POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Visual servoing control and modeling of a soft robotic endoscope

Laurea Magistrale in Biomedical Engineering - Ingegneria Biomedica

Author: Lorenzo Civati

Advisor: Prof. Elena De Momi

Co-advisor: Chun-Feng Lai

Academic year: 2021-2022

1. Introduction

Urolithiasis, stone formation in kidneys and urinary tract, is a very common disease, in Italy there are 100.000 new cases every year and the number of people affected by this disease is increasing. Flexible Ureteroscopy (fURS) is a min-invasive technique among the first choice procedures for stone removal. This procedure consists in the insertion in the urethra up to the kidneys of a flexible endoscope in order to assess stone position and either remove them or destroy them. The surgeon inserts and steers the endoscope in order to reach the stone and perform the treatment exploiting laser fibers to dust or baskets to grasp the stone.
The endoscope control by the surgeon is difficult due to the reduced visual information and the disturbances due to the environment of the urinary system. In order to help surgeons in these tasks, an autonomous device able to solve basic tasks can decrease surgeon's mental burden as well as improve accuracy and clinical outcomes. These robotic devices must adapt to the patient anatomy to prevent damages to biological structures. The constrained environment and the unpredictable disturbances make the robot control be a challenge. Moreover compliance required for safety makes difficult to model the device.
In this research line the European project ATLAS (AuTonomous intraLuminAl Surgery) is set aiming to build an autonomous intraluminal endoscope called Atlascope [3].
In this work we will face the control challenge and modelling for the soft robotic Atlascope. We will exploit the visual information available by the camera to build a visual servoing controller and at the same time we will search for a model to predict non linearities.

2. State of the Art

Visual servoing is a closed loop control technique that exploits the visual feedback coming from one or more cameras. According to the space in which visual servoing is realised we will have a Position-Based Visual Servoing (PBVS) that realizes visual servoing in operational space and an Image-Based Visual Servoing (IBVS), that realizes visual servoing in the image space.
IBVS is a natural choice for endoscopic devices with an hand Eye-in-Hand (EIH) configuration. Boeheler et al. build a robotic tendon driven endoscope called RALITY [1] implementing an IBVS control. They extract from the image the position of a landmark and after defining the desired position in the image they control the motor velocity with a PID to solve a centering

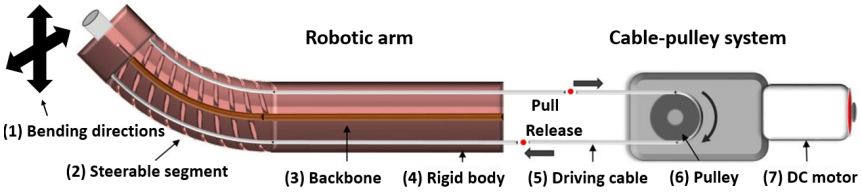


Figure 1: Actuation system adapted by [2]: DC motor (7) steers the cable (5) so that the steerable segment (2) bends vertically. Arrows (1) show the bending directions corresponding to the robot DoFs.

task. They faced problems due to backlash of the actuation system limiting their model accuracy.

An interesting IBVS approach is exploited also by Ott. et al. [4]. They built a controller to solve a centering task driving a tendon driven robot. Authors faced backlash and attempt to solve this problem by implementing an adjustment in the control scheme in order to solve this non-ideality in robot behaviour.

Continuum robots are characterised by an elastic structure able to continuously bend along its structure. Most of the robots are tendon-driven: wires running along the structure are pulled and released to move the robot arm. This can generate non-idealities such as death band, backlash and tendon coupling often resulting in hysteresis.

Modelling this non linear behaviour might allow to predict the non linearity and compensate for it. A promising approach to non linearities modeling is offered by deep learning as reported by Wu et al. [5]. Authors exploited an LSTM neural network to model the hysteresis of a pneumatic artificial muscle catheter.

3. Materials and methods

Here we report a description of Atlascope and its hardware connections. Then we will analyse the control system (Figure 2): the low level PID control and the high level controller implemented in ROS environment. Finally we will describe our neural network based model proposal and the metrics we will assess to test our controller and our model performances.

3.1. Atlascope

Atlascope is a continuum tendon driven robotic endoscope characterised by a proximal rigid segment of 58 mm length and by the a flexible robotic arm of 70 mm length [2]. An endoscopic camera is set on the tip of the arm. Cables on the opposite side of the shaft are anchored antagonistically on one pulley (Figure 1). Each pulley is rotated by a DC motor. Two DC motors (JGY370-30RPM, Walfront) characterised by self-locked worm gearboxes are exploited. Each motor pulley (thus each motor) is responsible for the steering of an antagonistic couple of cables controlling one Degree of Freedom (DoF). Each DC motor is provided by an hall effect incremental encoder. The robotic arm is then characterised by two bending DoFs. In order to supply the motors and acquire the data from the encoder and the camera a compact box is realised containing Arduino boards and motors driver.

3.2. Low level control

The separation between low and high level control is implemented because of the limited frequency of the visual feedback loop (30 Hz maximum). Separating the high level and low level control we can drive the motors with an higher frequency (up to 150-200 Hz). Low level control is implemented exploiting *ros_control_package*. We implemented two independent PID controllers, one for each motor. Variables associated to the bottom motor, called motor 1, are referred to with index 1 while index 2 is used when variables are referred to the right motor, called motor 2. The actual angle position of the motor ($q(t)$) is acquired by the encoders in degrees and multiplied by the constant C_{dr} in order to convert it in radians. We report the general PID equation we implemented (Equation (1)):

$$\tau(t) = K_P e_q(t) + K_D \dot{e_q}(t) + K_I \int e_q(t)dt. \quad (1)$$

We defined the error ($e_q(t)$) as the difference between the desired motor angle ($q_d(t)$) and the actual motor angle ($q(t)$) expressed in radians:

$$e_q(t) = q_d(t) - C_{dr} \cdot q(t). \quad (2)$$

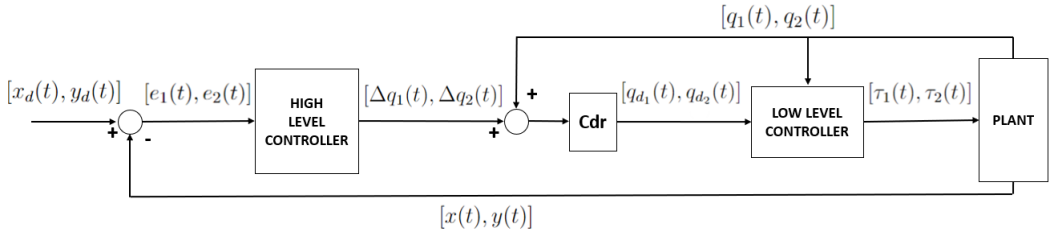


Figure 2: IBVS control scheme: the target position ($[x(t), y(t)]$) in the image is exploited to compute the error ($[e_1(t), e_2(t)]$) in the image space that will be exploited to compute the variation of the angle motor position ($[\Delta q_1(t), \Delta q_2(t)]$) that summed to the current position ($[q_1(t), q_2(t)]$) will make the robot adjust its position step by step so that the target appears in the desired position in the image ($[x_d(t), y_d(t)]$).

The PID output ($\tau(t)$) is the torque to be applied at motor level to reach the desired rotation angle. We see the classical PID coefficients K_P, K_I, K_D that weight the error, its integral and its derivative.

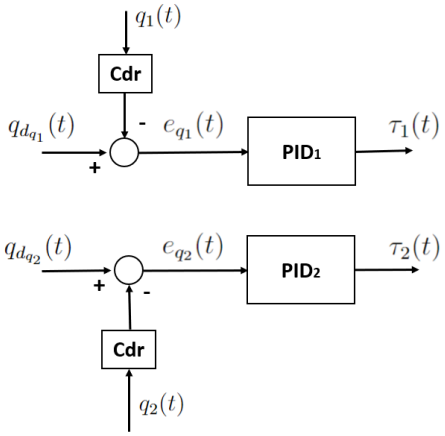


Figure 3: Low level PIDs.

After a manually tuning process performed searching for a trade off between overshoot and rise time, we set the PID parameters (K_P,K_I,K_D): $(3500, 500, 25)$ for PID_1 and $(3000, 600, 15)$ for PID_2. In order to limit the wind up effect at PID level we empirically set a limit to the maximum integral contribution.

3.3. High level visual servoing control

We implemented a visual sevoing controller composed of two independent P visual servoing controllers. The target position is relative to the camera and the robot tip position is unknown. We consider motor 1 to be mainly responsible for the endoscope bending in the horizontal direction thus for the x axis image position change of a target appearing in the endoscopic image space. Motor 2 will be instead the main actor who bends the soft robotic arm in vertical direction thus it will change the y coordinate of a target in the image space.
First of all we define a target in our image space and we detect the target position thanks to an edge detection algorithm available in the *OpenCV* library ($[x(t), y(t)]$). We define a desired position we want the target to be in the image ($[x_d(t), y_d(t)]$) and we get the error ($[e_1(t), e_2(t)]$) in the image space as difference between the desired and the actual position of the target in the image:

$$e_1(t) = x_d(t) - x(t) \tag{3}$$
$$e_2(t) = y_d(t) - y(t). \tag{4}$$

The desired angle position at motor joint level is updated as:

$$q_d(t) = C_{dr} \cdot (q(t) + C_{calib} \cdot K_P \cdot e(t)). \tag{5}$$

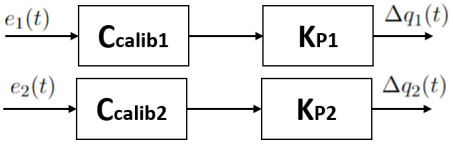


Figure 4: High level P visual servoing controllers: controller 1 working on motor 1 and controller 2 working on motor 2.

Where as we can see in Figure 4 the variation at each step is:

$$\Delta q(t) = C_{calib} \cdot K_P \cdot e(t). \tag{6}$$

The update is given by the product of constant value ($C_{dr} \cdot C_{calib} \cdot K_P$) multiplied by the error.

We will describe the tuning of these constants (both for controller 1 and controller 2) and describe the controllers performances in both directions in the next section.

3.4. Modeling

As further step of the work during experiments we faced one of the most common non linear effects in tendon driven continuum robots: hysteresis. This effect is present both for the bottom motor angle - x axis image position and for the right motor angle - y axis image position. These relationships will be considered independently and we will face the modeling of this behaviour exploiting deep learning techniques. In detail we will build a fully connected neural network in order to come up with a predictive model able to predict the motor angle position. Our network will provide the prediction of the motor angle associated to the last image position given the current target image position and the past angle and target position in the image as showed in Equation (7) and Equation (8):

$$q_1(t) = F_1[\boldsymbol{k_1}(t)] \tag{7}$$

$$q_2(t) = F_2[\boldsymbol{k_2}(t)]. \tag{8}$$

Where:

$$\boldsymbol{k_1}(t) = [x(t) \dots x(t-n_b), q_1(t-1) \dots q_1(t-n_a), 1]. \tag{9}$$

$$\boldsymbol{k_2}(t) = [y(t) \dots y(t-n_b), q_2(t-1) \dots q_2(t-n_a), 1]. \tag{10}$$

n_a and n_b are parameters that define respectively the number of past angle values and the number of the past camera values provided to the network that here is referred to as F. We will have two models F_1 and F_2, one for each motor - direction association we previously enhanced.

3.5. Performance evaluation

In order to assess the performances of our visual servoing controller we will test it in two tasks: a centering task and a tracking task. We will take into account Steady State Error (SSE) and overshoot for the direction involved in the target jump. Moreover to keep trace even of the effect of the P controllers interactions, we considered the tracking error ($E_{tracking}$) defined as the module of the vector composed by the error in the image space along x direction and y direction:

$$e_{tracking}(t) = (e_1(t)^2 + e_2(t)^2)^{\frac{1}{2}}. \tag{11}$$

We will compute the SSE and record the maximum value of $e_{tracking}$ ($E_{tracking_{max}}$). If we consider that in a recording we have N samples, $RMSE_{tracking}$ is defined as:

$$RMSE_{tracking} = \sqrt{\frac{\sum_{1=1}^{N} e_{tracking}(i)^2}{N}}. \tag{12}$$

Finally to evaluate our models performances we will consider the error between the real angle at motor level ($q(i)$) and the predicted angle ($q_P(i)$):

$$e_P(i) = q(i) - q_P(i). \tag{13}$$

Considering N predictions, we computed the Mean Squared Error (MSE) and Mean Absolute Error (MAE). Moreover we consider also the maximum absolute error ($|e_m|$). MSE and MAE are respectively defined as:

$$MSE = \frac{\sum_{1=1}^{N} e_P(i)^2}{N} \tag{14}$$

$$MAE = \frac{\sum_{1=1}^{N} |e_P(i)|}{N}. \tag{15}$$

4. Experiments

We performed four experiments to tune and assess the high level visual servoing controller. We chose a black circle considering its center as target position. Moreover we performed data acquisitions to collect data to train our deep learning model.

4.1. Experiment 1

This experiment is performed to tune the K_P constant for both the P visual servoing controllers. We will exploit the setup shown in Figure 5 considering a free space, removing the black obstacle on the right of the soft arm. A black circle is shown on a screen in different positions named according to Figure 5.
In detail the circle will jump between Pos_3, Pos_4 and Pos_5 to tune the P controller handling the bottom motor setpoint while the circle will jump between Pos_1, Pos_4 and Pos_7 to tune the P controller handling the right motor setpoint. Keeping the not involved P controller with K_P set to 0 we inspected several combinations. We empirically found from experiment

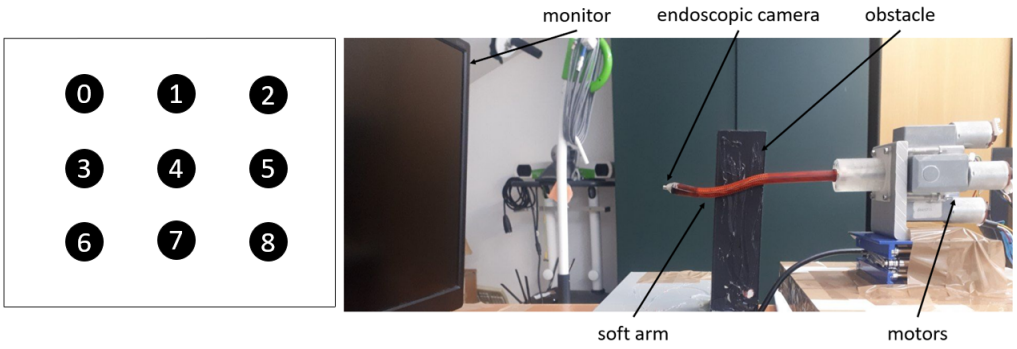


Figure 5: Experimental setup: Atlascope and the screen exploited in the experiments are shown on the right; circle positions on the monitor are shown on the left (each position is numbered from 0 to 8 and it will be referred to as *Pos_number* where number is the value associated in this figure).

1 the desired (K_{P_1}, K_{P_2}) values as the values $(0.5, 0.2)$. C_{dr}, C_{calib_1}, C_{calib_2} are fixed constants empirically set to 0.017, 0.21 and -0.36 respectively. The high level P controller will adjust the motor angle position to center the circle in the middle of the image. After setting (K_{P_1}, K_{P_2}) we performed experiments to assess the performance parameters of the chosen P controller.

4.2. Experiment 2

Here we exploit the same set up as in Figure 5 considering a free space as in experiment 1. We make the circle jump in different positions according to Table 1. Circle rests in each position for 15 seconds. This sequence sequence will be performed 5 times. When the circle jumps a step appears in the error image and we can inspect the step response of our controller.

4.3. Experiment 3

We studied the controller performances in tracking a target in free space. We move the circle on the monitor in a continuous way. The circle will displace: right and left, right-down and left-up, down and up and finally left-down and right-up always coming back to the center of the monitor. It will be repeated seven times.

4.4. Experiment 4

In this experiment we worked in a constrained environment (Figure 5). Every time the robot bends rightwards it touches the obstacle. We performed experiment 2 reducing the jumping sequence to left and right directions, the first row of Table 1. We repeated the sequence for 5 times.

Circle displacement sequence
$Pos\_4 \rightarrow Pos\_3 \rightarrow Pos\_4 \rightarrow Pos\_5 \rightarrow Pos\_4$
$Pos\_4 \rightarrow Pos\_1 \rightarrow Pos\_4 \rightarrow Pos\_7 \rightarrow Pos\_4$
$Pos\_4 \rightarrow Pos\_2 \rightarrow Pos\_4 \rightarrow Pos\_6 \rightarrow Pos\_4$
$Pos\_4 \rightarrow Pos\_0 \rightarrow Pos\_4 \rightarrow Pos\_8 \rightarrow Pos\_4$

Table 1: Circle jumping sequence.

4.5. Data acquisition protocol

In order to acquire the dataset to train and test the neural network, we exploit the same setup of Figure 5 considering a free environment without obstacles. We keep the circle still on the monitor so that it will appear in the camera field of view and we send an input to low level PID of the robot. The input is a series of set points at joint level. We will preform this procedure separately for the two neural networks. We will send set points to motor 1 and record the position x of the target in the image space while we will move motor 2 and record the coordinate y of the target in the image. We will collect four datasets both for x and y direction . Dataset 1 (1_test) is built sending sinusoidal inputs with constant amplitude and constant frequency to the motor; Dataset 2 (2_test) is built sending varying amplitude sinusoidal inputs; in Dataset 3 (3_test) sinusoidal inputs with varying frequency are exploited while for dataset 4 (4_test) a sinusoidal input with constant frequency constant amplitude is exploited, but the circle position on the monitor is different for each recording composing the dataset. We set $n_a = 7$, $n_b = 5$ and exploited a fully connected neural network composed by a single hidden layer composed of twelve neurons characterised by a Rectifier Linear Unit (ReLU)

activation function. We choose a mean squared error loss function and an output layer composed by a single linear neuron. We trained and validate it on a subset of dataset 1. Data are filtered by a Gaussian filter to remove measurement noise.

5. Results and Discussion

After discarding the experiment repetitions not correctly recorded, where for example some data were missing, we analysed the results.
The proposed P visual servoing controller is able to successfully center the target in the middle of the image space reaching a steady state error aligned with data in literature. Experiments performed in Section 4 showed good performances in centering task (Experiment 2 and 4). In free environment (Experiment 2) the circle jumps and the robot controller is able to center the target in the middle of the image space for each step direction with an average $SSE_{tracking}$ always lower than 12 pixels with a maximum standard deviation lower than 10 pixels. Instead a critical parameter is overshoot that depending on the direction reaches values higher than 20%. Lower performances are reached in tracking task (Experiment 3), our controller shows a mean $RMSE_{tracking}$ of 59.64 pixels.
In Table 2 we report the performance metrics for the centering task after steps in horizontal directions comparing free and constrained environment (Experiment 2 and 4 respectively). As we can see from the table when the robot is moving against the obstacle in the constrained environment we record an higher steady state value and variance for the the tracking error. However the controllers are still able to center the target. During the data recording we faced hysteresis and backlash. Our network model for x direction is able to predict the joint angle associated to the desired camera value as we can see in Table 3. The maximum absolute error ($|e_m|$) shows high values. Inspecting the predictions we notice that this error happens at the junctions point of the trials we joined together to build the dataset. Here the previous data are not linked to the current camera and joint value causing this wider error. We see that metrics are higher for model y with respect to the model for the x direction. We notice that predicting capabilities of model y are affected by the target displacement.

Free space		
Parameter	**Mean [pixel]**	**Std [pixel]**
$SSE_{tracking}$	7.98 px	4.75 px
SSE_1	−0.13 px	5.1 px
$E_{tracking_{max}}$	170.04 px	2.79 px
$E_{overshoot_1}$	26.5%	19%
Constrained space		
Parameter	**Mean [pixel]**	**Std [pixel]**
$SSE_{tracking}$	15.13 px	7.27 px
SSE_1	1.92 px	5.79 px
$E_{tracking_{max}}$	172.34 px	7.59 px
$E_{overshoot_1}$	22.2%	12.5%

Table 2: Performance in step direction 4 to 5.

	Model X		
Dataset	**MSE**	**MAE**	$\lvert e_m \rvert$
1_*test*	0.23	0.25	5.03
2_*test*	0.13	0.27	2.82
3_*test*	0.07	0.22	1.24
4_*test*	0.28	0.28	13.23
	Model Y		
Dataset	**MSE**	**MAE**	$\lvert e_m \rvert$
1_*test*	0.35	0.30	8.51
2_*test*	0.19	0.32	3.27
3_*test*	0.16	0.33	1.20
4_*test*	1.08	0.62	9.50

Table 3: Models performance metrics.

6. Conclusion

In this work a IBVS P controller has been implemented for Atlascope. This control system is able to track a target moving in space and center it in the middle of the image space. Better performances are reached for centering tasks since this allows the controller to adjust the position and correct the overshoot. Atlascope shows common tendon driven robots non linearities. We modelled these non linearities in specific conditions with the target set at a specific distance from the robot. The model shows promising predictive capabilities. In future works this model can be extended for targets at different distances from the robot and integrated into the controller here implemented in order to predict non linearities and adapt the proportional constant of the controller according to the model predictions.

References

[1] Quentin Boehler, David S Gage, Phyllis Hofmann, Alexandra Gehring, Christophe Chautems, Donat R Spahn, Peter Biro, and Bradley J Nelson. Realiti: A robotic endoscope automated via laryngeal imaging for tracheal intubation. *IEEE Transactions on Medical Robotics and Bionics*, 2(2):157–164, 2020.

[2] Jorge F Lazo, Chun-Feng Lai, Sara Moccia, Benoit Rosa, Michele Catellani, Michel de Mathelin, Giancarlo Ferrigno, Paul Breedveld, Jenny Dankelman, and Elena De Momi. Autonomous intraluminal navigation of a soft robot using deep-learning-based visual servoing. *arXiv preprint arXiv:2207.00401*, 2022.

[3] Finocchiaro Martina, Xuan Thao Ha, Lazo Jorge, Chun-Feng Lai, Ramesh Sanat, Hernansanz Albert, Borghesan Gianni, Dall'Alba Diego, Tognarelli Selene, Rosa Benoit, et al. Multi-level-assistance robotic platform for navigation in the urinary system: Design and preliminary tests. In *Conference on New Technologies for Computer/Robot Assisted Surgery*, pages 90–91, 2022.

[4] Laurent Ott, Florent Nageotte, Philippe Zanne, and Michel de Mathelin. Robotic assistance to flexible endoscopy by physiological-motion tracking. *IEEE Transactions on Robotics*, 27(2):346–359, 2011.

[5] Di Wu, Mouloud Ourak, Kenan Niu, Yao Zhang, Mirza Awais Ahmad, Jenny Dankelman, and Emmanuel Vander Poorten. Towards modeling of hysteresis in robotic catheters based on lstm. In *32nd Conference of the International Society for Medical Innovation and Technology (iSMIT), Date: 2020/12/03-2020/12/10, Location: Chicago, USA*, 2020.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Visual servoing control and modeling of a soft robotic endoscope

Tesi di Laurea Magistrale in
Biomedical Engineering - Ingegneria Biomedica

Author: **Lorenzo Civati**

Student ID: 920599
Advisor: Prof. Elena De Momi
Co-advisors: Chun-Feng Lai
Academic Year: 2021-22

Abstract

Urolithiasis is the formation of stones in the urinary system. This pathology is getting more and more spread due to low fluid intake, wrong diets and life styles. Flexible ureteroscpy is getting a primary choice for stone removal thanks to improvement in available devices. In this proceedure surgeon exploits fluoroscopic and mainly endoscopic visual feedback to drive the flexible ureteroscope inside urethra, bladder and ureter up to kidneys. Flexible endoscope control requires an high degree of dexterity and training to avoid damage that might create complication and prevent positive clinical outcomes. European ATLAS (AuTonomous intraLuminAl Surgery) project aims to support surgeons by the development of a continuum soft endoscopic robot called Atlascope able to navigate autonomously through fragile lumens. Soft continuum robots introduce challenges in the control design. A continuum and flexible structure can' t be easily modelled and simplifying assumptions are often exploited in order to come up with an analytical model of these devices. These assumptions reduce model accuracy and controller performances. In this thesis the design and implementation of a visual servoing controller for Atlascope is accomplished and deep learning modelling for Atlascope in inspected based on the experimental recorded data. After building the hardware connections to drive Atlascope's actuation system we implemented a visual servoing controller that is able to successfully detect, track and center a target in the endoscopic camera image. Atlascope showed a non linear behaviour due to friction and backlash translated in an hysteresis behaviour. A deep learning model is proposed and tested. Promising modelling results suggest that this model might be exploited to improve controller performance.

Keywords: continuum robots, robotic endoscope , visual servoing, deep learning model

Sommario

L'urolitiasi è la formazione di calcoli nel sistema urinario. Questa patologia si sta diffondendo sempre più a causa del basso apporto di liquidi, di diete e stili di vita sbagliati. L'ureteroscopia flessibile sta diventando una scelta primaria per la rimozione dei calcoli grazie al miglioramento della strumentazione disponibile. In questa procedura il chirurgo sfrutta il feedback visivo delle immagini ottenute tramite fluoroscopia e soprattutto l' informazione visiva della telecamera endoscopica per guidare l'ureteroscopio flessibile all'interno dell'uretra, della vescica e dell'uretere fino ai reni. Il controllo dell'endoscopio flessibile richiede un alto grado di destrezza e pratica per evitare danni che potrebbero creare complicazioni e impedire esiti clinici positivi. Il progetto europeo ATLAS (AuTonomous intraLuminAl Surgery) mira a supportare i chirurghi tramite la sviluppo di un robot endoscopico continuo e flessibile chiamato Atlascope in grado di navigare autonomamente attraverso lumi fragili. I robot continui introducono delle sfide nella progettazione del sistema di controllo. Una struttura continua e flessibile non può essere facilmente modellizzata e spesso vengono introdotte ipotesi semplificative per elaborare un modello analitico di questi dispositivi. Queste ipotesi riducono l'accuratezza del modello e le prestazioni del controllore. In questa tesi viene realizzata la progettazione e l'implementazione di un controllore a retroazione visiva per Atlascope. Inoltre la modellizzazione tramite deep learning per Atlascope viene analizzata sulla base dei dati sperimentali registrati. Dopo aver costruito le connessioni hardware per guidare il sistema di attuazione di Atlascope, abbiamo implementato un controller di servocomando visivo in grado di rilevare, tracciare e centrare con successo un obiettivo nell'immagine della telecamera endoscopica. Atlascope ha mostrato un comportamento non lineare dovuto ad attrito e gioco tradotto in un comportamento di isteresi. Qui viene proposto e sottoposto a verifica un modello di deep learning basato su reti neurali. I risultati del modello ottenuto sono promettenti e suggeriscono che questo possa essere utilizzato in futuro per migliorare le prestazioni del controllore ad ora implementato.

Parole chiave: robot continui, endoscopio robotico, visual servoing, modello deep learning

Contents

1 | INTRODUCTION

1.1. Clinical background

A patient specific treatment and a min-invasive approach are becoming essential [32]. Everyday these goals get more achievable thanks to advancement in technology and medicine. Main research lines are focusing on improving imaging techniques and creating robotic devices that can assist surgeons or autonomously perform some routine tasks [41]. There are many fields of medicine which benefit of these improvements and in particular urology, branch of medicine focused on the study and treatment of the urinary system diseases.

1.1.1. Urinary system

Urinary system (Figure 1.1) can be represented in a simplified way as composed of:

- kidneys;
- bladder;
- ureters;
- urethra.

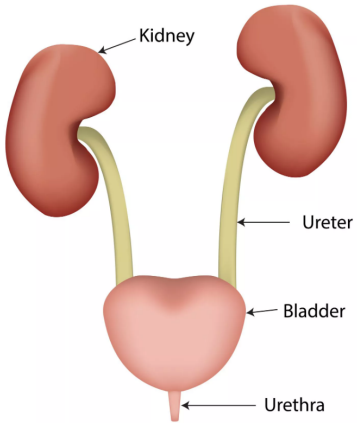


Figure 1.1: Urinary system structure.

Kidneys filter blood by removing waste substances and regulate blood pressure. Waste substances dissolved in water (urine) flow into the bladder thanks to the ureters. Ureters are characterized by a diameter lower than 1 cm and an average length of 25-30 cm. Bladder is an elastic reservoir able to expand to collect up to 300-400 ml of urine. Finally from the bladder, urine can be released through the urethra, a tube with average diameter of 1 cm [20].

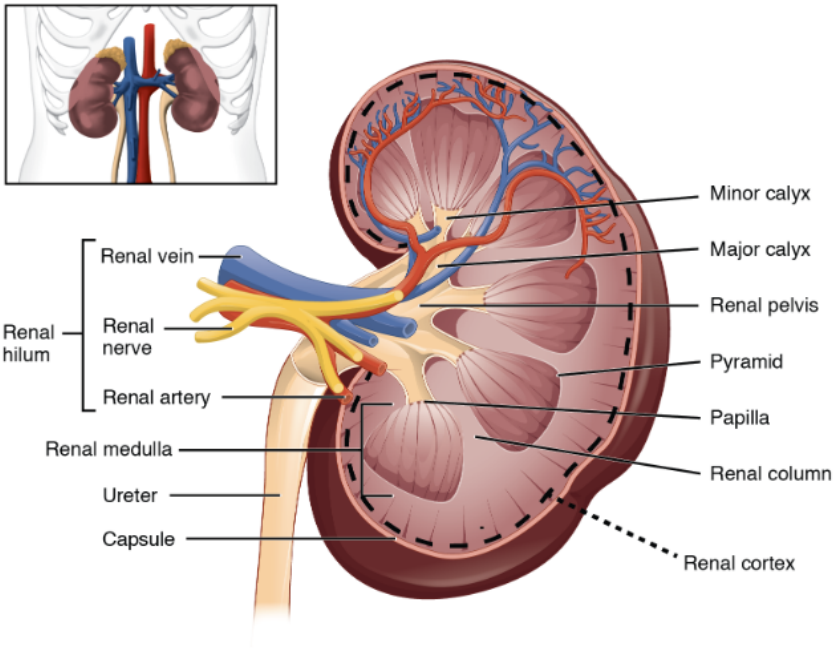


Figure 1.2: Kidney's structure.

Several diseases might affect the urinary system. Inflammations, infections and Urolithiasis are common. The latter one is a very good example of disease that benefited from technology advancement and there will be a wide step forward thanks to autonomous or partially autonomous devices [35].

1.1.2. Urolithiasis

Urolithiasis, or nephrolithiasis, is the formation of stones in the kidneys and urinary tract. During the filtering process, the waste material can get accumulated forming solid structures called calculi. When the stones are small they can pass on their own through the urinary tract. If this is not possible, some complications can appear For example, block of the ureter causes severe pain in the abdomen. In Italy, there are 100.000 new cases every year and the number of people affected by this disease is increasing due to sedentarism, unhealthy diet and low intake of fluids. Moreover, 50 percent of people who

had experienced the disease are going to be affected again during lifetime [38].

Nowadays several treatments are available:

- Extra-corporeal Shock Wave Lithotripsy (ESWL);
- Per-Cutaneous Nephrolithotomy (PCNL);
- Flexible Ureteroscopy (fURS);
- open surgical and laparoscopic removal.

EWSL is a minimally invasive surgical technique that uses ultrasound waves to break up kidney stones in small pieces that can be then expelled in the urine. PCNL consists of removal of kidney stones using a scope inserted through a small incision performed from the back of the patient. fURS consists of the passage of a flexible endoscope through the urethra and the bladder up to the ureter to remove the stones.

According to the size, the location and the chemical composition of the kidney stones different treatments are used. However, minimally invasive approach has become the gold standard. Minimally Invasive Surgeries (MIS) have lots of advantages compared to their open-counterparts:

- lower risk of infection, lower recovery time;
- lower pain and discomfort;
- lower bleeding.

