Master thesis

A first step to visual servoing based autonomous handovers with soft robots

J.G.A. Vervloet

TU Delft

Master thesis

A first step to visual servoing based autonomous handovers with soft robots

by

J.G.A. Vervloet

to obtain the degree of

Master of Science

in Robotics and Systems & Control

April 29, 2025

Student number:	4686276	
Project duration:	September, 2023 – May, 2025	
Thesis committee:	Associate Professor C. Della Santina,	Cognitive Robotics Department, TU Delft
	Assistant Professor A. Dabiri	Delft Center for Systems and Control, TU Delft
	Assistant Professor L. Peternel	Cognitive Robotics Department, TU Delft
	F. Stella, PhD	Cognitive Robotics Department, TU Delft & CREATE Lab, EPFL

This thesis report is confidential and cannot be made public until June, 2025.

TU Delft

A first step to visual servoing based autonomous handovers with soft robots

J.G.A. Vervloet
Supervisors:
Associate Professor Dr. C. Della Santina, Department of Cognitive Robotics, TU Delft;
Assistant Professor A. Dabiri, Delft Center for Systems and Control, TU Delft;
F. Stella, PhD

***Abstract* —**
To make interactions between humans and robots safer, soft robots may offer a solution. The autonomous closed loop control of these robots so far, however, is not accurate enough to perform specific tasks as handovers. The purpose of this paper is to propose a control algorithm that can make the control of soft robots accurate enough for human-to-robot handovers. The main focus within this research was on the state estimation and the Jacobian based control. Due to gravity, the internal system state is not an accurate representation of the actual system behavior. The optimized pose estimation solves this problem. In order to test the proposed algorithm, the complete control architecture has been implemented including the object and end-effector detection. Experiments have shown that the algorithm works with different step sizes within the Jacobian based control, consistently resulting in a successful handover. A second experiment has shown that the handovers are still successful and faster when a human guides the robot toward the right position. Lastly, a possible use case has been shown.

soft robotics | autonomous control | visual servoing | handovers | Helix

1. Introduction

Robots have been around for a while and these robots operate in all kind of disciplines, including agriculture, automotive, logistics and healthcare where both simple tasks (within a controlled environment) and challenging tasks (with lots of uncertainties) are performed [14]. This research focuses on a complex task: Automated human-robot handovers.

These handovers could be done by 'traditional' rigid robots, like the KUKA robots [10] and the TIAGo robot of PAL robotics [12]. These robots are composed of connected rigid segments. Due to the rigid segments, these robots generally do not deform (significantly) under loads. They can be modeled relatively easy and (partly because of this) controlled with high accuracy [13].

There are, however, also downsides to these rigid robots. First of all, these rigid robots have limited efficiency when the encountered situations are unknown or uncertain, with unknown environments and possible external disturbances from real world interactions. This comes forth due to the lack of compliance. Besides this, the rigidness of the robot can make it unsafe for humans to interact with it. This risk can be reduced by separating robots and humans as much as possible, but it will inevitably be present during human-robot interactions like handovers [13].

This is where soft robots, inspired by the compliance and softness of animals, come into play. The vast majority of animals are soft bodied and these soft structures enables them to move effectively in complex natural environments [2].
Soft robotics try to mimic this behavior in a certain way; creating compliant structures to adaptively perform operations in interactions within unknown environments [11]. This has two main advantages:

- Soft robots can handle environmental impacts. Due to the high compliance, the soft structures will elastically deform rather than break or plastically deform. This has been shown on physical systems [4, 5].
- Soft robots by design can safely interact with humans. This is a big advantage over rigid robots.

Based on this idea, a lot of different soft robots have been designed. One important distinguish that can be made, is the difference between *articulated soft robots* and *continuum soft robots*.

Articulated soft robots are quite similar to vertebrate muscle-skeletal system [1], with elastic components, making them much more compliant than regular rigid robots. Although these are soft systems due to the incorporated elastic components, these systems still include rigid elements. The modeling and control of such articulated soft robots is very similar to the modeling and control of (multibody) rigid systems.

Continuum soft robots, on the other hand, are based on invertebrates. As a consequence, these continuum soft robots (or continuous robots) are generally much more compliant as they in essence don't have any rigid parts. Where rigid robots and articulated soft robots do have a certain number of degrees of freedom, a continuous soft robot has in theory infinite degrees of freedom, as they can bend at any point. In a certain way it can therefore be seen as a continuum of joints. This makes the modeling and control significantly different than that of rigid systems.

It has been noticed that automated handovers have been done with rigid robot arms, but not yet with continuum soft robot arms. There is, to the knowledge of the author, currently no research on soft robots used for automated handover tasks. This is probably not done because handovers are a complex task incorporating accurate computer vision, accurate control and the interacting with a human which lead to an uncertain environment. It is, however, important to research this topic because soft robots are, due to their compliance, generally very safe for interaction with humans, therefore they can make human-robot collaborative tasks safer and possibly more pleasant for the interacting human.

As a result, this research is based on a single research question:

How can a control-architecture for soft robots combined with a visual servoing setup be developed to perform automated handovers?

This question reflect the urge for improving the overall algorithm for the autonomous control of soft robotic system, by not only incorporating the whole integrated system (from camera setup and computer vision algorithm to accurate control), but also enhancing the control accuracy, making it accurate enough for the handovers to take place.

Within this research, the main focus will be on the model and the control algorithm, rather than the computer vision for

the detection of the end-effector and object to handover.

The background information will be provided in section 2. The method will be discussed in section 3, followed by the results (section 4) and the discussion (section 5). Section 6 contains the conclusion. After the references four appendices are attached.

2. Background: Model

This section contains the background information concerning the used model. There are various different modeling methods for soft robots (including continuum mechanics models, discrete material models and surrogate models). This research focuses on the kinematic piecewise constant curvature model, as it has a relatively low amount of degrees of freedom compared to the other models, while it still provides a sufficiently accurate representation of the actual system. Research also shows that this model can be used for different kinds of soft continuum robots, including tendon driven systems [8].

2.1. Constant curvature model.

The constant curvature model assumes that the robot will create a constant moment along the arm, and as a consequence this leads to a constant curvature. Although this approximation is not accurate, it can lead to satisfying results when used on actual robots [7, 9]. One of the main advantages is the Degrees of Freedom (DOF). As can be seen in Figure 1b, the model can be described using only 3 parameters. These can be the angle ϕ that specifies the plane and direction of the curvature, the robot length l (or L) and segment curvature angle θ. There are multiple variations in choosing the state. One could also use the curvature radius $r = \frac{l}{\theta}$ or the curvature $\kappa = \frac{\theta}{l}$.

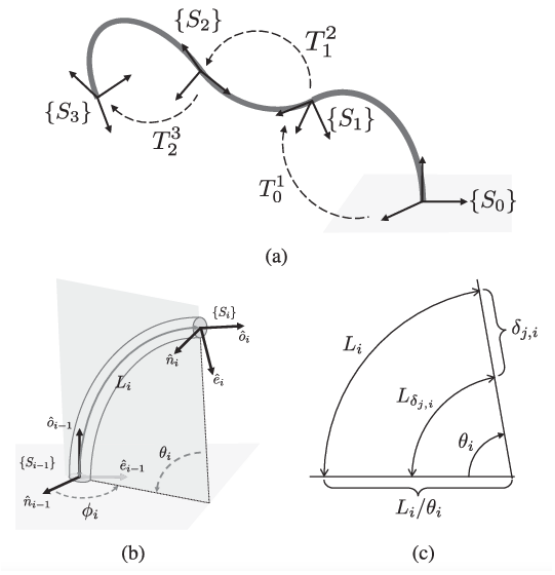


Fig. 1. Schematic overview of the piecewise constant curvature model. Figure is derived from [3].

The actuator space is different for every soft robot. For tendon driven robots this actuator space can include the various tendon lengths or the motor torques while for a pneumatic robot, the applied pressures are important in order to describe the robot state. Each robot that is modeled with the constant curvature model, will have its own specific transformation $q = g^{-1}(.)$ towards the configuration space. In this configuration space, the robot kinematics can be described with variables that can be used for every constant curvature model. These include the robot length l, segment curvature κ and the curvature-plane rotation ϕ. In that case the state is specified using arc geometry [15].

The configuration space can be used to find the state in the task space (using $X = f(q)$), which is usually (but not necessarily) the 3D position in Cartesian space. This transformation is no longer dependent on the actual robot properties, but only on the kinematic properties described in the configuration space.

2.1.1. Piecewise constant curvature.

When the robot consists of multiple segment, or is actuated on various points, it can be useful to model the robot as multiple segments with each its own constant curvature. This leads to the piece-wise constant curvature model (PCC). As a single constant curvature model can be expressed with 3 parameters in the configuration space, each added segment will consequently lead to 3 additional parameters, as indicated in the top image of Figure 1a.

2.1.2. Mathematical representation.

When using the constant curvature model, one can construct a homogeneous transform from the local frame at the end of a segment with respect to the local frame at the start of the segment. With multiple segments, these can be multiplied as the end frame of the previous segment is the starting frame for the next. The rotation matrix is shown in Equation 1, and the translation vector in Equation 2. In these equations, C_x represents $\cos(x)$ and S_x represents $\sin(x)$.

$$R_{i-1}^{i} = \begin{bmatrix} C_{\phi_i}^2(C_{\theta_i}-1)+1 & S_{\phi_i}C_{\phi_i}(C_{\theta_i}-1) & C_{\phi_i}S_{\theta_i} \\ S_{\phi_i}C_{\phi_i}(C_{\theta_i}-1)+1 & S_{\phi_i}^2(C_{\theta_i}-1) & S_{\phi_i}S_{\theta_i} \\ -C_{\phi_i}S_{\theta_i} & -S_{\phi_i}S_{\theta_i} & C_{\theta_i} \end{bmatrix} \tag{1}$$

$$t_{i-1}^{i} = \frac{l}{\theta}\begin{bmatrix} C_{\phi_i}(1-C_{\theta_i}) & S_{\phi_i}(1-C_{\theta_i}) & S_{\theta_i} \end{bmatrix}^T \tag{2}$$

It should be noted that the the angle θ can be expressed as a function of the curvature κ and the length l, so $\theta = \kappa l$.

