also be observed that the pull force, just like the bending profile, can be divided into three parts: first the third joint moves at the highest rate. Secondly, actuation of the second joint occurs and a slower rate is observed. Finally, the first joint is in movement and a decrease in force is found.

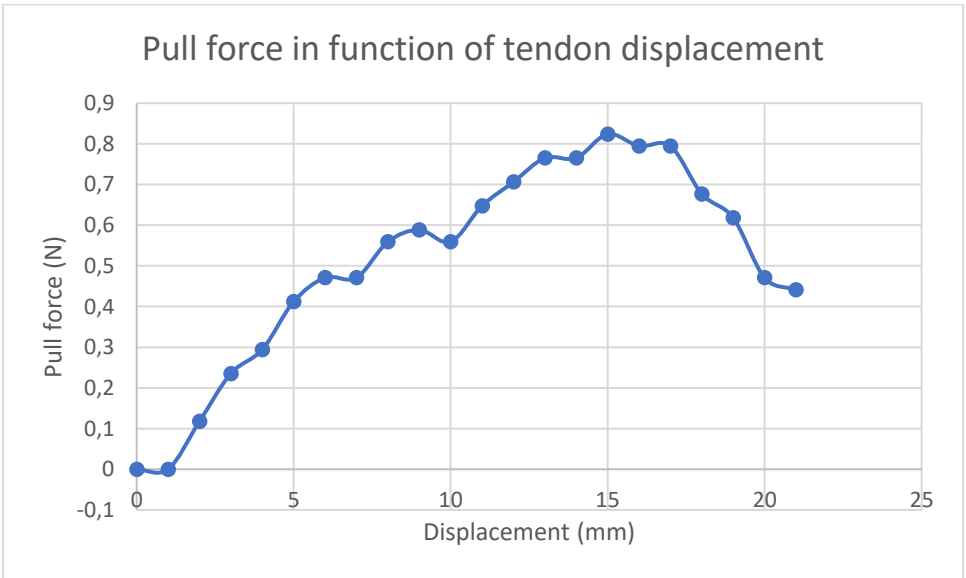


Figure 48 The pull force on the tendon given in function of the tendon displacement for the outer actuator.

3.5.2: Characterization inner actuator – one self-healing material

Just like for the outer actuator, the bending profile of the inner actuator built out of one self-healing material can be divided into 3 parts as visible in Figure 49. Movement starts in the third joint up to one mm after which the second joint joins the movement. After a tendon displacement of 6 mm all joints are actuated. The rate of the bending angle also changes throughout the three parts. Overall, the shape of the joint angles in function of displacement are very similar to the ones obtained for the outer actuator. The main difference are the rates in which the joint angle changes. The rates of the first and second joints are lower while the rate of the third joint increased. At the moment the first joint is actuated, there is still movement in the second joint explaining the lower rate of the first joint. The order in which the joints can be actuated can be explained by looking at the inertia that has to be overwon before the joint can be actuated. As the link lengths increase, the inertia increases as well. This way, the inertia of the first joint is a combination of three links while the third joint only relies on the inertia of the last link, illustrating the specific order in which the joints start moving. As only this last link has an impact, the least force is required to move the third joint so it will actuate first followed by the second and finally the first joint.

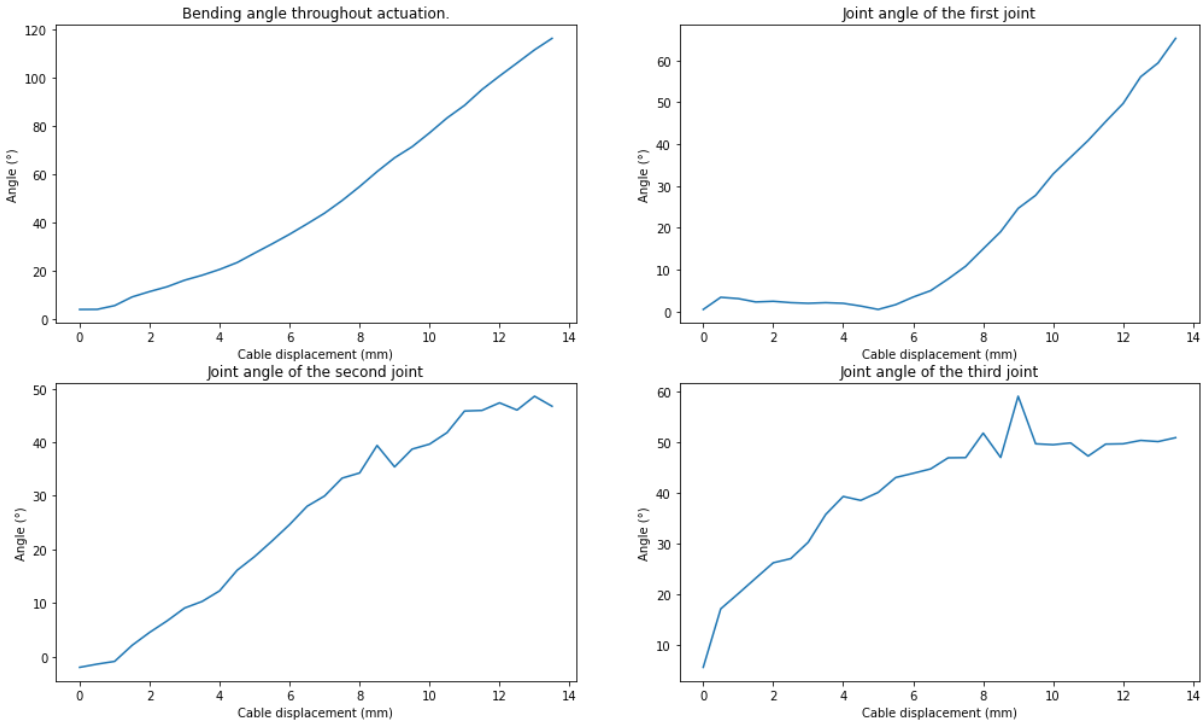


Figure 49 The evolution of (a) the bending profile, (b) the first joint, (c) the second joint and (d) the third joint for the inner actuator made out of one self-healing material.

Looking at the force required to pull the tendon (Figure 50), once again the three parts can be distinguished. However, instead of a decrease in force, the force remains constant for the final part of the curve. This can be explained the same way as mentioned in the previous paragraph: once the inertia is overwon, the joint will behave as a hinge and thus the required force to further move the joint does not increase any further.

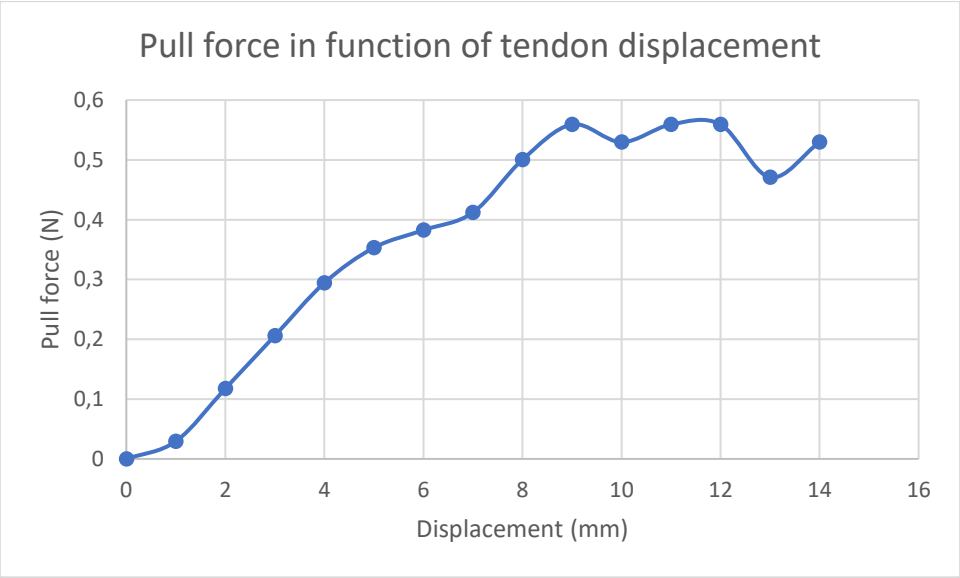


Figure 50 The pull force on the tendon given in function of the tendon displacement for the inner actuator made out of one self-healing material.