According to this principle, the open surgical and laparoscopic removal now is avoided if possible and ESWL and fURS are the first line treatment option for renal stones $<$10 mm. In Table 1.1 we summarized some general guidelines provided by European Association of Urology (EAU) [34].

EAU treatment algorithm for renal calculi

Stone dimension	Treatments
<10 mm	1. ESWL or fURS 2. PCNL
10–20 mm	ESWL endourology (fURS or PCNL)
>20 mm	1. PCNL 2. fURS or ESWL

Table 1.1: EAU treatment algorithm for renal calculi (10–20 mm lower pole renal calculi excluded) adapted by Dolzi and Taxer [13].

We have just seen that one of the most common urological disease must be solved by a minimally invasive surgery performed thanks to a flexible ureteroscope. Let' s analize in detail the Modern Flexible Ureteroscopy to see which is the standard armamentarium, how a standard proceedure is performed and how engineering and robotic devices introduction can help surgeons to improve clinical outcomes.

1.2. Flexible ureteroscopy

Flexible ureteroscopy is a minimally invasive technique exploited both to diagnose diseases and to perform treatments in urinary system. The main medical device necessary to perform a ureteroscopy is an endoscope, called ureteroscope, due to its specific use. Flexible refers to its ability to bent in a passive or active way.

1.2.1. Flexible Ureteroscope

The Flexible Ureteroscope (FU) consists of 3 main elements [6]:

- optical system;
- deflection mechanism;
- working channel.

Figure 1.3: Flexible ureteroscope.

The optical system can be characterised by:

- fiberoptic bundles, fibers of glass are bundled with identical orientation at each end (coherent) and lenses are used at the extremities to allow image magnification and to increase field of view and focusing ability;
- light source with digital camera (CCD or CMOS digital camera) on the tip of the ureteroscope.

The deflection mechanism can be realised in different ways but most deflecting mechanisms are cable-driven: control wires running down the length of the ureteroscope are attached on the proximal end to a manually or automatic operated lever mechanism.

The working channel is essential since it allows instrumentation to pass and allows irrigation. Among the accessories (ancillary equipment) there are biopsy forceps, stone graspers, baskets (Nitinol baskets are the standard for stone retrieval), laser fibers (Holmium:Yttrium-Aluminum-Garnet, Ho:YAG, is the most used because it works with any types of calculi). Due to its specific application, some requirements must be fulfilled by this device. The ureter has an average diameter of 1.8 mm (SD: 0.9,range 1-6 mm), usually being < 3 mm in asymptomatic people [57]. Therefore, the diameter of this endoscope must be small: it is usually asked to be smaller than 9 Fr (1 mm = 3 Fr). As we will see during the analysis of a fURS procedure other requirements are optimal image quality of the image/video delivered, good irrigation, bidirectional maneuverability, minimal decrease of the urine flow rate with the administration of endoscopic tool, ergonomic and user-friendly handles to allow for torque and easily handling.

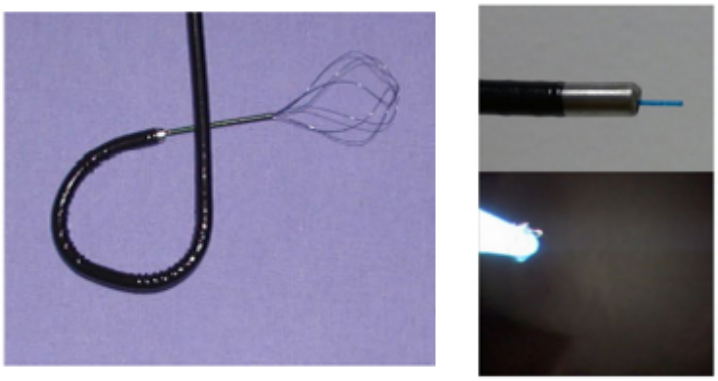


Figure 1.4: Ancillary equipment adapted by Doizi and Traxer [13].

1.2.2. Standard modern Flexible Ureteroscopy

In this paragraph we will describe the techniques for fURS focusing on the treatment for renal stone by laser lithotripsy. This will allow us to point out how robotic approach can help surgeons in their task.

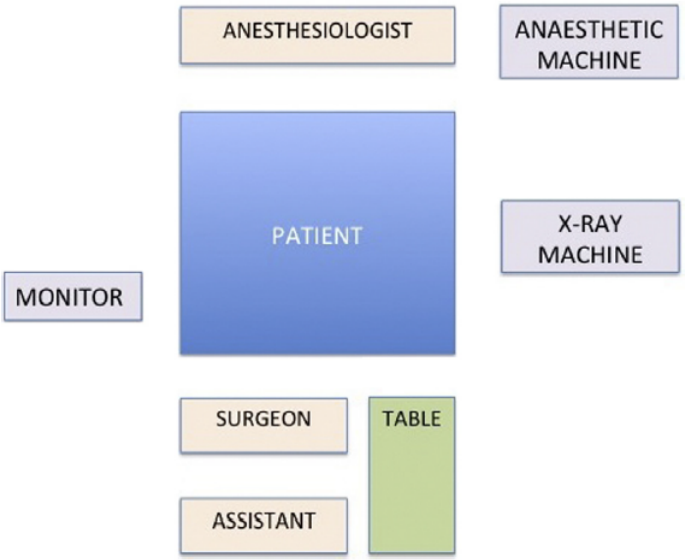


Figure 1.5: Operative room (OR) set-up adapted by Giusti et al. [18].

The procedure [13, 18] starts with patient positioning in lithotomy position. At this point, general anesthesia is delivered to reduce kidney's movement due to respiration, but even spinal anaesthesia is a viable option. Urologists will perform an inspection of the bladder to place a guide-wire into the kidney or alternatively, urologists will place an Ureteral Access Sheath (UAS) under fluoroscopic control and insert the FU through the UAS. Now navigation into the Upper Urinary Tract (UUT) starts. Once the stones have been

reached the endoscope must be straightened in order to insert the optical fiber. Ho:YAG laser delivers pulsatile energy and operates at a wavelength of 2100 nm in the infrared spectrum.

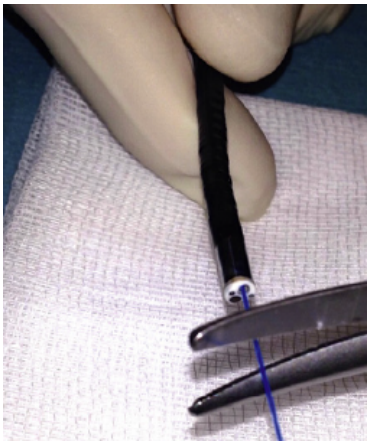


Figure 1.6: Laser fiber adapted by Giusti et al. [18].

To prevent the functional deterioration of the flexible ureterosccope is advisable to keep the laser fiber tip out of the scope as far as one-quarter of the screen diameter. This is enough to see a minimal part of the working tool going out from the Working Channel (WC) to be sure that it is far enough from the scope tip. Moreover, excessive and prolonged deflections should be avoided in case of lower pole stones and relocation to a more favorable upper pole calyx with a basket is suggested.

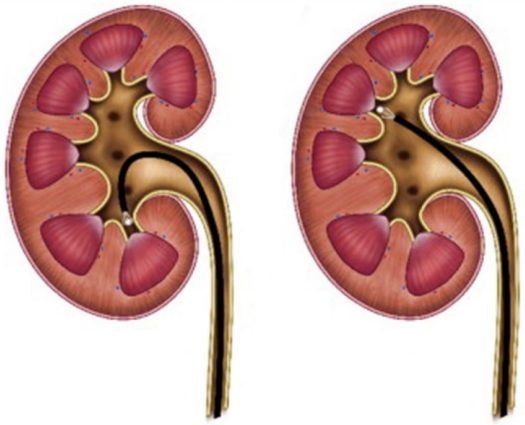


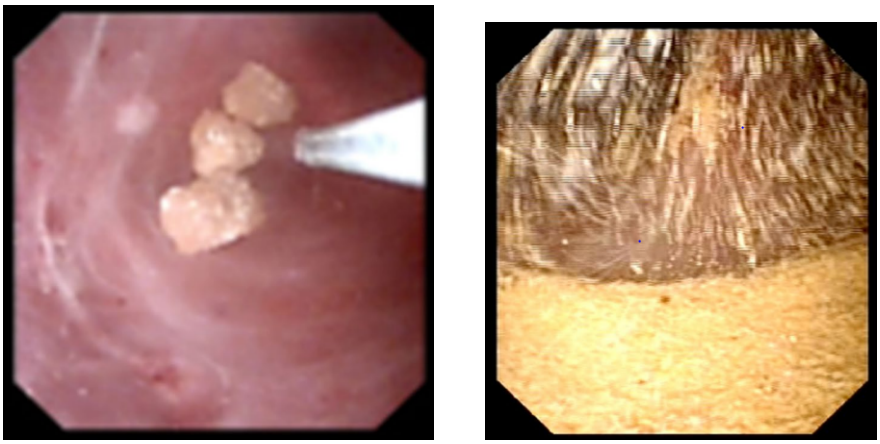
Figure 1.7: Lower pole renal stone relocation adapted by Doizi and Traxer [13].

The aim of lithotripsy is always to maximize pulverization of the stone and pulse energy is one of the most important factors affecting ablation and procedure speed. Finally we underline that large diameter laser fibers are more resistant to fiber tip degradation

or the 'burn-back' effect, although their stiffness reduces FU maximal deflection and their diameter decreases irrigation flow. On the other hand, small diameter laser fibers are more fragile and prone to fiber tip degradation, but they allows better irrigation, better endoscope deflection and less retropulsion. This displacement of the stone can be decreased lowering the pulse energy or shifting to long-pulse mode. In case of need a basket can be used to relocate the target in a better position. At the end of the procedure, after stones have been dusted or fragmented and fragments have been removed, a careful ureteral exploration should be performed to retrieve the scope being sure no injuries are present. All these procedures need high level of coordination with the scrub nurses and the assistants, it is of utmost importance to work in safety and to minimize operative time.

1.2.3. Problems

During the description of this surgical procedure there are some issues that once solved, may potentially improve safety and outcomes of the treatment. First of all, although min-invasive, it needs several operators to be present and work in a constrained environment. Several instruments are present around the patient and movements are constrained. C-arm for X-ray fluoroscopy is among the instrumentation. This implies exposure of operators especially the surgeon to X-rays. Usually during a procedure the surgeon is exposed to a radiation dose of 1.7-56 microSv [15]. There are problems due to patient movement due to respiration that move the anesthesia choice towards general anesthesia. This is also due to the possibility of exceeding the time of spinal anesthesia forcing then the anesthesiologist to switch to general anesthesia. Surgeon mind burden is very high. There are many aspects that must be taken into account and often whole procedure is made harder by difficulties in understanding tip orientation inside the kidney, need of compensating patient movements, the retropulsion effect of the laser (the displacement of the stone due to the impinging laser) or in general stone migration due the treatment and a visual information not always clear due to debris created by the stone dusting. In Figure 1.8, we can see two examples of the endoscopic view during lithotripsy that clearly show the burden surgeon faces during the procedures due to a sub-optimal visibility.



(a) Endoscopic view.

(b) Endoscopic view during dusting procedure.

Figure 1.8: Endoscopic view provided to the surgeon adapted by Doizi and Traxer [13].

Moreover this procedure is often performed in a standing position with movements and poses that might create discomfort and orthopedical issues to surgeons [15], these issues increase the fatigue of the operator and decrease ones performances.

Table 1.2 summarizes the main problems of fURS procedure.

Issues of fURS procedure

fURS issue	Causes
Radiation exposure	1. Fluoroscopy to access endoscope position
Ergonomic problems	1. Standing position 2. Not natural Wrist movement
Extra tasks	1. Difficult space orientation 2. Sub-optimal visibility due to debris 3. Patient movement compensation 4. Target and tip of the endoscope displacement
OR organisation	1. Many instruments and operators are needed

Table 1.2: Problems of fURS procedure that might benefit from a robotic approach.

1.3. Robotic solution

Introducing robotic devices might solve some of the issues enhanced in the previous subparagraph such as the exposure radiation, ergonomic problems and difficulties in instrument control rising from the reduced visibility and external disturbances. Manipulators or robotic endoscopes might also decrease the mental burden of the operator. They are characterized by several advantages:

- high geometric accuracy;
- stability;
- capability of operating at different scales (even microscale in mini-invasive systems);
- radiation resistance (image-guided robotic surgery);
- sensors presence (chemical, acoustic, electro-mechanical, optical).

There are three autonomy levels for robots in surgery[2]:

- no autonomy (passive devices, the surgeon controls the motion of manipulators from a console, a typical application in min-invasive surgery);
- partial autonomy (the device imposes geometric/motion constraints which ultimately increase accuracy and safety);
- complete autonomy (the device realises a surgical plan with no human intervention).

We are dealing with minimally invasive surgeries in the ureteroscopy. In minimally invasive surgery application, Robotic master–slave systems are used in clinical applications [40, 46]. The systems consist of a console unit and a patient cart at bedside. One of the last examples of this type of devices built especially for urology and Retrograde Intra Renal Surgery (RIRS) is Avicenna Roboflex (ELMED, Turkey).

1.3.1. Avicenna roboflex

Avicenna robotic platform was developed starting from 2012 [40, 46]. It is composed of two elements:

- surgeon's console;
- manipulator of the flexible ureterorenoscope.

Figure 1.9: Avicenna platform adapted by Saglam [46]

The surgeon's console is characterized by adjustable seat with armrests, two joysticks (the left one controls upwards and downwards deflection, the right one controls rotation in both directions and advancement and retraction of the endoscope), a regulation wheel (for fine adjustments of deflection), a touch screen control monitor (with four functionalities among which changing speed, advancement and retraction of the laser fiber are present) and two foot pedals to control the laser device and fluoroscopy.

Figure 1.10: Avicenna console adapted by Saglam [46]

The Manipulator consists of the motor system and the robotic arm which holds and moves the endoscope. The robotic arm can be rotated by 210° in each direction. Small

motors move the steering lever for deflection and enable motion scaling of the string based movement tip of the endoscope.

Figure 1.11: Avicenna manipulator adapted by Saglam [46]

In papers [40, 46], authors analyse clinical results of fURS performed by expert surgeons with Avicenna device. Their analysis shows that surgeons after being trained on phantoms and after few surgeries are able to reproduce their performances with classical fURS. In particular, it is enhanced how ergonomic position of the Avicenna platform produces less orthopedical discomfort in surgeons. Avicenna is able to solve many of the issues associated to the fURS procedure (Table 1.2): Surgeons' physical fatigue problem is almost completely solved thanks to the ergonomic position of the console and the console allows operators to be distant from C-arm reducing X-ray exposure to zero. Moreover the endoscope control is very precise and the absence of haptic feedback is not an issue: increased quality image, lower fatigue and fine movement control compensate this absence.

1.3.2. Avicenna limitations

Although these improvements shown in the previous section, many issues are still present. Above all, mental load faced by surgeon is still very high. Avicenna console is improving precision and ergonomics but whole procedure and decisions are up to the surgeon. Improvements in the image quality is not enough to decrease mental fatigue and efforts the surgeon has to face in order to drive the endoscope and compensate all unexpected situations. In recent years an increasing number of research lines are growing with the aim of making robots more and more capable of accomplishing tasks autonomously in order

to let the surgeons focus on the main and more delicate aspects of the surgery. For example, existing endoscopes and instruments are adapted to be actuated by motors driven by autonomous control systems and new devices are built by scratch in order to reach the best coupling between physical device and control system [26, 30]. A nice example of a combination between rigid manipulator and soft continuum endoscope is provided in [28] were they combined a rigid four degrees of freedom with a soft endoscope with two degrees of freedom controlled exploiting the visual information from a camera on the tip of the device. On this research line the European project ATLAS (AuTonomous intraLuminAl Surgery) is set.

1.3.3. Atlas project

ATLAS project [39] is an European project which aims to develop techniques to automate complex surgical intraluminal therapies. Navigation through narrow and mostly fragile and deformable lumens requires considerable skill and dexterity. Unfortunately, due to phenomena such as slack, backlash and compliance the controllability of the instruments is reduced. This might bring to surgical risks such as internal bleeding, tissue damage, puncture or rupture even when expert surgeons are operating. ATLAS project aims to solve these problems applying automation and novel technology to acquire, exploit and elaborate information to allow the device to be able to accomplish task in autonomous way. A first prototype, called Atlascope, has already been realised. This endoscope consists of a cable-driven soft robotic endoscope which has a backbone and a helical structure with two bending directions. It is actuated thanks to metallic cables attached to pulleys rotated by motors. Authors embedded several type of sensors, such as Electromagnetic (EM) tracking sensors, multicore Fiber Bragg Grating (FBG) and endoscopic camera with the aim of creating a complete robotic platform to help surgeons in accomplishing their tasks. The first results and tests are very promising and authors aim to proceed improving both hardware and software of the platform [31].

1.3.4. Challenges in autonomous robotic approaches

Exploiting autonomous devices to perform a treatment, high level of accuracy and safety margin of error must be guaranteed. Being able of providing surgeon with these data is essential especially in navigation. In order to be able to achieve this accuracy, a robust control system must be realised. If we were able to model patient anatomy and robot-human interactions we would be able to build a correct model to control and perform the treatment, but we are not provided with a global patient model and even modeling of robotic device is not trivial. Endoscopic robots' structure is more and more often

built with a continuum shape and compliant materials to be able to safely interact with tissues. This introduces several issues for example leading to unknown modifications due to external disturbances.

1.3.5. Control challenge

In order to deal with the difficulties in modelling and sensing, especially in case of intraluminal navigation due to the continuum structure of the devices, many authors approach the soft continuum robotic device challenges. Starting from the design, every attempt will aim either to collect as much information as possible in order to be able to sense disturbances and unmodelled behaviours exploiting for example shape sensing and visual information fusion as reported in [31] or realising new control techniques able to adapt themselves to these disturbances and being stable despite a lack of complete of a-priori knowledge of the system. The absence of a accurate and precise model is one of the main issues that researchers faces in different ways. As we will see in next chapter very often assumptions are considered to build an analytical model, while other authors attempts in modelling the robot stating from the data exploiting deep learning opportunities.

1.4. Aim of the work

In this thesis work we were provided with Atlascope, a soft continuum robot built in the ATLAS project framework as we previously introduced. Starting from the analysis of the main challenges in continuum robots control, we will focus on the design and implementation of a control system for this autonomous endoscope. Following the idea of ATLAS project, a controller will be realised to perform some routine tasks in autonomous way such as keeping a target in the centre of the endoscopic camera view in order to decrease surgeon mental burden. At the same time we will study the robot behaviour and search for a model able to predict its behaviour exploiting the opportunities offered by deep learning.

Summarising the cores of this work will be the design and testing of a visual servoing control algorithm and a novel modeling method that are able to deal with unknown and unpredictable nonidealities of our prototype. The control system will be implemented in ROS (Robot Operating System) framework in order to guarantee the best modularity and scalability of the system.

The thesis is composed as follow:

- Chapter 1 introduces briefly clinical background, robotic approach contribution and

challenges and Atlas project;

- Chapter 2 reports a detailed analysis of continuum robot control techniques focusing on visual servoing control approaches for tendon driven devices;
- Chapter 3 describes in detail Atlascope enhancing its characteristic, the hardware and software setup and all the methods exploited to build our visual controller and our model for Atlascope;
- Chapter 4 describes the experimental setup and protocol exploited to verify and test the controller performances and the modelling performances over the experimental collected data;
- Chapter 5 presents experimental results discussing controller and the modeling performances;
- Chapter 6 discusses the limits of the proposed control approach and the predictive capability of our model introducing the future work.

2 | STATE OF THE ART

In this chapter, we will start from the definition of a soft robotic manipulator to enhance the main issues about control systems driving these devices and we will focus our attention on tendon driven continuum robots control. In detail, the main visual servoing techniques and approaches for tendon driven soft robots will be inspected to come up in the end with the most suitable control system choice for Atlascope.

2.1. Soft robotics

First of all we must be detailed in the definition of soft robotics. Soft robotics is associated with two different branches [17]:

- compliant joints within rigid link robots;
- Continuum Robotic manipulators (CRs).

As previously described, we are dealing with this latter case. We will have to handle mechanisms without rigid links but with an elastic structure able to continuously bend along their structure. These devices guarantee an higher safety in human- robot interactions but rise problems in controller design.

In rigid manipulators there is no deformation, every joint is characterized by a specific number of Degrees of Freedom (DoFs) and it is finely controlled thanks to a precise modelling of the structure. In soft robotic the first clear difference is deformation. These devices are compliant, whenever a force is applied their shaft bends. This elastic behaviour has to be taken into account with the joint variables values in order to be able to correctly define the end-effector pose. In addiction, several non-idealities in the robot behaviour are present and external disturbances can' t be fully sensed and compensated with classical approaches leading to inaccuracy at control system level that might damage the robot. According to the design and in particular to the actuation system of the continuum robot, there might be different aspects to be handled. Each robot is becoming unique and needs a specific controller to be finely controlled [33].

The general design and structure of CRs can be classified from different perspectives such as mechanism designs, the number of DoFs, application and actuation modality of the robot. According to the actuation we can divide them in [5, 33]:

- Tendon-Driven Continuum Robot (TDCR) [7, 24];
- Concentric-Tube Continuum Robot (CTCR) [14, 50];
- Pneumatic Muscle-Actuated Continuum Robot (PMACR) [8];
- Hydraulic Muscle-Actuated Continuum Robot (HMACR) [22];
- Magnetically Driven Continuum Robot (MDCR) [27].

Concentric-tube robots are composed of multiple, precurved, elastic tubes that are nested inside of each other. The base of each tube can be axially translated and rotated to control the shape of the robot structure. This pre-curvature is often exploited to model the characteristics of the device to build the model. In papers [14, 50] a nice description of the design and modeling of these kind of continuum robots is reported. An interesting example of PMACR is reported in [8]. Authors built a robotic manipulator (Colobot) composed of a unique silicone rubber unit with three active pneumatic chambers regularly disposed at 120° apart. After the formulation of the kinematic model which relates actuators input and position and orientation of the distal end of Colobot in a Cartesian frame exploiting the information obtained by 3 optical sensors, they build a controller to drive the robot during colonoscopy. In [22] a new robot based on hydraulic pressure chambers and valves was build. This device was able to drive acting on a pressure signal several segments independently. This approach allowed to remove any metallic element from the inserted portion of the robot allowing its use in MRI room. In [27] a novel steerable robotic ablation catheter system is presented. It is equipped with a set of current-carrying micro-coils which are actuated by the magnetic forces generated by the magnetic field of the MRI scanner.

There are many examples of continuum robots for each of the previous actuation methods but TDCR is one of the most common actuation methods exploited in endoscopy and robotic devices [24].

2.2. Tendon driven soft robots control

TDCRs exploit the displacement of cables running along the length of the device and anchored to the shaft in order to bend the structure when they are pulled or released. An example of this actuation mechanism is presented in [26].

2.2.1. Operating space

Cables are actuated by motors and theoretically we should be able to know the degrees of freedom of the robot knowing how cables and motors are arranged.

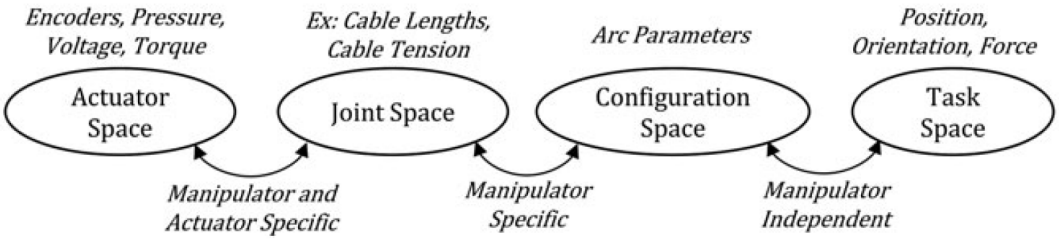


Figure 2.1: Operating spaces adapted by George Thuruthel et al. [17]

Unfortunately non idealities like slackening, tendon load coupling, friction effects, bulking, death zones, torsion, hysteresis, backlash will result in non linear behaviours that are difficult to predict. Number of DoFs gets higher and infinite degrees of freedom should be introduced to take into account every stochastic non ideality [33, 55]. To deal with pulling and releasing cable we will work with torque or cable displacement at low level control, motor position can be sensed thanks to motor encoders, potentiometers and these quantities will be mapped to the configuration space as we can see in Figure 2.1. In the following table we summarize the operating spaces for tendon driven continuum robots.

Operating spaces

Operating space	Definition	Description
Actuator space	$q \in \Re^k$	q proportional to eg. motor position or torque k = number of actuators
Joint space	$\varsigma \in \Re^l$	ς proportional to eg. cable length or tension $l \geq$ number of actuators
Configuration space	$\zeta \in \Re^m$	ζ proportional to eg. motor position or torque m = number of actuators
Task space	$x \in \Re^n$	x proportional to position or pose or forces applied at the end-effector n = dimention of the target variable

Table 2.1: Operating spaces definition for tendon-driven continuum robots adapted by George Thuruthel et al. [17].

2.2.2. Main non-linearities

In order to make this description more complete, we provide a brief description of non-idealities we referred to in the previous paragraph.

Slackening occurs when the tension is decreased and tendons can go out of the rails. Tension on cables is essential and we can find control approaches focusing on tendon tension optimisation [55].

Backlash is caused by clearance, or play, between mating parts, which introduces a dead band when the direction of travel is reversed [37]. In the dead band, no movement occurs until the clearance between mating parts is eliminated. In general dead zone is a range of inputs at which the output is insensitive (we rotate the shaft but there is no displacement of the robot)[37].

Buckling is a sudden change of properties. The system releases suddenly causing instability [56].

Friction effects due to the displacement to the cable in its guide [1, 23].

Coupling of actuation between tendons or segments of the robot can appear. Tendons can couple their pulling action on the shaft giving birth to unmodelled and non-linear effects or for example coupling actuation between sections can cause a change of the shape without encoders detecting a change in tendon length or tension [10].

Hysteresis is most often associated with magnetic systems and manifests in electrical motors as hysteresis loss. Simply stated, hysteresis is the relationship between the reaction of a material to an initial load (or magnetizing force) and the material's recovery once the load (or magnetizing force) is removed. Hysteresis also affects the behavior of drive shafts in mechanical systems. When torque (a torsional force) is applied to a shaft, it produces an internal stress and causes the shaft to change shape. This change in shape is referred to as strain (or, torsional strain, in the case of torsional loading). In perfectly elastic materials, the relationship between stress and strain is linear. But few materials are perfectly elastic, and the in-elasticity of materials gives them a non-linear stress-strain curve. This non-linear behavior as forces increase and decrease is referred to as hysteresis. For exmaple, piezo-actuators, which rely on material strain to produce motion, can experience hysteresis of 10 to 15 percent of the commanded movement. Operating piezo-actuators in a closed-loop system can reduce or eliminate hysteresis effects [9, 58].

All these non-idealities will negatively act on control systems decreasing both accuracy and stability. In detail hysteresis deeply affects tendon driven continuum robot. It appears as

result of backlash, friction and other non-linear effects. We will describe how the research is facing these issues and challenges exploiting visual feedback information.

2.3. Feedback information

A crucial aspect in the control definition is the information we have available from the task space. Given these modelling issues and high number of extra degrees of freedom sensors information becomes essential since feed-forward model control approaches might lead to huge inaccuracies [33]. Among these sensors several viable options are available:

- Optical sensors based on fibre Bragg grating (FBG) [19];
- Electromagnetic tracking-based sensing [54];
- Imaging sensors (as we will see deeply in the next paragraphs).

FBG sensors does not provide high accuracy in three dimensional (3D) deflections due to challenges in their accurate placement and they decrease the lower strain transfer of the robot [48]. EM tracking sensors are very sensitive to interference in presence of electronic circuits and ferrous instruments, their information would be no more accurate. Moreover they are accurate only within their limited working envelope [48]. As final consideration any sensor or additional element will modify the properties of the robot, especially its dynamic. Therefore exploiting a source of information already present by definition within the device we are developing can be a good choice [33].

Imaging sensors has noticeable importance, particularly in medical scenarios. Since they are contactless sensors already inside the majority of the minimally invasive instruments such as endoscopes, they don't require extra hardware or modifying the device characteristic. In this perspective, exploiting visual information feedback to provide real-time feedback to the controller to drive the robot appears among the best choices to build controllers with robustness to uncertainties in kinematic and dynamic modelling.

2.4. Visual sevoing control

Visual servo control of robots is a control approach where the robot is controlled based on visual feedback provided by the imaging device/system [33]. The error between the provided and desired visual information is calculated to generate the control command for the robot's actuators. Most of the visual servoing approaches process the image in order to gain some information. In detail, features of the image are considered [21]. An image feature can be defined as any structural feature than can be extracted from an image

(such as edges, corners...). Starting from these, we can compute real valued quantities called feature parameters such as coordinates of points in the image. Thus given an image $\boldsymbol{I}$ with k image feature parameters, we can define the space F of feature parameters so that $\boldsymbol{f} = [f_1, f_2 \cdots f_k]^T \in \Re^k$. Where f_i is the $i-th$ feature parameter. For example if we consider a bi-dimensional image and a point position in the image space, we will get $f = [u, v]$ as feature vector with (u, v) coordinates of the extracted point in the image reference frame.

According to the space in which visual servoing is realised, vision-based control schemes can be divided into [21, 33, 47]:

- Position-Based Visual Servoing (PBVS), that realizes visual servoing in operational space;
- Image-Based Visual Servoing (IBVS), that realizes visual servoing in the image space;
- hybrid visual servoing that combines the characteristics of the previous two approaches

The main difference lies in the fact that the schemes of the first category use visual measurements to reconstruct the relative pose of the object with respect to the robot, or vice-versa, while the schemes of the second category are based on the comparison of the feature parameters of the image of the object between the current and the desired pose.

IBVS establishes a non-linear mapping relationship between the image feature error and the posture of the robotic manipulator. Since we exploit image features and the control law is in the image feature space, IBVS control is 2D and is not sensitive to calibration error. Instead PBVS is a 3D control method: the control law is designed in Cartesian space utilising a 3D interpretation of the image features extracted from the captured images. The 3D interpretation is often called pose estimation or 3D reconstruction. PBVS defines the error signal for the posture between the end-effector of a robot and the target object in a Cartesian coordinate system and calculates the position and posture of the end-effector against the target object using the robotic kinematics model and camera calibration. Thus small errors in the image measurements can induce quite different results.

Before inspecting them we point out an interesting aspect that will appear in the analysis. According to the configuration, that is where the camera is positioned in the control loop to observe the scene of interest, we distinguish [21]:

- Eye-In-Hand (EIH);

- Eye-To-Hand (ETH).

In EIH, the imaging device is mounted on the end-effector or the robot's tip to observe the scene. In ETH, the imaging device is mounted on a fixed base to observe the scene of interest, including the robot. As we will see both these configuration are exploited by several authors. During the analysis we will point out which configuration will be exploited and its advantages bearing in mind that in case of intraluminal navigation that is the aim of ATLAS project ETH approach won't be feasible.

2.4.1. PBVS

In order to formalise what we stated earlier, we define for example a positioning task. In this approach, the information is translated in the task space T, which is defined as the set of positions and orientations that the robot tool can attain. It is the configuration space of the robot tool that for sake of generality we will consider of dimension m. In detail positioning task is represented by a function $\boldsymbol{E} : T \rightarrow \Re^m$, called kinematic error function. This task is fulfilled with the end effector pose $\boldsymbol{x}_e$: $\boldsymbol{E}(\boldsymbol{x}_e) = 0$. In this approach, we estimate the end effector and the target poses and we compute the error function searching for end effector pose which make the error function null. Let's inspect some examples of this approach for continuum tendon driven robots.

We report three examples [10, 49, 53] of PBVS approach, these are all characterised by an ETH approach since this makes easier the definition of the problem in the task space.