The obtained transform is accurate for constant curvature robots, but for systems without curvature ($\theta = \kappa l = 0$) the transformation will become singular. To prevent this, the coordinates can be remapped to other coordinates, using $\Delta_{x,i}$, $\Delta_{y,i}$ and the length l, as was proposed by Della Santina et al. [3]. These describe the rotation along the relative x and y axis and relate to the previously introduces parameters ϕ_i and θ_i as in Equation 3.

$$\begin{aligned} \phi_i(q_i) &= arccos(\frac{\Delta_{x,i}}{\Delta_i}) = arcsin(\frac{\Delta_{y,i}}{\Delta_i}) \\ \theta_i(q_i) &= \frac{\Delta_i}{s_i} \\ \Delta_i^2 &= \Delta_{x,i}^2 + \Delta_{y,i}^2 \end{aligned} \tag{3}$$

When this is used, one obtains the rotation matrix and translation vector as in Equation 4 and 5. This configuration has no singularities with a straight segment and therefore this representation was used to model the system.

$$R_{i-1}^{i} = \begin{bmatrix} (\frac{\Delta_{xi}}{\Delta_i})^2(C_{\frac{\Delta_i}{d_i}}-1)+1 & \frac{\Delta_{yi}\Delta_{xi}}{\Delta_i^2}(C_{\frac{\Delta_i}{d_i}}-1) & \frac{\Delta_{xi}}{\Delta_i}S_{\frac{\Delta_i}{d_i}} \\ \frac{\Delta_{yi}\Delta_{xi}}{\Delta_i^2}(C_{\frac{\Delta_i}{d_i}}-1)+1 & (\frac{\Delta_{yi}}{\Delta_i})^2(C_{\frac{\Delta_i}{d_i}}-1) & \frac{\Delta_{yi}}{\Delta_i}S_{\frac{\Delta_i}{d_i}} \\ -\frac{\Delta_{xi}}{\Delta_i}S_{\frac{\Delta_i}{d_i}} & -\frac{\Delta_{yi}}{\Delta_i}S_{\frac{\Delta_i}{d_i}} & C_{\frac{\Delta_i}{d_i}} \end{bmatrix} \tag{4}$$

$$t_{i-1}^{i} = \frac{l}{\theta}\begin{bmatrix} l_i d_i(\frac{\Delta_{xi}}{\Delta_i^2})(1-C_{\frac{\Delta_i}{d_i}}) & l_i d_i(\frac{\Delta_{yi}}{\Delta_i^2})(1-C_{\frac{\Delta_i}{d_i}}) & \frac{l_i d_i}{\Delta_i}S_{\frac{\Delta_i}{d_i}} \end{bmatrix}^T \tag{5}$$

By multiplying the homogeneous transforms (T_{i-1}^{i}) of the consecutive segments, the transform from the base to the end-effector (T_0^N) can be computed as in Equation 6.

$$T_O^N = T_O^1 T_1^2 \cdots T_{N-1}^N \tag{6}$$

Here the state can be written as in Equation 7.

$$q = \begin{bmatrix} \Delta_{x,1} & \Delta_{y,1} & l_1 & \cdots & \Delta_{x,N} & \Delta_{y,N} & l_N \end{bmatrix}^T \quad (7)$$

3. Method

In this section the methods used during this research will be discussed. First, the used robot will be discussed, followed by the computer vision system. After that, the control architecture is presented. Lastly, the experiments that have been performed are discussed.

3.1. Continuum soft robot: the Helix.

During the experiments, the Helix robot was used [6]. A photo of the Helix is shown in Figure 2. This continuum soft robot consists of three actuated segments, followed by a gripper. Due to its design, it has a big workspace, which is useful for interacting tasks. As there are three segments, the system is modeled with nine state variables.

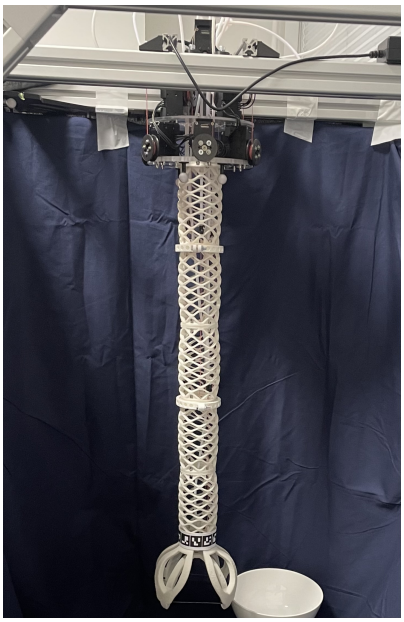


Fig. 2. The Helix robot. The system contains 3 actuated segments. The two lower segments are themselves built up of 2 similar segments.

Each segment is actuated independently, using Bowden cables. Actuation is done by 3 tendons evenly distributed around the segment (with an angle of 120 degrees between the tendons). The gripper is also tendon driven and can simply be opened and closed by pulling and letting go of this single tendon. The transform from the system state q to the tendon lengths l_t is a nonlinear function $g(.)$ based on the known kinematic model of the Helix system, where $l_t = g(q)$.

Before the experiments start, the arm is calibrated. After this calibration process all three segments are hanging down, without any angle and the initial lengths as described in the model. The first/top segment has a length of 0.12 meter, the other two have a length of 0.24 meter. This results in an initial state $X_0 = [0, 0, 0.12, 0, 0, 0.24, 0, 0, 0.24]$. Consequently, the initial length vector can be described as $l_{t,calib} = [0.12, 0.12, 0.12, 0.24, 0.24, 0.24, 0.24, 0.24, 0.24]$, as each segment has 3 tendons. The motor positions are saved as calibration values.

The nine driving motors for the arm motion can be controlled by publishing the desired change of the tendon lengths compared to the calibrated lengths ($\Delta l_t = l_{t,desired} - l_{t,calib}$) on a ROS-topic. The Raspberry Pi 5 inside the Helix system uses these values together with the motor radii and calibration values to find the new motor position.

3.1.1. System limits.

It is important to stress out that the system has some limits in its workspace. These are needed as too much tension on the tendons can lead to a system failure and no tension will lead to an unresponsive system.

The limits are represented in the (6D) DL-space (with state q_{DL}), as shown in Equation 8.

$$q_{DL} = \begin{bmatrix} d_1 & l_1 & d_2 & l_2 & d_3 & l_3 \end{bmatrix}^T \quad (8)$$

Whereas the state q includes the variables dx_i and dy_i, the DL-space uses $d_i = \sqrt{\Delta_{x,i}^2 + \Delta_{y,i}^2}$, as indicated in Equation 9.

$$\begin{aligned} q_{DL} &= \sqrt{A_{q,q_{DL}} q^2} \\ A_{q,q_{DL}} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (9)$$

As dx_i and dy_i both represent an angle along their own unit vector direction, and this direction is chosen arbitrarily, limiting these parameters independently does not make sense. As d_i represents the magnitude of the total angle, this is a logical variable to limit the systems curvature.

By staying in the limits, as indicated in Equation 10, the tendons will not become too short or too long, preventing both a system failure as well as segments without enough tension. The used limits are shown in Equation 11.

$$L_{sys,LB} \leq q_{DL} \leq L_{sys,UB} \quad (10)$$

$$\begin{bmatrix} 0 \\ 0.08 \\ 0 \\ 0.16 \\ 0 \\ 0.16 \end{bmatrix} \leq \begin{bmatrix} d_1 \\ l_1 \\ d_2 \\ l_2 \\ d_3 \\ l_3 \end{bmatrix} \leq \begin{bmatrix} \frac{\pi}{4} \\ 0.14 \\ \frac{\pi}{2} \\ 0.26 \\ \frac{\pi}{2} \\ 0.26 \end{bmatrix} \quad (11)$$

3.2. Camera system.

The camera system is used to detect items of interest, which are twofold:

- A green colored ball. This is the object to grasp.
- The end-effector pose. This will be used for estimating the pose of the full robot arm as well as for the control of the arm to the desired location.

The camera system includes various different elements. It includes the camera placement, distortion correction of the separate cameras and calibration. Moreover, this system describes how the ball and end-effector are detected and how the info is processed to get an estimation of the current position of the objects.

3.2.1. Camera placement.

The first part of the camera system is the placement of the cameras. For the system, 2 cameras (Logitech Brio) are used. There are many different options on how the cameras can be placed. A few things should be noted.

The two cameras are used together to get an 3D position estimation of the ball and end-effector. If the two cameras are places close to each other, the depth estimation can become more uncertain. A small baseline, leads to small disparity (as baseline and disparity are proportional), and as depth is inversely proportional to disparity, the depth estimation becomes more uncertain. Therefore, it is desired to place the camera's far apart from each other.

Another aspect that needs to be considered for the camera placement is occlusion. As the robot arm moves, it is possible that the arm blocks the view of one camera, which as a consequence can not detect the ball. On the other end, it is also possible that the object to grasp or the human hand is blocking the view, making it not possible for the camera to detect the pose of the arm.

Different camera placements can be chosen to minimize the occlusion and maximize the (depth) accuracy of the detected element. Within this research it was was chosen to use two cameras beside each other. They were positioned behind the arm, at approximately the hight of the gripper. Figure 3 shows the setup. As a result the cameras can capture both the object in the hand as well as the end-effector. The exact placement of the cameras is not needed to be known, as the camera system is calibrated. This makes it easier to modify the camera placement if needed.

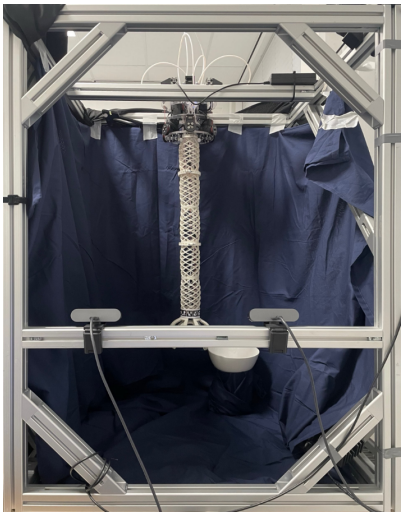


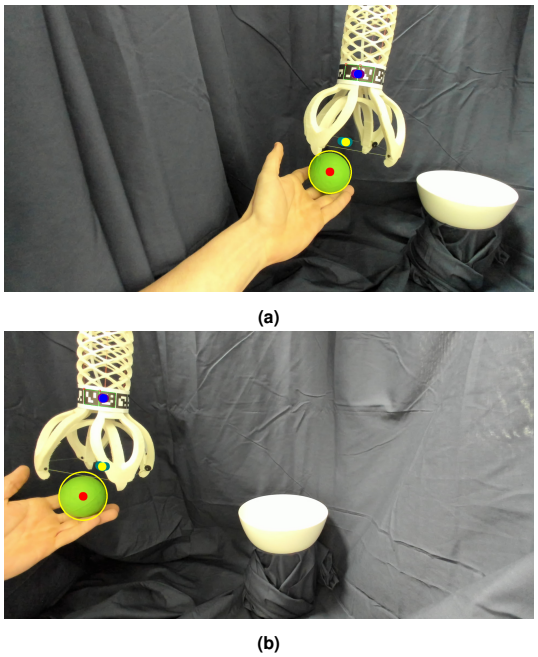
Fig. 3. The setup. Picture from behind.