3.5.3: Characterization inner actuator – three self-healing materials

Looking at the bending profile of this actuator constrained at one side, the bending angle, second and third joint have a near constant rate (Figure 51). The first joint angle remains rather low which results in a an unmeaningful profile roughly Indicating no movement of the first joint. The rates of the second and third joint are close to each other which is desired to mimic the grasping of an object. During grasping, the actuator will first be actuating the second and third joint until it encloses the object after which the first joint will move to complete the grasping motion. The main difference between this actuator and the one built out of one material is that the profile of the third joint has a more constant rate. The fact that the second and third joint start moving from the beginning of the motion is a benefit. Overall, it can be concluded that this actuator made out of three self-healing materials imitates best the grasping motion of a finger.

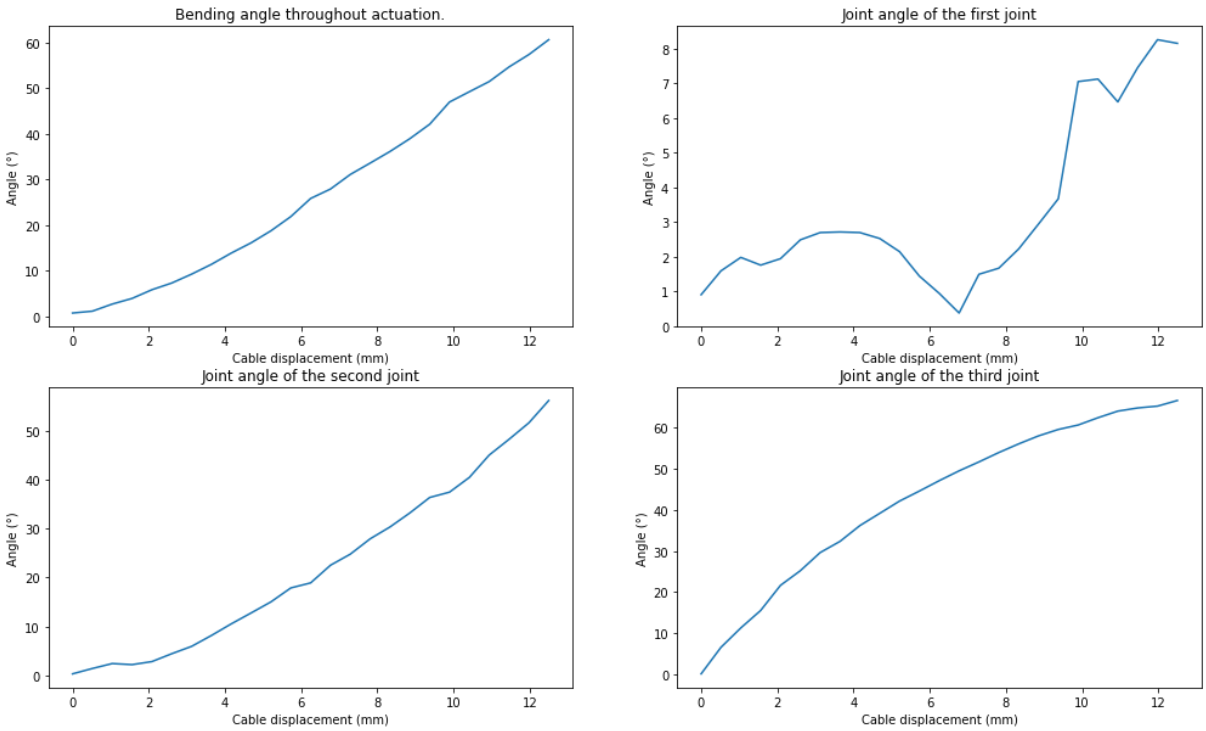


Figure 51 The evolution of (a) the bending profile, (b) the first joint, (c) the second joint and (d) the third joint for the inner actuator made out of three self-healing materials.

The fact that this actuator shows a more constant rate, can also be observed in the curve of the pull force of the tendon (Figure 52). The three different parts as described in the sections above for the other actuators can no longer be set apart from one another. The maximum force required remains equal to that of the actuator built out of one self-healing material.

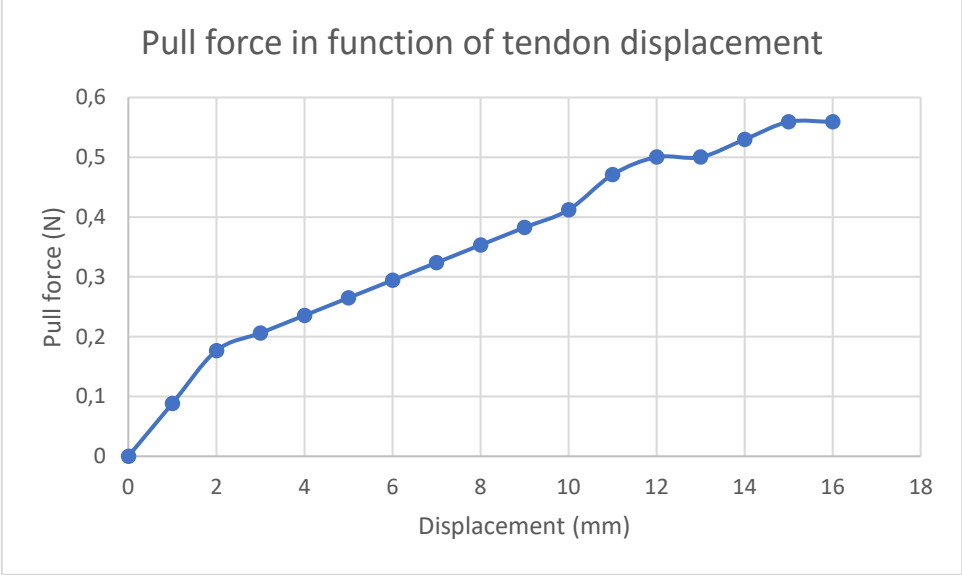


Figure 52 The pull force on the tendon given in function of the tendon displacement for the inner actuator made out of three self-healing materials.

To demonstrate the similarity between the actuator and the human finger, pictures of different bending poses of the finger can be seen in Figure 53. The actuator gives a very similar profile as the finger.