In [10] authors considers the Clemson Elephant Trunk composed of 16 joints each one characterised by 2 degrees of freedom divided in 4 tendon driven sections each one thought to bend with a constant curvature. They exploit a fixed camera (ETH configuration) to be able to detect the position and shape of their tendon driven continuum robot. They adapt the D-H convention for rigid manipulators to in order to be able to exploit the classical D-H homogeneous transformations between links and build a kinematic model of the robot. They set the reference systems at the beginning and at the end of each section and they come up with an end-effector position and orientation in the task space of the robot (end effector pose, $\boldsymbol{p}$) as function of the joint space variables ($\boldsymbol{q}$) as we see in:

$$\boldsymbol{p} = \boldsymbol{f}(\boldsymbol{q}). \tag{2.1}$$

and computing the first derivative:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}(q) \times \dot{\boldsymbol{q}}. \tag{2.2}$$

Where:

- $\boldsymbol{f}(\boldsymbol{q})$ correspond to the forward kinematics;
- $\boldsymbol{q}(t)$ is the joint variable vector;
- $\boldsymbol{p}(t)$ is the end-effector pose;
- $\dot{\boldsymbol{q}}(t)$ is the joint velocity vector;
- $\boldsymbol{J}(q)$ by definition is the Jacobian matrix equal to $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}$

Exploiting a pinhole model for the camera and an euclidean reconstruction they are able to get the vector $\boldsymbol{q}$ for the camera image so now they know the Jacobian matrix. They define the error in the task space as:

$$\boldsymbol{e} = \boldsymbol{p} - \boldsymbol{p}_d. \tag{2.3}$$

Where:

- $\boldsymbol{e}$ is the error;
- $\boldsymbol{p}$ is the actual end effector pose;
- $\boldsymbol{p}_d$ is the desired end-effector pose (constant);

Thus:

$$\dot{\boldsymbol{e}} = \boldsymbol{J} \times \dot{\boldsymbol{p}}. \tag{2.4}$$

They solved the equation providing a solution exploiting the pseudo-inverse matrix and an additional element tailored on the specific task. They provided a simulation to show the convergence of the task error to zero. They underlined how this approach might need multiple cameras in real applications to avoid occlusions of the line of sight.

In [53] authors exploits an ETH configuration by using a stereo-camera system to collect information about their robot tip in order to increase the accuracy in positioning and control. As we can see in Figure 2.2 their device is a tendon driven continuum robot characterised by 2 degrees of freedom each one. These segments are assumed to bend with a constant curvature. Moreover they are attached on a rigid segment at the bottom which provides a translation and rotation degree of freedom.

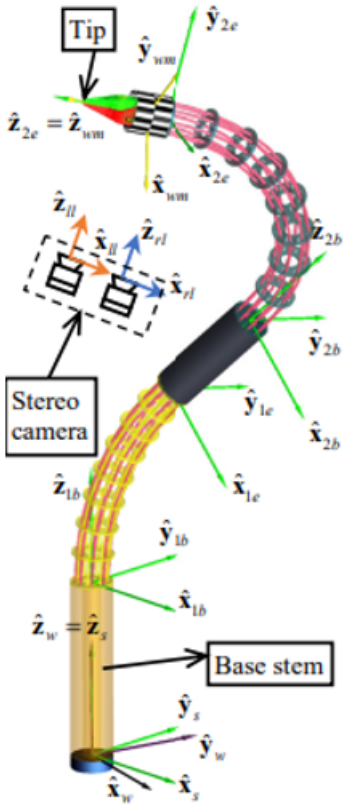


Figure 2.2: Schematic representation of the device adapted by Yang et al. [53].

They model and compute the kinematics of the robot and thanks to calibration they know the transformation matrix linking the camera reference frame with the reference frame of the laboratory. To detect the tip of the device they exploit a custom check-board wrist marker. To extract the marker position from the camera images they pre-process the image and then exploit a corner detection algorithm based on a corner likelihood method. After locating the corners in the left and right images of the stereo endoscopic camera they exploit the known dimension of the wrist marker to estimate the tip position in the 3D space by optimisation.

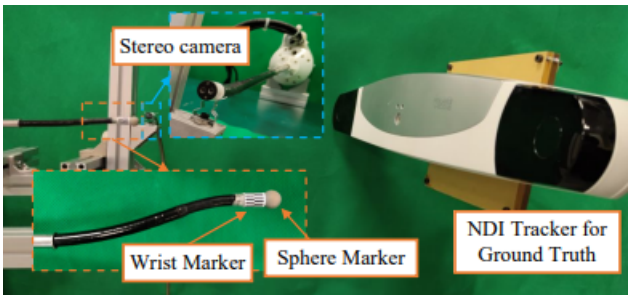


Figure 2.3: Experimental set-up adapted by Yang et al. [53].

Now they are able to compute thanks to the camera the actual tip position in the 3D space ($\boldsymbol{p}_c$) while the desired tip position ($\boldsymbol{p}_d$) is provided by the surgeon. Moreover they know the jacobian matrix linking the velocty vector of the end effector in the task space and the velocity vector in the configuration space. At each feedback loop iteration they solve the system searching for the Moore-Penrose pseudo-inverse matrix of the jacobian and, computed the configuration velocity vector, they compute the new configuration vector (composed of the joint variables) adding to the previous vector a value equal to the just computed configuration velocity multiplied by a temporal variation equal to the servo cycle time (Figure 2.4).

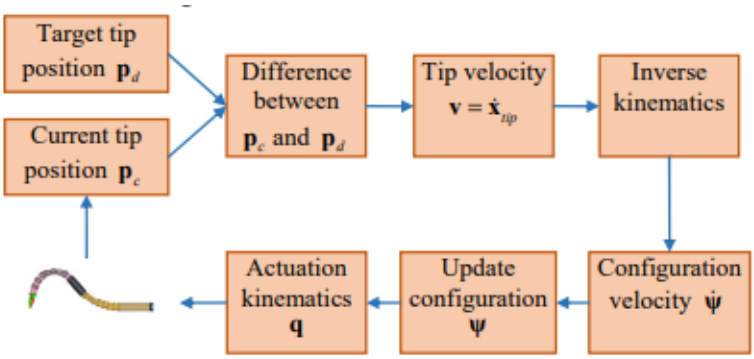


Figure 2.4: Closed loop control algorithm adapted by Yang et al. [53].

As we see in Figure 2.3 they tested their control algorithm. First they tested the position estimation accuracy. By measuring the real tip position thanks to a passive marker and an NDI optical system they were able to compare the real 3D position of the tip provided by the NDI system with the position computed by the stereo-camera. They measured an average sub-millimetric error. Then compared the trajectory tracking performances between an open loop and the suggested closed loop control configuration. The latter one decreased both the mean and maximum tracking error to 25.23% although overshoot during direction changes is present probably due to a too long control cycle.

In [49] authors built a vision-based motion control framework with ETH configuration for their flexible surgical robot. They robot is actually composed of a rigid robot (KUKA LWR) and a flexible probe attached at the end effector of the rigid arm. The flexible tip is tendon driven actuated and it is characterised by a single bending degree of freedom.

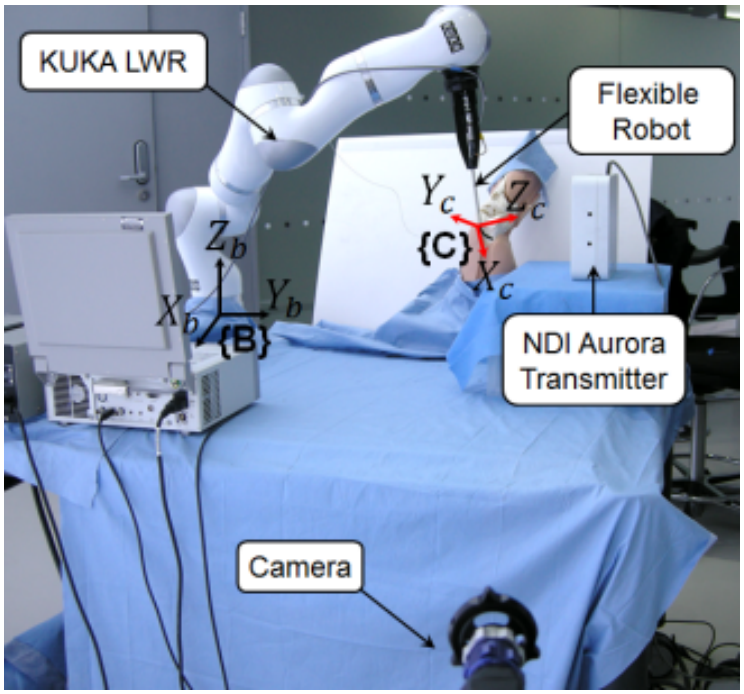


Figure 2.5: Experimental set-up by Vandini et al. [49].

Authors split the control task. They handle the positioning of the rigid arm with a Cartesian impedance control to guarantee safe interaction of the rigid arm with tissues. While they handle the positioning task of the flexible part with a PID controller taking as error (e) the difference between the desired motor velocity (θ_d) and the actual motor velocity (θ). Thus the command torque (τ_M) at motor level is:

$$\tau_M = K_P e + K_D \dot{e} + K_I \int e(t)dt. \tag{2.5}$$

The desired velocity is the velocity needed to bend the flexible portion to a desired angle β_d. By exploiting a vision algorithm authors compute the shape of the flexible probe from the camera image. Since the 3D position of the base of the probe is known thanks to the KUKA forward kinematics and the probe movement is limited to a plane due to the single bending degree of freedom, they can reconstruct the shape and the position of the tip of the probe computing the actual bending angle β. They performed some experiments setting a desired bending position β_d and comparing the actual angle β computed both

with the visual technique (VS angle) and EM Aurora sensor (EM angle). As we see in Figure 2.6 they reach a good accuracy for the angle estimation (VS and EM angle are very close to eachother) and the control algorithm activated at frame 10 with a frequency of 3 Hz reaches a steady state error inside the tolerance band (red lines).

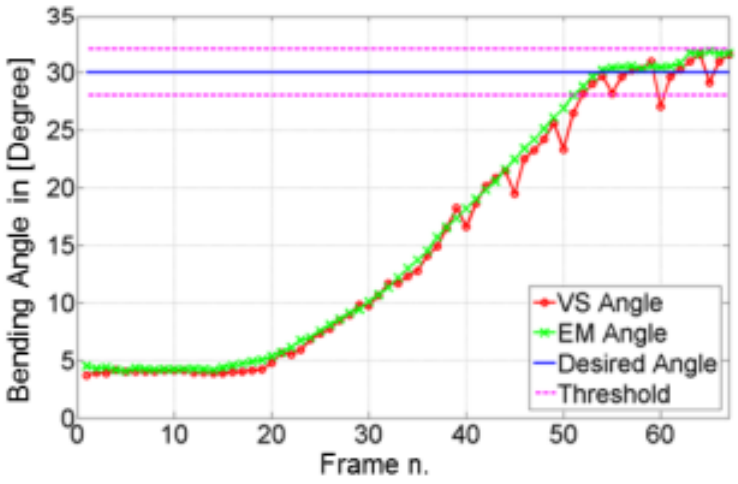


Figure 2.6: Experimental results by Vandini et al. [49].

2.4.2. IBVS

IBVS is a very common approach that doesn't consider the error signal in the task space coordinates, instead it we will be defined directly in terms of feature parameters. The image-based visual servoing task is represented by an image error function $\boldsymbol{e} : F \rightarrow \Re^l$ with $l \leq k$. This function must be defined in such a way that $\boldsymbol{e} = 0$ when the task is achieved. For example in a centering task we can consider $\boldsymbol{f}$ as the actual feature vector parameter and $\boldsymbol{f}_d$ as the desired feature vector parameter: for the centering task they will correspond to the position of the target object in the image and the centre of the image respectively. Now we define an error function $\boldsymbol{e} = \boldsymbol{f} - \boldsymbol{f}_d$ in the feature image space F[21].

The error, $\boldsymbol{e}$, is defined on the image parameter space, but the manipulator control input is typically defined either in joint coordinates or in task space coordinates. Therefore, it is necessary to relate changes in the image feature parameters to changes in the position of the robot. In order to do this a new jacobian matrix is defined: the image Jacobian ($\boldsymbol{J}_v$). If we consider a task or actuation space of dimention m and a feature parameter space of dimention k, we have:

$$\dot{\boldsymbol{f}} = \boldsymbol{J}_v \times \dot{\boldsymbol{r}}. \tag{2.6}$$

$$\boldsymbol{J}_v = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{r}}\right]. \tag{2.7}$$

$$\boldsymbol{J}_v = \begin{bmatrix} \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{r}_1} & \cdots & \frac{\partial \boldsymbol{f}_1}{\partial \boldsymbol{r}_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{r}_1} & \cdots & \frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{r}_m} \end{bmatrix}. \tag{2.8}$$

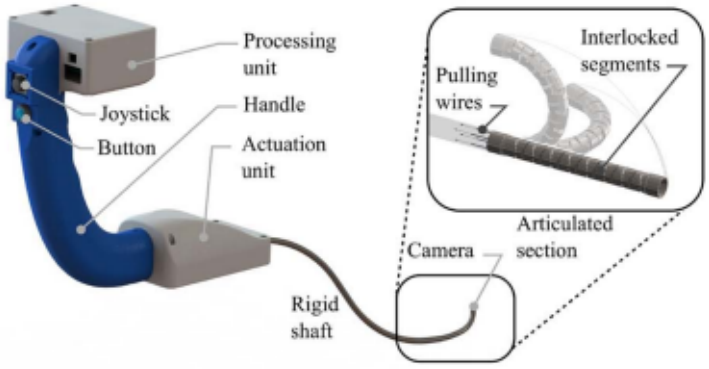
Where:

- $\boldsymbol{r}$ is the end effector position vector;
- $\boldsymbol{f}$ is the vector of feature parameters;
- $\dot{\boldsymbol{r}}$ is the end effector velocities vector (velocity screw);
- $\dot{\boldsymbol{f}}$ is the vector of the feature parameters rate of change;

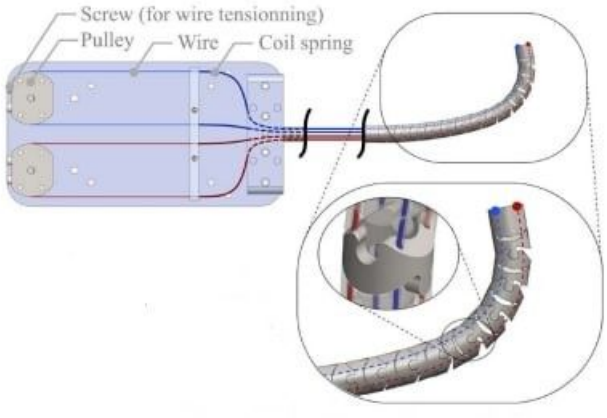
This relation is describing how image parameter change with respect to changing manipulator pose. This will allow us to come up with a control law function of the error in the feature space. Let's start with some applications of this approach.

This approach is robust against calibrations errors since it doesn't require to compute the transformation matrix describing the position of the target in Cartesian space with respect to the end effector pose.

The first example we describe is REALITI, A Robotic Endoscope Automated Via Laryngeal Imaging for Tracheal Intubation [3]. Boeheler et al. built a continuum tendon driven endoscope actuated by two pairs of antagonistic pulling wires pulled by two DC motors provided by optical encoders for position feedback and an EIH camera.



(a) Real device appearance.



(b) Schematic view of the actuation elements.

Figure 2.7: REALITI device by Boeheler et al. [3].

This design allows two bending degrees of freedom and the precise manufacturing allows authors to derive a Jacobian matrix ($\boldsymbol{J}$) to map motors velocty into the task space, distal angular velocity of the articulated section. Authors define a positioning task in the image space. In detail, a visual processing of the endoscopic camera images extracts the position of an anatomical landmark of interest, the glottis. They establish then a centering task where feature parameters vector is the position vector of the anatomical structure in the image ($\boldsymbol{p}_f$) and the desired position is the central point of the image ($\boldsymbol{p}_c$). At this point they are able to define an error function ($\boldsymbol{e}_p$) expressed in pixel. The visual controller is a PID with anti-windup.

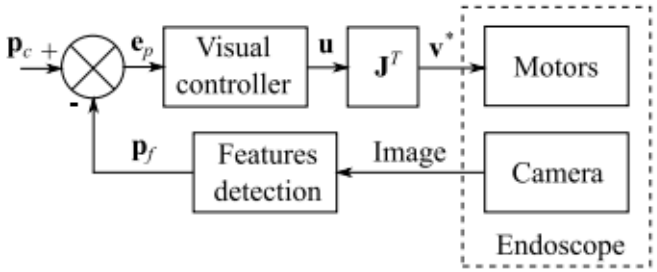
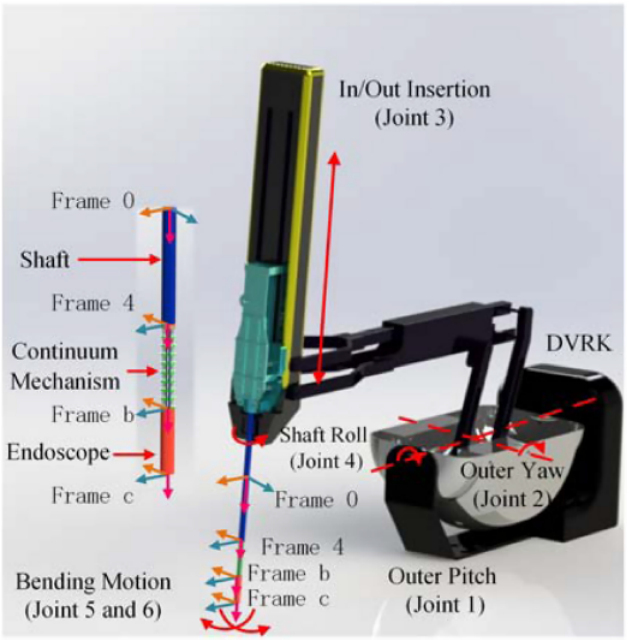


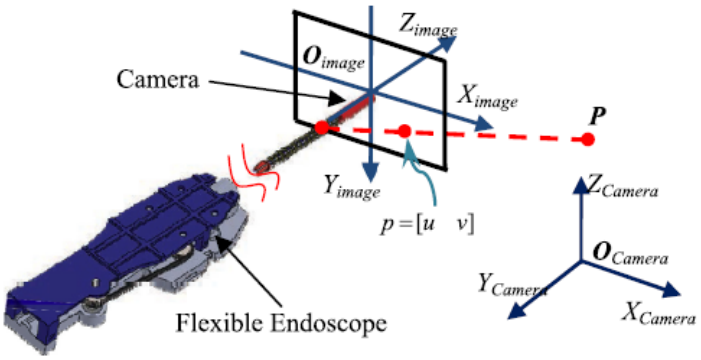
Figure 2.8: Visual servoing control scheme by by Boeheler et al. [3].

They analyse the performances keeping fixed the handle and inspecting the response of the controller to a step. The target is set in a shifted position from the center and then the visual servoing control is activated. The controller is stable and reaches a steady state error of 5 pixels, being able to center the glottis in 3 seconds . When the target is lost, the desired motor velocity is forced to 0 in order to solve the problem of missing target. The main problem is the backslash in the wire transmission and the play in the articulated structure of the device. Thus the motion in response to the step input starts with a delay and motor have to rotate the pulley of 4° to make the robot start bending. So nonlinearities are causing the main issues about performances since their jacobian matrix and controller are linear.

Another example of IBVS approach exploited to control a soft continuum robot is provided by Ma et al. [28]. Authors built a IBVS control algorithm for 6-DoF flexible endoscope. Their platform consists of flexible tendon driven endoscope with 2 degrees of freedom (called tendon-driven continuum mechanism, TCM) linked to a 4 DoF rigid manipulator (dVRK, da Vinci Research Kit).



(a) Complete platform.



(b) Flexible endoscope and image plane reference system.

Figure 2.9: Continuum endoscopic device adapted by Ma et al. [28].

As we see from Figure 2.9 a EIH configuration was exploited. The camera is set at the tip of the endoscope. Authors exploited green markers to be able to extract the position of a surgical instrument in the image space. So in this case their feature parameters vector is the position of the markers in the image space (u, v), their desired feature is the centre of the image since they aim to track the target and the error is defined as the distance between the centre of the image and the markers position in the image space. Exploiting a kinematic model of the device and a pinhole model to describe how a point in the physical space is projected on the image space they are able to estimate the image Jacobian. The robot is clearly redundant so they set an optimisation problem: they search for a visual

servoing control law which is able to minimize the absolute motion in the joint space. Thanks to the optimisation approach they are able to track the target and minimize the error around 10 pixels in less than 2 seconds. Target depth not provided by the 2D image is now estimated by an optimisation problem and then they aim to compute the next joint vector variable value solving/reducing the image error and at the same time forcing the robot to occupy less space possible. Due to assembly and manufacturing errors, to cable frictions and plastic deformation of the wires there might appear non-idealities like unexpected deviations of the view of the endoscope when it is rotated. Authors tried to solve this problem adding a second endoscopic camera on the tip of the device [29] exploiting a similar optimisation approach. Now they are able to computed the distance between the target and the camera in a precise way since a depth and rotation control are added to make the system more robust.

An interesting approach to the controller design is carried out by Ott et al. [37]. They built a IBVS visual servoing control with an EIH configuration in order to face motion breathing oscillations to keep a target in the desired position in the image space. They started by the observation the breathing pattern and found out it can be modelled as sinusoidal signals if patients are under anesthesia and breath control. They built a controller able to reject disturbances within a specific frequency band. They experienced and faced the backlash problem. When motors are submitted to forth and back motion there is no change of the tip of the robot inside a certain range of motor input values, called death band. This is due to the mechanic coupling, to the friction of wires in the guidelines. Moreover it depends also on the soft shaft configuration. Thus it is necessary to quantify the backlash for the different working points and it is not possible to assume an ideal model of the backlash as the black straight lines in Figure 2.10 since non linear behaviour is present. In this case the endoscope has two bending degrees of freedom and authors here reported the position of a fixed target in the image while changing the motor input which causes a motion in the image space parallel to the y direction of the image space. The red and blue curve are experimental data recorded at different values of shaft bending so for different values of the other motor.

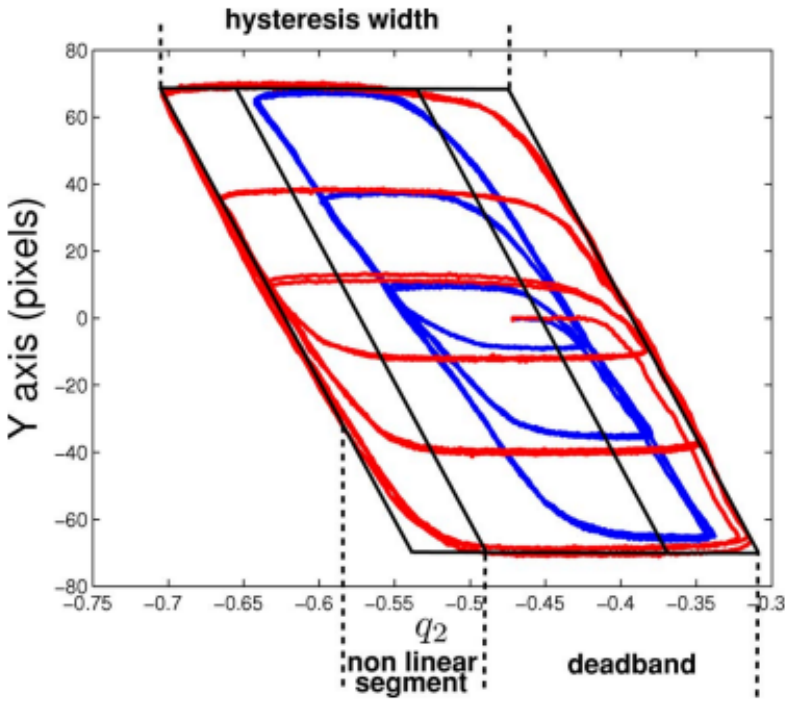


Figure 2.10: Backlash hysteresis adapted by Ott et al. [37]

In order to solve this problem they included a compensation in the control scheme. Their controller was able to converge to the correct position but this backlash added an error every time a direct inversion of the direction was present. Moreover to correct this additional error, a lag in the control loop is introduced. To solve this problem they states that before every use, backlash calibration must be performed. An open-loop back and forth movement recording the position in the image of a fixed target for each axis independently to compute the hysteresis band which will be added to correct the input on the motor.

Finally we cite the work by Yip et al. [55]. They explored a different approach avoiding the definition of a kinematic model for their tendon driven continuum manipulator in order to avoid the approximations due to non-idealities and external disturbances that are almost impossible to be accurately modelled. They enhanced how it is very difficult to model a continuum robot in a constrained environment since it has infinite degrees of freedom and sensing all of them is impossible. This can lead to a scaling in the singular values of the Jacobian matrix and to an orientation change of the column vectors as we can see in Figure 2.11. We see how in the constraint motion since we can't sense the obstacle the model assume the robot to be in a pose different from the real one since the only values that are available are the cable displacements. This might bring to a positive

feedback loop damaging the robot.

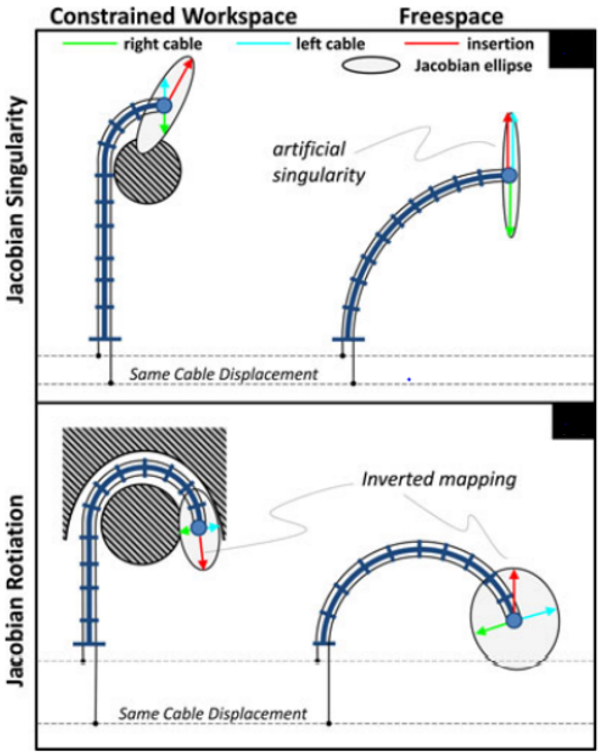


Figure 2.11: Jacobian scaling and rotation adapted by Yip et al. [55]

In order to solve these issues, authors exploits a ETH camera configuration to acquire the end effector position and pressure sensors to sense tension on cables. They built a tendon driven robot where two tendons are independently actuated and a separated insertion mechanism is exploited. They established a visual servoing control technique solving an optimisation problem. At each cycle they estimate the Jacobian matrix for the next step as the old Jacobian matrix value plus a variation obtained by the minimization of the Froebenius norm of the Jacobian matrix. So at the next step, they have the jacobian to compute the next actuation input searching for the input variation which pre-multiplied by the estimated jacobian gives the desired change in the position output. This approach was proved to be able to deal with external perturbances that can rotate and scale the Jacobian matrix. They performed experiments in free space proving that the method is able to reach a null steady state error in the camera space. They tested the controller even in a constrained environment where the robot is able to adapt the cable tension to reach the final target position even when touching an obstacle. Unfortunatly the controller is affected by a phase lag due to the estimation of the jacobian on past data and in case of hard contacts of the tip with an obstacle velocity can suddenly drop to zero resulting in an ill conditioned jacobian. In order to better handle contact interaction in a following work

[56] authors added a force sensor to be able to integrate a contact estimation introducing an hybrid position/force control.

2.4.3. Hybrid methods

This kind of approach aims in exploiting both IBVS and PBVS techniques. This can be realised in several ways for example switching from one approach to the other according to some parameters values or splitting the joint variables in two subsets and defining a PBVS approach for one subset and a IBVS approach for the other. Here one example of hybrid approach built to control a continuum tendon driven robot is reported. In detail authors [59] focus on the visual servoing control of an endoscopic tendon driven robot actuated by two couples of wires providing two bending degrees of freedom. In addiction a probe able to provide Optical Coherence Tomography (OCT) images is inserted in the working channel and is characterised by three degrees of freedom (rotation, bending and insertion). Thus the total system has five degrees of freedom.

The Authors want to exploit IBVS and PBVS at the same time. Their goal is reaching a point on a surface in order to acquire OCT images of that point of the tissue. In order to acquire good images with OCT techniques the OCT camera must be parallel to the surface and as close as possible at the surface point (ideally touching it). In Figure 2.12 we can see a schematic representation of a typical configuration with the tissue surface with a point we want to reach. The target point is the surface point set at the origin of the unitary vector $\boldsymbol{m}$ orthogonal to the surface. We can notice even $\boldsymbol{n}$ which is the unitary vector orthogonal to the direction of the acquisition range (OCT-camera acquires data on the direction orthogonal to $\boldsymbol{n}$) and points $\boldsymbol{p}_{i,k}$ and $\boldsymbol{p}_{i,d}$ which are the projection on the endoscopic image plane of the OCT-camera and of the desire point on the surface respectively.

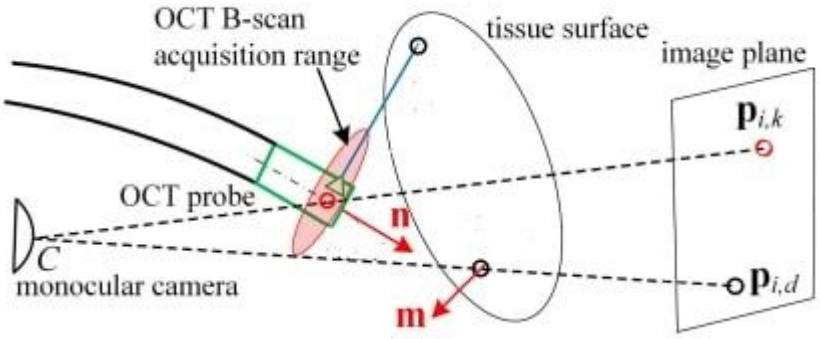


Figure 2.12: Scheme of the endoscopic camera and OCT-probe by Zhang et al.[59]

The visual servoing positioning task is set as an optimisation problem. They split the problem in several tasks. Each task is translated in an optimisation problem and finally they are combined together in a unique quadratic programming problem. The two main tasks as previously stated are:

- moving the OCT camera as close as possible to the target point on the surface (reducing the distance along the vector $\boldsymbol{m}$ ideally to 0);
- forcing the correct orientation so that $\boldsymbol{m}$ is orthogonal to $\boldsymbol{n}$ and reducing the distance in the $\boldsymbol{n}$ direction.

In order to solve the first task an OCT-based approaching speed control for contact detection is implemented defining the error in the task space while to solve the latter one they exploit an error function defined in the image feature space with an image jacobian. With this approach Zhang et al. are able to reach sub-millimetre accuracy in terms of average position error. The main issue which increases the error is the non linear behaviour of the device. They work considering the device behavior as it is linear but they face the backlash nonlinearity effect during changing direction which increases the positioning error that is later recovered.

Combining PBVS and IBVS approaches shows an excellent accuracy both in simulation and experimental evaluations in the example we analysed. Unfortunately even combining the advantages specific of the two approaches the issues due to the nonlinearity of the tendon driven continuum robot are still decreasing controller performances. Thus in the next paragraph we will report some new trends and techniques exploited to solve this issue introducing for example deep learning to model some non-linear behaviours of TDCR.