3.2.2. Duo camera calibration.

After the camera images are undistorted, the stereo camera setup is calibrated to find the 3D position based on the pixel positions in the camera frames. This is executed with a calibration box, placed at a specific position. 6 corners, of which the 3D position is known in a specified world frame are selected in the two camera images. Using 'cv2.solvePnP()' the transform from the world frame to the camera frame can be found. Together with the intrinsic camera matrix this leads to a transform from the world frame to the pixel frame. Using these transforms it is also possible to find the 3D location of a point based on two pixel values. The first step is to represent both pixel values in their own camera pixel frame as a line in the 3D world frame space. This can be done by using the obtained transforms. Secondly, the intersection of these two lines can be used as the 3D location. As it is generally not the case that these two lines meet at one point, the algorithm finds the point that is closest to both lines. It is assumed that this point is the location of the object of interest.

3.2.3. Ball detection.

As mentioned, the object that will be used for the handover is a green colored ball. The color was chosen to make the detection easier. The ball detection consists of a few steps. First, the image is blurred, which smoothens the image. As a result, adjacent pixels will more likely be classified the same in the next step. In this next step, the images are filtered to only select pixels in which the HSV values are within some bounds. The bounds have been chosen so the green color of the ball is within these bounds.



(a)

(b)

Fig. 4. A typical image of how the cameras detect the ball and end-effector. Image (a) shows the result of the left camera, image (b) the right camera. The detected ball is indicated with the red dot at the center and a yellow circle around it. The dark blue indicates the middle of the end of the third segment and the yellow point indicate the gripper position.

The resulting image is a black and white mask. Next, the contours of this image are found and used to find the minimum enclosing circle. This leads to the most likely place and radius (in pixels) of the detected ball based on the single image. Using the algorithm discussed in section 3.2.2 this leads to a 3D position estimation of the ball. Figure 4 shows an example of how the ball is detected in the separate images.

3.2.4. End-effector detection.

The end-effector is detected using Aruco markers. A ring of Aruco markers (20mm) is places around the end of the third segment, denoted as the wrist. The cameras detect both the position and the direction of these markers. The detected position of these markers is shifted along their own z-axis to get an estimation of the middle point of the wrist. In addition, these positions can be shifted along their own y-axis to get an estimation of the gripper location, which is located 0.1 meters apart from the wrist.

It is important to note that for the detection of the Aruco markers only one camera is needed, as the size of the markers is known. Both cameras are, however, still used to detect the markers. The detected position of the wrist is based on the average of all separately estimated locations. The direction estimation is simply the average of the detected directions. Together with the position of the wrist, this direction is used to get the estimation of the position of the gripper. Figure 4 shows an example of how the Aruco markers are detected and how this leads to an estimation of the gripper position.

The measured gripper pose X_{meas} can therefore be described as in Equation 12, where the first and second part of the pose represent the 3D wrist location and the 3D gripper location, respectively. These two locations must stick to Equation 13, as the gripper has a fixed length.

$$X_{meas} = \begin{bmatrix} x_{\text{wrist}} & y_{\text{wrist}} & z_{\text{wrist}} & x_{\text{grip}} & y_{\text{grip}} & z_{\text{grip}} \end{bmatrix}^T \quad (12)$$

$$\left\| \begin{bmatrix} x_{\text{wrist}} \\ y_{\text{wrist}} \\ z_{\text{wrist}} \end{bmatrix} - \begin{bmatrix} x_{\text{grip}} \\ y_{\text{grip}} \\ z_{\text{grip}} \end{bmatrix} \right\|^2 = d_{\text{gripper}} \quad (13)$$

3.2.5. Filter.

So far, we have described how the ball and end-effector are detected. It should however be noted that it can occur that one or both of these are not detected at a time instance. On the other hand there will always be some uncertainty in the position estimations.

Therefore, a Kalman filter has been added for the position estimation of the ball, wrist and vector from the wrist to the gripper. In these three filters, an new incoming measurements (X_{new}) is filtered based on the estimated measurement uncertainty (R) and the model uncertainty (Q), in order to get an unbiased estimation of the state X with minimum variance P. This is obtained by using the Kalman gain K_{kalman}. The Kalman filter contains two updates:

- During the prediction update, a model is used to predict the next state X based on the current state. As no specific movement is expected from the ball nor from the gripper, the prediction of the next state in this research is just the same as the current state. The variance (P) of the state, however, will slightly increase as the system can move between the current step and the next, and therefore the uncertainty of the state increases. The prediction update is shown in Equation 14, where Q represents the model uncertainty.

$$\begin{aligned} X &\leftarrow X \\ P &\leftarrow P + Q \end{aligned} \tag{14}$$

- The second update is the measurement update. When a new measurement X_{new} comes in, the state X and variance P are updated based on the Kalman gain K_{kalman}. This gain specifies the weight given to both the current state estimation X and the new measurement X_{new} based on their uncertainties (P and R, respectively). The calculation of the measurement update is shown in Equation 15.

$$\begin{aligned} K_{kalman} &\leftarrow P(P+R)^{-1} \\ X &\leftarrow X + K_{kalman}(X_{new} - X) \\ P &\leftarrow P - K_{kalman}P \end{aligned} \tag{15}$$

The position estimations that come out of the Kalman filter (for the ball, wrist and gripper) are used in the rest of the algorithm. For future references, this Kalman filter is denoted as *filter*.

3.3. Control architecture.

Apart from the detection, the control architecture is build up with two main components:

- A state estimation and update
- A Jacobian based control

Both parts are crucial for the system, as will be discussed in this section.

3.3.1. State estimation.

With the cameras and object detection in place, the next step is to estimate the actual pose of the robot arm. This is not needed in the general robotics case. For many rigid robots the state and endpoint position can be accurately obtained if the actuator inputs are known.

This is however not generally the case with soft robots. If all tendons are under tension, the lengths of these tendons can give a fairly good indication of the system state and the gripper pose, using forward kinematics. The gripper pose X can simply be computed with $X = f(q)$, where f represents the forward kinematics of the system, discussed in Section 2.1.2.

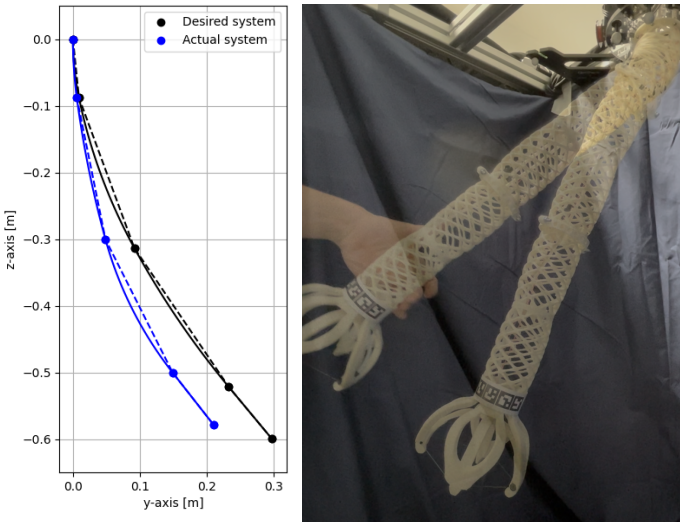


Fig. 5. Illustrations of the difference between the desired (model-based) state and the actual state of the system. (a) shows a measurement, where the black graph illustrates the model based system and the blue dots represents the measured data points based on a motion capture system. The dots (connected by the linear dashed lines) represent the segment links and the curve shows the PCC model. (b) shows a illustrative example of this type of behavior on the Helix. The arm that is supported by the hand is in an configuration where all tendons are under tension (and therefore a representation of the internal model), whereas the other arm shows how the system deforms without support. In this configuration not all tendons are under tension.

Due to gravity, however, the arms will tend to fall down. As the tendons can only exert tensile forces and no compressive forces, the arm will fall down at the locations where the tendons are under compressive load. In these cases the forward kinematics is no longer an adequate representation of the system.

When the internal system state q_{sys} is then still used, the gripper pose based on the model X_{sys} and the actual (measured) pose X_{meas} will be completely different. This is shown in Figure 5 with both a graph and an illustration with the Helix. Using the model-based pose in a Jacobian based control approach will not be effective as the Jacobian will be evaluated at a pose that is different from the actual pose.

With this paper we propose that this problem can be solved by adding an additional state representation which is based on the observed gripper pose. With this method the system can be described with two states. q_{sys} represents the internal system state. This is used in the model to compute the tendon lengths. Consequently, this state representation should adhere to the system limits as described in Section 3.1.1.

The second state, q_{est} represents the estimation of the system state based on how the arm is observed by the cameras. When this state is used in the forward kinematics, the resulting estimated gripper pose $f(q_{est,n}) = X_{est,n}$ should be close to the actual gripper pose.

The pose estimation is done based on a weighted and constraint squared error optimization function as shown in Equation 16. The function is adressed as the state optimization function (SOF) for later references. The following optimization is done at every timestep n:

$$\begin{aligned} \text{SOF:} \quad \min_{q_{est,n}} \quad & \|A_{pose,upd}(q_{est,n} - q_{sys,n})\|^2 \\ \text{s.t.} \quad & \|f(q_{est,n}) - X_{meas,n}\|^2 \leq \epsilon_{pose,bound} \end{aligned} \tag{16}$$

As can be seen, the optimization function find the state $q_{est,n}$ which minimizes the difference between the system state $q_{sys,n}$ and the state estimation $q_{est,n}$. $A_{pose,upd}$ is a (9x9) scaling matrix, indicating how important it is for different parts of the state estimation to be close to the system state. Within this research this was chosen to be a diagonal matrix. As the higher

segments have more weight hanging on them, their curvatures are more influenced by the gravity. As a consequence their weight in $A_{pose,upd}$ is lower than for the lower segments.

The constraint makes sure that the gripper pose of the resulting state is close to the estimated actual gripper pose, where $\epsilon_{pose,bound}$ represents the bound on the difference between the measured gripper pose $X_{meas,n}$ and the estimation of the gripper pose based on the froward kinematics $f(q_{est,n})$.

The found state estimation $q_{est,n}$ will be used within the controller. This is discussed in the next section.