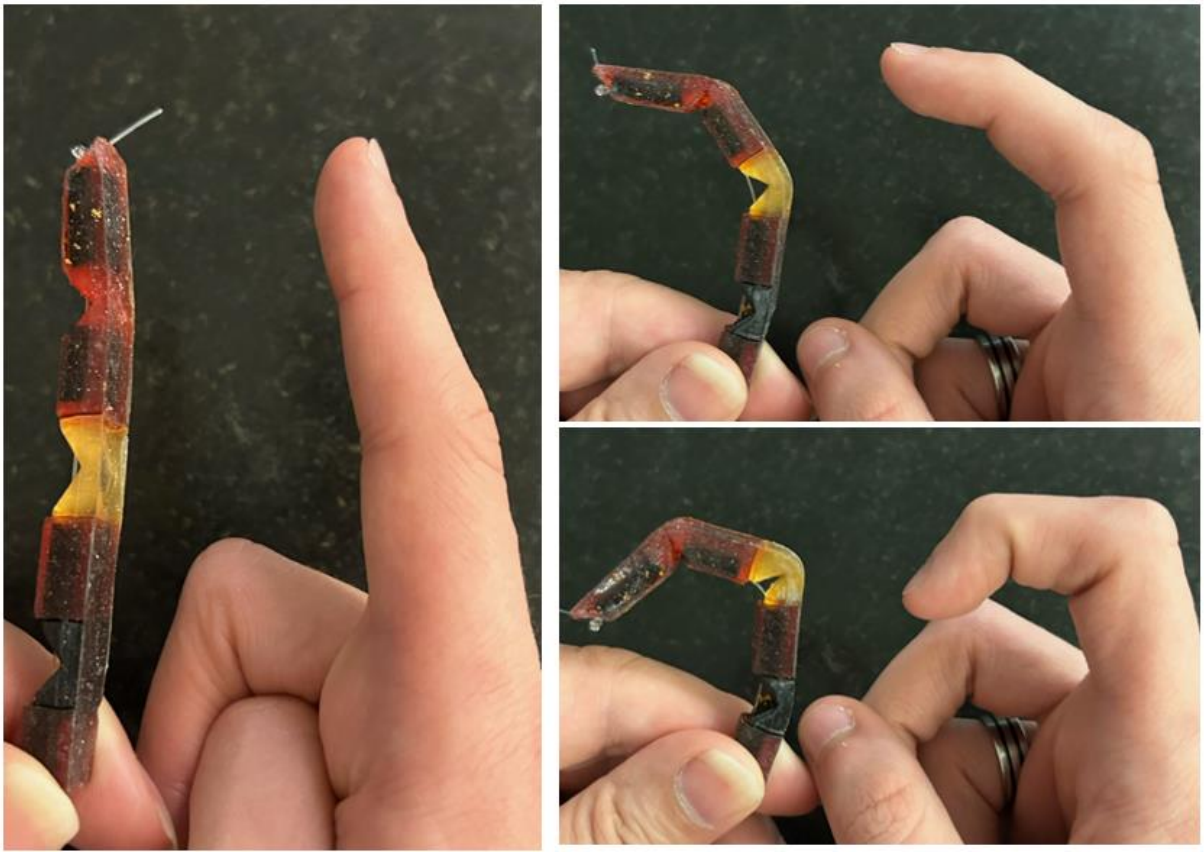


Figure 53 Comparison of a human hand and the final hand for different bending poses.

Chapter 4: Conclusion and future work

4.1: Conclusion

During this thesis, both an outer and inner tendon-driven actuator were developed for a soft exoskeleton hand. The actuator was constructed out of self-healing material allowing more freedom in the design of the actuator.

First, data was obtained to determine the grasping profile of a finger. Using this data, a design of an actuator was made and simulated. The simple design featuring one material was upgraded to a multi-material approach where the links contained rigid parts to secure the length and shape of the links and increase the output force. Based on this multi-material design, an outer- and inner actuator were manufactured. For this, multiple casting methods were discussed, tested and compared resulting in the selection of a method including fusing multiple single-stage casts using the properties of the material. Furthermore, the inner actuator was improved by providing each joint a different stiffness thus optimizing the bending profile of this actuator.

The bending profiles of the manufactured actuators were experimentally determined and compared to simulation results. Even though multiple error sources are introduced by both the experimental setup and the simulations, which in turn are reinforced by the selected material model and assumptions, a maximum error of 11% on the bending angle was observed. To verify the validity of the applied simulations, the RMS error was determined. The simulation of the outer actuator can be deemed valid, while those of the two inner actuators displayed an excessive error underscoring the need for a Neo-Hookean hyperelastic material model to ensure their validity.

Furthermore, the actuators were characterized via their bending profile when constrained at the bottom, simulating the scenario where the actuator is attached to a finger and allowing the pull force on the tendon to be obtained. These bending profiles were compared to those of the grasping motion of a finger whereby the final actuator composed of three self-healing materials demonstrated superior performance, closely resembling the results of the reference data.

Further improvement of the design for the outer actuator was considered unnecessary due to its lower required force output. The force requirement solely pertains to returning the finger from a bent position to a straight state and is lower than the force needed for gripping an object, hence making the design of the actuator less crucial.

4.2: Future work

This study primarily focussed on designing an actuator with a bending profile closely resembling that of a finger during grasping. Advantages include the straightforward implementation of sensors. Although the development of sensors was not explored in this thesis, incorporating sensor feedback on a hand exoskeleton can offer significant benefits by providing data on exoskeleton movement. One potential sensor that could be implemented is a strain sensor, which enables the determination of the exoskeleton's bending angle. This information could in turn be utilized to estimate the size of the held object and measure the

bending profile during grasping. Additionally, the inclusion of a pressure sensor could measure the grasping force of the glove, ensuring a secure grip on objects while minimizing the risk of damage to the object. Another possibility is the integration of a pain sensor, capable of detecting potential damage to the actuator and if possible identifying any injuries on the finger. By utilizing a tree branch structure for the sensor, it could not only indicate whether the actuator is damaged but also provide insights into the specific location of the damage.

Applying the actuator in real life requires further research to ensure its secure attachment to the finger. Even though misalignment concerns are less significant in soft robotics, it remains essential to affix the actuator in a manner that prevents any relative movement between the actuator and the finger, playing a vital role in providing good wearer comfort and optimizing the force output efficiency of the actuator. Once this attachment process is successfully accomplished, all five fingers can be manufactured, enabling the determination of the gripping force of the exoskeleton hand.

The control aspect of a soft exoskeleton hand holds significant importance. A viable approach to achieve control is by acquiring a thorough understanding of tendon displacement in function of the bending angle. By combining this knowledge with a force sensor capable of detecting grasping and ensuring a secure grip, effective control of the exoskeleton hand can be achieved. Alternatively, strain sensors could be applied instead of tendon displacement to gather the necessary data. However, these sensors often exhibit considerable drift and relaxation over time, possibly demanding the implementation of machine learning algorithms to address these effects.

Further improvements include the recommendation to delve into more comprehensive investigations on specific elements discussed in this thesis. For example, in order to compare casting methods, conducting experiments to explore the strength of the actuator under different load scenarios would provide valuable information. In addition, once the bug in the SOFA software, related to the interaction between the SparseLDL solver and hyperelastic materials, is resolved, it is advised to rerun the simulations using a more accurate material model. As previously mentioned, the self-healing material can be most accurately modelled as a Neo-Hookean hyperelastic material. To obtain the necessary parameters for this model, the experimental data obtained from a tensile test can be inputted into Abaqus. Moreover, developing a more professional setup to measure the bending profile of the actuator would contribute to improving the quality and reliability of the validation.

Chapter 5: Sources

[1] 'Soft Robotics: Academic Insights and Perspectives Through Bibliometric Analysis | Soft Robotics'. https://www-liebertpub-com.myezproxy.vub.ac.be/doi/10.1089/soro.2017.0135 (accessed Nov. 16, 2022).

[2] T. Shahid, D. Gouwanda, S. G. Nurzaman, and A. A. Gopalai, 'Moving toward Soft Robotics: A Decade Review of the Design of Hand Exoskeletons', *Biomimetics*, vol. 3, no. 3, Art. no. 3, Sep. 2018, doi: 10.3390/biomimetics3030017.

[3] R. A. Bilodeau and R. K. Kramer, 'Self-Healing and Damage Resilience for Soft Robotics: A Review', *Front. Robot. AI*, vol. 4, 2017, Accessed: Nov. 16, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2017.00048

[4] S. Kumar, C. Savur, and F. Sahin, 'Survey of Human–Robot Collaboration in Industrial Settings: Awareness, Intelligence, and Compliance', *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 1, pp. 280–297, Jan. 2021, doi: 10.1109/TSMC.2020.3041231.

[5] 'Actuators for Soft Robotics | SpringerLink'. https://link.springer.com/chapter/10.1007/978-3-319-32552-1_21 (accessed Nov. 17, 2022).

[6] C.-H. Liu, C.-H. Chiu, T.-L. Chen, T.-Y. Pai, Y. Chen, and M.-C. Hsu, 'A soft robotic gripper module with 3d printed compliant fingers for grasping fruits', presented at the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM, 2018, pp. 736–741. doi: 10.1109/AIM.2018.8452420.

[7] N. R. Sinatra, C. B. Teeple, D. M. Vogt, K. K. Parker, D. F. Gruber, and R. J. Wood, 'Ultragentle manipulation of delicate structures using a soft robotic gripper', *Sci. Robot.*, vol. 4, no. 33, p. eaax5425, Aug. 2019, doi: 10.1126/scirobotics.aax5425.