2.4.4. Deep learning to model nonlinearities

We want to spend some word about non-idealities and the advance of a new trend to deal with them. As we were able to notice from previously examples all these nonlinearities give birth to non linear effects as we can well understand from Figure 2.10. As reported by [51], the study of a complex non-linear multi-valued relationship between input commands and the end-effector might benefit by deep learning. In detail Wu et al. inspected the use of Long-Short Term Memory (LSTM) neural network to model hysteresis effect in a Pneumatic Artificial Muscle (PAM) continuum catheter [51]. Hysteresis is a phenomenon which takes into account of the present value and past values. LSTM is able to keep memory of events and predict future values given a sequence of past values. The authors built an LSTM network and tested it on simulated data obtained building an hysteresis analytical model with random noise superimposed. They found that LSTM is able to

predict correctly so the trained and tested the network on real data giving inputs varying frequency and amplitude to the catheter actuation. In a next work [52] they inspect deeper in this approach. Exploiting the same hardware they sent different pressure inputs to the actuation and recorded with a laser photoelectric sensor the bending angle of the catheter. They collected data, trained the network and compared its prediction performances over new data with the predictions obtained by the Dependent Prandtl-Ishlinskii analytical model and a Support Vector Regression (SRV) model for hysteresis. LSTM gave smaller error with all metrics they exploited. Thus deep learning method appears to be a promising approach to model history dependant phenomena.

2.4.5. Final considerations

Thank to the overview reported in the previous paragraphs, we understand how continuum robots introduce many challenges about controller design. This kind of devices increases compliance and improves safety in minimally invasive-surgery but Continuum and compliant structures increase the number of degrees of freedom making the description of an exact analytical model almost impossible. Sensing these degrees of freedom is cumbersome and non idealities like backlash, friction introduce non-linearities that are hard to be predicted. The most evident nonlinearity for TDCRs is hysteresis effect since this problem causes lags and decreases the accuracy of the control system. Hysteresis can somehow be modelled thanks to analytical approaches in order to compensate for it and this compensation and modelling can be improved by the advance of deep learning.

In the next chapters of this work we will firstly describe our continuum robot Atlascope, build the electronic hardware and the software infrastructure in ROS environment to drive it and after that we will design a controller exploiting the visual feedback information to build a control system for Atlascope and build a deep learning model to be able to predict non ideal behavior of our robot. Camera sensor is embedded in our device by definition of endoscope and as we discussed visual information has been largely studied and it is proved to be effective in the continuum tendon driven control. Thus we will build and validate a visual servoing control system to accomplish centering and tracking task and we will define and trained a deep learning model in order to be able to predict for hysteresis effect.

3 | MATERIAL AND METHODS

In this chapter Atlascope prototype and the electronic hardware driving the robot are firstly presented. After this the software environment used to handle the data communication and the implementation of the control algorithm will be introduced. The controller design and implementation for the robotic platform will be described in detail. Finally the deep learning model exploited to predict the robot behavior will be defined. Moreover we report the metrics we will exploit to evaluate our controller and our model performances.

3.1. Atlascope

As it was earlier anticipated, Atlascope is a tendon driven continuum robotic endoscope [25]. It is a prototype built inside the European project ATLAS.

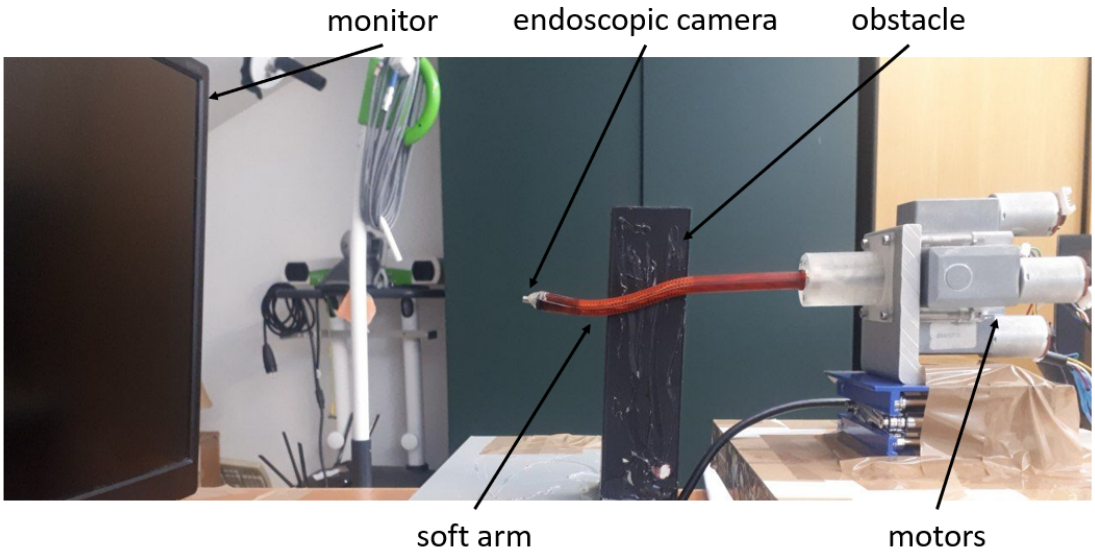


Figure 3.1: Complete set up.

As we can see in Figure 3.1 and in Figure 3.2 the endoscope is characterised by a proximal rigid segment of 58 mm length and by the a flexible robotic arm of 70 mm length. The robotic arm is a soft non assembly 3D-printed structure based on the HelicoFlex

design mechanism [12]. The diameter of the arm is 10 mm in order to make easier the installation of the desired sensors and the camera. In detail, four channels running along the arm are manufactured in order to allow the passage of cables and a cup at distal tip is designed to host the camera module. Currently an endoscopic camera of 1280x720 pixel of resolution and an acquiring frequency of 30 Hz is mounted. Arm dimensions are currently matched with a colonoscope device but the size can be easily reduced exploiting the 3D-manufacturing technology.

Figure 3.2: Flexible arm.

Focusing on the actuation of Atlascope, it is a TDCR characterised by four driving cables inserted trough the cable guides and glued at tip level. These cables of 0.4 mm diameter section are free to move inside the guides and at proximal level they reach the motors.

Cables on the opposite side of the shaft are anchored antagonistically on one pulley (Figure 3.3). Each pulley is rotated by a DC motor. Two DC motors (JGY370-30RPM, Walfront) characterised by a self-locked worm gearboxes with a nominal torque of 7.4 Kg-cm and maximum 30 RPM are exploited. Each motor pulley (thus each motor) is responsible for the steering of an antagonistic couple of cables controlling one Degree of Freedom. Each DC motor is provided by an hall effect incremental encoder. In addiction the whole structure is fixed onto a linear stage with 20 mm (EBX1204-200, Garosa) stroke and a stepper motor (23SSM2440-EC1000, ACT MOTORS) to provide a degree of freedom to allow back and forth insertion movement.

The robotic arm is then characterised by three degrees of freedom, two in bending directions and one in the insertion.

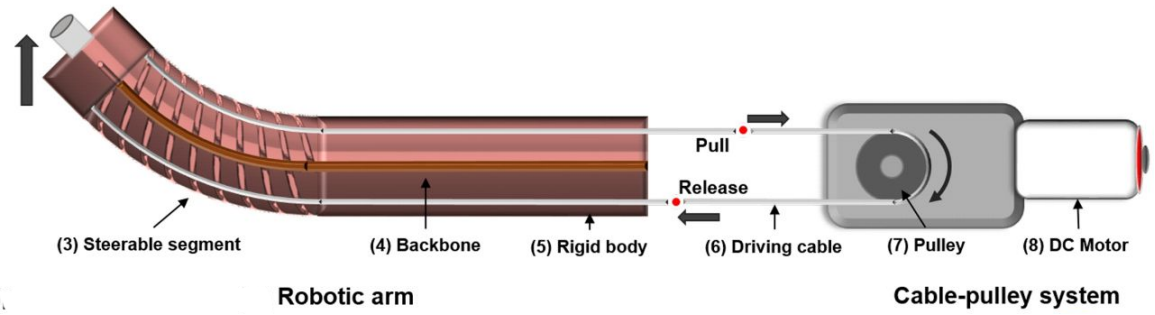
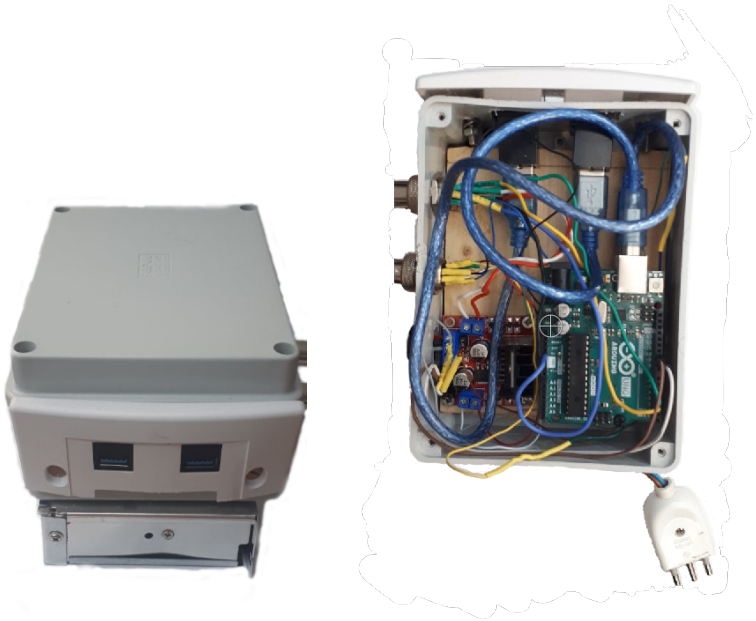


Figure 3.3: Actuation scheme for a single bending degree of freedom adapted by [25].

3.2. Electronic hardware

The hardware is characterised by the two motors and the camera that must be connected to acquire data from sensor and to give power and drive the actuation.



(a) External case. (b) Internal view.

Figure 3.4: Compact hardware box.

In detail, camera is supplied and images are acquired thanks to a USB port. To acquire data and drive the motors we built a compact control box. A DC power supply scaling receives the 220 AC voltage and convert it to DC voltage scaling it to 12 V with a maximum current of 10 A. These current and voltage values are suitable for the motors

driver board L298n. As we can see in Figure 3.5 L298n module (a double H bridge driver) receives the power supply and drives both the motors providing a PWM signal.

Bottom motor (motor 1) allows the robot to move in horizontal direction (x direction, left and right). It is connected to the connector named S (left one) and its USB port is the left one too. The upper Arduino Uno Rev3 handles the motor 1 inputs and the information coming from the Hall-effect incremental encoder of the motor 1 thanks to the uploaded script called *arduino_code_1*.

Right motor (motor 2) allows to displace the robot in vertical direction (y direction, up and down). It is connected to the connector named D (right one) and its USB port is the right one too. The Arduino board handling it is the lower board and the code uploaded is *arduino_code_2*.

We decided to split the control of the two motors exploiting an Arduino Uno Rev3 board for each of the DC brushed motor. Each Arduino board drives a motor and receives information by the incremental encoder.

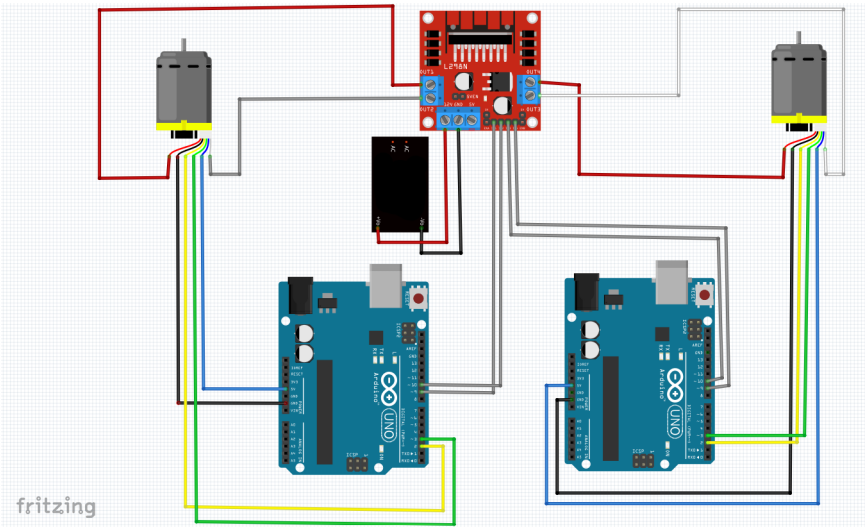


Figure 3.5: Hardware connections.

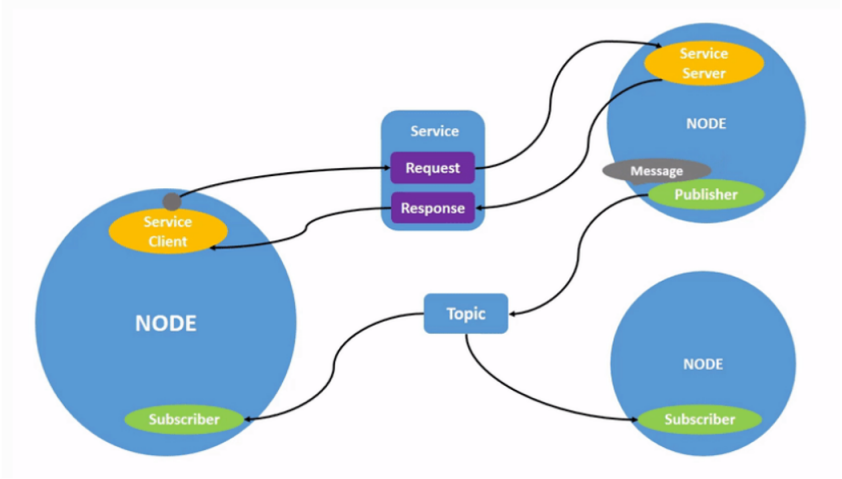
This choice allows us to build an architecture where each Arduino acts as a node sending and receiving information and makes it easier the implementation of two separate low level controllers.

3.3. ROS environment

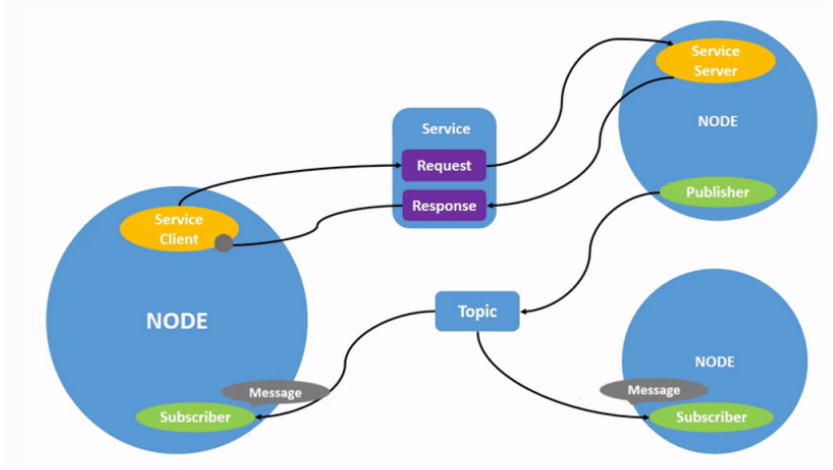
We decided to implement the communication and control architecture in ROS (Robot Operating System) environment. ROS is an open source software development kit for robotics applications. It offers a standard software and several libraries allowing to handle all information in an organised way, considering each device able to send and receive information as a black box called node. ROS environment allows to build a communication infrastructure that is higher with respect to the physical implementation of the device which will then exploit the information. This allows even to solve the issue of different programming languages since ROS has specific packages for all the common programming languages. In this work we worked on Ubuntu 16.04 and the ROS_kinetic version is installed.

3.3.1. Introduction to ROS common terminology

Before going deeper in the communication architecture we implemented we will report some basic definitions exploited in the next paragraphs [43].



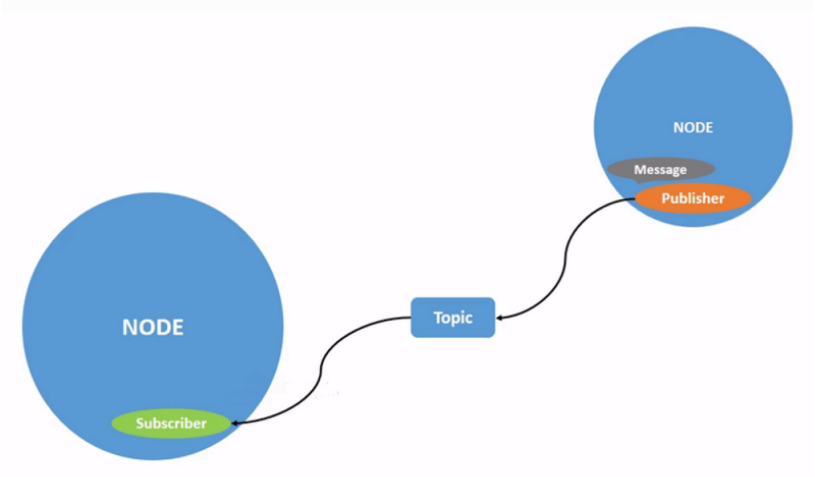
(a) Start.



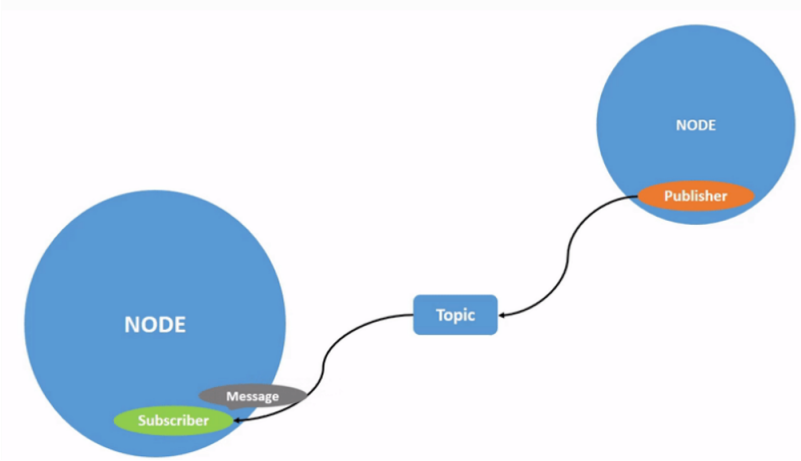
(b) End.

Figure 3.6: Ros communication architecture adapted by [42].

We will refer to a node (ROS node) as an entity able to send and/or receive information (visually represented as a circle). This information is composed of data which can be of different type (integer, floats...). We can define data even as instances of a specific topic. If we imagine we are communicating our position in space, the data we provide is an instance of the position. So we will refer to a topic (visually represented as a square) as the kind of information, a collector where a specific information can be published by a speaker (a node) or where a listener (a node) can get the information. In general a node which publishes an information on a topic is called a publisher while a node which asks a topic for information is called a subscriber (Figure 3.7).



(a) Start.



(b) End.

Figure 3.7: Ros publisher and subscriber nodes adapted by [42].

Several nodes can publish on the same collector topic and a single node can publish over several topics. In the same way several subscribers can read data by a single topic and a single subscriber can read data by several topics (Figure 3.8).

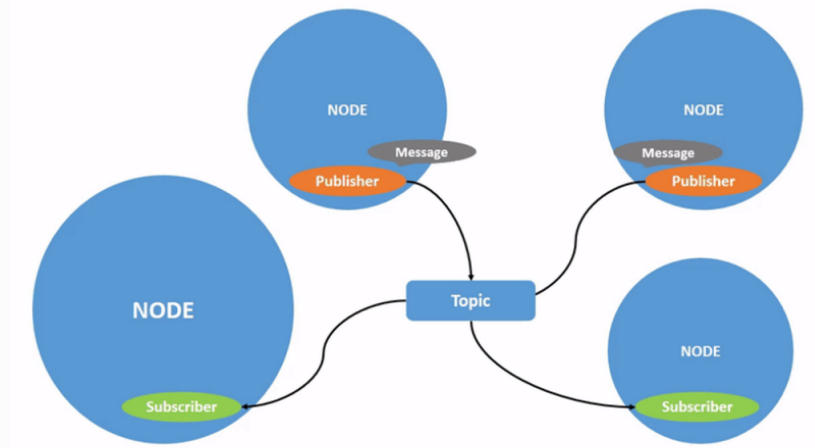
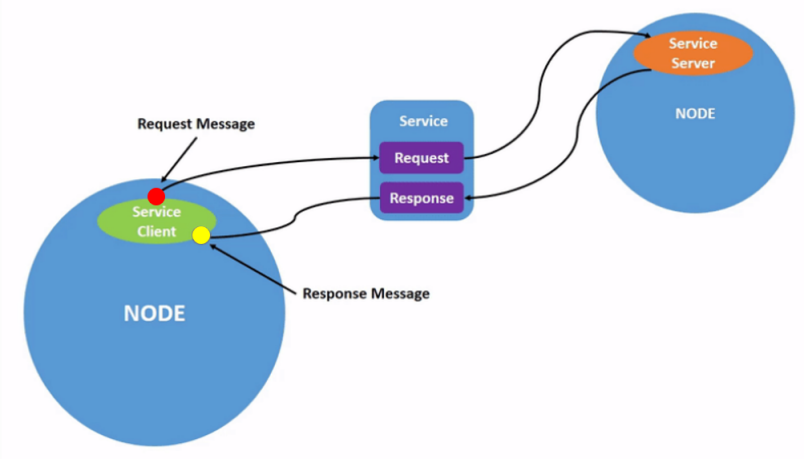
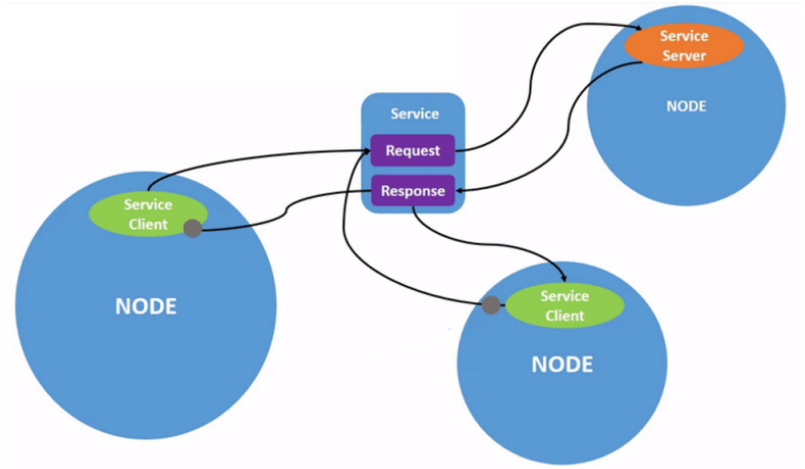


Figure 3.8: Ros multiple publishers and subscribers adapted by [42].

In ROS, it is possible to define more complex entities, here we report two particular node types called service and client. A service node is a node that is able to perform a task every time it is called. For example we can define a service node that is able to acquire two numbers and give back its sum. A client node is a node which typically calls a service, waits for the service to be ready and reads the data provided by the service. Several client can interact calling the same service node (Figure 3.9).



(a) Single couple service-client nodes.



(b) Extension to multiple nodes.

Figure 3.9: Ros service and client nodes adapted by [42].

A node can perform several tasks at the same time, it can publish and subscribe data at the same time and act as a service as we can appreciate in the Figure 3.6.

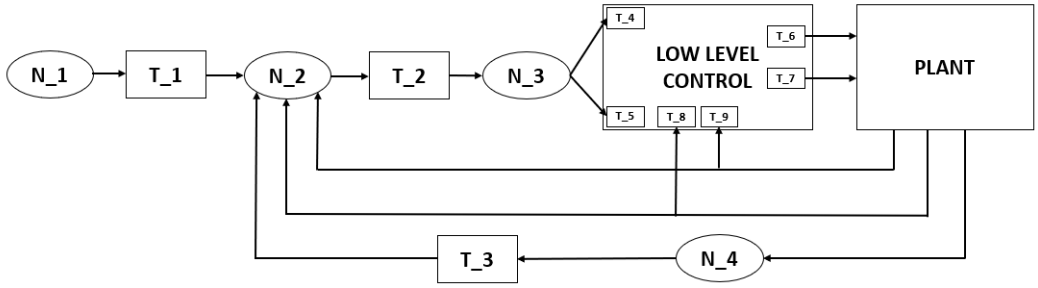
Each publisher and subscriber are characterised by a specific type of data, if a subscriber and a publisher node are acting on the same topic they must be defined to deal with data that have the same type of the topic they are working on. These nodes will keep publishing or reading data with a fixed frequency that can be specified during their definition. Of course the subscriber will wait in case no data have been published yet.

Now we are ready to describe the architecture to handle data communication for Atlascope. We will point out the type of information and the frequency of publishing and subscribing of each node, since frequency is essential in control especially the feedback loop frequency, and all the information needed such as functions exploited to synchronise

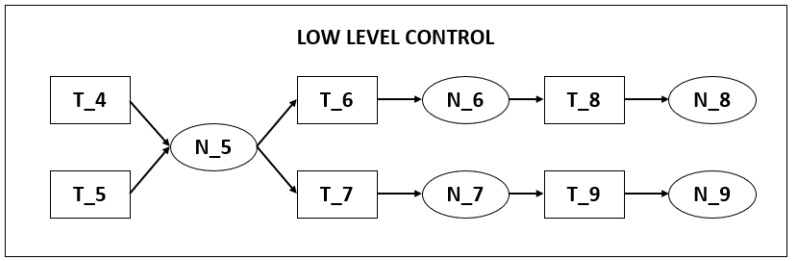
data from different sensors.

3.3.2. Communication architecture

We started creating a workspace and a specific package for our application called *ros_control_example* [11]. A workspace in ROS is typically the folder where all scripts are collected and where ROS will add all the libraries and files needed. Here we set all our scripts (written in C++, Yaml and python). We report here the complete description of the structure with all nodes and topics we exploited. We can see in Figure 3.10 the complete architecture with all nodes and topics. We will exploit abbreviations in Table 3.1 and Table 3.2 to make the notation lighter.



(a) ROS communication architecture.



(b) Detail of the low level control structure.

Figure 3.10: Ros nodes and topics.

We aim to build a visual servoing controller. We started by handling and integrating all devices and sensors in ROS. First of all, creating an interface that allows the communication and handles the flow of information from the robotic platform to ROS platform and vice versa is needed to handle an hardware device with ROS. Fortunately ROS has already implemented libraries both to integrate Arduino boards as nodes in the communication architecture both libraries that are able to handle the low level control for example implementing a PID low level control.

We must send commands to the motors and acquire information from the encoders and

the camera.

Starting from the motors we built what we called as low level control structure. Here built an hardware interface node (N_5), it is the core of the low level information handling. Here, the number of joints of our robot and the type of the controller for each joint are defined. The hardware interface should provide ROS with the information about joints, their type and all information to model the robot structure. We decided to provide a very light hardware description providing only the basic information needed to allow ROS to acquire and send information. In detail we defined our device as a rigid base with two parallel revolute joints. In order to prevent extreme bending angles of the shaft and limit the joint velocities, we set some constraints to these joints. We set a joint position limit of $|q_{max}|$ and a velocity boundary of $|\dot{q}_{max}|$ for both joints. The joint position constraint is settled considering the maximum bending of the shaft we will need in our experiments. This information is required by $ros\_control\_package$ [11] to know the hardware that it will handle, the hardware is not a classical rigid robot, but this is not an issue, we exploited this light hardware description only to make the interface work and force a position at motor level, then we built the real visual servoing controller as we will see in the next paragraphs. This node (N_5) is provided by the desired joint values in radians. We see that it subscribes to the topic T_4 and T_5 which contains respectively the desired joint value for the joint 1 (motor 1) and joint 2 (motor 2). Inside N_5 there are the low level controllers for the two motors so it needs to acquire data from the encoders and send commands to the motors. In order to achieve this communication, we integrated Arduino boards in ROS environment, now each board is a node. In detail node N_6 is the Arduino board handling the motor 1 while the node N_7 corresponds to the board handling the motor 2. N_6 subscribes at the topic T_6 containing the PWM value so the torque to be applied by the motor 1 (discrete values from -255 up to 255) while N_7 subscribes at T_7 to acquire the new torque value that it will be to applied at motor 2. We see from Figure 3.10b how Arduino boards publish on a topic too. They publish the angle value in degrees read by the encoder (T_8 and T_9 receive the joint values in degrees of the motor 1 and motor 2 respectively). To complete the low level structure description we have to enhance how N_5 collects data from T_8 and T_9. We see that there are two nodes N_8 and N_9 that subscribe at T_8 and T_9 respectively. These two nodes are service nodes, they collect the information and whenever N_5 calls the service they give back the stored values. Actually we acquire both the position and the velocity from the motors so N_5 can know even the actual velocity.

Now we can start considering the camera, we acquire and process images from the USB port thanks to the *OpenCV* library, and finally we get the target position in the image.

We integrate this acquisition and processing inside the node N_4 which publishes all information on topic T_3. Here we publish the actual target position and other useful information like the acquisition time and the target dimension. To build a visual servoing controller we will need a desired position of the target in the image and node N_1 is actually publishing this information on topic T_1.

We have the desired position, the joint values and the camera values (respectively in T_1, T_9, T_8 and T_3). These data are published at different frequencies and must be synchronised. Node N_2 subscribes to all these topics, matches the information according to the time they where acquired, computes the error in the image space as difference between desired position and actual position of the target in the image and publishes all this information to the node N_3 where the high level visual servoing controller is implemented. Here we compute the next joint values that are then published on topic T_4 an T_5.

After this general explanation we go into details about data type and publishing and subscribing rate.

N_1 is a node publishing on the topic T_1 at 40 Hz. It publishes on a *Floats* data topic. In detail the desired position of the target in the image $[x_d, y_d]$ is published.

N_2 subscribes to:

- T_1;
- T_3;
- T_8;
- T_9;

These are all Floats data types. The node N_2 also publishes at 40 Hz on topic T_2. This is a topic characterized by *Floats* data type. We publish an array containing in the right order:

- x_d, desired position of the target x (in pixel);
- y_d, desired position of the target y (in pixel);
- x, position of the target in the image x (in pixel);
- y, position of the target in the image y (in pixel);
- e_x, the error (in pixel) along x corresponding to the difference between the position of the target in the image x and the desired position of the target x;

- e_y, the error (in pixel) along y corresponding to the difference between the position of the target in the image y and the desired position of the target y;
- q_1, angle value for motor 1 (in degrees);
- q_2, angle value for motor 2 (in degrees);
- t_s, time in ros environment in which the circle was detected (seconds);
- t_{ns}, time in ros environment in which the circle was detected (nanoseconds).

N_3 is the node containing the core of the high level control, it subscribes to T_2 and publish at 40 Hz on topics (data type $float64data$):

- T_4;
- T_5.

N_4 is a node which publishes at 40 Hz on a topic called T_3 (we set 40 Hz but the camera can reach only 30 Hz maximum). It publishes data on a topic of type $Floats$ ($float32[]data$). In detail an array of 5 elements $[x, y, r, t_s, t_{ns}]$ is published. In position 1 there is the position of the center of the detected circle in x direction. In position 2 there is the position of the center of the detected circle in y direction. In position 3 there is the radius of the detected circle. In position 4 and 5 detection time of the circle in the image (expressed in ROS time) is reported (in position 4 seconds and in position 5 nanoseconds).

N_5 is the hardware node which is characterised by 2 threads one for the real loop handling the hardware communication running at 40 Hz and an other one that runs slower and it is exploited to handle the communication of the information messages of ROS. This node calls the service of nodes N_8 (N_7) collecting the actual joint and velocity values of the motor 1 (motor 2) and publishes the new torque to be applied by the motor 1 (motor 2) on the topic T_6 (T_7). Here we read and publish joint and velocity values at 40 Hz and these values are of type $Floats$ ($float32[]data$).

N_6 (N_7) is the node associated to the Arduino board handling motor 1 (motor 2). It subscribes at the topic T_6 (T_7) acquiring the torque to be applied and publishes on topic T_8 (T_9) the actual angle (in degrees) and velocity values. This is performed at 40 Hz and data are of type $Floats$ ($float32[]data$).

N_8 (N_9) is a service node which subscribes at T_8 (T_9) collecting the new angle position and velocity and when it will be called it will give back these values.