3.3.2. Jacobian based control.

Jacobian based control is a widely used control method. Based on the error of the end effector pose X and the Jacobian $J(q)$, the change of the state Δq is computed as $\Delta q = J^{-1}(q)X_{error}$, where the state error is the difference between the goal state and the measured current state ($X_{error} = X_{goal} - X_{meas}$). As the inverse of the Jacobian does not exist in the general case with a Jacobian that is not generally squared, the pseudo inverse $J^{+}(q)$ can be used, where $J^{+} = J^{T}(JJ^{T})^{-1}$.

The state is updated by adding this Δq to the previous state q. In the general case this is done with some scaling factor α, with $\alpha \in (0, 1]$.

$$q_{n+1} = q_n + \alpha J^{+}(q_n)X_{error} \tag{17}$$

In the case of the control of the Helix robot, this leads to a control as shown in Equation 18, where both the new system state $q_{sys,n+1}$ and the estimated state $q_{est,n+1}$ are computed. The Jacobian that is used, is calculated with the obtained estimation of the current state q_{est}, as this gives the best representation of the actual current state of the system.

$$\begin{aligned} q_{est,n+1} &= q_{est,n} + \alpha J^{+}(q_{est,n})(X_{goal,n} - X_{meas,n}) \\ q_{sys,n+1} &= q_{sys,n} + \alpha J^{+}(q_{est,n})(X_{goal,n} - X_{meas,n}) \end{aligned} \tag{18}$$

Although this is a relatively simple way to compute the new system state q_{sys}, there is no guarantee that the system stays within the system limits.

3.3.3. Jacobian based control in loop.

One way to make the system stay within the system limits, is by using the weighted pseudo-inverse matrix (19) within a loop. This is based on previous work in the research group. Here, K is a (9x9) matrix, which represents a virtual stiffness of the system. Normally, this is a diagonal matrix. If the first three diagonal values are higher than the others, this indicates that the first segment is relatively stiffer than the other two segments. As a consequence the change in tendon lengths for the first segment will be relatively smaller than for the other two segments. The new state will be computed similarly, as can be seen in Equation 20.

$$J_K^{+} = K^{-1}J^{T}(JK^{-1}J^{T})^{-1} \tag{19}$$

$$\begin{aligned} q_{est,n+1} &= q_{est,n} + \alpha J_K^{+}(q_{est,n})(X_{goal,n} - X_{meas,n}) \\ q_{sys,n+1} &= q_{sys,n} + \alpha J_K^{+}(q_{est,n})(X_{goal,n} - X_{meas,n}) \end{aligned} \tag{20}$$

To find a suitable new state which reaches the goal and adheres to the system limits, these equations can be used in a loop, while the virtual stiffness matrix K is varied.

As the step size α is normally smaller than 1, this step size basically shifts the goal location closer to the current location. When used in a loop, however, this goal location is shifted to a different new location every iteration of the loop. To prevent this, it is needed to get α out of the loop, which leads to an update of the desired gripper pose X_{goal} based on the step α. This leads to $X_{goal,\alpha}$, as shown in Equation 21. Equation 20 can then be written as in Equation 22. This can be used in the control algorithm.

$$X_{goal,\alpha,n} = \alpha X_{goal,n} + (1 - \alpha)X_{meas,n} \tag{21}$$

$$\begin{aligned} q_{est,n+1} &= q_{est,n} + J_K^{+}(q_{est,n})(X_{goal,\alpha,n} - X_{meas,n}) \\ q_{sys,n+1} &= q_{sys,n} + J_K^{+}(q_{est,n})(X_{goal,\alpha,n} - X_{meas,n}) \end{aligned} \tag{22}$$

The control loop is shown in Algorithm 1, where its called *JBC* for future references. First, the state representations of the system q_{sys} and the estimated model q_{est} are initialized, as well as the virtual stiffness matrix K, the goal pose and measured gripper pose. Before the system enters the first loop, the corrected goal pose $X_{goal,\alpha}$ is computed, followed by the error e.

When the system enters the loop (line 5), the desired change in state vector Δq_J is computed using the weighted pseudo inverse of the Jacobian J_K^{+} and the error e. This is added to both the state estimation of the model q_{est} and the state of the system q_{sys} to obtain the constraint free desired states.

In line 12, the newly found desired system state $q_{sys,J}$ is compared with the constraints. This function is named *SKU* and its pseudo-code is shown in Algorithm 2. If the state is within bounds, nothing changes (so $q_{sys} \leftarrow q_{sys,J}$. If $q_{sys,J}$ is not within bounds, the parameters are updated:

- If the length of one of the segments is out of bound, this length is cropped to fit within the boundaries. Next the corresponding index on the diagonal inverse stiffness matrix K^{-1} is divided by 100. In the next iteration, this segment will be considered much stiffer along the longitudinal direction. In Algorithm 2, the subscript l denotes that the code only uses the indices that represent a length.

- If the angle of one of the segments is too big, the code checks which element of the angle ($\Delta_x = Dx$ or $\Delta_y = Dy$) is the biggest. This one is cropped to make sure the total angle is cropped to fit within the boundaries. Next the corresponding index on the diagonal inverse stiffness matrix K^{-1} is divided by 100. In the next iteration, this segment will be considered much stiffer along this angular direction. In Algorithm 2, the subscript D denotes that the code only uses the indices that represent segment rotations.

In line 13 of Algorithm 1, the change in the system state as the result of the limits is computed. Based on this the state estimation q_{est} is updated as well. The computed new state estimation is used in the forward kinematics model to find its corresponding end effector pose X_{est}, which is used to determine the next estimated error e compared to the goal location $X_{goal,\alpha}$.

This code loops till the found error is within some bounds $\epsilon_{control}$, or till the the maximum iteration $N_{max,iter}$ is reached. In this last case the system did not find a new system state and corresponding estimation state that adheres to the system limits and have a small enough error from the goal position. Because the error is computed based on the estimation of the state, it is important that the model evaluated at the state estimation q_{est} is an accurate representation of the actual pose ($X_{meas} \approx X_{est}$). With a more accurate state estimation (and a more accurate model), the resulting control action will be more effective.

3.4. Integrated system.

In order to perform the handover tasks, all previously discussed methods have been implemented into one control system. This is shown in Algorithm 3. First, all parameters are initialized, for both the gripper pose estimation of the gripper and ball as well as for the system state and state estimation.

Algorithm 1 JBC: Jacobian based control approach with virtual stiffness

$q_{est} \leftarrow q_{est,cur}$
$q_{sys} \leftarrow q_{sys,cur}$
$K^{-1} \leftarrow K^{-1}_{initialize}$
$X_{goal} \leftarrow X_{goal,initialize}$
$X_{meas} \leftarrow X_{meas,initialize}$
$X_{goal,\alpha} \leftarrow \alpha X_{goal} + (1-\alpha) X_{meas}$
$e \leftarrow X_{goal,\alpha} - X_{meas}$
for $m = 1, \ldots, N_{max,iter}$ **do**
  $\Delta q_J \leftarrow J^{+}_{K^{-1}}(q_{est}, K^{-1}) e$
  $q_{est,J} \leftarrow q_{est} + \Delta q_J$
  $q_{sys,J} \leftarrow q_{sys} + \Delta q_J$
  $q_{sys} \leftarrow \mathrm{SKU}(q_{sys,J}, K^{-1})$
  $\Delta q_L \leftarrow q_{sys} - q_{sys,J}$
  $q_{est} \leftarrow q_{est,J} + \Delta q_L$
  $X_{est} \leftarrow f(q_{est})$
  $e \leftarrow X_{goal,\alpha} - X_{est}$
  if $\|e\|^2 < \epsilon_{control}$ **then**
    $q_{est,new} \leftarrow q_{est}$
    $q_{sys,new} \leftarrow q_{sys}$
    return $q_{est,new}, q_{sys,new}$

Algorithm 2 SKU: State and stiffness update based on limits

$q_{sys} \leftarrow q_{sys,initialize}$
$K^{-1} \leftarrow K^{-1}_{initialize}$
$A_{q,q_{DL}} \leftarrow A_{q,q_{DL},initialize}$
$q_{DL} \leftarrow \sqrt{A_{q,q_{DL}} q^2_{sys}}$
if $L_{sys,LB} \leq q_{DL} \leq L_{sys,UB}$ **then**
  continue
else
  for index i with instance $q_{sys,l,i}$ in $q_{sys,l}$ **do**
    if $q_{DL,l,i} < L_{sys,LB,l,i}$ or $q_{DL,l,i} > L_{sys,UB,l,i}$ **then**
      $q_{sys,l,i} \leftarrow \mathrm{crop}(q_{sys,l,i}, (L_{sys,LB,l,i}, L_{sys,UB,l,i}))$
      $K^{-1}_{l,i} \leftarrow K^{-1}_{l,i}/100$
  for index i with instance $q_{DL,D,i}$ in $q_{DL,D}$ **do**
    if $q_{DL,D,i} > L_{sys,UB,D,i}$ **then**
      $q_{DL,D,i} \leftarrow \mathrm{crop}(q_{DL,D,i}, (0, L_{sys,UB,D,i}))$
      if $q_{sys,Dx,i} > q_{sys,Dy,i}$ **then**
        $q_{sys,Dx,i} \leftarrow \sqrt{q^2_{DL,D,i} - q^2_{sys,Dy,i}}$
        $K^{-1}_{Dx,i} \leftarrow K^{-1}_{Dx,i}/100$
      else
        $q_{sys,Dy,i} \leftarrow \sqrt{q^2_{DL,D,i} - q^2_{sys,Dx,i}}$
        $K^{-1}_{Dy,i} \leftarrow K^{-1}_{Dy,i}/100$
return q_{sys}

Algorithm 3 Integrated system

$X_{meas} \leftarrow X_{meas,init}$
$X_{ball,meas} \leftarrow X_{ball,meas,init}$
$q_{est} \leftarrow q_{est,init}$
$q_{sys} \leftarrow q_{sys,init}$
while $e > \epsilon_{threshold}$ **do**
  $X_{meas} \leftarrow filter(X_{meas}, X_{meas,new})$
  $X_{ball,meas} \leftarrow filter(X_{ball,meas} X_{ball,meas,new})$
  $e \leftarrow \|X_{ball,meas} - X_{meas}\|^2$
  if $e < \epsilon_{threshold}$ **then**
    break
  $q_{est} \leftarrow \mathrm{SOF}(q_{sys}, X_{meas})$
  $q_{sys}, q_{est,new} \leftarrow \mathrm{JBC}(q_{sys}, q_{est})$
  $l_{t,sys} \leftarrow g(q_{sys})$
  Publish on ROS-topic: $\Delta l_{t,sys} = l_{t,sys} - l_{t,calib}$
Close gripper

In the main loop, the system first updates the measured position of both the gripper and the ball. This follows the procedure of Section 3.2. The filtered measurement of the gripper position is used to make a new estimation of the state q_{est} (as discussed in Section 3.3.1). This is used within the controller to find the desired new system state q_{sys} and corresponding new state estimation q_{est} (Section 3.3.3). Using the Helix model, this is transformed to the desired new tendon lengths.
The difference between these lengths and the calibration lengths are then published on the specific ROS topic, where the Raspberry Pi 5 transforms this into the motor rotations, as discussed in Section 3.1. The main loop stops when the error between the desired gripper pose and the measured gripper pose is smaller than a set threshold. As can be seen in Algorithm 3, this error is specified as the l_2-norm between the two gripper pose vectors. The gripper closes by publishing the tendon retraction to the specific gripper-topic.