[8] J. Jørgensen, 'Interaction with Soft Robotic Tentacles', in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, in HRI '18. New York, NY, USA: Association for Computing Machinery, Mar. 2018, p. 38. doi: 10.1145/3173386.3177838.

[9] J. Saldien, B. Vanderborght, and D. Lefeber, 'The social Robotplatform Probo', in *Proceedings of the 28th Annual European Conference on Cognitive Ergonomics*, in ECCE '10. New York, NY, USA: Association for Computing Machinery, Aug. 2010, pp. 363–364. doi: 10.1145/1962300.1962386.

[10] H. Wang, R. Zhang, W. Chen, X. Wang, and R. Pfeifer, 'A cable-driven soft robot surgical system for cardiothoracic endoscopic surgery: preclinical tests in animals', *Surg. Endosc.*, vol. 31, no. 8, pp. 3152–3158, Aug. 2017, doi: 10.1007/s00464-016-5340-9.

[11] B. Mazzolai *et al.*, 'Roadmap on soft robotics: multifunctionality, adaptability and growth without borders', *Multifunct. Mater.*, vol. 5, no. 3, p. 032001, Aug. 2022, doi: 10.1088/2399-7532/ac4c95.

[12] C. Zhang, P. Zhu, Y. Lin, Z. Jiao, and J. Zou, 'Modular Soft Robotics: Modular Units, Connection Mechanisms, and Applications', *Adv. Intell. Syst.*, vol. 2, no. 6, p. 1900166, 2020, doi: 10.1002/aisy.201900166.

[13] W. Yang *et al.*, 'Multifunctional Soft Robotic Finger Based on a Nanoscale Flexible Temperature–Pressure Tactile Sensor for Material Recognition', *ACS Appl. Mater. Interfaces*, vol. 13, no. 46, pp. 55756–55765, Nov. 2021, doi: 10.1021/acsami.1c17923.

[14] S. Terryn *et al.*, 'A review on self-healing polymers for soft robotics', *Mater. Today*, vol. 47, pp. 187–205, Jul. 2021, doi: 10.1016/j.mattod.2021.01.009.

[15] R. B. N. Scharff, J. Wu, J. M. P. Geraedts, and C. C. L. Wang, 'Reducing Out-of-Plane Deformation of Soft Robotic Actuators for Stable Grasping', in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2019, pp. 265–270. doi: 10.1109/ROBOSOFT.2019.8722823.

[16] L. Ionov, 'Hydrogel-based actuators: possibilities and limitations', *Mater. Today*, vol. 17, no. 10, pp. 494–503, Dec. 2014, doi: 10.1016/j.mattod.2014.07.002.

[17] J. Walker *et al.*, 'Soft Robotics: A Review of Recent Developments of Pneumatic Soft Actuators', *Actuators*, vol. 9, no. 1, Art. no. 1, Mar. 2020, doi: 10.3390/act9010003.

[18] S. Terryn, J. Brancart, D. Lefeber, G. Van Assche, and B. Vanderborght, 'Self-healing soft pneumatic robots', *Sci. Robot.*, vol. 2, no. 9, p. eaan4268, Aug. 2017, doi: 10.1126/scirobotics.aan4268.

[19] *Adaptive Gripper Fingers*, (Oct. 31, 2018). Accessed: Nov. 17, 2022. [Online Video]. Available: https://www.youtube.com/watch?v=jOc3e5O5OPM

[20] T. Takuma, 'Design of Tendon-Driven Mechanism Using Geometrical Condition', *Actuators*, vol. 9, no. 3, Art. no. 3, Sep. 2020, doi: 10.3390/act9030048.

[21] H. Wang, S. Cui, Y. Wang, and C. Song, 'A Hybrid Electromagnetic and Tendon-Driven Actuator for Minimally Invasive Surgery', *Actuators*, vol. 9, no. 3, Art. no. 3, Sep. 2020, doi: 10.3390/act9030092.

[22] E. Roels, S. Terryn, J. Brancart, G. Van Assche, and B. Vanderborght, 'A Multi-Material Self-Healing Soft Gripper', in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2019, pp. 316–321. doi: 10.1109/ROBOSOFT.2019.8722781.
[23] E. Roels *et al.*, 'Processing of Self-Healing Polymers for Soft Robotics', *Adv. Mater.*, vol. 34, no. 1, p. 2104798, 2022, doi: 10.1002/adma.202104798.
[24] N. Guo and M. C. Leu, 'Additive manufacturing: technology, applications and research needs', *Front. Mech. Eng.*, vol. 8, no. 3, pp. 215–243, Sep. 2013, doi: 10.1007/s11465-013-0248-8.
[25] 'Additive Manufacturing for Self-Healing Soft Robots | Soft Robotics'. https://www-liebertpub-com.myezproxy.vub.ac.be/doi/10.1089/soro.2019.0081 (accessed Nov. 17, 2022).
[26] M. Bhuvanesh Kumar and P. Sathiya, 'Methods and materials for additive manufacturing: A critical review on advancements and challenges', *Thin-Walled Struct.*, vol. 159, p. 107228, Feb. 2021, doi: 10.1016/j.tws.2020.107228.
[27] G. Alici, 'Softer is Harder: What Differentiates Soft Robotics from Hard Robotics?', *MRS Adv.*, vol. 3, no. 28, pp. 1557–1568, Jun. 2018, doi: 10.1557/adv.2018.159.
[28] S. Terryn, E. Roels, J. Brancart, G. Van Assche, and B. Vanderborght, 'Self-Healing and High Interfacial Strength in Multi-Material Soft Pneumatic Robots via Reversible Diels–Alder Bonds', *Actuators*, vol. 9, no. 2, Art. no. 2, Jun. 2020, doi: 10.3390/act9020034.
[29] R. A. T. M. van Benthem, W. (Marshall) Ming, and G. (Bert) de With, 'Self Healing Polymer Coatings', in *Self Healing Materials: An Alternative Approach to 20 Centuries of Materials Science*, S. van der Zwaag, Ed., in Springer Series in Materials Science. Dordrecht: Springer Netherlands, 2007, pp. 139–159. doi: 10.1007/978-1-4020-6250-6_7.
[30] H. M. Jonkers, 'Self Healing Concrete: A Biological Approach', in *Self Healing Materials: An Alternative Approach to 20 Centuries of Materials Science*, S. van der Zwaag, Ed., in Springer Series in Materials Science. Dordrecht: Springer Netherlands, 2007, pp. 195–204. doi: 10.1007/978-1-4020-6250-6_9.
[31] S. Xu, A. García, J. Su, Q. Liu, A. Tabaković, and E. Schlangen, 'Self-Healing Asphalt Review: From Idea to Practice', *Adv. Mater. Interfaces*, vol. 5, no. 17, p. 1800536, 2018, doi: 10.1002/admi.201800536.
[32] E. Calabrese *et al.*, 'Design of self-healing catalysts for aircraft application', *Int. J. Struct. Integr.*, vol. 9, no. 6, pp. 723–736, Jan. 2018, doi: 10.1108/IJSI-12-2017-0077.
[33] R. Frei, R. McWilliam, B. Derrick, A. Purvis, A. Tiwari, and G. Di Marzo Serugendo, 'Self-healing and self-repairing technologies', *Int. J. Adv. Manuf. Technol.*, vol. 69, no. 5, pp. 1033–1061, Nov. 2013, doi: 10.1007/s00170-013-5070-2.
[34] S. J. Benight, C. Wang, J. B. H. Tok, and Z. Bao, 'Stretchable and self-healing polymers and devices for electronic skin', *Prog. Polym. Sci.*, vol. 38, no. 12, pp. 1961–1977, Dec. 2013, doi: 10.1016/j.progpolymsci.2013.08.001.
[35] Y. Chen, X. Tan, D. Yan, Z. Zhang, and Y. Gong, 'A Composite Fabric-Based Soft Rehabilitation Glove With Soft Joint for Dementia in Parkinson's Disease', *IEEE J. Transl. Eng. Health Med.*, vol. 8, pp. 1–10, 2020, doi: 10.1109/JTEHM.2020.2981926.
[36] O. A. van Nierop, A. van der Helm, K. J. Overbeeke, and T. J. P. Djajadiningrat, 'A natural human hand model', *Vis. Comput.*, vol. 24, no. 1, pp. 31–44, Jan. 2008, doi: 10.1007/s00371-007-0176-x.
[37] Y. Xue, Z. Ju, K. Xiang, J. Chen, and H. Liu, 'Multimodal Human Hand Motion Sensing and Analysis—A Review', *IEEE Trans. Cogn. Dev. Syst.*, vol. 11, no. 2, pp. 162–175, Jun. 2019, doi: 10.1109/TCDS.2018.2800167.
[38] S. Panchal-Kildare and K. Malone, 'Skeletal Anatomy of the Hand', *Hand Clin.*, vol. 29, no. 4, pp. 459–471, Nov. 2013, doi: 10.1016/j.hcl.2013.08.001.
[39] M. A. Abdul Wahit, S. A. Ahmad, M. H. Marhaban, C. Wada, and L. I. Izhar, '3D Printed Robot Hand Structure Using Four-Bar Linkage Mechanism for Prosthetic Application', *Sensors*, vol. 20, no. 15, Art. no. 15, Jan. 2020, doi: 10.3390/s20154174.
[40] G. ElKoura and K. Singh, 'Handrix: Animating the Human Hand', p. 11.
[41] R. H. Chowdhury, M. B. I. Reaz, M. A. B. M. Ali, A. A. A. Bakar, K. Chellappan, and T. G. Chang, 'Surface Electromyography Signal Processing and Classification Techniques', *Sensors*, vol. 13, no. 9, Art. no. 9, Sep. 2013, doi: 10.3390/s130912431.
[42] B. B. Kang, H. Choi, H. Lee, and K.-J. Cho, 'Exo-Glove Poly II: A Polymer-Based Soft Wearable Robot for the Hand with a Tendon-Driven Actuation System', *Soft Robot.*, vol. 6, no. 2, pp. 214–227, Apr. 2019, doi: 10.1089/soro.2018.0006.
[43] Y. Chen, Z. Yang, and Y. Wen, 'A Soft Exoskeleton Glove for Hand Bilateral Training via Surface EMG', *Sensors*, vol. 21, no. 2, Art. no. 2, Jan. 2021, doi: 10.3390/s21020578.