Summarizing all topics we have:

- T_1, topic of type $Floats$ ($float32[]data$);
- T_2, topic of type $Floats$ ($float32[]data$);
- T_3, topic of type $Floats$ ($float32[]data$);
- T_4, topic of type $Float64$ ($float64data$);
- T_5, topic of type $Float64$ ($float64data$);
- T_6, topic of type $Floats$ ($float32[]data$);
- T_7, topic of type $Floats$ ($float32[]data$);
- T_8, topic of type $Floats$ ($float32[]data$);
- T_9, topic of type $Floats$ ($float32[]data$);

Finally we report the services:

- S_1, custom service of type $Floats\_array$;
- S_2, custom service of type $Floats\_array$;

Nodes Abbreviations

Abbreviation	Real name
N_1	$/desired\_position\_generator$
N_2	$/data\_collector\_for\_error\_computation\_sync$
N_3	$/high\_level\_controller\_synch$
N_4	CV
N_5	$/robot\_hardware\_interface$
N_6	$/ArduinoOne$
N_7	$/ArduinoTwo$
N_8	$subscriber\_1\_py$
N_9	$subscriber\_2\_py$

Table 3.1: Nodes abbreviations vs real names.

Topics and services Abbreviations

Abbreviation	Real name
T_1	$/desired\_command$
T_2	$/error\_and\_data\_for\_error\_computation\_test$
T_3	$/detecting\_point$
T_4	$/single\_joint\_actuator/joint1\_position\_controller/command$
T_5	$/single\_joint\_actuator/joint2\_position\_controller/command$
T_6	$/joints\_to\_arduino\_1$
T_7	$/joints\_to\_arduino\_2$
T_8	$/joint\_state\_from\_arduino\_1$
T_9	$/joint\_state\_from\_arduino\_2$
S_1	$/read\_joint\_state\_1$
S_2	$/read\_joint\_state\_2$

Table 3.2: Topics and services abbreviations vs real names.

3.3.3. Data synchronisation

Data are acquired by sensors and published on their topics in different time instants. It might happen that due to a bad illumination condition the camera will acquire image at 15 Hz instead of 30 Hz. So we will acquire data with different acquisition frequencies (Figure 3.11). In order to correctly handle the information flow we must match messages according to the time of acquisition. Otherwise we would end up computing an error in the image space for example which is older than the actual angle position.

ROS provides some functions acting as data filters which allow to handle data according to the time of acquisition. In detail since we will have data acquired in different instants of time and at a different rate, messages won't be characterised by the same instant of time and we will match them according to a minimum time distance criterion. ROS provides a function called *ApproximateTime* filter [44]. Given a number N of subscriptions to several topics, this filter creates sets of N messages, in each set there is at least one instance of each topic. The element who arrives first will be called *pivot* (red points in Figure 3.11). When a set is completed the filter will give in output this set of messages. Each message can be exploited only once and if the time distance between messages in a set is grater than a threshold the eldest messages will be discarded. In Figure 3.11 we

can see four time lines, one for each topic in input to our filter. Messages (circles) are published at varying frequencies. We can see how filter selects pivots (red circles) and matches messages according to the algorithm we earlier explained in order to create sets of synchronised messages that will be our filter outputs (circles linked by dashed lines).

Reminding Figure 3.10 we perform time matching filtering in N_2 synchronising all data we will exploit in the high level controller set in node N_3.

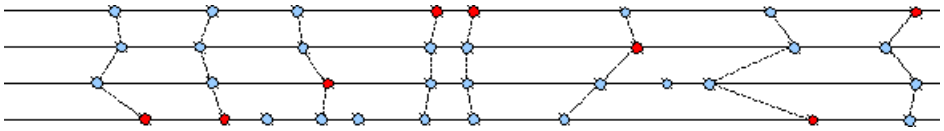


Figure 3.11: Approximate Time filter association example adapted by [44].

We are synchronizing data coming from both motor encoders and camera. In this way we are sure that our data will be correctly matched even in case of a drop in camera publishing rate. Moreover we match the desired position in the image space with the current position so we are able to compute the correct error in the image space.

3.4. Image feature extraction

We are able to acquire images and we have already anticipated that we are able to extract the target position coordinates $(x(t), y(t))$ in the image space I. In Section 2.4 we underlined the importance of feature parameter extractions in visual servoing approaches thus we will describe in detail our choice. Many opportunities are available to detect targets starting from basic segmentation methods up to image filtering thanks to convolutional neural networks (CNN) that properly trained are able for example to detect the center of a lumen (as reported in [25]). We decided to exploit a target easily detectable, a black circle on a white background changing position on a screen. Here we decided to exploit an edge detection method to detect the circle and get the coordinates and the radius of the circle in the image. The *OpenCV* library offers a function able to detect circles in images thanks to the Hough Transform [36]. A circle is unambiguously defined by the center position (x_c,y_c) and the radius r. *OpenCV* implements a detection method slightly trickier than the standard Hough Transform: The Hough gradient method, which is made up of two main stages. The first stage involves edge detection and finding the possible circle centers and the second stage finds the best radius for each candidate center. The $cv.HoughCircles$ function receives as input the acquired image coming from the camera converted in gray scale image and blurred by a Gaussian filter, acceptable radius values (minimum and maximum values) and other parameters such as canny threshold values.

It gives back a vector composed of three elements in our case. It will be our feature parameters vector $\boldsymbol{f} = [x_c(t), y_c(t), r(t)]$ expressed in pixels.

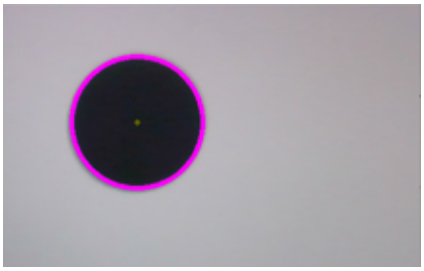


Figure 3.12: Hough transform for circle detection.

However ROS architecture allows to change the code in a node without need of touching the other element of the architecture, thus it must be easily added in the CV node the detection CNN Algorithm ([25]) in order to achieve lumen detection and centering.

3.5. Problem formulation

We aim to build a controller for Atlascope. There are several tasks that a surgeon can perform, but as it appears from Section 2.4 of the most classical tasks that can help surgeon is the automatic centering of a target in the centre of the image space and target tracking (for example keeping the target in the centre of the image and reject disturbances such as patient's movements due to respiration [37]). In order to accomplish these tasks we will exploit information coming from Atlascope's endoscopic camera. This information is processed according to Section 3.4 in order to come up with image feature parameters that can be exploited to move the desired target in the centre of our image space. Our device is characterised by three degrees of freedom, but our image space is bi-dimensional. One possible way to deal with this is to split the task dealing in a separate way the insertion task and the image centering task. Following this idea we think of moving the endoscope forward or backward only when the error in the image space will be lower a fixed threshold. The insertion degree of freedom can be kept constant until the target is centered and then activated to insert or retract the device by small steps as reported in the next algorithm.

Algorithm 3.1 General control algorithm

1: Initial instructions
2: **while** $Automatic\_mode == TRUE$ **do**
3: **if** $error_{image} > error_{threshold}$ **then**
4: Visual servoing control loop instructions
5: **end if**
6: **if** $Insertion == TRUE \ \& \ error_{image} \leq error_{threshold}$ **then**
7: Insertion instructions
8: **end if**
9: **end while**
10: Final instructions

According to this idea we won' t consider the insertion degree of freedom and focus only on the two bending degrees of freedom of the flexible arm of Atlascope.

3.6. Global control design

We build an IBVS control scheme in order to overcome camera calibration errors that might rise from an inaccurate robot model. In detail we chose a IBVS technique where we acquire camera images, we extract desired image feature parameters and we exploit it to solve our task. Our camera has a 30 Hz frequency acquisition as specification but can easily drop down to 15 Hz in case of bad light conditions, the average frequency is 25 Hz. This frequency is too low to handle motors directly so we decoupled the low level control and the low level motor control from the high level IBVS. Recalling Figure 3.10, we highlight the low level control and the IBVS control architecture with the feedback loop.

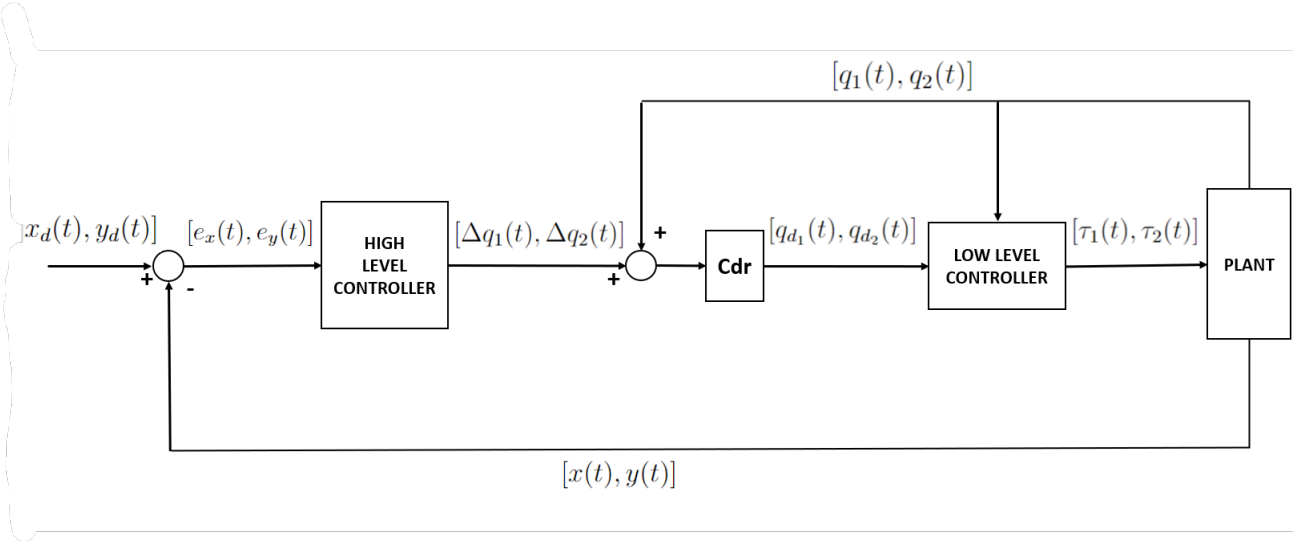


Figure 3.13: General IBVS control scheme.

In Figure 3.13 we schematise the general control architecture. We set the desired position of the target in the image at the time instant t. The image space is a bi-dimensional space composed of a coordinate x and a coordinate y. Thus our desired position of the target in the image at time t is $[x_d(t), y_d(t)]$ expressed in pixels. We acquire images thanks to EIH endoscopic camera and after processing them we are able to get the actual target position in the image space at time t. Recalling that our target will be a circle (Section 3.4), we consider as target position the coordinates of the centre of the circle that we previously called $[x_c(t), y_c(t)]$. Renaming $[x_c(t), y_c(t)]$ as the vector $[x(t), y(t)]$ to simplify the notation, we can define our error vector in the image space $\boldsymbol{e(t)} = [e_x(t), e_y(t)]$ such as when it is equal to zero we realise our task. In detail:

$$e_x(t) = x_d(t) - x(t) \tag{3.1}$$

and

$$e_y(t) = y_d(t) - y(t). \tag{3.2}$$

This error is the input for the high level controller which will provide as output an increment ($[\Delta q_1(t), \Delta q_2(t)]$) expressed in degrees to be summed to the current angle values ($[q_1(t), q_2(t)]$) read by the encoder in degrees in order to make a small variation in the joint space direction needed to decrease the error in the camera space. This sum is multiplied by the conversion constant C_{dr} in order to be expressed in radians and it will be the angle

value ($[q_{d_1}(t), q_{d_2}(t)]$) we should force at motor level to make the circle move towards the desired position of the image. In our case centring and tracking are the main tasks. These tasks requires the target to be always in the middle of the image provided by the camera. Our image space is a rectangular space of dimensions $(640, 480)$ pixels. The origin of the reference frame is set in the upper left corner of the image thus the desired position vector will be constant and characterised by $x_d = 320$ and $y_d = 240$ expressed as number of pixels (Figure 3.14).

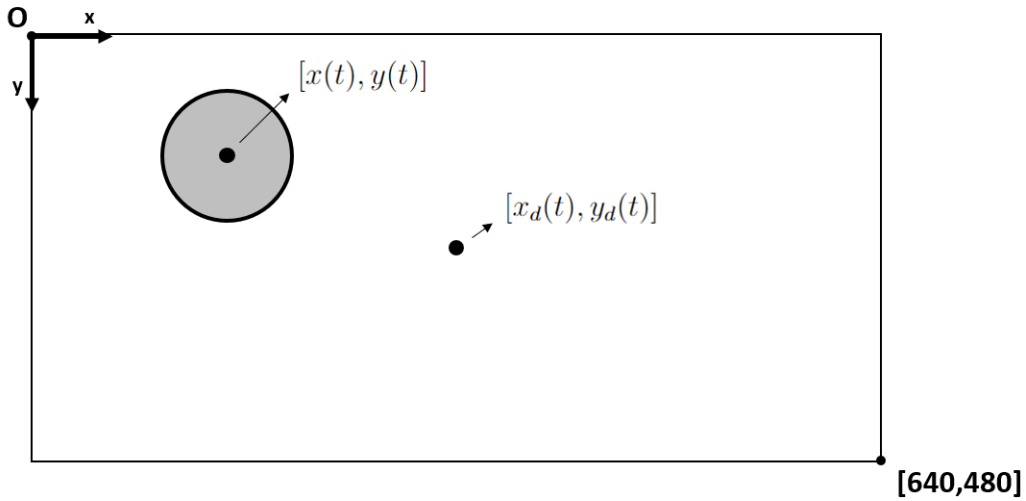


Figure 3.14: Image space.

The low level control will then control the motors applying the torque ($[\tau_1(t), \tau_2(t)]$), expressed as PWM signal) needed to rotate the motors to reach the desired angle values. Let's start inspecting the low level controller in detail.

3.6.1. Low level controller

This controller is composed of two independent PID controllers implemented thanks to ROS control package. ROS control package provides a standard interface to build the hardware interface node. In detail it offers different kind of basic controller. Inside node N_5 we chose a position-effort joint interface that accept the desired and the actual position values, which is the motor angle in radians, and gives back torque values to be applied at joint level as output, that for us is the PWM activation level for the motors. We built two independent PID controllers in order to handle the two motors separately. PID_1 will handle motor 1 while PID_2 will handle motor 2 both running at 200 Hz as reported in Figure 3.15.

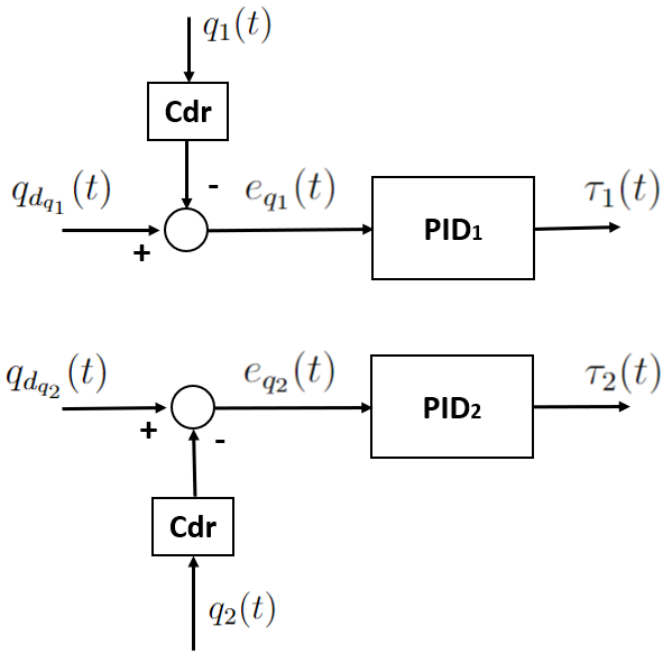


Figure 3.15: Low level control scheme.

The following description is valid for both motor 1 (bottom motor) and motor 2 (right motor). Parameters and variables will be indicated with number 1 for motor 1 while with number 2 for motor 2.

The classical PID equation [4] is:

$$\tau(t) = K_P e_q(t) + K_D \dot{e_q}(t) + K_I \int e_q(t)dt. \tag{3.3}$$

Here the error ($e_q(t)$) is defined in the joint space. In detail for each joint we send the desired angle position $q_d(t)$ in radians to the hardware interface node, here the node calls the service reading the actual angle position, $q(t)$ in degrees, and after converting it in radians computes the error:

$$e_q(t) = q_d(t) - C_{dr} \cdot q(t). \tag{3.4}$$

We give desired joint position expressed in radians and we get as output the torque, the PWM activation level for our motor to reach that position. ROS provides tools like *rqt_guipackage* that allows to change dynamically all PID parameters while the robot is active, so we were able to tune our PID constants in order to fit our device. To tune parameters we opened the visual feedback loop and built a square wave position input

node publishing the desired position (acting as it was the high level control output) for each PID. This step is publishing the desired position at a rate of 40 Hz.

We are not provided with a model of the robot so we started calibrating each PID independently exploiting the empirical closed loop calibration method of Ziegler and Nichols [16]. This method leads to some undesired oscillations so we decided to calibrate manually the parameters giving a square wave input, recording data and searching for the best combination of (K_P, K_D, K_I). We will describe in detail our results in Section 3.4.

3.6.2. High level controller

Our high level controller is thought to work in the image space in the attempt of decreasing our position error in image space. It receives as input the error in the image space $[e_x(t), e_y(t)]$, this is the main feedback branch of the visual servoing algorithm. We compute the error as the difference among the desired image position $(x_d(t))$ and the actual position $(x(t))$ of the detected target in the image (the same is valid with y direction where we will have $y_d(t)$ and $y(t)$ respectively). We can set the desired position, but for our tasks we will keep this desired position constant and corresponding to the centre of the image. In order to let the system work even in case no target is detected, we will consider the vector of errors $([e_x(t), e_y(t)])$ as a null vector setting both $e_x(t)$ and $e_y(t)$ equal to zero in order to keep the current tip position waiting for the target to be detected. Let' s open the box of this controller.

3.7. P visual servoing controller

In this controller the idea is improving the positioning of the tip of the robot with small steps. Starting from the current angle values we update them independently as in Figure 3.16.

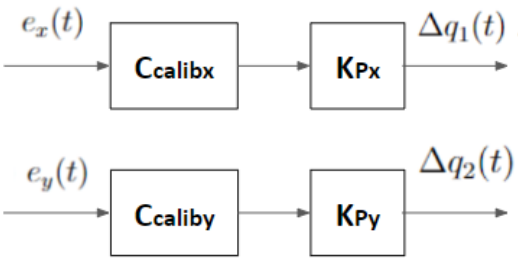


Figure 3.16: High level P controllers.

We will exploit the law:

$$\Delta q(t) = C_{calib} \cdot K_P \cdot e(t). \tag{3.5}$$

And the low level control will receive:

$$q_d(t) = C_{dr} \cdot (q(t) + C_{calib} \cdot K_P \cdot e(t)). \tag{3.6}$$

Here we have the error $e(t)$ in the image space expressed in pixel. We will consider $e_x(t)$ for the motor 1 and $e_y(t)$ for the motor 2 obtaining respectively $q_{d_1}(t)$ and $q_{d_2}(t)$. The desired joint value $q_d(t)$ ($q_{d_1}(t)$ and $q_{d_2}(t)$ for motor 1 and motor 2 respectively) is equal to the current joint value to which we add a correction proportional to the error image (Equation (3.5)). The constant value multiplying the error is composed by the product of a calibration constant (C_{calib}, degrees over pixels), which converts the error from pixels to angle values (in degrees), and a proportional constant coefficient (K_P) that modulates the correction. Given a fixed distance between the robot and the screen where the circle is moving we compute the average ratio between a joint variable variation and the variation in the image position of the target, this will be our calibration coefficient. The degree to radian constant C_{dr} corresponds to $\pi/180$ converting values from degrees to radians while the C_{calib} is empirically set to 0.21 for the P controller x (C_{calibx}) and to -0.36 for the P controller y (C_{caliby}).

The constant proportional parameter K_P is tuned searching for the best performance parameters. Feedback loop will run at the lowest frequency among the frequency of the topics that will come in input to the *Approximatetime* synchroniser filter. In our case camera acquisition rate is the bottleneck. The visual servoing loop will provide information with an average rate of 25 Hz. For deeper detail see the next chapter where we will speak about the experimental procedure to search for the best K_P parameter value.

3.8. Modelling soft robot using deep learning

Analysing the first recorded data we recognise immediately the most common non linear behavior for TDCRs: hysteresis. This effect is hard to model and we will face this issue exploiting deep learning techniques. Predicting the hysteresis curve will allow in the future to apply a correction to controller in order to improve its performances. As we saw in the state of art, there are attempts in the direction of modelling the hysteresis behaviour thanks to neural networks. We developed a neural network inspired to the idea presented

by Cheng et al. [9]. We build a compact neural network to model the relationship between image and joint space.

3.8.1. Neural network structure

We build a fully connected neural network as we can see in Figure 3.17. This neural network is characterised by 3 layers: an input layers, a single hidden fully connected layer and an output layer composed of a single neuron. We developed our network in the Google-Colab environment exploiting Tensorflow. Thanks to this network we implement a NARMAX system modelling the hysteresis. Our input layer is composed of $s = n_a + n_b + 1$ input neurons, each neuron receives a single value as input. We build an input vector composed of the past n_a joint values, the actual target position and the target position in the image in the previous n_b instants of time. Thus referring to the Figure 3.17 we will have $u(t) = x(t)$ and $f(t) = q_1(t)$ in order to model the function linking x and q_1 while we will have $u(t) = y(t)$ and $f(t) = q_2(t)$ in order to model the function linking y and q_2. Our input vector will be:

$$\boldsymbol{k_1}(t) = [x(t), x(t-1) \dots x(t-n_b), q_1(t-1), q_1(t-2) \dots q_1(t-n_a), 1]. \tag{3.7}$$

$$\boldsymbol{k_2}(t) = [y(t), y(t-1) \dots y(t-n_b), q_2(t-1), q_2(t-2) \dots q_2(t-n_a), 1]. \tag{3.8}$$

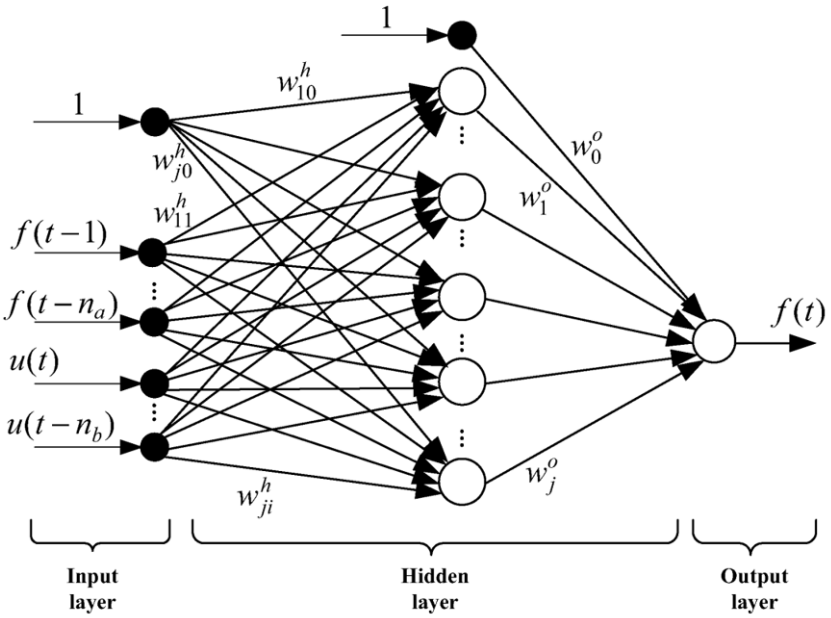


Figure 3.17: Neural network internal structure adapted by [9].

This will allow us to predict the joint value $q(t)$ associated to the last position of the target in image space.

3.9. Performance evaluation

In order to evaluate our visual servoing controller, we will test it in two tasks: centering and tracking task. In both the tasks the main parameter is the distance between the desired position of the target and the actual position of the target in the image space. It will be defined as:

$$e_{tracking}(t) = (e_x(t)^2 + e_y(t)^2)^{\frac{1}{2}}. \tag{3.9}$$

We will consider the module of the vector $\boldsymbol{e(t)}$ of Section 3.6 and its components to point out the behaviour of the P controllers when they are both active.

In detail the following parameters will be reported:

- Steady State Error (SSE) and overshoot for the x component of the image space error Equation (3.1) called SSE_x and $E_{overshoot_x}$ respectively.
- SSE and the overshoot for y component of the image space error Equation (3.2) called SSE_y and $E_{overshoot_y}$ respectively.
- SSE and maximum value for the module of the image space error vector, referred to as tracking error Equation (3.9), called $SSE_{tracking}$ and $E_{tracking_{max}}$ respectively.

Instead in order to analyse model performances the prediction error (e_P) will be consdered. Prediction error $i-th$ associated to the $i-th$ prediction is defined as:

$$e_P(i) = q(i) - q_P(i). \tag{3.10}$$

We will exploit three metrics: we compute the mean squared error (MSE), Mean Absolute Error (MAE) and record the absolute value of the maximum prediction error ($|e_{Pmax}|$). MSE is computed as the mean of the square values of prediction error e_P over the considered dataset. Given a dataset of N rows MSE is given by:

$$MSE = \frac{\sum_{1=1}^{N} e_P(i)^2}{N}. \tag{3.11}$$

MAE is mean of the absolute prediction error values. Considering N predictions we have:

$$MAE = \frac{\sum_{1=1}^{N} |e_P(i)|}{N}. \tag{3.12}$$

4 | EXPERIMENT

In this chapter we will report a detailed description of the experimental setup and procedure exploited to:

- to tune the low level PID controllers and access their performances;
- to collect the datasets we will exploit to train, validate and test our neural network models;
- to tune the high level P controllers and access their performances.

Let's start with the low level controllers.

4.1. Low level PID tuning

As reported in (Section 3.6) we split the task in an high level control task which will be solved searching for the new joint value (desired set point for each motor) found by adding step by step a small correction to the actual joint values at an average rate of 25 Hz and the low level control task aiming to provide the torque needed to reach the desired joint value.

As stated in [4] in the most of control system applications 90-95% of control loops are PID form because of their simple structure and robust performance. We decided to build two independent PID controllers (one for each motor). Fortunately ROS provides a nice package called *ros_control_package* that allows the user to implement PID controllers for the low level control of joints. This package needs to be provided with information about the joint it will control and about the controller we will implement. We provided ROS with the lightest description possible, avoiding a complex modelling. Each joint is described as a revolute joint with some constraints:

- maximum joint angle value is set to be 3 radians
- minimum joint angle value is set to be -3 radians
- maximum absolute joint velocity value is set to be 0.5 radians/s

- maximum effort is set to 255
- minimum effort is set to -255

The effort is expressed as integer numbers whose absolute value is lower or equal to 255. The absolute value express the duty cycle value and the sign encodes for the direction of rotation. The boundaries on position and velocity are set empirically in order to allow a safe movement of the robot preventing too high bending angles and keeping a reasonable maximum velocity to prevent damages of tendon level according with the idea that the robot will never reach high velocities due to its final working environment (urinary system).

Among the available control interfaces we chose a position controller proving a torque as output (position to effort controller). So as reported in [45] the classical PID equation is implemented:

$$\tau(t) = K_P e_q(t) + K_D \dot{e}_q(t) + K_I \int e_q(t)dt. \tag{4.1}$$

Where recalling Equation (3.4):

$$e_q(t) = q_d(t) - C_{dr} \cdot q(t). \tag{4.2}$$

Here we see a classical parallel type formulation for the PID and collecting K_P we get the ideal representation of the PID controller:

$$\tau(t) = K_P \cdot \left(e_q(t) + T_D \cdot \dot{e}_q(t) + \frac{1}{T_I} \cdot \int e_q(t)dt \right). \tag{4.3}$$

We can recognise:

- q_d, the set point (desired joint value in radians);
- q, the process value (the current joint value in degrees);
- e_q, the error of the process computed as difference between the target (set point, q_d) and the current state (process value, q) after being converted to radians;
- τ, the controller output (torque) sent to the plant which will return the process value q;
- K_P, the proportional gain;

- K_I, the integral gain ($K_I = K_P/T_I$);
- K_D, the derivative gain ($K_D = K_P * T_D$)
- T_I, the integral time;
- T_D, the derivative time.

As we can see in the previous equations the PID output is composed of the sum of 3 contributions:

- $P_{term} = K_P e_q(t)$
- $I_{term} = K_D \dot{e}_q(t)$
- $D_{term} = K_I \int e_q(t)dt$

The integral term generates the effects called windup. When the output saturates, the system will then take longer time to respond to a step input. This longer response time will result in a greater integral contribution that will produce an overshoot that will be then reduced dissipating the extra integral contribution. This phenomenon is called (integrator) windup. There are methods to minimise this effect and ROS is provided with the clamping (anti-windup) method. It consists in setting boundaries to the integral contribution setting a minimum and maximum thresholds ($i_{clamp_{min}}$ and $i_{clamp_{max}}$).

Before starting with the experiments we tuned the PID controller parameters in order to reach the better performances for low level control. We have to define the suitable values for K_P,K_I,K_D,$i_{clamp_{min}}$ and $i_{clamp_{max}}$.

We set the a reasonable symmetric clamping threshold at 130.0, so we set $i_{clamp_{min}} = -130$ and $i_{clamp_{max}} = 130$.

An important aspect is the frequency of the PID loop. We set this frequency at 200 Hz, so nodes N_5,N_7,N_8 and N_9 are working at 200 Hz. Unfortunately we notice some drops of the frequency but it is always above 130 Hz.

In order to set the gain parameters we exploited the Zieger-Nichols (ZN) closed loop step response method and a manual tuning searching for a trade of between steady-state error, overshoot and rising time.

Steady state error is the difference between the input (set point) and the output of a system when the response has reached steady state. Rise time is the time required for the system output to rise from a lower level to a higher level of the steady-state value. Typical range is 10% - 90%. Percent overshoot is the percent by which a system exceeds its final steady-state value.

We built two independent PIDs: PID_1 handling motor 1 (bottom motor, left and right direction) and PID_2 handling motor 2 (right motor, up and down direction). Since they are independent we tune each PID separately.

We built a node called *square_wave_generator* that will substitute the high level controller node N_3 and publish the desired setpoint to the PID (it will publish on T_4 for PID_1 tuning and on T_5 for PID_2 tuning). We will send to the PID a square wave oscillating from 0.7 and -0.7 radians, with a period of 30 seconds and a duty cycle of 50%. We record 5 complete cycles for each parameters combination inspected.

The following procedure will be repeated for both PID_1 and PID_2. We start with the ZN tuning method. We set $(K_P, K_I, K_D) = (0, 0, 0)$ and we start increasing K_D till we reach the ultimate gain K_U and the ultimate period time t_U. K_U is defined as the lowest value of the proportional gain so that given the parameter set $(K_U, 0, 0)$ the system shows a permanent oscillation in the output. T_U is the period characterising this permanent oscillation. Once we found the K_U and t_U values the PID gains are computed according to the expressions in Table 4.1

Ziegler Nichols tuning formula for PID controller

Parameter	Value
Proportional gain	$K_P = 0.6 * K_U$
Integral time	$T_I = 0.5 * t_U$
Derivative time	$T_D = 0.125 * t_U$

Table 4.1: Topics and services abbreviations vs real names.

In order to collect the data we recorded the topic */single_joint_actuator/joint1_position_controller/state* and */single_joint_actuator/joint2_position_controller/state* which provides the state of PID_1 and PID_2 respectively. This topic allows us to record all the variables and parameter sample by sample.

Manually tuning method will be performed considering one motor at time and testing several combination of parameters in order to reach the best performances.

The best PID parameters computed by these procedures will be exploited during all the following steps both for the data acquisition for the networks and in the visual servoing control structures.