3.5. Experiments.

To investigate the performance of the proposed control architecture, three experiments has been executed. These consist of 2 qualitative experiments focused on the handover and one experiment to demonstrate a simple use case in which a handover is performed after which the arm moves to a desired location.

For the experiments, the proposed algorithm has been implemented in Python. During the experiments, the Helix robot starts in its calibrated position, hanging down. A human hand holds the ball in the camera frames. When the operator starts the algorithm, the robot arm will move. As explained in Section 3.5, the arm-movement stops when the error becomes small enough. After that, the gripper closes, where the amount of retraction was predefined and took 10 seconds.

In order to make the actual handover possible, two decisions have been made. First of all, it was specified that the Helix should grip the ball from above. The orientation of the gripper was therefore fixed during the execution of the algorithm. Secondly, the control towards the ball was done in two steps. First the desired location was set to be $d_{\mathrm{grasp-offset}} = 0.05$ meter on top of the ball. When the l_2-norm error was below the threshold ($\epsilon_{threshold} = 0.05$ m), the next desired location was set to be the location of the ball. This two step control strategy was implemented to make sure the gripper does not bump into the object to grasp, but approaches the object from above.

3.5.1. Experiment 1: Handovers.

The first and main experiment has been performed to test the automated human-to-robot handover. It has been performed with three different values for the step size α. These were set to $\alpha = [0.1, 0.2, 0.4]$. The bigger α is, the bigger the change in tendon lengths will be. This means that the system will probably be at the desired location within a shorter time. On the other had, the Jacobian will be evaluated fewer times, which can possible lead to overshoot of the system.

For every condition of α, 8 runs of the system have been executed. The human hand (with ball) was held at 4 different locations, with two runs per location. The locations are, observed from the camera looking toward the Helix: front, front-left, left & back-left. As an actual human hand was holding the ball, the hand (and so object) locations were never completely the same. As there are 8 measurements per condition, this can however still give valuable results.

All acquired data (including images and calculated variables) is stored per timestep.

3.5.2. Experiment 2: Human guidance.

In addition to the fully automated handovers, a second experiment was added to see if and how the implemented system could cooperate with a human to perform the task.

The basis of the experiment was the same as with the automated handovers (in the first experiment). The only difference was that a human was now involved in the handover task, with the goal to guide the arm to the desired position.

This experiment has been performed with 2 step sizes $\alpha = [0.1, 0.2]$ and with 2 runs of the system for both of the conditions.

3.5.3. Experiment 3: Use case demonstration.

Lastly, experiment 3 was performed to show a use case, where the handover is only a part of a task. The use case was simple: Start form the initial position, perform the human-to-robot handover, move the object to a specified location (where a bowl is located) and drop the object into the bowl.

As the bowl was put at a specified hard-coded location, no additional camera detection was needed. Therefore, the proposed and implemented code would suffice for this experiment. This experiment has been performed eight times with step size $\alpha = 0.1$.

4. Results

This section shows the results of the experiments. A discussion of the result will follow in the next section.

As mentioned in Section 3.5.1, there are 8 measurements per condition for the first experiment. As each measurement has the goal position at a different location, the results will focus on the l_2-norm of the obtained data, as this not only maps the 6-dimensional pose X toward a single number which can easily be plotted. It also takes away the directional component, which is not needed as the system performance should not be dependent on the radial angle.

It should be noted that all 24 runs completed the handover task successful. Figure 6 shows the evolution of one of these runs.

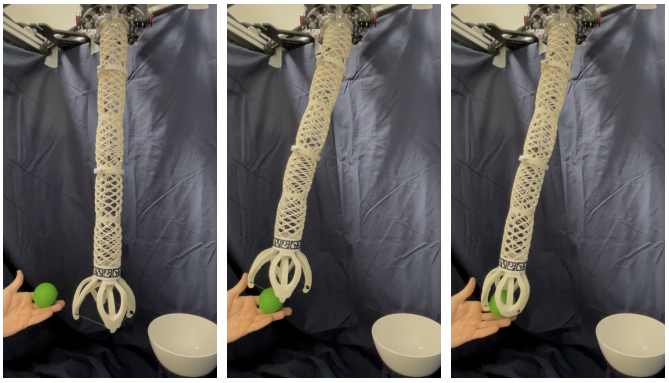


Fig. 6. The evolution of one of the performed handover tasks. The first image shows the starting position, the second shows the system where the gripper is just above the ball, and the last shows the system when the handover task is just finished.

As the measurements are uncertain, the discussed results will focus on the average values of those measurements. The length of these average values will be the median length of the measurements. Figure 7 shows how the average of the 8 measurements is obtained for the measured error (with $\alpha = 0.1$). As can be seen, there is one run that took much longer than the others. Because the median length is used for the average line, outliers have less influence on the resulting time.

4.1. Comparison of different step size conditions.

The average of the measured errors for the three conditions of the step size α over time, are shown in Figure 8. It should be noted that these graphs represent the time for the motion of the Helix arm to the desired position. As mentioned in Section 3.5, closing the gripper takes 10 seconds. As the physical handover is only finished when the gripper is closed, these 10 seconds need to be added to the time for moving the arm in order to find the total handover-time. Table 1 shows the median and average time needed to move the Helix arm to the correct position. The measurement with the fastest (minimum) and slowest (maximum) time are also shown, as well as the average (and standard deviation) of the timestep length.

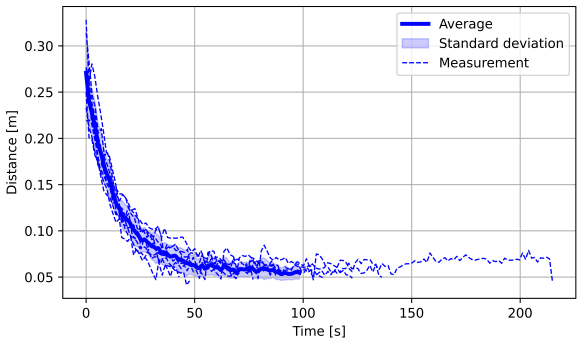


Fig. 7. The single measurements and the average and standard deviation of the error between the measured and desired position for step size $\alpha = 0.1$. The length of the average ling is the median of the 8 runs.

Table 1. The time (in seconds) for the arm movement with the different step sizes α. The minimum and maximum time of the 8 different runs are presented, as well as the mean and median. All minimum, maximum and median values have an uncertainty of 0.01 seconds. It should be noted that the time for the handover task takes 10 seconds longer, as this was the time needed to grasp the object. The average timestep duration is shown in the last row.

α [-]	0.1	0.2	0.4
Minimum	42.9	28.4	14.6
Median	98.3	39.6	22.3
Maximum	214.9	57.6	29.5
Mean	104.8 ($\pm$ 52.0)	42.2 ($\pm$ 10.8)	22.4 ($\pm$ 4.3)
Timestep	1.38 ($\pm$ 0.02)	1.51 ($\pm$ 0.18)	1.45 ($\pm$ 0.05)

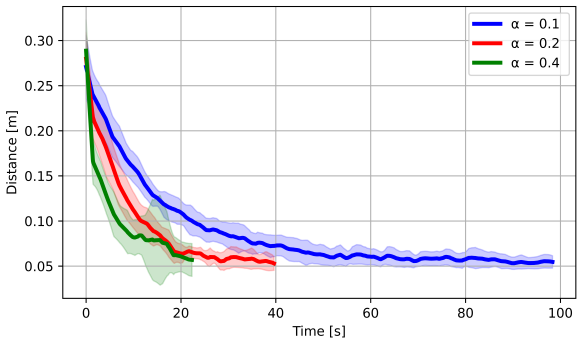


Fig. 8. The averages and standard deviation of the error between the measured and desired position for step sizes $\alpha = [0.1, 0.2, 0.4]$. The length is the median of the 8 runs.

4.2. Comparison of measurements with system model state and state estimation.

The pose estimation discussed in Section 3.3.1 is one of the main elements in the control architecture. This is based on the idea that the internal system model does not provide an accurate estimation of the actual position of the end-effector. To check this, the model based end-effector position and the (filtered) measured position using $\alpha = 0.1$ are compared in Figure 9. The shown distance represents the distance between the 3D gripper position $[x_{\text{grip}}, y_{\text{grip}}, z_{\text{grip}}]$ with respect to the calibrated starting position $[0, 0, -0.7]$. The shown error is the distance between the model based position and the measured position. The results using different step sizes α are very similar, as can be seen in the appendix.

To see the effect of the state estimation, Figure 10 shows

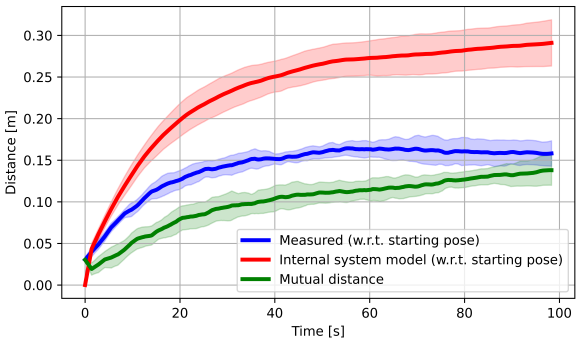


Fig. 9. The averages and standard deviation of the error between the measured and desired position (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.1$.