[44] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, and C. J. Walsh, 'Soft robotic glove for combined assistance and at-home rehabilitation', *Robot. Auton. Syst.*, vol. 73, pp. 135–143, Nov. 2015, doi: 10.1016/j.robot.2014.08.014.
[45] Z. Sun, Z. Guo, and W. Tang, 'Design of wearable hand rehabilitation glove with soft hoop-reinforced pneumatic actuator', *J. Cent. South Univ.*, vol. 26, no. 1, pp. 106–119, Jan. 2019, doi: 10.1007/s11771-019-3986-x.
[46] P. Ben-Tzvi, J. Danoff, and Z. Ma, 'The Design Evolution of a Sensing and Force-Feedback Exoskeleton Robotic Glove for Hand Rehabilitation Application', *J. Mech. Robot.*, vol. 8, no. 5, May 2016, doi: 10.1115/1.4032270.
[47] N. Takahashi, S. Furuya, and H. Koike, 'Soft Exoskeleton Glove with Human Anatomical Architecture: Production of Dexterous Finger Movements and Skillful Piano Performance', *IEEE Trans. Haptics*, vol. 13, no. 4, pp. 679–690, Oct. 2020, doi: 10.1109/TOH.2020.2993445.
[48] T. du Plessis, K. Djouani, and C. Oosthuizen, 'A Review of Active Hand Exoskeletons for Rehabilitation and Assistance', *Robotics*, vol. 10, no. 1, Art. no. 1, Mar. 2021, doi: 10.3390/robotics10010040.
[49] Q. Meng, S. Xiang, and H. Yu, 'Soft Robotic Hand Exoskeleton Systems: Review and Challenges Surrounding the Technology', presented at the 2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017), Atlantis Press, Apr. 2017, pp. 186–190. doi: 10.2991/eame-17.2017.45.
[50] J. Legrand, S. Terryn, E. Roels, and B. Vanderborght, 'Reconfigurable, Multi-Material, Voxel-Based Soft Robots', *IEEE Robot. Autom. Lett.*, vol. 8, no. 3, pp. 1255–1262, Mar. 2023, doi: 10.1109/LRA.2023.3236883.
[51] F. Faure *et al.*, 'SOFA: A Multi-Model Framework for Interactive Physical Simulation', in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Y. Payan, Ed., in Studies in Mechanobiology, Tissue Engineering and Biomaterials. Berlin, Heidelberg: Springer, 2012, pp. 283–321. doi: 10.1007/8415_2012_125.
[52] J. Allard *et al.*, 'SOFA - an Open Source Framework for Medical Simulation'.
[53] V. Vörös *et al.*, 'Comparison of 2D and autostereoscopic 3D visualization during mixed reality simulation', *Int. J. Comput. Assist. Radiol. Surg.*, Mar. 2023, doi: 10.1007/s11548-023-02876-4.
[54] M. Lerotic, S.-L. Lee, J. Keegan, and G.-Z. Yang, 'Image constrained finite element modelling for real-time surgical simulation and guidance', in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, Jun. 2009, pp. 1063–1066. doi: 10.1109/ISBI.2009.5193239.
[55] C. Duriez *et al.*, 'Framework for online simulation of soft robots with optimization-based inverse model', in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Dec. 2016, pp. 111–118. doi: 10.1109/SIMPAR.2016.7862384.
[56] K. Wu and G. Zheng, 'Simulation and control co-design methodology for soft robotics', in *2020 39th Chinese Control Conference (CCC)*, Jul. 2020, pp. 3910–3914. doi: 10.23919/CCC50068.2020.9189205.
[57] T. Morzadec, D. Marcha, and C. Duriez, 'Toward Shape Optimization of Soft Robots', in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2019, pp. 521–526. doi: 10.1109/ROBOSOFT.2019.8722822.
[58] P. Ferrentino, E. Roels, J. Brancart, S. Terryn, G. Van Assche, and B. Vanderborght, 'Finite Element Analysis-Based Soft Robotic Modeling: Simulating a Soft Actuator in SOFA', *IEEE Robot. Autom. Mag.*, pp. 2–12, 2023, doi: 10.1109/MRA.2022.3220536.
[59] P. Ferrentino, S. K. Tabrizian, J. Brancart, G. V. Assche, B. Vanderborght, and S. Terryn, 'FEA-Based Inverse Kinematic Control: Hyperelastic Material Characterization of Self-Healing Soft Robots', *IEEE Robot. Autom. Mag.*, vol. 29, no. 3, pp. 78–88, Sep. 2022, doi: 10.1109/MRA.2021.3132803.
[60] P. Ferrentino *et al.*, 'Quasi-Static FEA Model for a Multi-Material Soft Pneumatic Actuator in SOFA', *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7391–7398, Jul. 2022, doi: 10.1109/LRA.2022.3183254.