4.2. Data acquisition protocols for modelling

Our aim is to collect data to model the robot behaviour, we want to relate the motion of the target in the image space in response to different inputs at joint level. In order to achieve this aim we will follow four protocols for acquiring data. In order to keep trace of all system variables, in all the recording sessions we will record topics:

- */single_joint_actuator/joint1_position_controller/state*, the low level PID state for bottom motor;
- */single_joint_actuator/joint2_position_controller/state*, the low level PID state for right motor;
- */detecting_point*, the target position in the image;
- */error_and_data_for_error_computation_test*, the data from the camera, the encoders and the errors in image space after the sychronisation.

However what we will exploit will be the variables:

- x, actual position of the target in the image x;
- y, actual position of the target in the image y;
- q_1, actual angle value for motor 1;
- q_2, actual angle value for motor 2

in topic */error_and_data_for_error_computation_test*. We will repeat the procedures listed in the following paragraphs both for the bottom motor and the right motor separately. We will consider q_1 (q_2) values read by the encoder of the bottom (right) motor and the position x (y) of the target in the image along the horizontal (vertical) axis while moving the bottom (right) motor building in this way for each protocol data acquisition two datasets: one composed of values (q_1, x) and the other composed by values (q_2, y). Here we report a table with the complete list of data collected.

Datasets

Protocol	Direction	Name List
1	**X**	$DATASET\_1\_x\_training$
		$DATASET\_1\_x\_validation$
		$DATASET\_1\_x\_test$
1	**Y**	$DATASET\_1\_y\_training$
		$DATASET\_1\_y\_validation$
		$DATASET\_1\_y\_test$
2	**X**	$DATASET\_2\_x\_validation$
		$DATASET\_2\_x\_test$
2	**Y**	$DATASET\_2\_y\_validation$
		$DATASET\_2\_y\_test$
3	**X**	$DATASET\_3\_x\_validation$
		$DATASET\_3\_x\_test$
3	**Y**	$DATASET\_3\_y\_validation$
		$DATASET\_3\_y\_test$
4	**X**	$DATASET\_4\_x\_validation$
		$DATASET\_4\_x\_test$
4	**Y**	$DATASET\_4\_y\_validation$
		$DATASET\_4\_y\_test$

Table 4.2: Data collected for modelling.

4.2.1. Data acquisition protocol 1

In this protocol we set a screen in front of the robot at a fixed distance of 44 centimeters. We display on this monitor a black circle so that it will appear in the camera field of view and we send an input to the robot. The input is a series of set points at joint level ($q_d(t)$). We will record how the circle moves in the image space according to the end effector movement. We repeat this procedure for the two joints, keeping one of them still while changing the other. The circle will be kept still in the central position of the monitor (Figure 4.1).

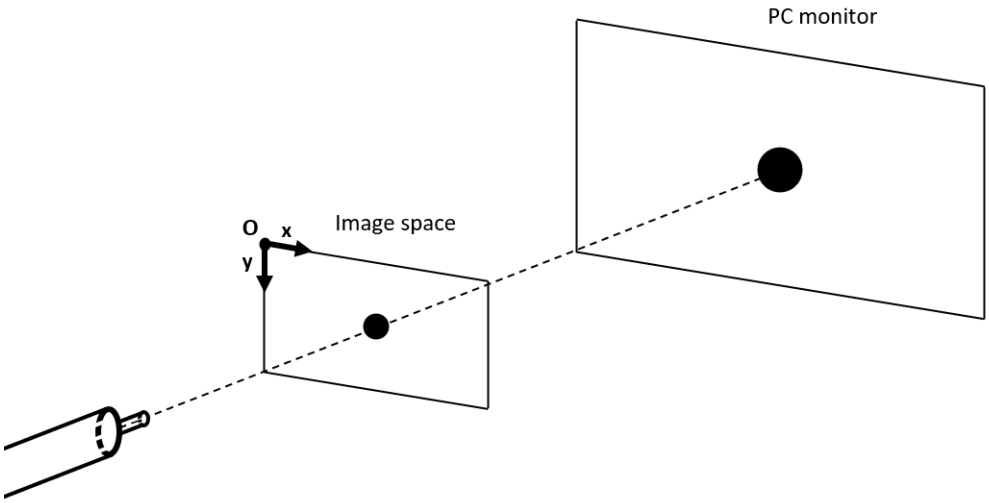


Figure 4.1: Expeimental setup for $DATASET\_1$, $DATASET\_2$ and $DATASET\_3$ recording.

We will consider q_1 (q_2) values read by the encoder of the bottom motor and the position x (y) of the target in the image along the horizontal (vertical) axis while moving the bottom (right) motor. These datasets (named $DATASET\_1\_x$ and $DATASET\_1\_y$) will be exploited to train, validate and test the neural network models.

For $DATASET\_1$ we will exploit a sinusoidal input function:

$$q_d(t) = a * \sin(2 * \pi * f * t). \tag{4.4}$$

Where $f = 1/T$ with:

- $a = 0.3, 0.5, 0.7$ rad and $T = 1, 2, 4, 8$ sec for the bottom motor;
- $a = 0.3, 0.45, 0.6$ rad and $T = 1, 2, 4, 8$ sec for the right motor since with an amplitude of 0.7 radians the target would get out of the camera view while acquiring data.

When we start each recording the robot is in a straight configuration for joint values $(0, 0)$ and the circle appears for this configuration in the middle of the endoscopic camera view. Recalling the nomenclature: q_{d_1} set point for motor 1 (bottom motor), q_{d_2} set point for motor 2 (right motor), q_1 process value for motor 1 (bottom motor), q_2 process value for motor 2 (right motor), x is the horizontal target position in the image space and y is the vertical target position in the image space. For each parameters combination we will acquire 3 recordings:

- a training file composed of 3 sinusoidal cycles;
- a validation file composed of a single sinusoidal cycle;
- a test file composed of a single sinusoidal cycle.

We exploit this data to train, validate and test a neural network model for our soft robot. We name $DATASET\_1\_training$ the collection of all training files, $DATASET\_1\_validation$ the collection of all training files and $DATASET\_1\_test$ the collection of all testing files acquired.

In order to further test the predictive capability of the model we collected other data to test our model.

4.2.2. Data acquisition protocol 2

In detail we collected data exploiting the same setup previously described but two different inputs: increasing amplitude sine and decreasing amplitude sine. We define the increasing amplitude sine as:

$$q_d(t) = a * (1 - e^{-\frac{t}{\tau}}) * \sin(2 * \pi * f * t). \tag{4.5}$$

with $a = 0.7$ radians for the bottom motor and $a = 0.6$ radians for the right motor, $f = 1/T$ with $T = 2$ seconds, $\tau = 5$ seconds.

We define the decreasing amplitude sine as:

$$q_d(t) = a * e^{-\frac{t}{\tau}} * \sin(2 * \pi * f * t). \tag{4.6}$$

with $a = 0.7$ radians for the bottom motor and $a = 0.6$ radians for the right motor, $f = 1/T$ with $T = 2$ seconds, $\tau = 5$ seconds.

4.2.3. Data acquisition protocol 3

In detail we collected data exploiting the same setup previously described but with a constant amplitude sine with varying period as input. We define the constant amplitude sine with varying period as:

$$q_d(t) = a * \sin(2 * \pi * f * t). \tag{4.7}$$

with $a = 0.7$ radians for the bottom motor and $a = 0.6$ radians for the right motor, $f = 1/T$ with $T = T_0 + (0.1 * t)$ where $T_0 = 3$ seconds.

4.2.4. Data acquisition protocol 4

Finally we will inspect the dependency of our network output with respect to the target position in space. We recorded an other dataset called $DATASET\_4$. We set the black circle in different positions before starting the recording. Each recording starts with the robot in a straight configuration, but now the circle is set in different positions on the monitor so that at joint configuration $(0, 0)$ radians the circle will appear shifted with respect to the center of the image space as we can appreciate in Figure 4.2.

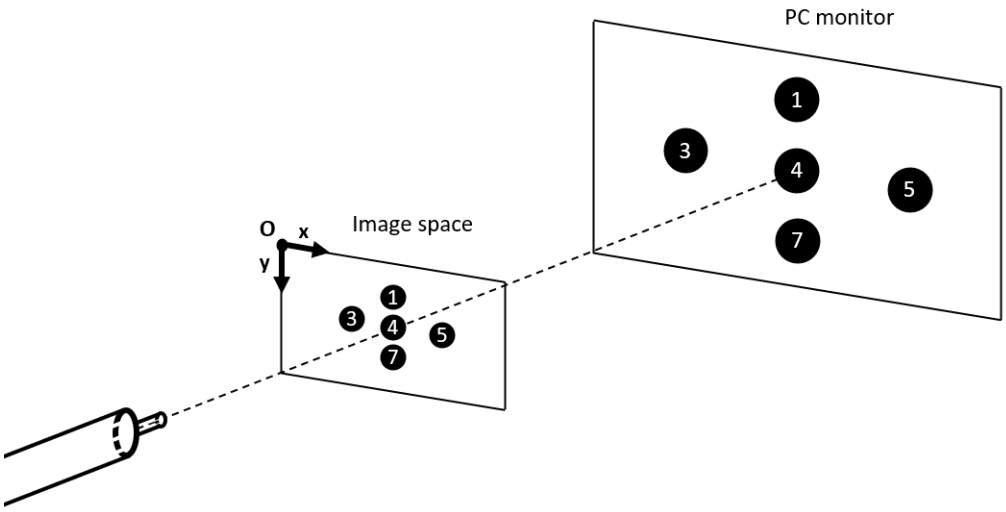


Figure 4.2: Experimental setup for $DATASET\_3$ recording.

So when the robot is straight the circle will appear in the image space in a different position according to the position on the monitor. We will refer to these positions as Pos_n, where n is the number associated to each circle position appearing in Figure 4.2 and Figure 4.5. Circle is still during the recording, the position is chosen before starting the recording. Since the circle will appear shifted in the image space we have to adjust the trajectory inputs according to the position of the circle on the monitor to be sure it won't go out of the camera view during the end effector movement. Thus when we will be recording data for the bottom motor we will exploit $Input\_1\_x$, $Input\_2\_x$, $Input\_3\_x$ while when we will be recording data for the bottom motor we will exploit $Input\_1\_y$, $Input\_2\_y$, $Input\_3\_y$.

We define $Input\_1\_x$ and $Input\_1\_y$ as:

$$q_d(t) = a * \sin(2 * \pi * f * t). \tag{4.8}$$

We define $Input\_2\_x$ and $Input\_2\_y$ as:

$$q_d(t) = |a * \sin(2 * \pi * f * t)|. \tag{4.9}$$

We define $Input\_3\_x$ and $Input\_3\_y$ as:

$$q_d(t) = -|a * \sin(2 * \pi * f * t)|. \tag{4.10}$$

Setting $a = 0.7$ radians for inputs of type x to the bottom motor, $a = 0.6$ radians for inputs of type y to the right motor and $f = 1/T$ with $T = 4$ seconds.

We will record data for each of the following combination of circle position and input in Table 4.3.

Combinations exploited in protocol 4

Input	Circle position
$Input\_1\_x$	Pos_1
$Input\_1\_x$	Pos_4
$Input\_1\_x$	Pos_7
$Input\_2\_x$	Pos_3
$Input\_3\_x$	Pos_5
$Input\_1\_y$	Pos_3
$Input\_1\_y$	Pos_4
$Input\_1\_y$	Pos_5
$Input\_2\_y$	Pos_7
$Input\_3\_y$	Pos_1

Table 4.3: Combinations for protocol 4.

For protocol 2, 3 and 4 we consider that inputs will be kept equal to zero when $t > T * 5$ seconds. For each of the inputs in protocol 2 and 3 and for each combination in Table 4.3 for protocol 4, we collected two files (one named validation and the other named test), each recording lasts $T * 5$ seconds in order to have 5 complete sinusoidal cycles in each recording. The collection of all validation files recorded according to protocol 2 will compose the dataset $DATASET\_2\_validation$ while the collection of all test files according to protocol 2 will compose the dataset $DATASET\_2\_test$. The same is

done for protocol 3 and 4 composing $DATASET\_3\_validation$, $DATASET\_3\_test$, $DATASET\_4\_validation$ and $DATASET\_4\_test$. These dataset will allow us to inspect the ability of the neural network model to predict the future joint when varying amplitude and frequency sinusoidal inputs are present. Moreover it will allow to test the robustness in case of an initial shift of the target position.

4.3. Data preparation

Data of datasets in Table (4.2) need of a specific data arranging according to the network we exploit. Before doing this we need to perform data cleaning and filtering.

During the recordings unfortunately it might happen that the circle end up being out of the camera view. In this case we will record camera values out of range. Image coordinate x belongs to the close set [0,640] and y can be in [0,480]. So the first step is outliers removal. Moreover we have many recordings and there is a resting period between one recording and the following. This resting period is not meaningful for the training of the network and to quantify correctly the performances in testing phase. We will reduce this period keeping only 20 samples of the waiting time.

After this we filtered our data. We consider two kinds of filters: Gaussian filter and Moving Average filter (MA filter) applied both on joint and camera values. We will implement the filters exploiting the Gaussian filtering function and a moving average filter window, manually averaging data withing a window of a desired size. We will vary the filtering parameters (window width, $width = 3, 5$ for MA filter and standard deviation, $sigma = 1, 2$ for Gaussian filter) looking for the filter that fits to our application and data. We filter the data in order to remove the noise around the measurements. For example we notice that the the detected circle position is characterised by small oscillations even if the target and the robot are both still. In Figure 4.3 and in Figure 4.4 we show the target position x in the camera space in $DATASET\_1\_x\_test$. We can see the Gaussian filter with $sigma$ set to two is able too remove the dispersion around the correct measurements.

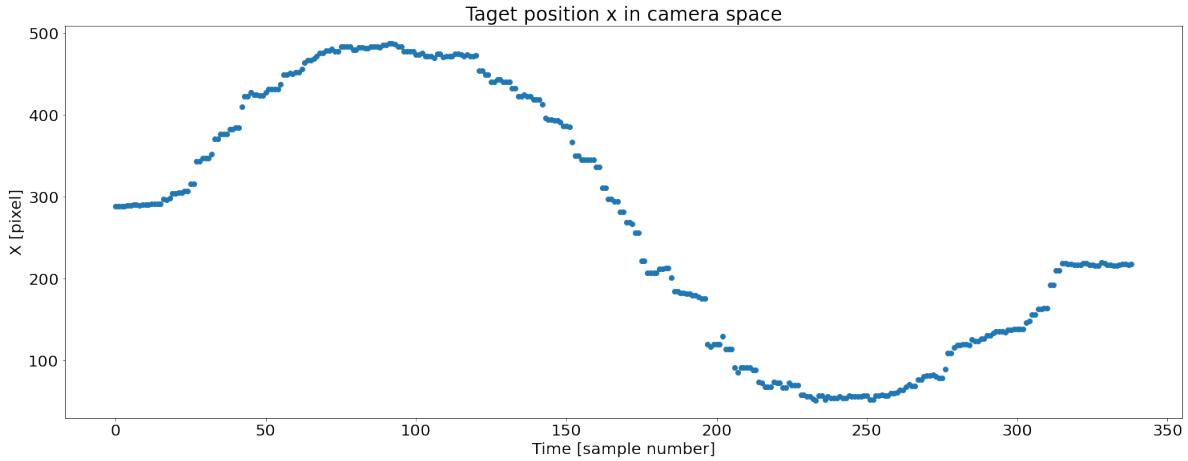


Figure 4.3: Raw camera recording of one sinusoidal cycle from $DATASET\_1\_test$ (a = 0.7 rad and T = 8 sec).

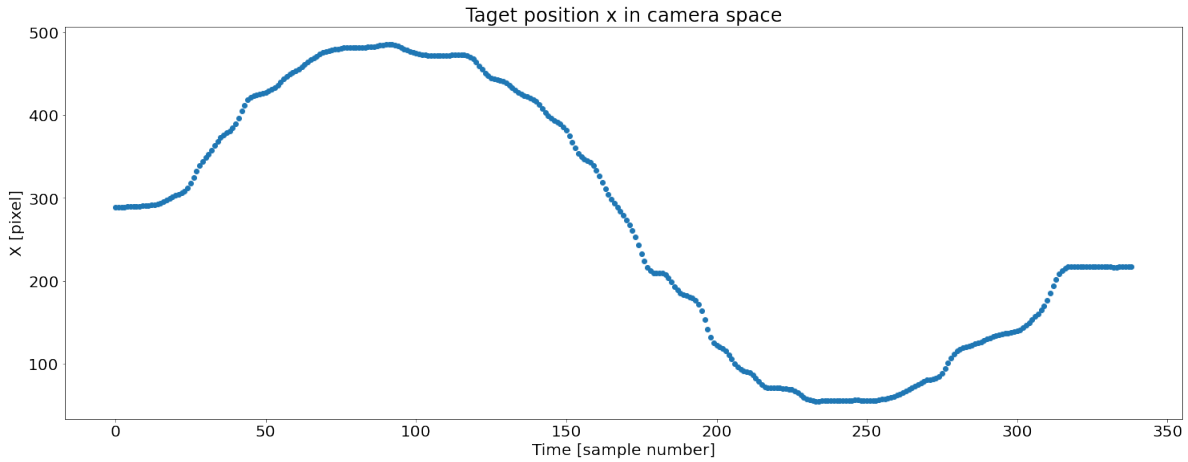


Figure 4.4: Filtered camera recording of one sinusoidal cycle from $DATASET\_1\_test$ (a = 0.7 rad and T = 8 sec).

We consider now the dataset preparation specific for the network. According to the inputs and outputs of the network described in Section 3.8.1 we must rearrange the datasets.

4.3.1. Fully connected network

Recalling what we stated in paragraph Section 3.8.1 we want to create a network that will model our system providing us $q(t)$. In detail:

$$q_1(t) = F_x(\boldsymbol{k_1}(t)). \tag{4.11}$$

$$q_2(t) = F_y(\boldsymbol{k_2}(t)). \tag{4.12}$$

Where $\boldsymbol{k_1}$ and $\boldsymbol{k_2}$ are given by Equation (3.7) and Equation (3.8) respectively.

Fully connected neural networks are built as sequential models built calling the dense layer function in Google-Colab environment. Dense layer allows to to add a fully connected layer of neurons. It allows to specify the number of neurons as well as the activation function. About the activation function we test sigmoidal, Hyperbolic tangent and Rectifier Linear Unit (ReLU) functions choosing the one showing the best predictive performances. We will consider a dense layer composed of 6, 8 neurons and 12 neurons. We will change the number of hidden layers (1 or 2). We considered Adam optimiser and Mean Squared Error (MSE) as loss function for the training process and we will vary the number of training epochs testing even the early stopping function provided by Tensorflow. We will search for the best predictive performance. We will train the network considering the data in their original scale.

We worked on datasets listed in Table 4.2 in order to have input datasets fitting the input requirements of the input layer of our network. In detail for each dataset of the table we arrange the data so that now the datasets are composed by rows of type $(\boldsymbol{k_1}(i), q_1(i))$, for datasets with data $(q_1(i), x(i))$, and of type $(\boldsymbol{k_2}(i), q_2(i))$, for datasets with data $(q_2(i), y(i))$, where i is the row number. From now on we will refer to the dataset listed in Table 4.2 as the dataset obtained after arranging the data in order to make the dataset be composed by $(\boldsymbol{k}(i), q(i))$. Selecting only values $\boldsymbol{k}(i)$ we have the input datataset while selecting only $q(i)$ we have the output dataset. For example we will train our network F_x with the arranged $DATASTE\_1\_x\_training$. It is composed by $(\boldsymbol{k_1}(i), q_1(i))$ rows. So we will get the $DATASTE\_1\_x\_training\_input$, composed of rows of type $(\boldsymbol{k_1}(i))$, selecting the first column of $DATASTE\_1\_x\_training$ and the $DATASTE\_1\_x\_training\_output$ composed of $(q_1(i))$ rows selecting the second column of $DATASTE\_1\_x\_training$.

4.4. P visual servoing controller

This last section will describe four series of experiments. The first experiment is carried out to tune empirically the proportional constant multiplying the error in the image space for our proportional visual servoing controller described in Section 3.7. Once we tuned the P constant for both x and y direction we activate both the P controllers asking the robot to keep the target in the centre of the image space. We performed three experiments to

test our controller performances. The target, our black circle on the monitor, will move in a jumping mode and a continuous mode so we will have respectively a centering and a tracking task to be performed. In last experiment we will take into account a constrained environment in order to test the robustness of the controller comparing its performances in free and constrained environment.

4.4.1. Experiment 1

In this experiment we exploit the setup of Figure 4.5 in order to search for the most suitable value of our P controller. In this case the set point will be set by the high level controller in node N_3. In detail for this for experiment N_3 will publish the set point keeping the set point for one of the two motors fixed to zero while varying the set point for the other motor in order to reduce the image error to zero. The experiment will deal with the bottom motor and the right motor separately under the assumption that the motion of the target position in the image along x axis is mainly due to a change of the joint value of the bottom motor while changes in y coordinate of the target in the image is mainly due to a change in the joint value at the right motor level. Recalling Equation (3.6) we have that:

$$q_{d_1}(t) = C_{dr} \cdot (q_1(t) + C_{calibx} \cdot K_{Px} \cdot e_x(t)). \tag{4.13}$$

and

$$q_{d_2}(t) = C_{dr} \cdot (q_2(t) + C_{caliby} \cdot K_{Py} \cdot e_y(t)). \tag{4.14}$$

We will set K_{Py} to zero while tuning the controller for the bottom motor and vice-versa we will set K_{Px} to zero while tuning the controller for the right motor. We will perform the experiments reported Table 4.4 and Table 4.5.

Experiment 1 - Bottom motor P tuning

Setpoint x	Setpoint y	Circle displacement sequence	Repetitions
$q_{d_1}(t)$	0	$Pos\_4 \rightarrow Pos\_3 \rightarrow Pos\_4$ $Pos\_4 \rightarrow Pos\_5 \rightarrow Pos\_4$	5

Table 4.4: Px controller tuning.

Experiment 1 - Right motor P tuning

Setpoint x	Setpoint y	Circle displacement sequence	Repetitions
0	$q_{d_2}(t)$	$Pos\_4 \rightarrow Pos\_1 \rightarrow Pos\_4$ $Pos\_4 \rightarrow Pos\_7 \rightarrow Pos\_4$	5

Table 4.5: Py controller tuning.

The circle disappears and appears in different positions on the monitor (it jumps from one position to the other, jumping mode). This type of change of the circle position will make the error in the image space increase as a step. Thus we will be able to analyse the response to a step input. Once the circle appears in a position, it rests in the current position for 15 seconds before jumping in the next position. We performs one experiment for each analysed combination (K_{Px}, K_{Py}). We inspect combinations:

- $K_{Px} = 0.05, 0.1, 0.2, 0.4, 0.5, 0.6, 0.8, 1$;
- $K_{Py} = 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8$;

We look for the best performance considering searching for an acceptable trade off between rising time and overshoot.

4.4.2. Experiment 2

This second experiment is performed after we identified the best (K_{Px}, K_{Py}). We will keep active both the P controllers and assess the performance of these controllers while they are working together in response to different combinations of step inputs. We will displace the circle in the monitor screen in a jumping mode as in experiment 1 but we will modify the jumping sequence. The resting time between one position and the next one will be 15 seconds as in the previous experiment. In detail we will follow the instructions of Table 4.6. The robot ideally will follow the red path we can appreciate in Figure 4.5. This jumping sequence will repeated 5 times. We will compute the performance parameters computing their average value and the standard deviation over the five repetitions for each step direction.

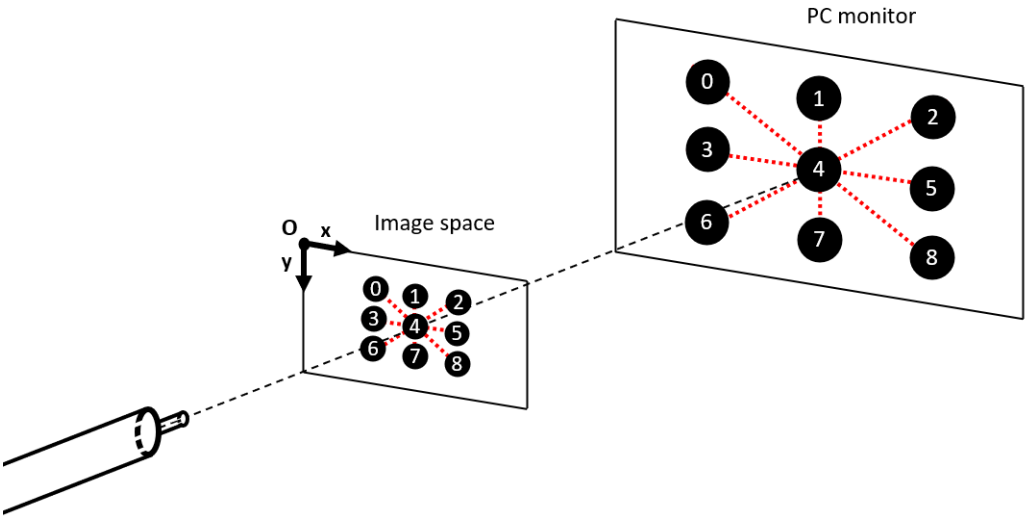


Figure 4.5: Experimental setup for Experiment 2 and 3.

Experiment 2

Setpoint x	Setpoint y	Circle dislacement sequence	Repetitions
$q_{d_1}(t)$	$q_{d_2}(t)$	$Pos\_4 \rightarrow Pos\_3 \rightarrow Pos\_4$	5
		$Pos\_4 \rightarrow Pos\_5 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_1 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_7 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_2 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_6 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_0 \rightarrow Pos\_4$	
		$Pos\_4 \rightarrow Pos\_8 \rightarrow Pos\_4$	

Table 4.6: Experiment 2 description.

4.4.3. Experiment 3

In this experiment we will move the circle on the monitor in a continuous way without any resting time. The circle will move along the four trajectories in Figure 4.6. The circle will displace: right and left, right-down and left-up, down and up and finally left-down and right-up always coming back to the center of the monitor. This path will be performed only once.

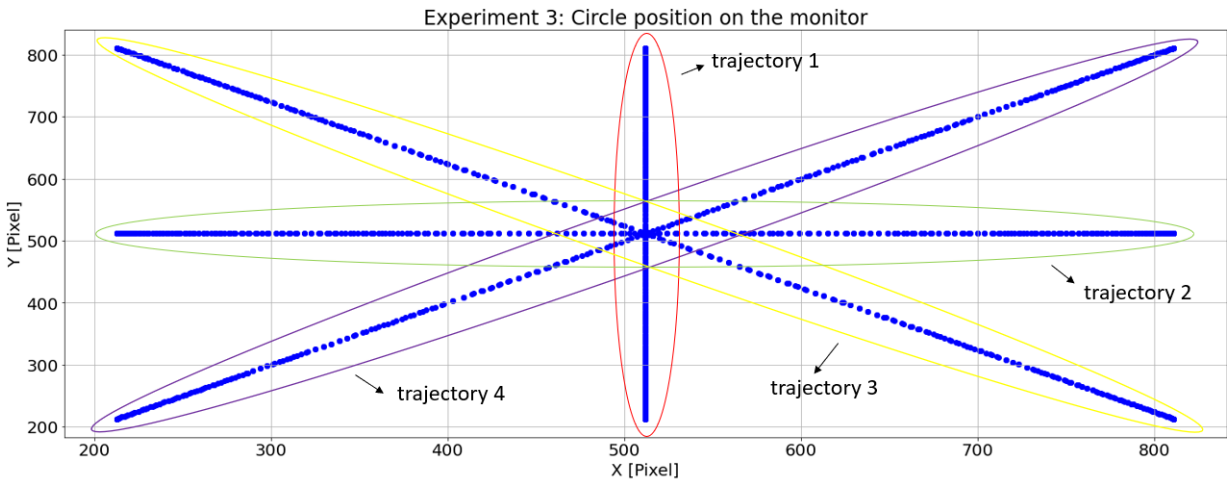


Figure 4.6: Circle displacement.

Trajectories 1 and 2 are obtained keeping one coordinate constant and making the other coordinate change like a sine wave centered in the coordinate of the central pixel of the image. Trajectory 3 and 4 are obtained varying like a sine wave centered in the coordinate of the central pixel of the image both coordinate x and y. We chose a sine wave of period of 10 seconds and amplitude 300 pixels. We will repeat this experiment nine times.

4.4.4. Experiment 4

In this experiment we aim to assess the controller performance in a constrained environment. We added an obstacle set next to the robot. The obstacle is set so that when the endoscope is in the straight configuration, the distance between the flexible part of robot shaft and the obstacle is lower than 5 millimeters assuring the contact during both the centering and tracking task. In detail to analyse these tasks we will carry out two version of this experiment: a jumping and a continuous way according to the movement type of the circle on the screen.

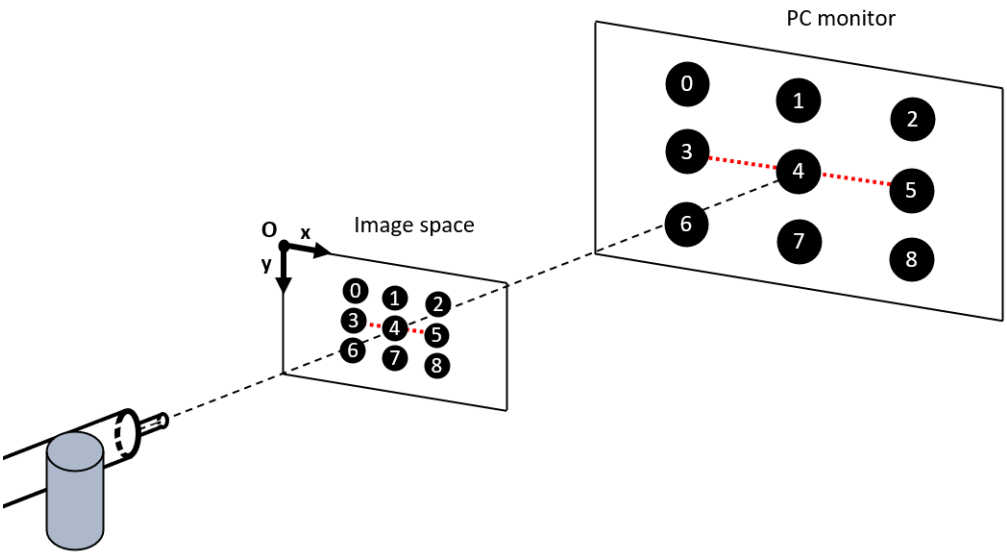


Figure 4.7: Experimental setup for Experiment 4.

In the continuous experiment we will displace the circle left and right in a continuous way, varying the x coordinate of the circle in the screen as a sine wave centered in the middle of the monitor of period 10 seconds and amplitude 300 pixels. We will repeat the movement several times. Our aim is to quantify the same performance parameters taken into account in experiment 3.

In the continuous experiment the circle will jump left and right with a resting time of 15 seconds as we saw for experiment 2. In this experiment we focus only on the left and right path to make the robot touch the obstacle. We will repeat this path five times.

5 | RESULTS AND DISCUSSION

In this chapter we will report a detailed description of the experimental results, pointing out our choices and performances of our model and controller.

5.1. Low level PID

Following the procedure described in Section 4.1 we searched for the PID parameter combination both for PID_1 and PID_2.

We started searching the best parameters combination according to ZN method for PID_1 (bottom motor PID). We recorded the first permanent oscillation at $K_P = 6500$. Thus we have that the ultimate gain k_U is $K_U = 3900$. Considering the step response we computed the ultimate time t_U. We took a single step response and computed the half ultimate time ($t_U/2$) as the average time distance between two following stationary points (where one point is a local maximum and the other is a local minimum). We obtained a $t_U/2 = 0.076$ thus $t_U = 0.152$ seconds. Recalling Table 4.1 we get $K_P = 3900$, $K_I = 51315$ and $k_D = 74$ by Ziegler-Nichols formulas.