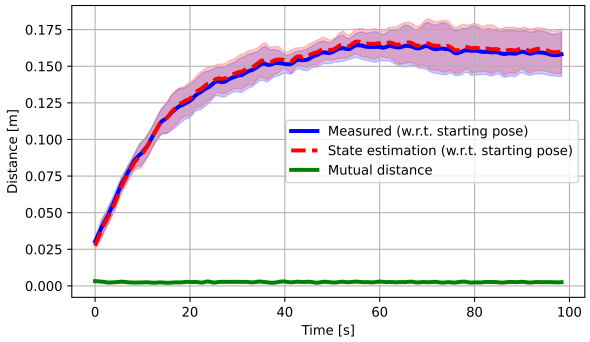


Fig. 10. The average and standard deviation of the norm of the gripper location with respect to the calibrated starting position using the state estimation in the model (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.1$.

the gripper position when the state estimation is used in the model. This is again compared with the measured values to get the error of the state estimation. Again, other step sizes lead to similar results.

4.3. Comparison between automated and human guided handovers.

When considering the experiments in which a human was present to guide the Helix arm towards the desired position, it should be noted that all 4 runs successfully completed the handover task. Figure 11 shows the evolution of one of these runs with human guidance. It can be seen that the human only guided the robot after the system initialization.

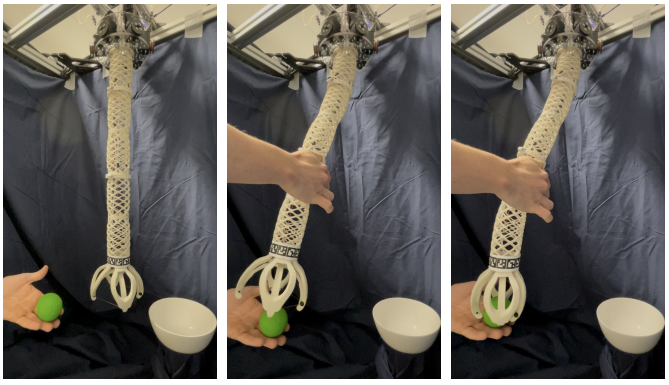


Fig. 11. The evolution of one of the performed handover tasks with human guidance. The first image shows the starting position, the second shows the system when the human has guided the arm towards the ball, and the last shows the system when the handover task is just finished.

Figure 12 shows the comparison between the automated handovers and the two runs where the handover was guided by

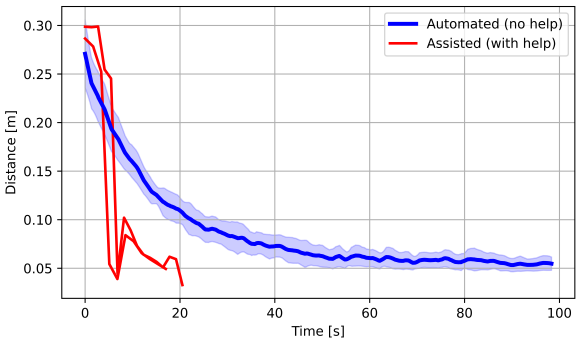


Fig. 12. The averages and standard deviation of the error between the measured and desired position for the 8 runs without human guidance (blue) and 2 runs with human guidance (red). The used step size $\alpha = 0.1$.

a human. As can be seen, both runs with the human guided handover are very similar and lead to a handover. that is significantly faster than the fully automated cases. A step size of $\alpha = 0.2$ leads to similar results, as can also be seen in the appendix.

4.4. Use case.

Figure 13 gives an impression on how the system perform the demonstration discussed in Section 3.2.3. It can be seen that the ball was taken from the hand and dropped in the bowl. For all 8 runs, the ball was taken from the hand and correctly dropped into the bowl.

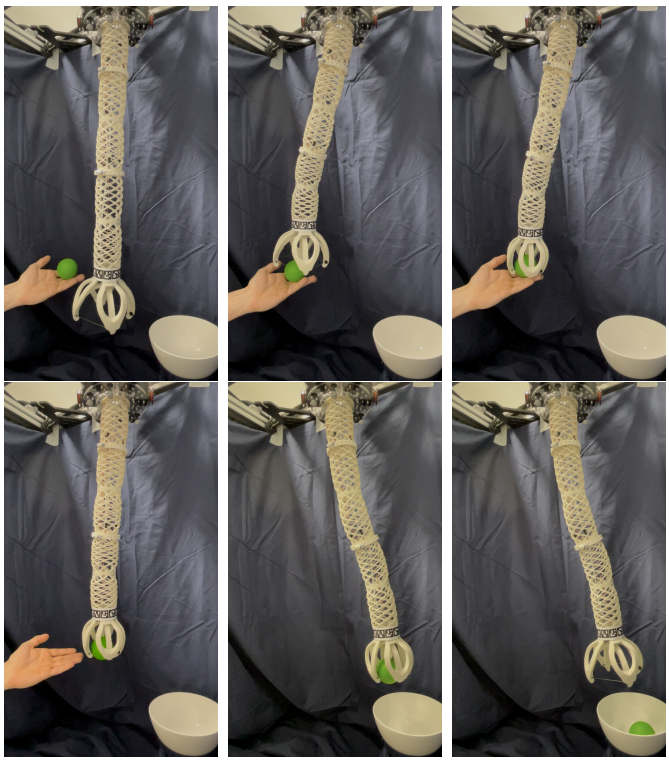


Fig. 13. The evolution of one of the performed use case tasks. The first image shows the starting position, the second and third show the handover task. The fourth and fifth image show the movement of the arm (with ball) towards to desired location. The last image shows the system when the task is just finished.

5. Discussion

In this section the results of the previous section will be discussed. After this, the non-idealities will be discussed and some improvements will be suggested.

5.1. Comparison of different step size conditions.

When Figure 8 is considered, a few things can be noticed. First of all, it can be seen that for all step sizes α the shape of the decent in error is similar. The slope of the decrease in error is the highest with the biggest error and gradually decreases as the gripper comes closer to the object to grasp. This is a logical behavior for Jacobian based control, as the control action is proportional to the error distance.

Secondly, it can be seen in both the figure and in Table 1 that the handover time decreases with an increasing step size α. This is in line with the expectation. It should be noted that the average errors start with a relatively smooth decrease, followed by a rougher decrease. This can be partially explained by the two step control strategy, as the error to the goal suddenly increases slightly with the transition from the first goal position (with the $d_{\text{grasp}-\text{offset}}$) to the second goal position (without the offset). Another aspect that plays a role in the rough behavior are the uncertainties present in the measurements of both the gripper pose and the ball.

It was expected that increasing the step size from 0.1 to 0.2 and from 0.2 to 0.4 would lead to a decrease in completion time by half. When comparing the different configurations, it can indeed be seen that the arm motion takes roughly twice as long if the step size is halved. This time is slightly shorter when comparing $\alpha = 0.2$ with $\alpha = 0.4$, but slightly higher when comparing $\alpha = 0.1$ with $\alpha = 0.2$. This shows that the control algorithm is not ideal, which is probably caused by the set of inaccuracies present in the model, the state estimation, and the measurements all together.

5.2. Comparison of measurements with system model state and state estimation.

When Figure 9 is considered, it can be seen that there is a big mismatch between the measured gripper location and the gripper location based on the internal system model. At the start, the model assumes that the system is hanging straight (with an offset of 0) while the measurements show that the system does not actually start at the desired starting location.

As the arm moves outwards, towards the object to grasp, the difference between the system model and the measured values increases. At the final position, when the gripper X_{meas} is about 16 cm away from the calibrated staring position, the internal model based gripper position X_{sys} is 29 cm away form the starting position, 14 cm away form the measured position. This behavior caused by gravity on the system, leading to a loss of tension in some tendons, was already discussed in Section 3.3.1.

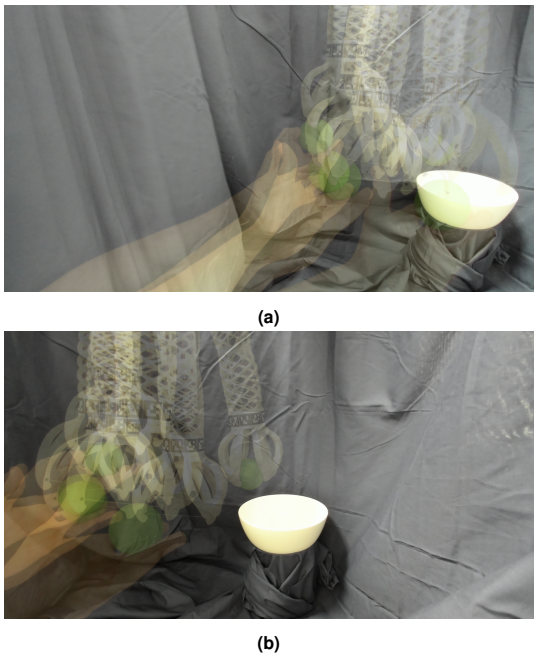
All together, Figure 9 clearly shows that using the systems internal model as assumption for the actual system behavior is not even close to a sufficient representation, when using the system in the whole workspace.

When the state estimation as discussed in Section 3.3.1 is used, and the resulting state is used in the model, this leads to a much more accurate representation of the measurements as can be seen in Figure 10. It can be seen that the error between the estimated state model and the measurements is generally very small. When the control error (X_{error}) is considered, this is always (for all conditions and all timesteps) equal to the bound $\epsilon_{pose,bound}$. This is not surprising based on how the optimization function was defined. As the error between the internal system model estimation X_{sys} and the measurements X_{meas} is always bigger that the chosen bound $\epsilon_{pose,bound}$, the constraint 'pushes' the estimation towards the measurements, but not further than the set bound.

5.3. Comparison between automated and human guided handovers.

As was shown in Section 4.3, the handovers with human guidance were significantly faster than the fully automated handovers. This is not surprising, as the fastest automated handover took 14.6 seconds, while a healthy human can do a similar task much faster. Therefore, having a human in the loop that guides to arm towards the desired position can significantly increase the handover time, as the implemented control algorithm only has to add small tendon length changes to move within the bounds of the desired position.

This experiment also shows that the implemented control algorithm is an adaptive system as it can handle unexpected and uncertain interactions with a human in the loop. This is a useful property as many systems will operate in an unknown and uncertain environment.



(a)

(b)

Fig. 14. An overview of the approximate workspace in the camera spaces. Image (a) shows the result of the left camera, image (b) the right camera.

5.4. Improvements.

When analyzing both the algorithm and the results, a few improvements can be suggested, ranging form the computer vision to the control architecture.

5.4.1. Camera and camera placement.

Within the used setup, the two cameras were placed behind the robot arm with an approximate distance of about 50 cm. For the algorithm it is needed that both the Aruco markers on top of the gripper and the object to grasp are visible by the cameras. In the current setup the handover workspace was not limited by the robot workspace, but by the space that was captured by the two cameras. As can be seen in Figure 14, the workspace was limited on the left side by the right camera and vice versa. Moving more in either one of these directions would suggest that one camera could not detect the scene. A lot of motion in the upward and downward direction was also limited as both the Aruco markers and the ball needed to be detected.