Chapter 6: Appendix

The code developed in section 2.1.2 **Image processing**, can be found in Figure 54 up to Figure 57. The specific functions explained there, are first presented, after which the main loop of the algorithm is coded. At the end, the saved frames are combined and put into a video.

```
def findneighbors(pixel,neighbors_number, threshold):
    bordercheck = False
    #Defining the neigbors of the pixel
    if neighbors_number == 4:
        neighbors = [[pixel[0]+1,pixel[1]],[pixel[0],pixel[1]+1],[pixel[0]-1,pixel[1]],[pixel[0],pixel[1]-1]]
    elif neighbors_number == 8:
        neighbors = [[pixel[0]+1,pixel[1]],[pixel[0],pixel[1]+1],[pixel[0]-1,pixel[1]],[pixel[0],pixel[1]-1]
                     ,[pixel[0]+1,pixel[1]+1],[pixel[0]+1,pixel[1]-1],[pixel[0]-1,pixel[1]+1],[pixel[0]-1,pixel[1]-1]]
    #filter out the neighbors out of frame
    neighbors = BoundaryCheck(neighbors)
    #set to true if a neighbour is part of the background
    for neighbor in neighbors:
        if(pixelvalues[neighbor[0]+width_image*neighbor[1]][2]<marker_threshold
           and pixelvalues[neighbor[0]+width_image*neighbor[1]][0]<threshold):
            bordercheck = True
    return bordercheck

def BoundaryCheck(neighbors):
    for neighbor in neighbors:
        if(neighbor[0]>=(width_image) or neighbor[0]<0 or neighbor[1]>=(height_image) or neighbor[1]<0):
            neighbors.remove(neighbor) #delete neighbours out of bounds
            neighbors = BoundaryCheck(neighbors)
    return neighbors

def DrawLine(point1,point2):
    if (point1[0]<point2[0]):
        X = np.arange(point1[0],point2[0],0.1)
        Y = (point2[1]-point1[1])/(point2[0]-point1[0])*(X-point1[0]) + point1[1]
        rico = (point2[1]-point1[1])/(point2[0]-point1[0])
    else:
        X = np.arange(point2[0],point1[0],0.1)
        Y = (point1[1]-point2[1])/(point1[0]-point2[0])*(X-point2[0]) + point2[1]
        rico = (point1[1]-point2[1])/(point1[0]-point2[0])
    return X,Y,rico
```

Figure 54 Implementation of the findneighbours, BoundaryCheck and DrawLine function.

```
def MarkerOrder(center_clusters,Initialize, mean_clust_prev):
    remove_pic = False
    dist_list = np.empty((0,1), int)
    temp_list = np.empty((0,2), int)
    if(Initialize):
        temp_list = np.sort(center_clusters,axis=0)
        mean_clust = [temp_list[3],temp_list[2],temp_list[1],temp_list[0]] #For first frame: sort along x-values
    else:
        #Determine distance between the points in this frame and from previous frame
        for points in mean_clust_prev:
            dist_list = np.empty((0,1), int)
            for pixels in center_clusters:
                    delta = ((points[0]-pixels[0])**2+(points[1]-pixels[1])**2)**0.5
                    dist_list = np.append(dist_list, delta)
            #If distance is not too big, the nearest point of the new frame correspond to the same marker of the previous frame
            if (np.min(dist_list)<50):
                temp = center_clusters[np.argmin(dist_list)]
                temp_list = np.append(temp_list, np.array([temp]), axis =0)
            else:
                print('Frame ' + str(q+1) + ' destroyed.' )
                remove_pic = True
                mean_clust = center_clusters
                break
        if(remove_pic == False):
            mean_clust = [temp_list[0], temp_list[1],temp_list[2],temp_list[3]]
    return mean_clust, remove_pic
```

Figure 55 Implementation of the MarkerOrder function.

```
def segmentation(threshold, Initialize, mean_clust_prev):
    region = np.empty((0,2), int)
    marker_region = np.empty((0,2), int)
    center_clusters = np.empty((0,2), int)
    center_points = np.empty((0,2), int)
    counter = 0
    for i in range(width_image):
        i = int(i)
        #if ((i%10)==0):
        #    print("i ",i)
        for j in range(height_image):
            j = int(j)
            r = pixelvalues[width_image*j+i][0]
            b = pixelvalues[width_image*j+i][1]
            g = pixelvalues[width_image*j+i][2]
            #Save the finger pixels
            if(r<=threshold and g>=marker_threshold and b >= marker_threshold):
                #Save only the border pixels of the finger
                bordercheck = findneighbors([i,j],4,threshold)
                if(bordercheck):
                    region = np.append(region, np.array([[i,j]]), axis=0)
            #Save the marker pixels
            if(r>=threshold):
                    marker_region = np.append(marker_region, np.array([[i,j]]), axis=0)
    #Use kmeans to split the marker pixels in their separate markers
    kmeans = KMeans(n_clusters=k, init='random')
    kmeans.fit(marker_region)
    labels = kmeans.labels_
    #Determine the center point of each marker
    for clusters in range(k):
        min_distance = 100000000000000
        point = np.empty((0,2), int)
        mask = (labels == clusters)
        marker_region_k = marker_region[mask] # the mask is used to select the clusters with a specific label
        center = np.median(marker_region_k, axis=0)
        center_points = np.append(center_points, np.array([center]), axis =0)
        #Save the pixel of the finger that is nearest to the center of the marker
        for pixel in region:
            distance = ((pixel[0]-center[0])**2 + (pixel[1]-center[1])**2)**0.5
            if (distance<min_distance):
                point = pixel
                min_distance = distance
        center_clusters = np.append(center_clusters, np.array([point]), axis =0)
    mean_clust, remove_pic = MarkerOrder(center_clusters, Initialize, mean_clust_prev)
    image = np.zeros((1280,720)).astype(int)
    for pixel in region:
        image[pixel[1],pixel[0]] = 1
    for pixel in marker_region:
        image[pixel[1],pixel[0]] = 2
    return image, mean_clust, remove_pic
```

Figure 56 Implementation of the Segmentation function.

```
img_array = []
frames_destroyed = 0
mean_clust_list = np.empty((0,2),int)
rico_list = np.empty((0,1),int)
teller =0
for q in range (frames//fps):
    image1 = Image.open('Finger' + str(q*fps) + '.jpg')
    width_image, height_image = image1.size
    pixelvalues = list(image1.getdata())
    threshold = 110
    marker_threshold = 57
    k = 4
    Initialize = False
    if(q==0):
        Initialize = True
        post, mean_clust, remove_pic = segmentation(threshold, Initialize, [0, 0])
    else:
        post, mean_clust, remove_pic = segmentation(threshold, Initialize, [mean_clust_list[4*teller-4]
                                                                            ,mean_clust_list[4*teller-3]
                                                                            ,mean_clust_list[4*teller-2]
                                                                            ,mean_clust_list[4*teller-1]])
    if(remove_pic == False):
        teller += 1
        fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(17,8))
        axes.imshow(post, cmap="CMRmap")
        #plot the four marker points
        for mean in mean_clust:
            axes.plot(mean[0],mean[1],"*",c='red')
        for i in range (k-1):
            X,Y,rico = DrawLine(mean_clust[i],mean_clust[i+1])
            rico_list = np.append(rico_list, rico)
            axes.plot(X,Y,c='red')
        axes.set_title("Postprocessed finger")
        #save iamge as jpg
        plt.savefig('Finger_post' + str(q*fps) +'.jpg')
        plt.close()
        image1.close()
        img = cv2.imread('Finger_post' + str(q*fps) + '.jpg')
        os.remove('Finger_post' + str(q*fps) + '.jpg')
        os.remove('Finger' + str(q*fps) + '.jpg')
        height,width, layers = img.shape
        size = (width, height)
        img_array.append(img)
        print('image ' + str(q+1) + ' out of ' + str(frames//fps+1) +
              ' with ' + str(frames_destroyed) + ' frame(s) destroyed.')
        mean_clust_list = np.append(mean_clust_list, mean_clust, axis = 0)
    else:
        frames_destroyed += 1

#Create a video out of the saved images
out = cv2.VideoWriter('Finger_new4_undist3.avi',cv2.VideoWriter_fourcc(*'DIVX'),30//fps,size)
for i in range(len(img_array)):
    out.write(img_array[i])
    cv2.destroyAllWindows()
out.release()
```

Figure 57 Implementation of the main loop of the algorithm and the creation of the post-processed video.

The designs explained in section 2.3 **Mechanical design** have been put in technical drawings containing the dimensions used for the design (Figure 58 and Figure 59). Using these technical drawings, the design can be recreated in any CAD software. These drawings also allow a more detailed view of the designs to as such get a better understanding of the final product.