For PID_2 (right motor PID), we recorded the first permanent oscillation at $K_P = 7600$. According to previously stated we computed the $t_U/2$ value getting $t_U/2 = 0.074$, thus $t_U = 0.148$ seconds. Recalling Table 4.1 we get $K_P = 4650$, $K_I = 102702$ and $k_D = 144$ by Ziegler-Nichols formulas.

Due to its soft nature, Atlascope showed a different behaviour during the repetitions of the step input although the PID parameters were kept constant both for PID_1 and PID_2 tuning. Here we reported the parameters combination considering the proportional gain at which we measured a continuum oscillation although during the repetitions we noticed a change in the response. ZN methods gives a parameter combination that is at the border of the stability region and might usually show some oscillations. Applying the parameter combinations by ZN, we faced continuous shaft vibrations both for PID_1 and PID_2. We think that being on the stability border a change in the shaft properties due to cable friction and backlash might cause this instability. In order to find a more stable

combination for our PIDs we switched to the manual tuning method.

Changing manually the PID parameters we started increasing the K_P till we visually saw the error roughly decrease towards zero and then we modified the K_I parameter in order to delete the steady state error and the K_D parameter finding a balance in order to reach a trade off between speed of response and stability. Manually tuning the K_P,K_I and K_D parameters for PID_1 we chose as $(3500, 500, 25)$ as best combination.

Manually tuning the K_P,K_I and K_D parameters for PID_2 we chose as $(3000, 600, 15)$ as best combination.

5.2. Network

Recalling Section 3.8 after a preliminary inspection we fixed our network parameters. Our model is composed of a fully connected neural network. We set as input parameters:

- $n_a = 7$;
- $n_b = 5$.

Thus our input layer is composed by fourteen neurons. We choose a compact network structure composed by a single hidden layer of twelve neurons with $ReLU$ activation function. The output layer is composed of a single linear neuron collecting the weighted output coming from the hidden layers. This model is trained on the $DATASET\_1\_training$ after filtering data with a Gaussian filter ($sigma = 2$) as in Section 4.3. We set Adam stochastic gradient descent method as optimisation method and considered MSE as loss function. Mean squared error is computed as the mean value of the square of the error on the prediction. In our case the error will be the difference between the real angle value recorded at joint level ($q(t)$) and the predicted joint angle ($q_P(t)$). We fixed a maximum number of iterations equal to one thousand exploiting the early stopping function that is a callback function which stops the training when the metric, chosen as the monitoring metric, improves of a value lower than a fixed threshold for more than a maximum number of iterations called patience (for us 0.01 squared degrees of decrease in the loss function and 30 times of patience respectively). We computed this metric on the $DATASET\_1\_validation$.

5.3. Neural network model performances

Atlascope showed a non linear behaviour (Figure 5.1,Figure 5.2). If we plot the joint angle and target position in the image, respectively $q_1(t)$ ($q_2(t)$) on the vertical axis and

$x(t)$ ($y(t)$) on the horizontal axis, we see how the target follows different paths according to the direction of movement. Moreover we can even see how backlash makes the robot not coming back exactly to the previous configuration. Our network have to face both these effects appearing during data recordings.

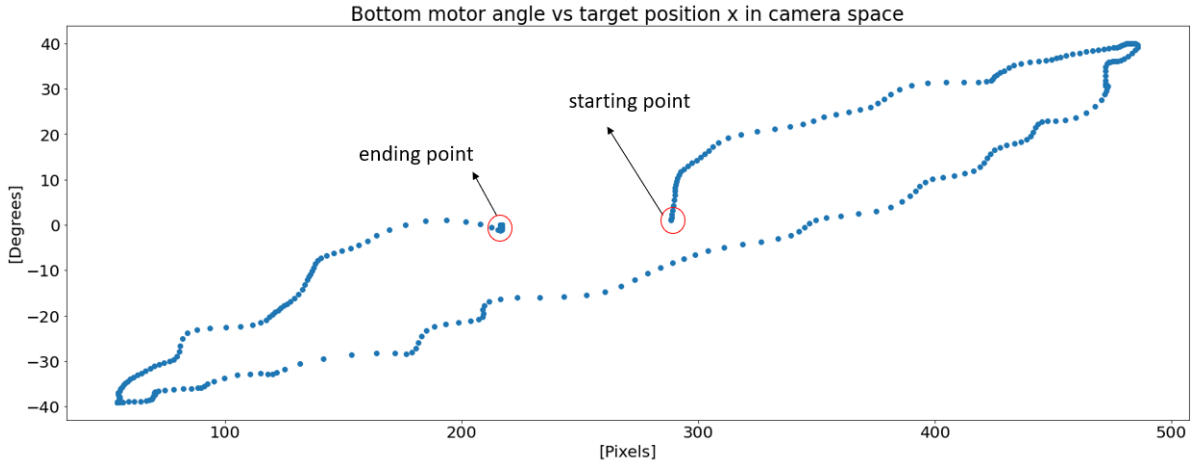


Figure 5.1: Hysteresis example during the recording of one sinusoidal cycle from $DATASET\_1\_x\_testing$ (a = 0.7 rad and T = 8 sec).

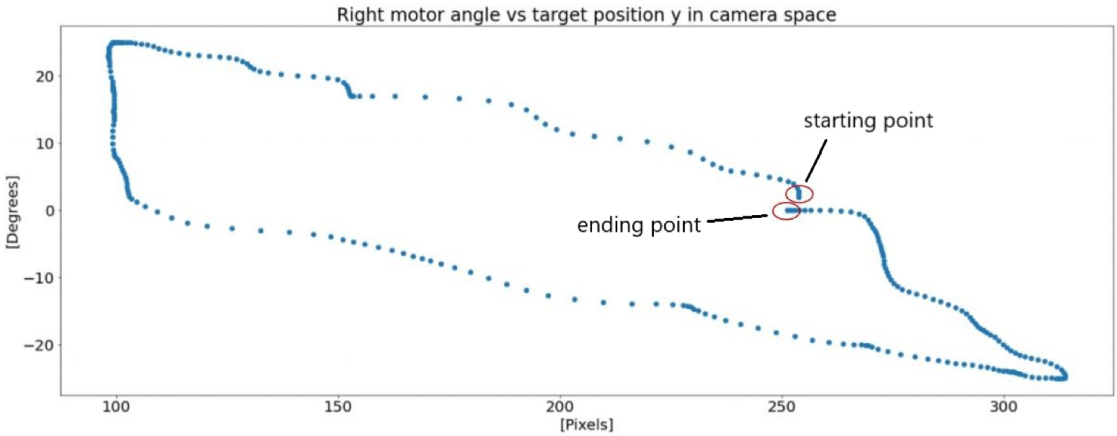


Figure 5.2: Hysteresis example during the recording of one sinusoidal cycle from $DATASET\_1\_y\_testing$ (a = 0.6 rad and T = 8 sec).

Here we report in Table 5.1 and Table 5.2 our network performance parameters according to the metrics defined in Section 3.9 both in modelling the bottom motor - x direction and right motor - y direction experimental curves for the datasets we collected.

Performance parameters for x direction

Dataset	MSE	MAE	$\lvert e_{Pmax} \rvert$
$DATASET\_1\_x\_testing$	0.2295	0.2477	5.0264
$DATASET\_2\_x\_testing$	0.126	0.2662	2.821
$DATASET\_3\_x\_testing$	0.0744	0.2176	1.2423
$DATASET\_4\_x\_testing$	0.2804	0.2797	13.2327

Table 5.1: Performance metrics x.

Performance parameters for y direction

Dataset	MSE	MAE	$\lvert e_{Pmax} \rvert$
$DATASET\_1\_y\_testing$	0.3523	0.3015	8.5112
$DATASET\_2\_y\_testing$	0.1927	0.3189	3.2693
$DATASET\_3\_y\_testing$	0.1622	0.33	1.1966
$DATASET\_4\_y\_testing$	1.0781	0.6168	9.5002

Table 5.2: Performance metrics y.

Considering the model for the x direction, we see in Table 5.1 that the MSE is lower than 0.3 (degrees squared) for all the datasets. MAE confirms that the mean absolute error is lower than 0.3 degrees for all the datasets. These parameters show that the network can predict the angle value associated to a given position of the target in the image. We notice an high maximum error values. To understand the reason of the $|e_{Pmax}|$ high values we inspected the datasets. Looking at the the absolute errors graphs we noticed that the highest error values that are shown by $|e_{Pmax}|$ happened in correspondence of the joining points between the recording composing a single dataset. In Figure 5.3 we see the prediction error module for the $DATASET\_4\_x\_testing$ where the higher maximum error appears. We see that the highest errors are limited to few points in specific locations where the files are joined. This can be explained with the fact that the last point of a file recording is not correlated with the first point of the following recording. Thus the network is not able to correctly predict the joint value in these points because we are giving in input measurement that are not related to the same path.

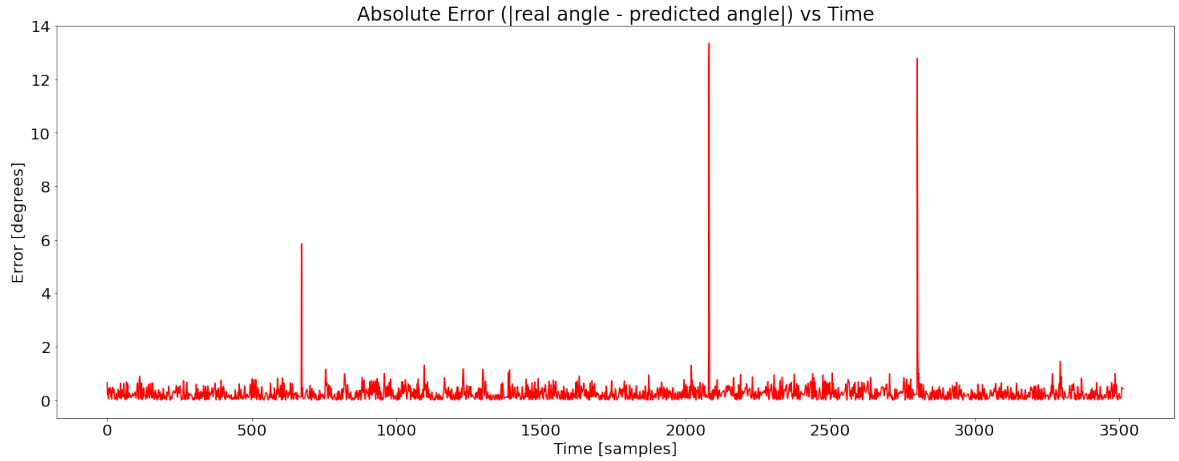


Figure 5.3: Absolute error between actual angle values and predictions for $DATASET\_4\_x\_testing$.

Thus we can say that our model is able to successfully predict the joint position. Here we show the predictions compared to the real points for the same dataset we considered to discuss the error peaks.

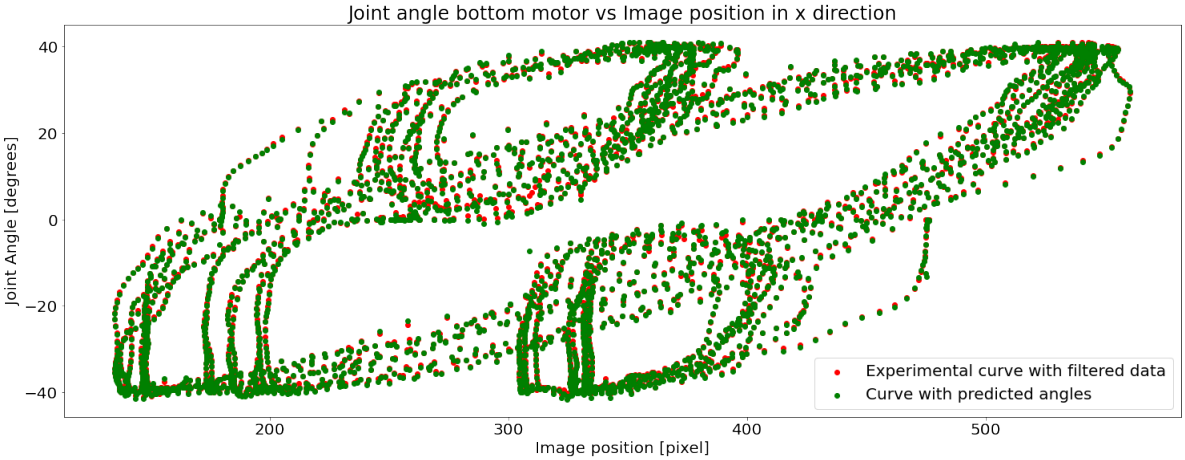


Figure 5.4: $DATASET\_4\_x\_test$ data and predictions.

The network is able to predict correctly for all the different testing datasets with similar performances over all datasets. This shows that the network is not affected by different angle motor trajectories neither by variations in amplitude nor in frequency. Is it interesting to underline the behaviour obtained over dataset 4 (Figure 5.4). We see three classical hysteresis shapes due to different position of the circle set for the recordings composing the dataset. We can see how the three elliptic shapes are shifted left and right due to the recordings where we varied the circle position on the screen showing it leftward or right-

wards (Pos_3 and Pos_5 Figure 4.2). The dimensions of the the three elliptic shapes are different due to the different inputs exploited. In Figure 5.5 we show the angle curve comparing real curve and predictive curve.

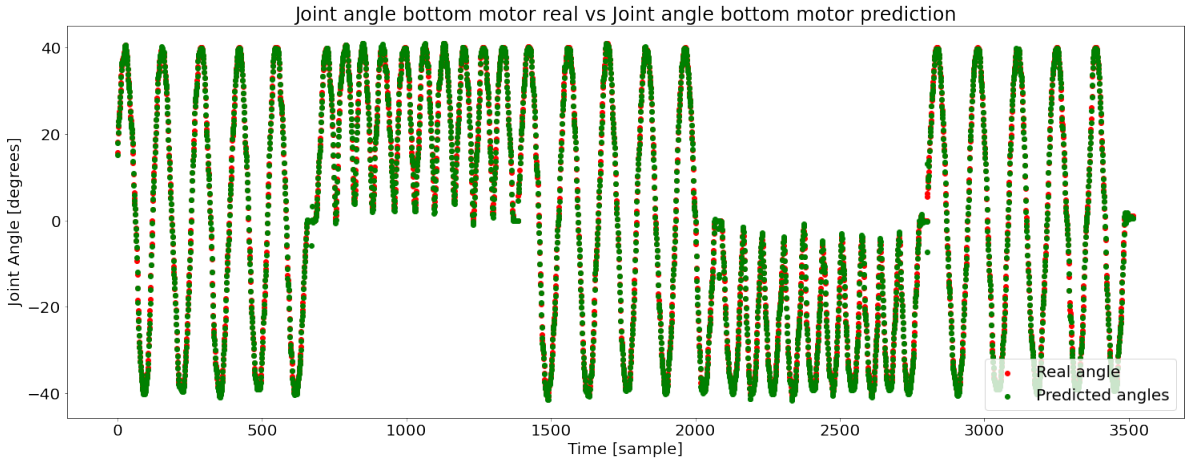


Figure 5.5: $DATASET\_4\_x\_testing$ angle data and and predicted angles.

Let's inspect the model for the y direction. According to the Table 5.2 metrics shows higher values with respect to the model x performances. Here we report the network behaviour over $DATASET\_4\_y\_testing$ in order to show the model behaviour. It appears that the worse performances are present in correspondence of the lower hysteresis shape when the circle is in Pos_1 and the $Input\_3$ is applied (Table 4.3).

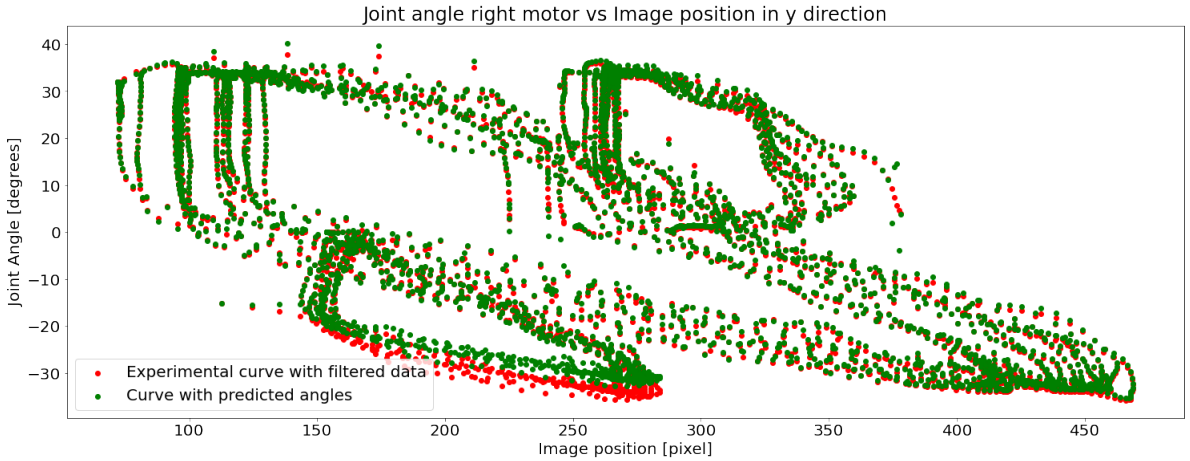


Figure 5.6: $DATASET\_4\_y\_testing$ data and predictions.

As we see the network is not able to follow the hysteresis curve, The network can' t reach the lowest angle values.

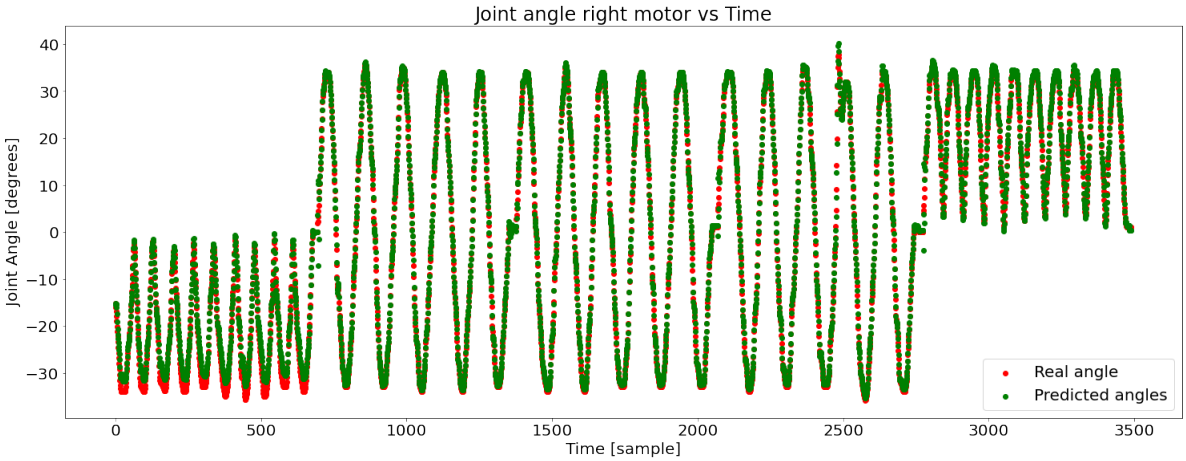


Figure 5.7: *DATASET_4_y_testing* angle data and and predicted angles.

This causes consistent errors when the shifted circle force shifts the data in the lower region of the graph.

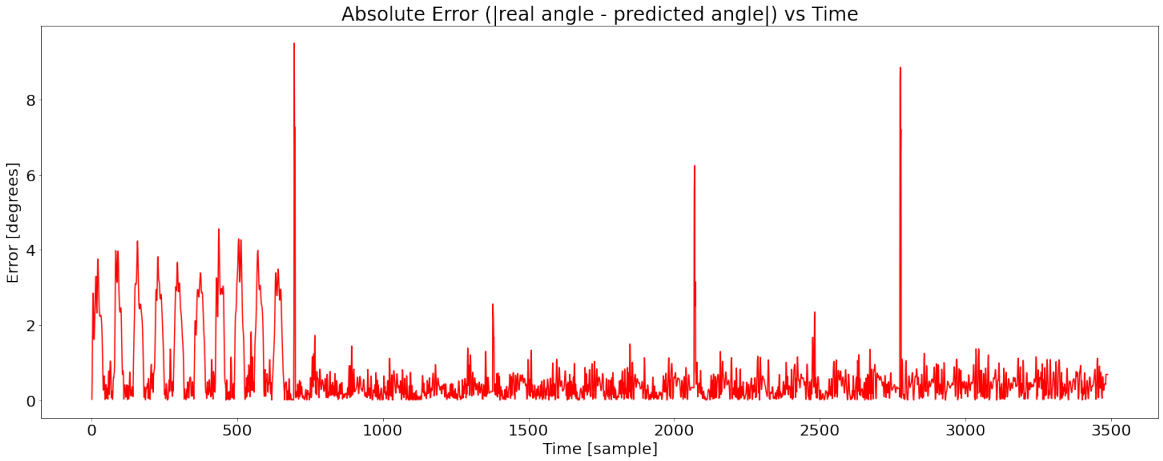


Figure 5.8: Absolute error between actual angle values and predictions for *DATASET_4_y_testing*

This behavior might be due to the fact that the network during the training met few points in this region to be trained or it is actually influenced by the different circle position on the screen. The first cause if it was correct it might be potentially solved normalising the data. Instead in the second case a different analysis should be performed.

5.4. P controller

As previously described we had to tune the (K_{Px}, K_{Py}). Thanks to the experiment 1 we found a suitable parameter combination with:

- $K_{Px} = 0.5$;
- $K_{Py} = 0.2$.

When we set parameters K_{Px} set to 0.5 and K_{Py} set to 0 we analysed the error behaviour. We search for a small steady state error and a limited overshoot. Here the horizontal component error $(e_x(t))$ is considered. In Figure 5.9 we see how a peak in the error rises when the circle jumps. The P controller for the bottom motor is trying to center the target in order to reduce the error $e_x(t)$ to zero.

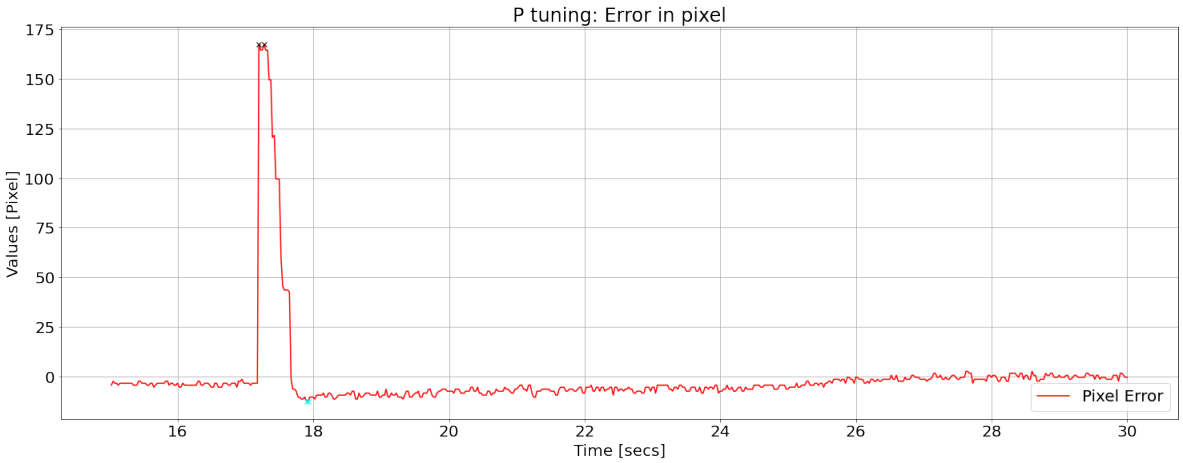


Figure 5.9: Error in the image space along x direction ($K_{Px} = 0.5$,$K_{Px} = 0$).

An analogue behaviour was recorded setting $K_{Px} = 0$ and $K_{Py} = 0.2$. Here we show the response of vertical component $(e_y(t))$ error.

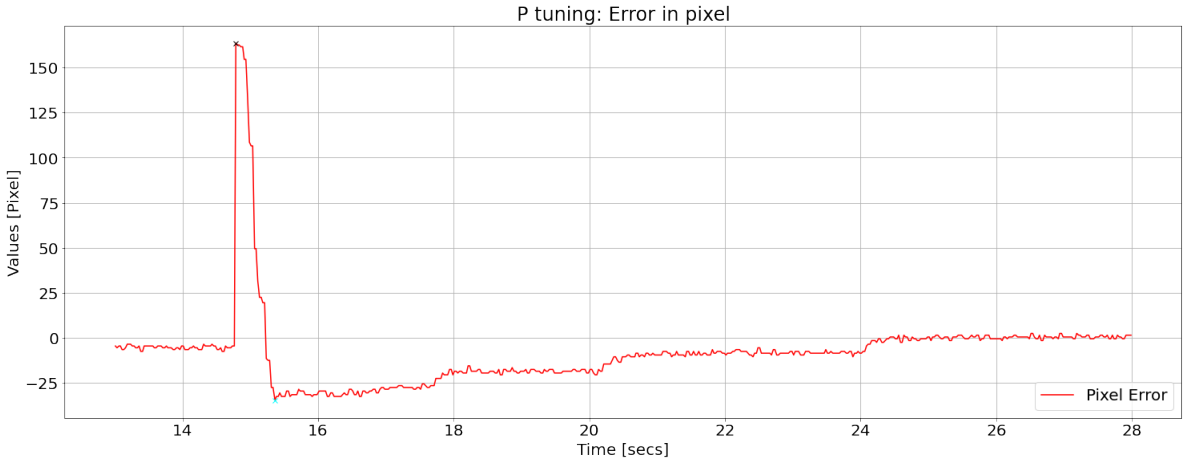


Figure 5.10: Error in the image space along y direction ($K_{Px} = 0$, $K_{Px} = 0.2$).

Let's inspect the controllers performances according to different types of input when both the controllers are active. We will inspect the centering and tracking task both in a free and in a constrained environment according to the metrics defined in Section 3.9.

5.4.1. Centering task in free space

Centering task in free space is analysed in experiment 2. We aimed to repeat this experiment only once repeating the steps sequence five times. After three sequences the robot faced an unusual behaviour that forced us to stop the experiment. At the end of the third repetition of the step sequence the robot stopped moving for some seconds and then bent exceeding the range limits for the joint values and we had to power off the robot. We repeated the experiment other two times but in one recording a topic was not recorded and in the third recording we faced the same problem immediately at the end of the first repetition thus we kept the first recording as valid one. We inspect the performances over the three repetitions we correctly recorded. The circle jumps as describes in Section 3.6 creating steps in the image error space. Considering Figure 4.5 when the circle jumps from positions 4 or to position 4 moving in the horizontal direction the step will ideally only in the e_x component, vertically the step will be ideally in the e_y component. Small variations happens even in the direction not directly involved even due to the action of the controllers working together. We will analyse the parameter for the error components that are involved in the step and for the module of the tracking error to keep trace of the overall action of the two P controllers. If the circle jumps in a diagonal way the step will appear instead in both the error components.

In the following tables we collected the performance parameters for each step direction.

We report the mean value and the standard deviation over the valid repetitions recorded for all the parameters.

Experiment 2.1 - Performance parameters

Step Direction	Parameter	Mean [pixel]	Std [pixel]	Repetitions
4 to 0	$SSE_{tracking}$	6.05 px	0.66 px	3
	SSE_x	−4.73 px	0.73 px	
	SSE_y	−3.46 px	0.33 px	
	$E_{tracking_{max}}$	236.63 px	0.66 px	
	$E_{overshoot_x}$	22.7%	7.3%	
	$E_{overshoot_y}$	12.3%	5.7%	
4 to 1	$SSE_{tracking}$	5.55 px	2.39 px	3
	SSE_x	–	–	
	SSE_y	−3.68 px	1.18 px	
	$E_{tracking_{max}}$	169.79 px	5.78 px	
	$E_{overshoot_x}$	–	–	
	$E_{overshoot_y}$	12.5%	0.6%	
4 to 2	$SSE_{tracking}$	11.25 px	9.26 px	3
	SSE_x	9.03 px	9.72 px	
	SSE_y	−4.88 px	3.31 px	
	$E_{tracking_{max}}$	242.24 px	7.65 px	
	$E_{overshoot_x}$	25.8%	8%	
	$E_{overshoot_y}$	19.1%	5.7%	
4 to 3	$SSE_{tracking}$	7.52 px	1.11 px	3
	SSE_x	−6.45 px	1.41 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	171.35 px	3.08 px	
	$E_{overshoot_x}$	25.65%	6.41%	
	$E_{overshoot_y}$	–	–	
4 to 5	$SSE_{tracking}$	7.98 px	4.75 px	3
	SSE_x	−0.13 px	5.1 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	170.04 px	2.79 px	
	$E_{overshoot_x}$	26.5%	19%	
	$E_{overshoot_y}$	–	–	

Table 5.3: P controller performances in centering task in free space.

Experiment 2.2 - Performance parameters

Step Direction	Parameter	Mean [pixel]	Std [pixel]	Repetitions
4 to 6	$SSE_{tracking}$	5.47 px	1.89 px	3
	SSE_x	−1.58 px	0.73 px	
	SSE_y	−3.46 px	1.41 px	
	$E_{tracking_{max}}$	241.49 px	3.22 px	
	$E_{overshoot_x}$	9.9%	2.3%	
	$E_{overshoot_y}$	25.6%	3.5%	
4 to 7	$SSE_{tracking}$	9.96 px	6.43 px	3
	SSE_x	–	–	
	SSE_y	3.1 px	6.94 px	
	$E_{tracking_{max}}$	176.41 px	0.63 px	
	$E_{overshoot_x}$	13.2%	6.4%	
	$E_{overshoot_y}$	12.3%	5.7%	
4 to 8	$SSE_{tracking}$	6.3 px	–	2
	SSE_x	–	–	
	SSE_y	−3.46 px	–	
	$E_{tracking_{max}}$	247.6 px	–	
	$E_{overshoot_x}$	–	–	
	$E_{overshoot_y}$	–	–	

Table 5.4: P controller performances in centering task in free space.

We see how in the step directions from position 4 to position 8 in Table 5.4 and from position 8 to position 4 in Table 5.6 only two recordings are available. This is due to the behavior the robot showed that made us stop the recording.

Experiment 2.3 - Performance parameters

Step Direction	Parameter	Mean [pixel]	Std [pixel]	Repetitions
0 to 4	$SSE_{tracking}$	9.62 px	1.37 px	3
	SSE_x	5 px	2.36 px	
	SSE_y	0.4 px	7.72 px	
	$E_{tracking_{max}}$	259.27 px	0.52 px	
	$E_{overshoot_x}$	18.6%	5.4%	
	$E_{overshoot_y}$	16.3%	2.1%	
1 to 4	$SSE_{tracking}$	7.49 px	4.77 px	3
	SSE_x	–	–	
	SSE_y	0.43 px	1.01 px	
	$E_{tracking_{max}}$	183 px	1.8 px	
	$E_{overshoot_x}$	–	–	
	$E_{overshoot_y}$	20.3%	3.9%	
2 to 4	$SSE_{tracking}$	9.71 px	4.84 px	3
	SSE_x	−9.01 px	4.75 px	
	SSE_y	−1.83 px	2.9 px	
	$E_{tracking_{max}}$	264.15 px	11.98 px	
	$E_{overshoot_x}$	25.1%	7.4%	
	$E_{overshoot_y}$	27.9%	5.6%	
3 to 4	$SSE_{tracking}$	7.05 px	2.6 px	3
	SSE_x	−9.01 px	4.75 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	179.64 px	1.56 px	
	$E_{overshoot_x}$	19.3%	9.6%	
	$E_{overshoot_y}$	–	–	
5 to 4	$SSE_{tracking}$	7.74 px	3.96 px	3
	SSE_x	−1.98 px	1.17 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	173.23 px	5.63 px	
	$E_{overshoot_x}$	27.2%	14.4%	
	$E_{overshoot_y}$	–	–	

Table 5.5: P controller performances in centering task in free space..