This limited workspace can be improved with various methods. First of all, different cameras can be used with a bigger field of view. This way, a bigger workspace can be observed while the cameras can stay at approximately the same location. Secondly, the cameras can be placed at a bigger distance from the robot. A combination of those is also possible.

A closely related topic is the camera placement. In the used setup, both cameras were beside each other behind the robot arm. It can be investigated how the detection performances changes when other camera placements are used with for instance one of the cameras looking down from the top or one of the cameras observing the system from one of the sides. This

might lead to a better position estimation of the object as the depth can possibly be better observed than with two cameras beside each other. In these cases, however, the risk of occlusion should be considered.

Other possibilities in setup that can be investigated are either setups with depth cameras, where one camera could suffice. Using more than 2 cameras (in a redundant system) can help against the risk of occlusion, but probably needs more processing time.

5.4.2. Detection system.

When the computer vision is analyzed, a few improvements can be made.

First of all, the detection does not always works flawless. As can be seen in Figure 15, there are cases where the ball is partially occluded by the gripper, leading to an inaccurate detection. This will lead to an inaccurate ball position estimation and therefore makes the handover task more challenging.

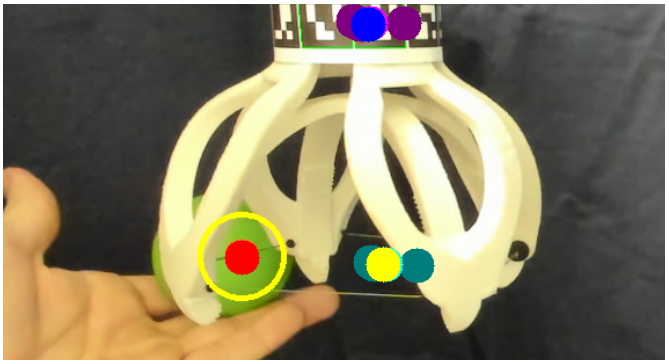


Fig. 15. An example of an inaccurate ball detection.

Secondly, the object to grasp was now specified to be a ball with a specific green color. The algorithm to detect this object uses this knowledge of both color and shape to detect the object. This thus means that the algorithm is very object specific. Using a computer vision model that is more general and does not need object specifics for a decent position estimation would significantly improve the overall performance towards real applications.

This is also related to the grasp angle. Within this report, the handover was simplified by prescribing that the ball was lying in the hand and that the desired orientation of the grasp would be from the top. In real world scenarios this is not generally the case as objects can have different sizes and shapes. An improved detection system could also provide the best grasp angle, which could be used by the control algorithm.

When considering the detection of the arm, the Helix arm was now observed with the use of Aruco markers. This is sufficient in a controlled environment, but will not always suffice outside of this environment. Therefore, a more robust detection system would be desired. This can include a more sophisticated detection algorithm that does not need any Aruco markers, but can also use different sensors in and/or on the Helix.

Lastly, the Kalman filter can be improved. Within the used system, the model and measurement uncertainties for both the gripper pose and the ball position were fixed. This can be improved by making the measurement uncertainties dependent on the estimation of these uncertainties, based on the detection algorithm. This way, an uncertain measurement will have less influence on the position estimation, and vice versa, leading to more accurate position estimations and therefore to more effective control and faster handovers.

5.4.3. Estimation method.

As discussed in Section 3.3.1, the constraint specifies that the distance between the measured gripper pose and pose computed with the the state estimation in the model should not be higher than a fixed value $\epsilon_{pose,bound}$. Apart from simplicity, there is however no good reason to set this as a constant value. Making this value dependent on the filtered measurement uncertainty will increase the accuracy of the system.

A similar situation holds for the diagonal scaling matrix $A_{pose,upd}$. The diagonal values of this matrix have been fixed by the programmer. This was set as a hyperparameter, but there might be values that lead to a better performance. The loss of tendon tension occurs to a bigger extent when the segment is longer (leading to less internal pressure to keep the tendons under tension) and when the segment and/or the segment(s) above are curved. Therefore, this scaling matrix can be made dependent on the internal system state q_{sys} to get a better representation of the actual system.

Is is also possible to use a completely different state estimation method, that is for instance completely computer vision based. Further research can investigate these options for soft robots.

5.4.4. Jacobian control.

As discussed in Section 5.1, the constraint Jacobian based control algorithm controls the soft robotic arm towards the goal position. This happened for all values of the step size α. During the experiments these step sizes were constant. In general a higher step size is useful when the error is big, since the goal is reached faster. As the error becomes smaller, however, a smaller step size can be desirable to prevent the risk of overshoot. In order to improve the overall performance for fast control and handover tasks, follow-up research can focus on a variable step size, which is dependent on the error.

5.4.5. Processing speed.

Lastly, it can be concluded that the processing speed of the whole system is not high, with each timestep taking more than a second. This should increase, which can be achieved in two ways. Firstly, a critical look on all the separate functions is needed as this can significantly improve the total algorithm speed. Secondly, the used hardware can be improved, as better hardware has a higher computational performances. Optimizing both the software and hardware can together result in a much faster control, which can make the interaction with humans much smoother and therefore potentially more pleasant.

6. Conclusion

Within this research, a control architecture to automated handovers with a soft robot using visual servoing has been proposed and tested. Although, the main focus was put on the pose estimation and control algorithm, the research also included the detection algorithm to detect both the robot arm and the object to grasp.

The two main contributions within the system that made the control accurate enough for automated handovers (given the detection algorithm) are firstly the optimization based state estimation, leading to a much better representation of the system, which could be used for the Jacobian based control. The second contribution was the constraint jacobian based control, where the state estimation was used within the jacobian, while the internal system state was used to fit in the constraints.

All together, the proposed control architecture was able to consequently perform the human-to-robot handover task for all tested jacobian step sizes $\alpha = [0.1, 0.2, 0.4]$, where the system was clearly the fastest with the highest step size. It was shown that the system can adaptively deal with a human that guides the robot towards the object in the hand, resulting in a faster handover. Lastly, a demonstration of a simple use case was shown that performed as intended.

As such, this research proposes a baseline for automated handovers with soft robots where further research can build upon

in order to make the handovers more general (with different objects and grasp techniques) and faster.

Acknowledgements

Firstly, I would like to thank my project supervisors Cosimo Della Santina, Azita Dabiri and Francesco Stella for their help, support and enthusiasm for the subject. The meetings we had were generally short, but very useful and effective.

Secondly, I would like to thank my friends for their support and the fun we had throughout my studies. Especially I would like to thank my wonderful girlfriend Anna. You are amazing! In addition to this, I would like to thank my pets (and the pets I have had) for their immense support for me as a person. Thirdly, I would like to thank my coffee machine, I could not have done this without that precious thing.

Mostly, I would like to express my gratitude to my parents André and Jet and my sister Amber. You three have known me my entire life and supported me all the way. I never could have achieved this without the three of you, and therefore I am immensely proud that I can share this master thesis with you.

References

[1] Alin Albu-Schaffer, Oliver Eiberger, Markus Grebenstein, Sami Haddadin, Christian Ott, Thomas Wimbock, Sebastian Wolf, and Gerd Hirzinger. Soft robotics. *IEEE Robotics & Automation Magazine*, 15(3):20–30, 2008.

[2] Marcello Calisti, Giacomo Picardi, and Cecilia Laschi. Fundamentals of soft robot locomotion. *Journal of The Royal Society Interface*, 14(130):20170101, 2017.

[3] Cosimo Della Santina, Antonio Bicchi, and Daniela Rus. On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control. *IEEE Robotics and Automation Letters*, 5(2):1001–1008, 2020.

[4] Markus Grebenstein. The dlr hand arm system - damping control and robustness, 2011.

[5] Markus Grebenstein, Alin Albu-Schäffer, Thomas Bahls, Maxime Chalon, Oliver Eiberger, Werner Friedl, Robin Gruber, Sami Haddadin, Ulrich Hagn, Robert Haslinger, et al. The dlr hand arm system. In *2011 IEEE International Conference on Robotics and Automation*, pages 3175–3182. IEEE, 2011.

[6] Qinghua Guan, Francesco Stella, Cosimo Della Santina, Jinsong Leng, and Josie Hughes. Trimmed helicoids: an architectured soft structure yielding soft robots with high precision, large workspace, and compliant interactions. *npj Robotics*, 1(1):4, 2023.

[7] Michael W Hannan and Ian D Walker. Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. *Journal of robotic systems*, 20(2):45–63, 2003.

[8] Bryan A Jones and Ian D Walker. Kinematics for multisection continuum robots. *IEEE Transactions on Robotics*, 22(1):43–55, 2006.

[9] Robert K Katzschmann, Cosimo Della Santina, Yasunori Toshimitsu, Antonio Bicchi, and Daniela Rus. Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 454–461. IEEE, 2019.

[10] KUKA. Industriële robots van kuka, 2024.

[11] Georgios Methenitis, Daniel Hennes, Dario Izzo, and Arnoud Visser. Novelty search for soft robotic space exploration. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 193–200, 2015.

[12] Jordi Pages, Luca Marchionni, and Francesco Ferro. Tiago: the modular robot that adapts to different research needs. In *International workshop on robot modularity, IROS*, volume 290, 2016.

[13] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, 2015.

[14] Statistica. Robotics - worldwide, 2024.

[15] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

A. Parameters

The used parameters used in this report are listed below with their definition, value and unit.