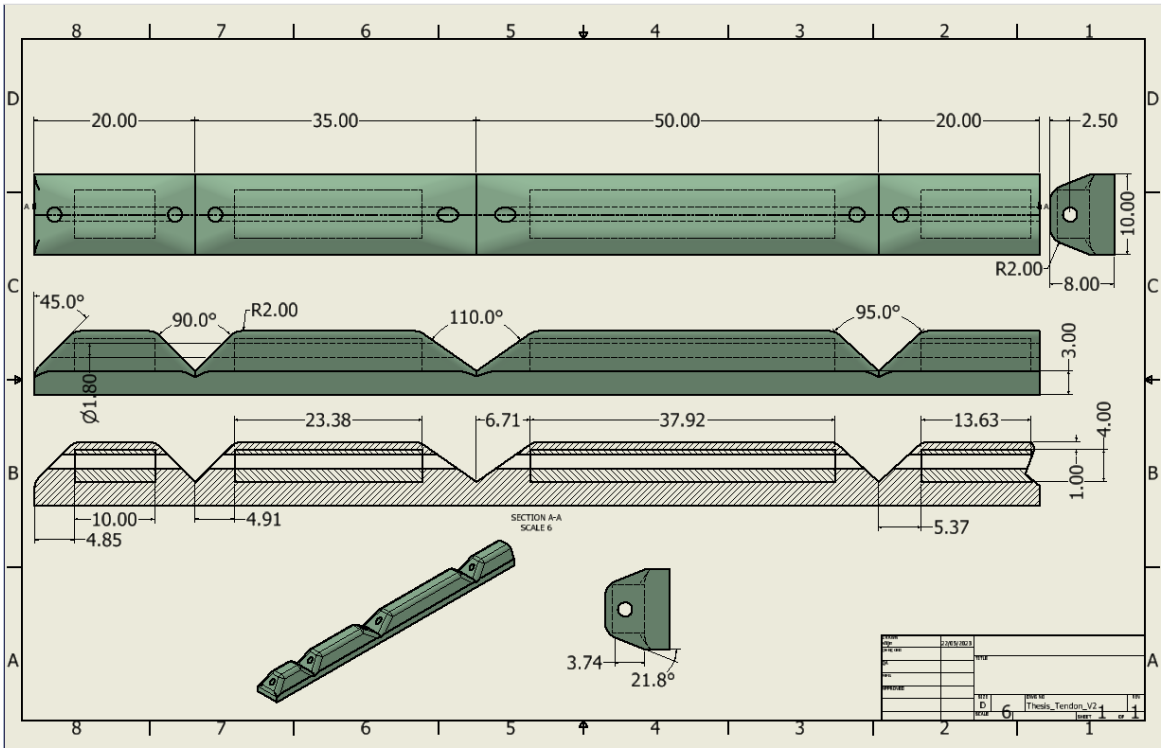


Figure 58 Technical drawing of the outer actuator.

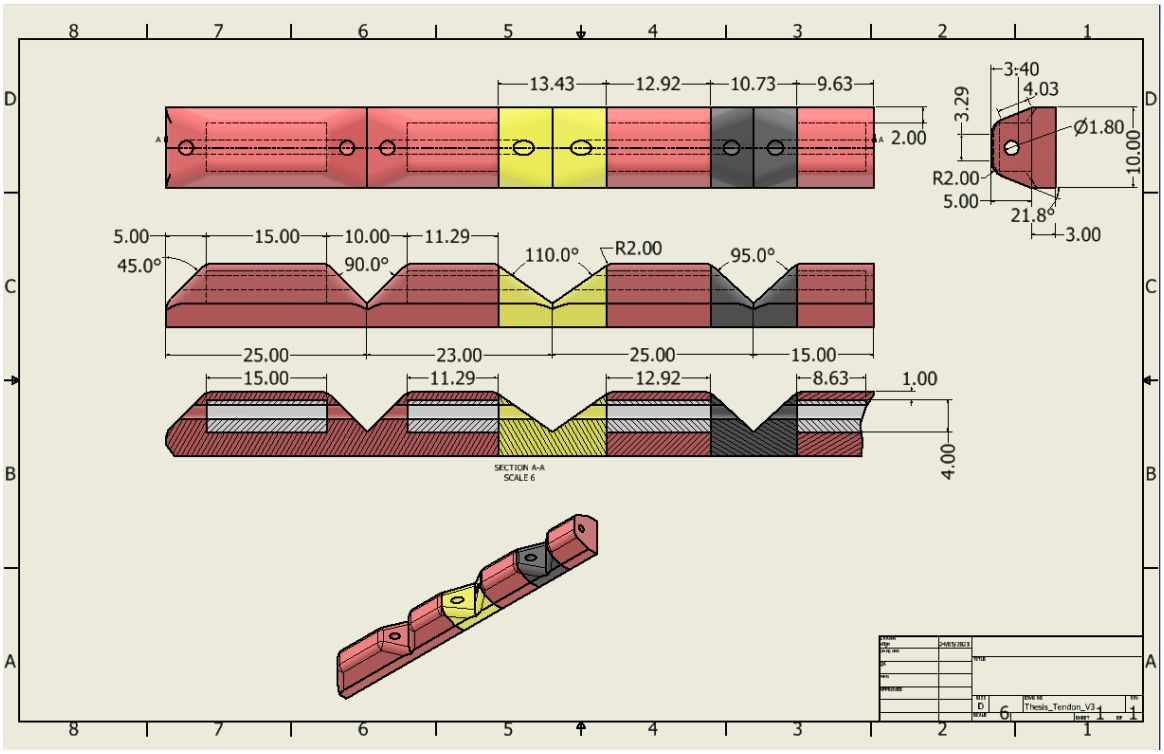


Figure 59 Technical drawing of the inner actuator.

Apart from the designs of the actuator, the technical drawing of the negative image of the moulds used throughout section 2.5 **Manufacturing of the actuator** are shown from Figure 60 up to Figure 64. These designs were 3D-printed and used as mould for the casting of the silicone moulds used for the manufacturing of the actuators. In these technical drawings, the design of the finger was not dimensioned and instead only dimensions for the mould itself are given. It is thus assumed the design of the actuator is already made.

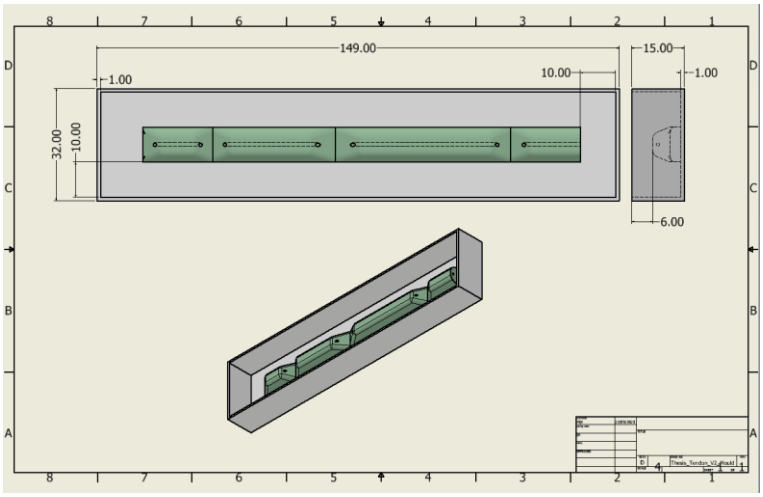


Figure 60 Technical drawing of the mould used for the first casting method.