Experiment 2.3 - Performance parameters

Step Direction	Parameter	Mean [pixel]	Std [pixel]	Repetitions
6 to 4	$SSE_{tracking}$	4.52 px	1.8 px	3
	SSE_x	0.1 px	2.22 px	
	SSE_y	−0.95 px	0.07 px	
	$E_{tracking_{max}}$	239.06 px	3.99 px	
	$E_{overshoot_x}$	25.3%	7.5%	
	$E_{overshoot_y}$	18%	3.2%	
7 to 4	$SSE_{tracking}$	9.10 px	7.78 px	3
	SSE_x	–	–	
	SSE_y	2.98 px	8.23 px	
	$E_{tracking_{max}}$	170.8 px	6.91 px	
	$E_{overshoot_x}$	–	–	
	$E_{overshoot_y}$	20.6%	19.4%	
8 to 4	$SSE_{tracking}$	11.1 px	–	2
	SSE_x	−7.65 px	–	
	SSE_y	2.47 px	–	
	$E_{tracking_{max}}$	237.27 px	–	
	$E_{overshoot_x}$	21.5%	–	
	$E_{overshoot_y}$	29.8%	–	

Table 5.6: P controller performances in centering task in free space.

Considering all the step directions where three repetitions are available in the previous tables, $SSE_{tracking}$ is under 12 pixel for all the step directions. The controller is always able to center the target in the middle of the image space. A critical parameter for our controller is the mean overshoot that is often higher than 20%. Here we show an example of the controller response after a circle step from position of how the tracking error changes in time after the circle on the screen jumped from position 4 to position 2 with a Figure 5.11.

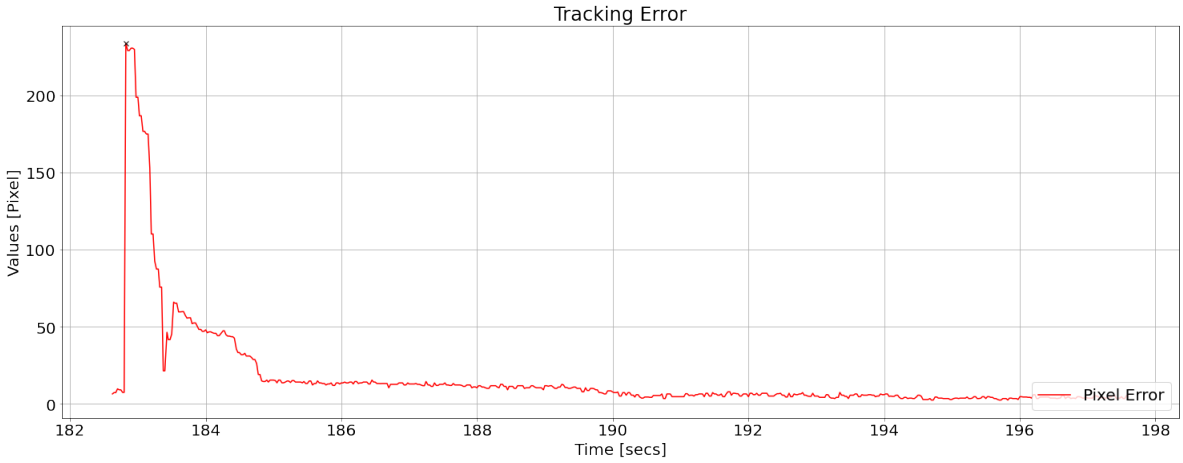


Figure 5.11: Tracking error curve in time after a step input.

5.4.2. Tracking task in free space

Tracking task is inspected in experiment number 3. We performed the experiment nine times. Among these nine recordings we discard recording number 7 and recording number 9 since during the inspection the appeared to be incomplete, recordings stopped before the target performed the whole trajectory. For each of the remaining recordings we will take into account the tracking error defined in Equation (3.9).This parameter tells us the distance of the target from the desired central point in the image space expressed in pixel. We compute the root mean squared value of this parameter over the recording. We consider as performance parameters the maximum value and Root Mean Squared value of the tracking Error($RMSE_{tracking}$ and e_{max} in Table 5.7) computing their mean and standard deviation over the recordings. If we consider that in a recording we have N samples, $RMSE_{tracking}$ is defined as:

$$RMSE_{tracking} = \sqrt{\frac{\sum_{1=1}^{N} e_{tracking}(i)^2}{N}}. \tag{5.1}$$

Experiment 3 - Performance parameters

Name	Mean [pixel]	Std [pixel]	Experiment repetitions
$RMSE_{tracking}$	59.6402	4.4046	7
e_{max}	167.3172	17.3206	7

Table 5.7: P controller performances in tracking task.

We see in (Figure 5.12 how both $RMSE_{tracking}$ and e_{max} are pretty high. Comparing how the circle moves on the monitor and the tracking error we see that the tracking error gets higher when the slope of the curves of the coordinates on the monitor reach its highest. Coordinates varies as sine-waves thus the circle speed vary and when it is at the highest value the controller need time to track the target. Instead when it slows down the error drops since the controller is able to recover.

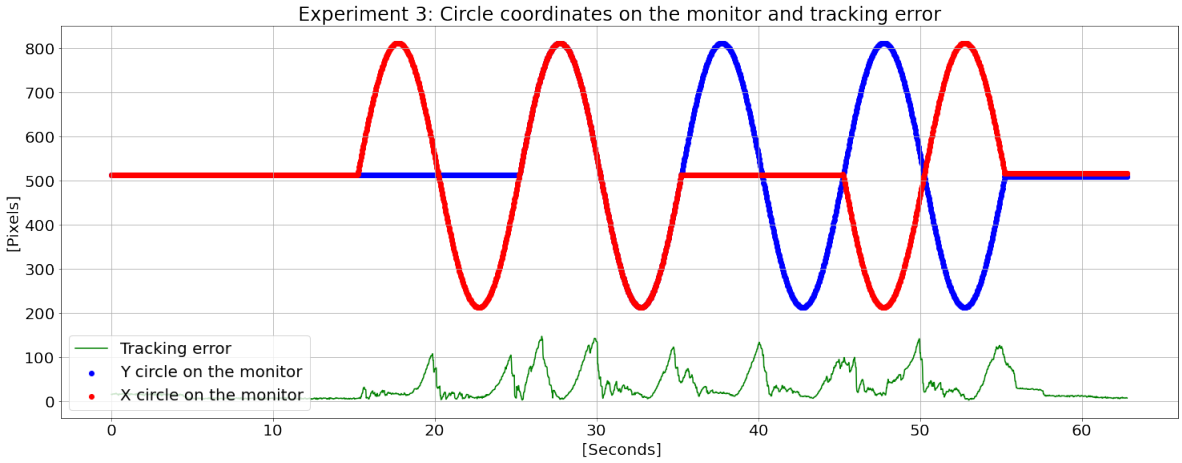


Figure 5.12: Tracking error curve in time and circle movement on the screen.

5.4.3. Centering task in constrained space

In order to access the robustness of our P controller we tested our controller in presence of an obstacle. The centering task is tested in experiment 4 in particular in the jumping version. Here we repeated the circle jumping sequence five times, we compute the same parameter performances analysed for experiment 2, each one with its mean and standard deviation over the five repetitions as we can see in Table 5.8. In step direction 4 to 5 we have four repetitions because of the early cutting of the recording.

Experiment 4 jumping - Performance parameters

Step Direction	Parameter	Mean [pixel]	Std [pixel]	Repetitions
4 to 3	$SSE_{tracking}$	10.19 px	11.69 px	5
	SSE_x	−8.43 px	10.47 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	175.01 px	9.43 px	
	$E_{overshoot_x}$	21.2%	4.9%	
	$E_{overshoot_y}$	–	–	
4 to 5	$SSE_{tracking}$	15.13 px	7.27 px	4
	SSE_x	1.92 px	5.79 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	172.34 px	7.59 px	
	$E_{overshoot_x}$	22.2%	12.5%	
	$E_{overshoot_y}$	–	–	
3 to 4	$SSE_{tracking}$	6.49 px	3.95 px	5
	SSE_x	0.22 px	5.70 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	182.34 px	11.16 px	
	$E_{overshoot_x}$	14%	5.1%	
	$E_{overshoot_y}$	–	–	
5 to 4	$SSE_{tracking}$	7.66 px	3.06 px	5
	SSE_x	−0.63 px	4.36 px	
	SSE_y	–	–	
	$E_{tracking_{max}}$	176.09 px	4.16 px	
	$E_{overshoot_x}$	17.1%	8.1%	
	$E_{overshoot_y}$	–	–	

Table 5.8: P controller performances in centering task in constrained space.

The robot touches the obstacle while it is moving rightwards then when the circle is displaced from position 4 to position 5. We can see how the steady state error is comparable to the one of steps in other directions although it appears slightly higher. In Figure 5.13 we show an example of how the tracking error changes in time after the circle on the

screen jumped from position 4 to position 5, that will make the robot hit the obstacle.

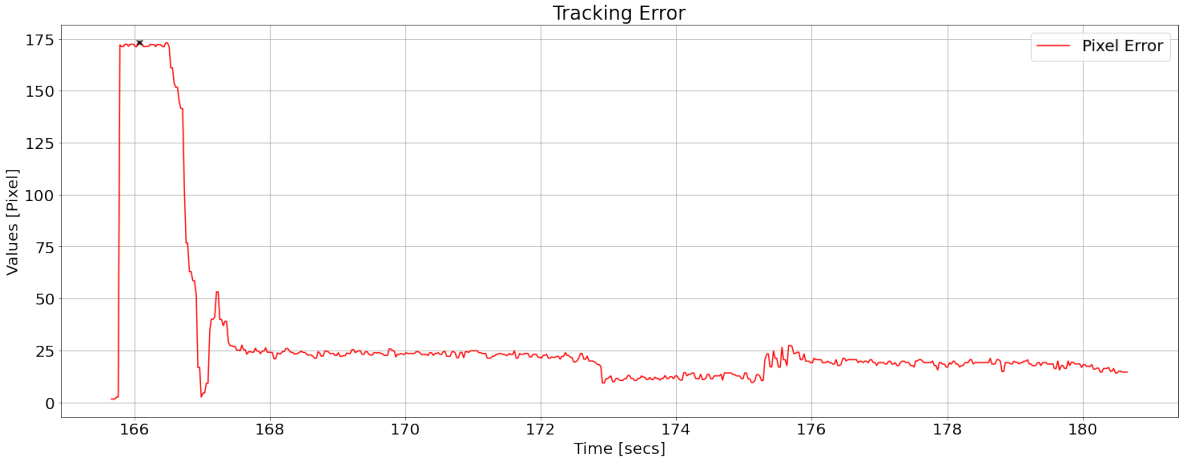


Figure 5.13: Tracking error curve in time after a step input.

We wanted to report this example because it shows how the controller is forcing the robot in order to have the target in the central position, but the obstacle is introducing a disturbance which might introduce oscillations. Although this effect considering all the repetitions the controller is able to force the steady state error around to an acceptable steady state error as shown in Table 5.8.

5.4.4. Tracking task in constrained space

We wanted to verify if the robot can track the circle even in presence of an obstacle so we report the results from the continuous version of experiment 4. We shifted the circle on the screen left and right direction for several times. We let the circle shift left and right on the monitor around the central position of the image varying the x coordinate only as a sine wave of amplitude 300 pixels and period 10 seconds. We completed 30 cycles. Excluding last portion of the recording where the robot lost completely the target and considering sixty second in the middle of the recording we can see in Figure 5.13 the evolution of the tracking error is time. We can recognise a similar behaviour as in the free space tracking experiment, but we were not able to acquire more data, thus this is a qualitative consideration and inspection.

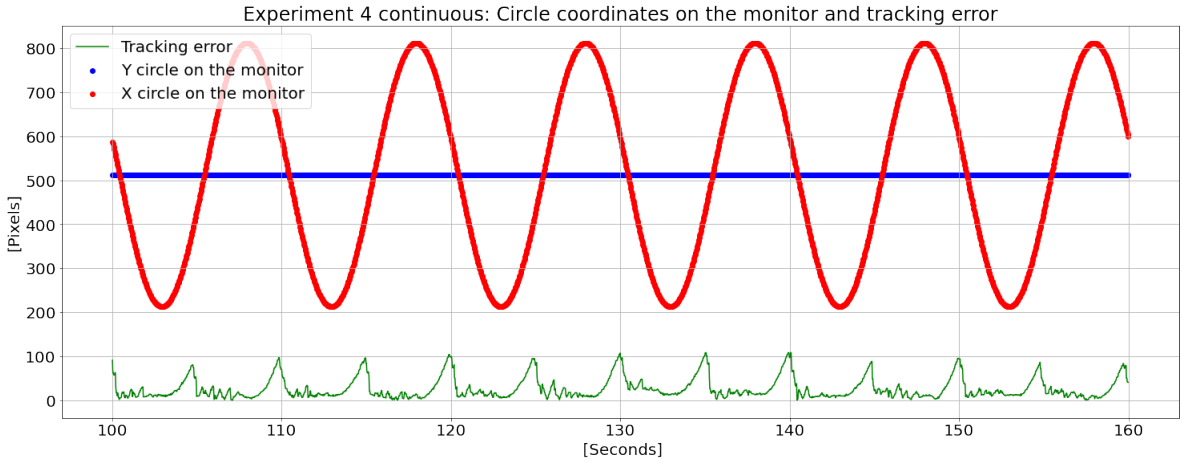


Figure 5.14: Tracking error curve and circle coordinates in time.

5.5. Controller Performances

We tested our controller in centering and tracking task inside different environments: free and constrained space. The controller is able to accomplish the centering task both in free and in a simple constrained environment showing comparable performances although the mean tracking error and its variance increased. In detail all directions are correctly handled. Comparing experiment 2 and 3 in Figure 5.15 for the same step direction from position we can see that the mean tracking increases keeping a a good accuracy even if some oscillations appeared before the steady state was reached.

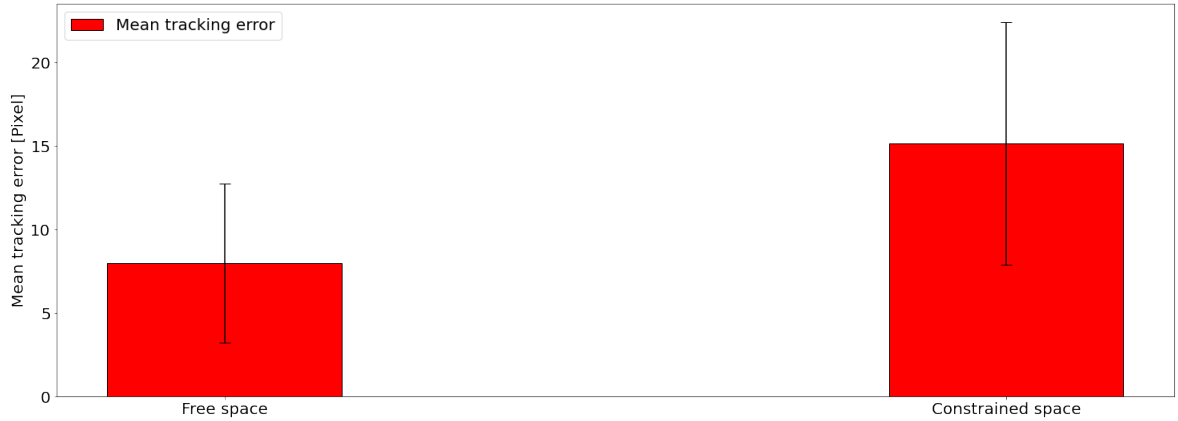


Figure 5.15: Performance comparison for centering task.

In the tracking task the controller appear weaker, the response time is pretty long with respect to the target movement when it reaches the highest speed. The velocity limitation might have enhanced this limitation but we decided to keep this limit imaging the robot

will move slowly in clinical application. We were not able to perform a deeper analysis for the constrained tracking task that will be considered in the future.

6 | CONCLUSION

This thesis work aimed to build a model and a controller to drive the new autonomous soft continuum endoscope developed in the European ATLAS project. In detail a visual servoing controller was chosen and we inspected the deep learning modelling. We a accomplished the following targets:

- modelling of the device non linear behaviour;
- design and implementation of a visual servoing controller.

In detail two models are considered: a model to link the x direction in the image and the bottom motor and a model to link the y direction in the image space and the right motor. The model x proved to be able to correctly predict the joint angle given the image position of the target and the history of the previous steps. It is not affected by different motor joint trajectories and robust against an initial shift of the target. The model for the y direction is weaker and affected especially by the target shift. A not correct data distribution might have caused this weakness of the model.

In parallel we implemented P controller composed of two independent P visual servoing controllers able to accomplish centering task with a reduced steady state error. The overshoot is still consistent and the response time showed to be high for tracking objects with a relevant speed. This is due even to the speed limitation we set because we focused our attention on slow tasks application. Tracking a slow target will reduce the tracking error in tracking tasks. The controller proved to be robust showing analogue performances in free and constrained environments. Thus our controller proves to be a valid starting point, being able to reach a steady state error in centering task aligned with the literature results. Its limitation about the speed of response in tracking and the overshoot might be improved exploiting the model presented in this work.

6.0.1. Future work

Atlascope has mechanical properties that can slightly modify during the tasks as the majority of TDCRs. This because of friction and movements of electric cables that reaches

the camera on the tip. In our case assembling proved to be essential for the performances of the robot. An incorrect cable mounting and fixing as well as not perfect alignment of the camera with the horizon cause an higher tendon coupling and nullify the assumption of independence between the two direction handling. A deep learning model as the proposed one might forecast the different robot behaviours allowing a parameter adjustment of the P controllers in order to prevent the overshoot or to decrease the response time. We started studying the model integration in order to build a P weighted controller that would adjust the C_{calib} constants but a damage in the electronic hardware made the control system loose the joint position forcing Atlascope to over bent beyond every safety limits and forced us to reassemble its flexible arm. Thus a future development will include the integration of the model inside the control architecture and the inspection of other promising deep learning techniques for example LSTM neural networks.

Bibliography

[1] V. Agrawal, W. J. Peine, and B. Yao. Modeling of transmission characteristics across a cable-conduit system. *IEEE Transactions on Robotics*, 26(5):914–924, 2010.

[2] G. Baroni. Notes of methods for biomedical imaging and computer aided surgery. 2022.

[3] Q. Boehler, D. S. Gage, P. Hofmann, A. Gehring, C. Chautems, D. R. Spahn, P. Biro, and B. J. Nelson. Realiti: A robotic endoscope automated via laryngeal imaging for tracheal intubation. *IEEE Transactions on Medical Robotics and Bionics*, 2(2):157–164, 2020.

[4] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar. A review of pid control, tuning methods and applications. *International Journal of Dynamics and Control*, 9(2): 818–827, 2021.

[5] J. Burgner-Kahrs, D. C. Rucker, and H. Choset. Continuum robots for medical applications: A survey. *IEEE Transactions on Robotics*, 31(6):1261–1280, 2015.

[6] M. Buscarini and M. Conlin. Update on flexible ureteroscopy. *Urologia Internationalis*, 80(1):1–7, 2008. ISSN 0042-1138. doi: 10.1159/000111721. URL `https://www.karger.com/DOI/10.1159/000111721`.

[7] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury. Mechanics modeling of tendon-driven continuum manipulators. *IEEE transactions on robotics*, 24(6):1262–1273, 2008.

[8] G. Chen, M. T. Pham, and T. Redarce. Sensor-based guidance control of a continuum robot for a semi-autonomous colonoscopy. *Robotics and autonomous systems*, 57(6-7): 712–722, 2009.

[9] L. Cheng, W. Liu, Z.-G. Hou, J. Yu, and M. Tan. Neural-network-based nonlinear model predictive control for piezoelectric actuators. *IEEE Transactions on Industrial Electronics*, 62(12):7717–7727, 2015.

[10] V. K. Chitrakaran, A. Behal, D. M. Dawson, and I. D. Walker. Setpoint regulation of continuum robots using a fixed camera. *Robotica*, 25(5):581–586, 2007.

[11] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2017. doi: 10.21105/joss.00456. URL `http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf`.

[12] C. Culmone, P. W. Henselmans, R. I. van Starkenburg, and P. Breedveld. Exploring non-assembly 3d printing for novel compliant surgical devices. *Plos one*, 15(5): e0232952, 2020.

[13] S. Doizi and O. Traxer. Flexible ureteroscopy: technique, tips and tricks. *Urolithiasis*, 46(1):47–58, 2018. ISSN 2194-7236. doi: 10.1007/s00240-017-1030-x. URL `https://doi.org/10.1007/s00240-017-1030-x`.

[14] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler. Design and control of concentric-tube robots. *IEEE Transactions on Robotics*, 26(2):209–225, 2009.

[15] M. Elkoushy and S. Andonian. Prevalence of orthopedic complaints among endourologists and their compliance with radiation safety measures. *Journal of endourology / Endourological Society*, 25:1609–13, Aug 2011. doi: 10.1089/end.2011.0109.

[16] L. Fan and E. M. Joo. Design for auto-tuning pid controller based on genetic algorithms. In *2009 4th IEEE Conference on Industrial Electronics and Applications*, pages 1924–1928. IEEE, 2009.

[17] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi. Control strategies for soft robotic manipulators: A survey. *Soft robotics*, 5(2):149–163, 2018.

[18] G. Giusti, S. Proietti, L. Villa, J. Cloutier, M. Rosso, G. M. Gadda, S. Doizi, N. Suardi, F. Montorsi, F. Gaboardi, and O. Traxer. Current standard technique for modern flexible ureteroscopy: Tips and tricks. *European Urology*, 70(1):188–194, 2016. ISSN 0302-2838. URL `https://www.sciencedirect.com/science/article/pii/S0302283816300112`.

[19] X. T. Ha, D. Wu, C.-F. Lai, M. Ourak, G. Borghesan, A. Menciassi, and E. Vander Poorten. Contact localization of continuum and flexible robot using data-driven approach. *IEEE Robotics and Automation Letters*, 2022.

[20] Humanitas. Urinary apparatus. 2022. URL `https://www.humanitas.it/enciclopedia/anatomia/`.

[21] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[22] K. Ikuta, Y. Matsuda, D. Yajima, and Y. Ota. Pressure pulse drive: A control method for the precise bending of hydraulic active catheters. *IEEE/ASME Transactions on Mechatronics*, 17(5):876–883, 2011.

[23] J. Jung, R. S. Penning, N. J. Ferrier, and M. R. Zinn. A modeling approach for continuum robotic manipulators: Effects of nonlinear internal device friction. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5139–5146. IEEE, 2011.

[24] M. D. Kutzer, S. M. Segreti, C. Y. Brown, M. Armand, R. H. Taylor, and S. C. Mears. Design of a new cable-driven manipulator with a large open lumen: Preliminary applications in the minimally-invasive removal of osteolysis. In *2011 IEEE International Conference on Robotics and Automation*, pages 2913–2920. IEEE, 2011.

[25] J. F. Lazo, C.-F. Lai, S. Moccia, B. Rosa, M. Catellani, M. de Mathelin, G. Ferrigno, P. Breedveld, J. Dankelman, and E. De Momi. Autonomous intraluminal navigation of a soft robot using deep-learning-based visual servoing. *arXiv preprint arXiv:2207.00401*, 2022.

[26] Z. Li, M. Zin Oo, V. Nalam, V. Duc Thang, H. Ren, T. Kofidis, and H. Yu. Design of a novel flexible endoscope—cardioscope. *Journal of Mechanisms and Robotics*, 8 (5):051014, 2016.

[27] T. Liu and M. C. Çavuşoğlu. Three dimensional modeling of an mri actuated steerable catheter system. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 4393–4398. IEEE, 2014.

[28] X. Ma, C. Song, P. W. Chiu, and Z. Li. Autonomous flexible endoscope for minimally invasive surgery with enhanced safety. *IEEE Robotics and Automation Letters*, 4(3): 2607–2613, 2019.

[29] X. Ma, C. Song, P. W. Chiu, and Z. Li. Visual servo of a 6-dof robotic stereo flexible endoscope based on da vinci research kit (dvrk) system. *IEEE Robotics and Automation Letters*, 5(2):820–827, 2020.

[30] M. Magro. Robotic actuation and autonomous control of a tendon-driven catheter for structural intervention cardiology. 2022.

[31] F. Martina, X. T. Ha, L. Jorge, C.-F. Lai, R. Sanat, H. Albert, B. Gianni, D. Diego, T. Selene, R. Benoit, et al. Multi-level-assistance robotic platform for navigation in

the urinary system: Design and preliminary tests. In *Conference on New Technologies for Computer/Robot Assisted Surgery*, pages 90–91, 2022.

[32] F. H. Moll and D. Schultheiss. How endoscopy founded modern urology. 2018.

[33] A. A. Nazari, K. Zareinia, and F. Janabi-Sharifi. Visual servoing of continuum robots: Methods, challenges, and prospects. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 18(3):e2384, 2022.

[34] E. A. of Urology. Eau guidelines on urolithiasis 2022. 2022. URL `https://uroweb.org/eau-guidelines`.

[35] E. A. of Urology. Eau pocket on urolithiasis 2022. 2022. URL `https://uroweb.org/guidelines`.

[36] Opencv. Opencv hough circle transform. 2022. URL `https://docs.opencv.org/4.5.3/d4/d70/tutorial_hough_circle.html`.

[37] L. Ott, F. Nageotte, P. Zanne, and M. de Mathelin. Robotic assistance to flexible endoscopy by physiological-motion tracking. *IEEE Transactions on Robotics*, 27(2): 346–359, 2011.

[38] F. Porpiglia, N. Buffi, and A. Et. *Urologia*. 2015.

[39] E. project. Autonomous intraluminal surgery. 2022. URL `https://atlas-itn.eu/`.

[40] J. Rassweiler, M. Fiedler, N. Charalampogiannis, A. S. Kabakci, R. Saglam, and J.-T. Klein. Robot-assisted flexible ureteroscopy: an update. *Urolithiasis*, 46(1):69–77, 2018. ISSN 2194-7236. doi: 10.1007/s00240-017-1024-8. URL `https://doi.org/10.1007/s00240-017-1024-8`.

[41] J. J. Rassweiler, R. Autorino, J. Klein, A. Mottrie, A. S. Goezen, J.-U. Stolzenburg, K. H. Rha, M. Schurr, J. Kaouk, V. Patel, P. Dasgupta, and E. Liatsikos. Future of robotic surgery in urology. *BJU International*, 120(6):822–841, Dec 2017. ISSN 1464-4096. URL `https://doi.org/10.1111/bju.13851`. https://doi.org/10.1111/bju.13851.

[42] ROS. Ros tutorial 2022. 2022. URL `https://docs.ros.org/en/rolling/Tutorials/Beginner-CLI-Tools.html`.

[43] ROS. Ros tutorial 2022. 2022. URL `http://wiki.ros.org/ROS/Tutorials`.

[44] ROS. Ros message filters. 2022. URL `http://wiki.ros.org/message_filters/ApproximateTime`.

[45] ROS. Ros control package, pid. 2022. URL `http://docs.ros.org/en/indigo/api/control_toolbox/html/classcontrol__toolbox_1_1Pid.html`.

[46] R. Saglam, A. Y. Muslumanoglu, Z. TokatlÄ?, T. ??a??kurlu, K. Sarica, A. Ä. Ta???§i, B. Erkurt, E. S??er, A. S. Kabakci, G. Preminger, O. Traxer, and J. J. Rassweiler. A new robot for flexible ureteroscopy: Development and early clinical results (ideal stage 1â??2b). *European Urology*, 66(6):1092–1100, 2014. ISSN 0302-2838. URL `https://www.sciencedirect.com/science/article/pii/S0302283814006216`.

[47] L. Sciavicco, B. Siciliano, L. Villani, and G. Oriolo. Robotics: Modelling, planning and control, ser. advanced textbooks in control and signal processing, 2011.

[48] C. Shi, X. Luo, P. Qi, T. Li, S. Song, Z. Najdovski, T. Fukuda, and H. Ren. Shape sensing techniques for continuum robots in minimally invasive surgery: A survey. *IEEE Transactions on Biomedical Engineering*, 64(8):1665–1678, 2016.

[49] A. Vandini, A. Salerno, C. J. Payne, and G.-Z. Yang. Vision-based motion control of a flexible robot for surgical applications. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6205–6211. IEEE, 2014.

[50] R. J. Webster, J. M. Romano, and N. J. Cowan. Mechanics of precurved-tube continuum robots. *IEEE Transactions on Robotics*, 25(1):67–78, 2008.

[51] D. Wu, M. Ourak, K. Niu, Y. Zhang, M. A. Ahmad, J. Dankelman, and E. Vander Poorten. Towards modeling of hysteresis in robotic catheters based on lstm. In *32nd Conference of the International Society for Medical Innovation and Technology (iSMIT), Date: 2020/12/03-2020/12/10, Location: Chicago, USA*, 2020.

[52] D. Wu, Y. Zhang, M. Ourak, K. Niu, J. Dankelman, and E. Vander Poorten. Hysteresis modeling of robotic catheters based on long short-term memory network for improved environment reconstruction. *IEEE Robotics and Automation Letters*, 6(2): 2106–2113, 2021.

[53] H. Yang, B. Wu, X. Liu, and K. Xu. A closed-loop controller for a continuum surgical manipulator based on a specially designed wrist marker and stereo tracking. In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 335–340. IEEE, 2020.

[54] Z. Yaniv, E. Wilson, D. Lindisch, and K. Cleary. Electromagnetic tracking in the clinical environment. *Medical physics*, 36(3):876–892, 2009.

[55] M. C. Yip and D. B. Camarillo. Model-less feedback control of continuum manipu-

lators in constrained environments. *IEEE Transactions on Robotics*, 30(4):880–889, 2014.

[56] M. C. Yip and D. B. Camarillo. Model-less hybrid position/force control: a minimalist approach for continuum manipulators in unknown, constrained environments. *IEEE Robotics and Automation Letters*, 1(2):844–851, 2016.

[57] N. Zelenko, D. Coll, A. T. Rosenfeld, and R. C. Smith. Normal ureter size on unenhanced helical ct. *American Journal of Roentgenology*, 182(4):1039–1041, 2004.

[58] X. Zhang, Y. Tan, and M. Su. Modeling of hysteresis in piezoelectric actuators using neural networks. *Mechanical Systems and Signal Processing*, 23(8):2699–2711, 2009.

[59] Z. Zhang, B. Rosa, O. Caravaca-Mora, P. Zanne, M. J. Gora, and F. Nageotte. Image-guided control of an endoscopic robot for oct path scanning. *IEEE Robotics and Automation Letters*, 6(3):5881–5888, 2021.

List of Figures

List of Tables

Acknowledgements

At the end of this work I would like and need to thank many people who supported me during this long and not always linear path.

I am very thankful to Professor De Momi for the oppurtunity of working at the Nearlab and her supevision during this work. I am really thankful to Chun-Feng that followed me week by week even being physically distant in the last months. I am glad for the opportunity to work with you and thankful for you patience and advises.

If these last months of work were beautiful it is thanks to the members of the lab with whom I shared this path. Mattia, Andrea, Chiara, Jessica, Ilaria I am very happy to have met you.

I am thankful to all my friends who supported me during these years. Marta, Mattia, Lorenzo, Andrea, Cristian, Serena, Johnathan, Matteo, Carmen, Edda and Rossella thank you for all time spent together. I can't forget to be thankful to the Baradello crew that is with me since I have memory.

I must express how deeply thankful I am to Stefano, Eleonora, Elena, Pres and Maurizio. You supported me in a way I can neither describe. Thank you for your patience and believing in me. Thank you Maurizio for literally pushing me in the last exams.

Thank you Cinzia for your help and friendship in these years and to Layla for her attention and kindness.

In the end I want to express my feelings to my family. I want to say that without you I wouldn't be who and where I am today thus I can just express here a small part of the gratefulness I feel for you. Thank you to my parents to support me in all ways possible and to my brother for his sometimes funny way of supporting me. I will try to give back at least a small part of you gave me. Thank you to my grandmothers and all my relatives for always inspiring me in doing my best.

Thank you