Parameter	Description	Value	Unit
l and L	Length of a CC segment	variable	m
l_i	Length of segment i in the PCC model	variable	m
ϕ	Curvature plane rotation in the CC model	variable	rad
ϕ_i	Curvature plane rotation of segment i in the PCC model	variable	rad
θ	Segment rotation in the PCC model	variable	rad
θ_i	Segment rotation of segment i in the PCC model	variable	rad
κ	Curvature of a CC segment	variable	m^{-1}
r	Radius of a CC segment	variable	m
R^i_{i-1}	Rotation matrix from the end of segment i with respect to the end of the previous segment $i-1$	variable	1
t^i_{i-1}	Translation matrix from the end of a segment i with respect to the end of the previous segment $i-1$	variable	1
T^i_{i-1}	Homogeneous transform from the end of a segment i with respect to the end of the previous segment $i-1$	variable	1
$\Delta_{x,i}$	x-component of the rotation of segment i in the PCC model	variable	rad
$\Delta_{x,i}$	y-component of the rotation of segment i in the PCC model	variable	rad
Δ_i and d_i	Rotation of segment i in the PCC model	variable	rad
q	State that describes the PCC model	variable	rad and m
q_{sys}	System state that is used to calculate the tendon lengths	variable	rad and m
$q_{sys,initialize}$	System state that is used to calculate the tendon lengths	$0.12[0,0,1,0,0,2,0,0,2]$	rad and m
q_{est}	Estimated state used for the Jacobian control	variable	rad and m
q_{DL}	6 dimensional state used for the limits	variable	rad and m
$A_{q,q_{DL}}$	Matrix used for calculating q_{DL} from q_{sys}	See Equation 9	1
$L_{sys,LB}$	Lower bound of the system limits	See Equation 11	rad and m
$L_{sys,UB}$	Upper bound of the system limits	See Equation 11	rad and m
l_t	Tendon lengths	variable	m
$l_{t,calib}$	Calibrated tendon lengths	$0.12[1,1,1,2,2,2,2,2,2]$	m
X	6 dimensional pose	variable	m
X_0	Calibrated system pose	$[0,0,0.6,0,0,0.7]$	m
X_{meas}	Measured gripper pose	variable	m
X_{ball}	Measured ball pose	variable	m
X_{new}	New measurement of the system pose	variable	m
X_{goal}	Goal pose	variable	m
$X_{goal,\alpha}$	Goal pose, corrected for step size α	variable	m
$d_{gripper}$	Distance between the gripper and wrist	0.10	m
$d_{\mathrm{grasp-offset}}$	Distance between the ball and the initial goal position above the ball	0.05	m
P	Variance of the pose X	variable	rad^2 and m^2
P_{init}	Initial variance of the pose	Ball, wrist and gripper direction: $2000I$	rad^2 and m^2
Q	Variance of the model	Ball: $0.1I$, Wrist: $0.01I$, Gripper direction: $0.1I$	rad^2 and m^2
R	Variance of the measurements	Ball: $0.01I$, Wrist: $0.1I$, Gripper direction: $0.01I$	rad^2 and m^2
K_{kalman}	Kalman gain matrix	variable	1
$A_{pose,upd}$	Weighted matrix used in the pose estimation optimization	$diag([1,1,1,1.05,1.05,1.05,1.1,1.1,1.1])$	1
$\epsilon_{pose,bound}$	Bound around the measurement for the pose estimation optimization	0.005	m
J	Jacobian matrix of the PCC model	variable	1
J_K	Jacobian matrix of the PCC model with virtual stiffness	variable	1
α	step size	$[0.1,0.2,0.3]$	1
K	Virtual stiffness matrix	variable	1
$K_{initialize}$	Initial virtual stiffness matrix	$diag([0.2,0.2,2,0.1,0.1,1,0.1,0.1,1])$	1
$N_{max,iter}$	Maximum of iterations for the Jacobian loop	20	1
$\epsilon_{control}$	Threshold for the Jacobian controller	0.005	m
$\epsilon_{threshold}$	Threshold for the measured values	0.05	m

B. Results for other step sizes

This appendix contains the relevant plots that were not shown in the main body of the paper.

B.1. Single measurements.

This section contains the plots with the single measurements and the average for the step sizes $\alpha = 0.2$ (in Figure 16) and $\alpha = 0.3$ (in Figure 17).

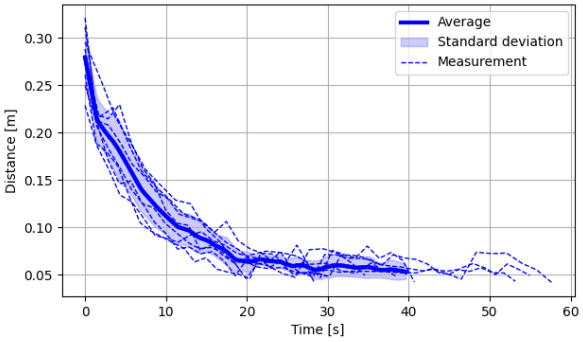


Fig. 16. The single measurements and the average and standard deviation of the error between the measured and desired position for step size $\alpha = 0.2$. The length of the average ling is the median of the 8 runs.

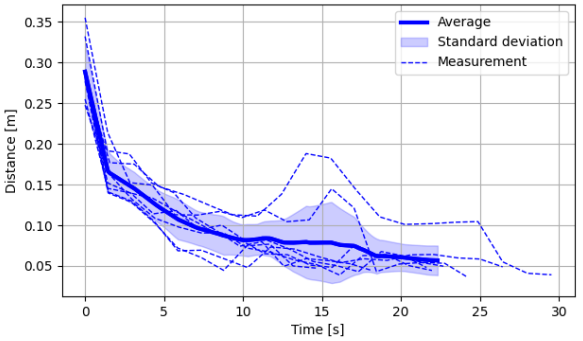


Fig. 17. The single measurements and the average and standard deviation of the error between the measured and desired position for step size $\alpha = 0.3$. The length of the average ling is the median of the 8 runs.

B.2. Comparison of measurements with system model state and state estimation.

This section contains the plots comparing the measured gripper location with the system model gripper location for the step sizes $\alpha = 0.2$ (in Figure 18) and $\alpha = 0.3$ (in Figure 20). It also shows the plots comparing the measured gripper location with the estimated model gripper location for the same step sizes in Figure 19 and Figure 21, respectively.

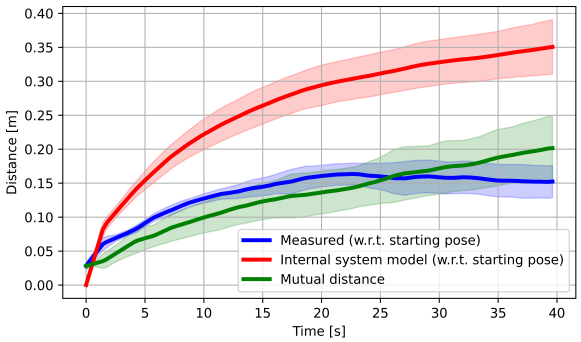


Fig. 18. The average and standard deviation of the norm of the gripper location with respect to the calibrated starting position using the system model (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.2$.

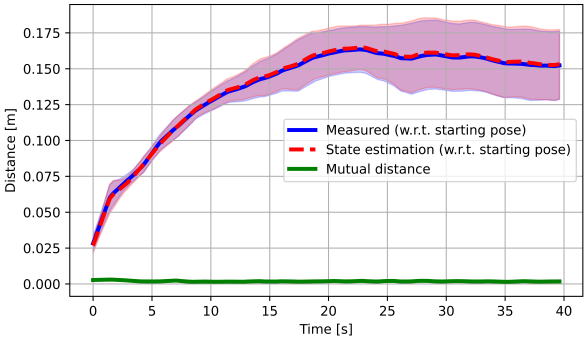


Fig. 19. The average and standard deviation of the norm of the gripper location with respect to the calibrated starting position using the state estimation in the model (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.2$.

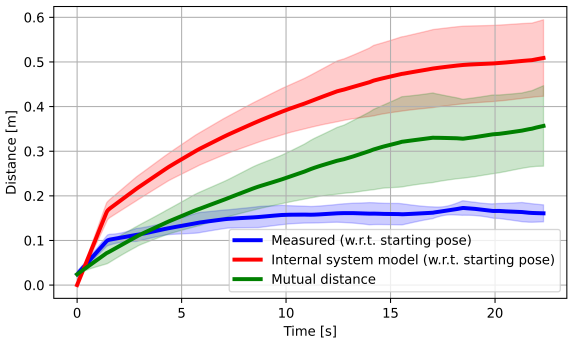


Fig. 20. The average and standard deviation of the norm of the gripper location with respect to the calibrated starting position using the system model (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.3$.

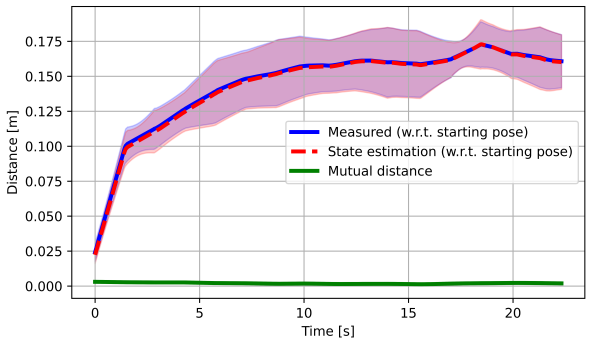


Fig. 21. The average and standard deviation of the norm of the gripper location with respect to the calibrated starting position using the state estimation in the model (blue) and the measurements (red). The distance between the two position estimations is shown in green. The used step size $\alpha = 0.3$.

B.3. Comparison of system with and without human guidance.

This section contains the plot comparing the automated system with the human guided system for step size $\alpha = 0.2$ in Figure 22.

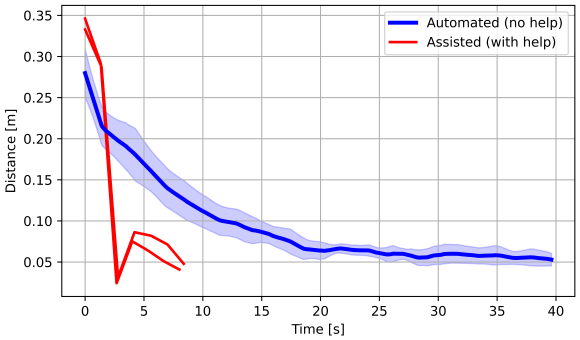


Fig. 22. The averages and standard deviation of the error between the measured and desired position for the 8 runs without human guidance (blue) and 2 runs with human guidance (red). The used step size $\alpha = 0.2$.

C. Hardware specifications

Raspberry Pi 5	Laptop: HP ZBook Studio G4
2.4 GHz quad-core	2.4 GHz quad-core
64-bit Arm Cortex-A76 CPU	Intel Core i7-7700HQ
4GB RAM	8GB RAM
VideoCore VII GPU	Nvidia Quadro M1200 4GB GDDR5
Wifi: 802.11ac, 2.4GHz and 5.0GHz	Wifi: 802.11ac, 2.4GHz and 5GHz

D. Code

For reproducibility, all source code used within the experiment that was executed on the laptop is available through the zip-file in this url: https://drive.google.com/file/d/1j45YhBTt1GUMbrJw8H3uS0gh6nbvFvVX/view?usp=drive_link. The zip-file also contains the results.

All code that was running on the Helix Raspberry Pi 5 is available through this url: https://github.com/helix-robotics-ag.