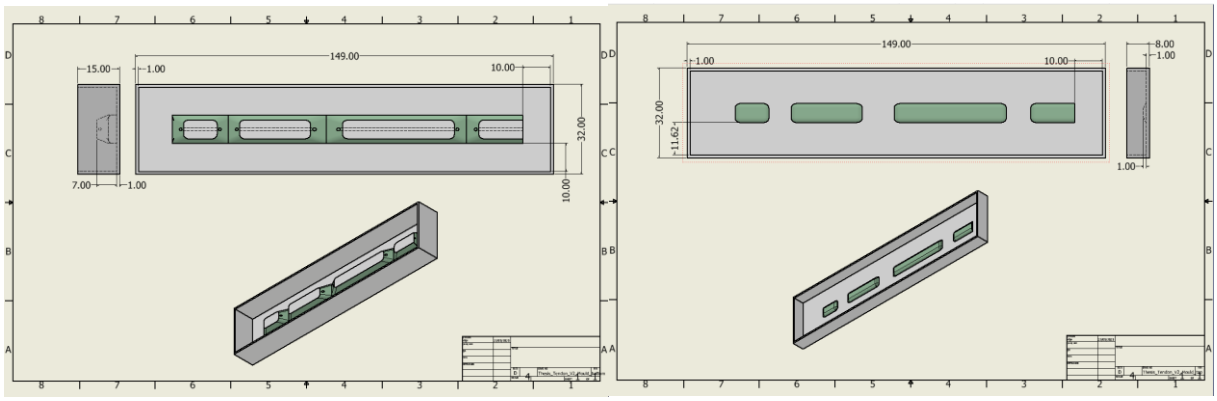


Figure 61 Technical drawing of the moulds used for the second casting method.

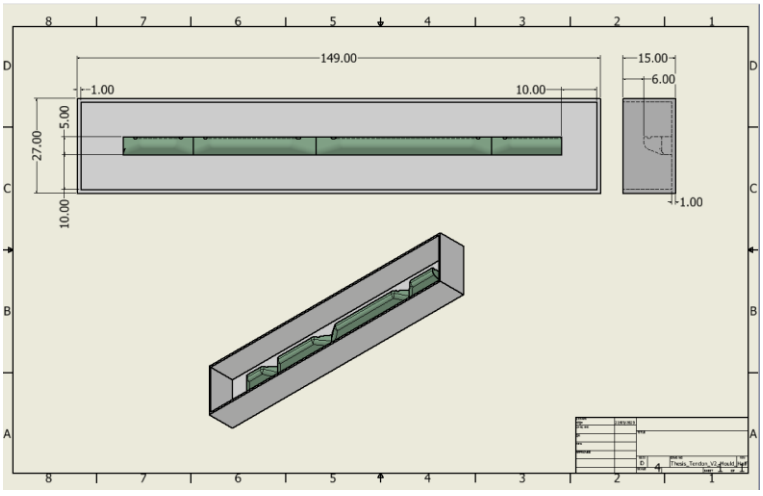


Figure 62 Mould used for casting one half of the actuator according to the third casting method.

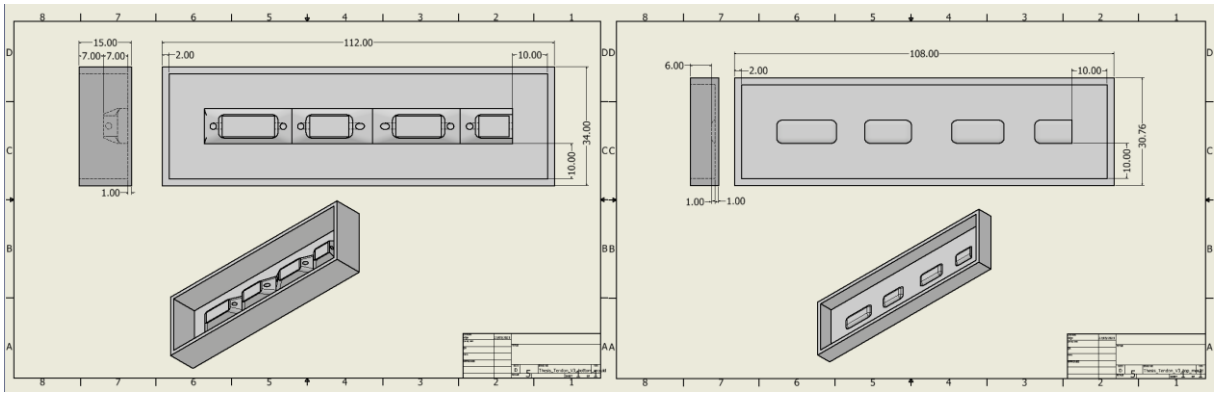


Figure 63 Technical drawings of the moulds used for the casting of the inner actuator made out of one self-healing material.

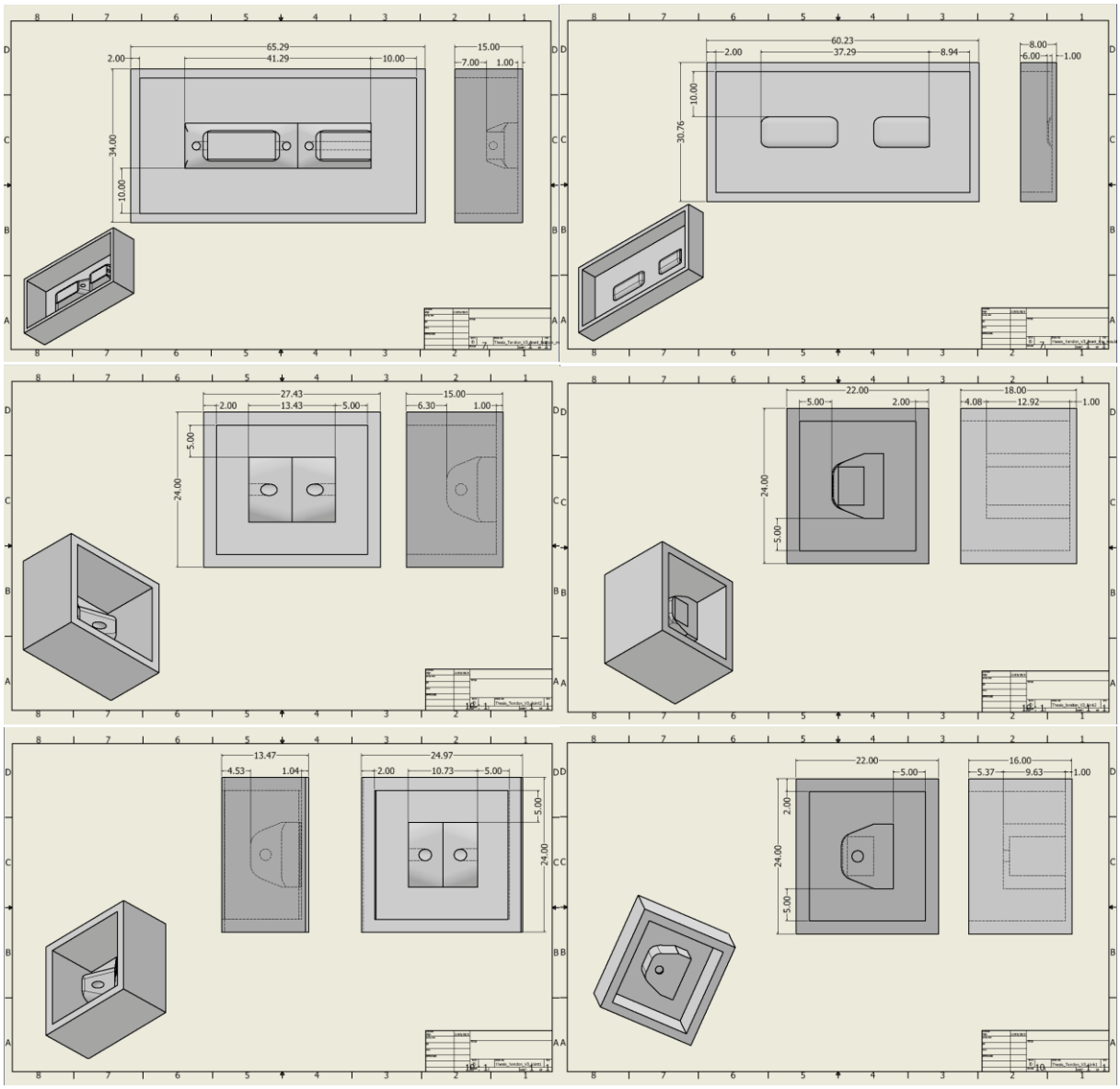


Figure 64 The technical drawings for the moulds used for the casting of the final inner actuator made out of three self-healing materials.