POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Towards Real-Time Tool Tracking for Autonomous Exoscope Control

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Author: **Diego Cattaneo**

Student ID: 967476
Advisor: Prof. Elena De Momi
Co-advisor: Elisa Iovene
Academic Year: 2022-23

Abstract

Exoscopes represent a promising 3D visual solution in the neurosurgical field aimed at offering an improved field of view and ergonomics to the surgeons compared with traditional surgical microscopes. However, the need for manual repositioning of the exoscope, each time a different viewpoint of the scene is required, may compromise the smoothness of the surgical procedure and bring to longer operation times. Researchers of the Nearlab Laboratory of the Politecnico di Milano proposed a working framework for an autonomous vision-guided camera holder (a redundant robotic manipulator) that tracked and followed a selected surgical instrument based on a markerless visual servoing technique. Unfortunately, the low processing speed of the Convolutional Neural Network, used to detect the tool in the images, causes a low system responsiveness with respect to tool movements and, as a consequence, a slow and inaccurate tracking of the surgical instrument.

In this thesis work, a novel hybrid tracking module is proposed to allow the autonomous exoscope system to achieve a real-time tool tracking. This module is composed of an Optical flow tracking method that substitutes the Convolutional Neural Network and by a modified Particle Filter predictor designed to estimate the future position of the tracked instrument on the basis of the previous movements. To control the robot a resolved velocity controller was employed in two different ways (Position control and Orientation control). The hybrid tracking module was validated employing simultaneously two robots, one to move the surgical tool in a repeatable, predefined trajectory at two different speeds and above two different backgrounds, and one as automatic camera holder. After the validation, the controller was fine-tuned enabling the autonomous exoscope to follow the surgical tool in real-time with high accuracy (tracking error in Position control mode approximately equal to 1 cm). Finally a User Study was conducted to investigate if the proposed system was able to reduce the users' workload with respect to the manual repositioning of the camera and the results confirmed the effectiveness of the automatic control, showing that it offers a better field of view and reduce the task duration.

Keywords: Visual Servoing, Tool Tracking, Optical Flow, Particle Filter, Exoscope, Collaborative Robotics

Sommario

Gli esoscopi sono una soluzione visiva promettente nel campo della neurochirurgia. Il loro scopo è offrire ai chirurghi una migliore visuale e una maggiore ergonomia rispetto ai microscopi chirurgici tradizionali. Tuttavia, la necessità di riposizionare manualmente l'esoscopio, ogni volta che è richiesto un punto di vista differente, può compromettere la fluidità della procedura chirurgica e allungarla. I ricercatori del Laboratorio Nearlab del Politecnico di Milano hanno proposto come soluzione il fissaggio dell'esoscopio a un supporto (un manipolatore robotico ridondante) mosso automaticamente sulla base delle informazioni visive derivanti dall'esoscopio stesso. Questo sistema inseguiva uno specifico strumento chirurgico tramite una tecnica di Markerless Visual Servoing, ma a causa della bassa velocità di elaborazione della Rete Neurale Convoluzionale (utilizzata per rilevare lo strumento), l'inseguimento dello strumento chirurgico era lento, impreciso e instabile.

In questa tesi, per consentire all'esoscopio autonomo un inseguimento in tempo reale, viene proposta un'innovativa tecnica ibrida di inseguimento. Questa prevede l'uso del flusso ottico come sostituto della rete neurale e l'impiego di un Particle Filter predictor per predire, in base ai movimenti precedenti, la posizione futura dello strumento. Per controllare il robot è stato utilizzato un Resolved Velocity Controller in due modalità (controllo in posizione e in orientamento). La tecnica ibrida sviluppata è stata validata ricorrendo simultaneamente a due robot, uno per spostare lo strumento chirurgico lungo una traiettoria ripetibile e predefinita a due differenti velocità e sopra due sfondi diversi, e l'altro per consentire alla videocamera di inseguire lo strumento. Dopo la validazione, i parametri del controllore sono stati messi a punto permettendo al sistema di seguire lo strumento chirurgico in tempo reale e con elevata precisione (errore di inseguimento intorno a 1 cm durante il controllo in posizione). Infine, è stato condotto uno studio con utenti per verificare che il controllo autonomo, rispetto a quello manuale, fosse efficace nel ridurre il carico di lavoro. I risultati confermano l'ipotesi: il controllo autonomo offre un migliore campo visivo e può ridurre il tempo necessario per eseguire le operazioni.

Parole chiave: Visual Servoing, Inseguimento dello strumento, Flusso ottico, Particle Filter, Esoscopio, Robotica Collaborativa

Contents

1 | Introduction

1.1. Motivation

Exoscopes represent a promising visual solution in the neurosurgical field aimed at offering an improved field of view and ergonomics to the surgeons compared with traditional surgical microscopes. These are 3D surgical video telescopes which enable a flexible and ergonomic working environment, together with improved image quality. They can be mounted above the surgical scene at a great distance by means of holding arms providing the surgeon with an extended workspace. Exoscopes usually project the images on an external monitor, enabling the surgeon to operate in a more comfortable position with respect to traditional microscopes that require the contact with the eyepieces as shown in Figure 1.1.

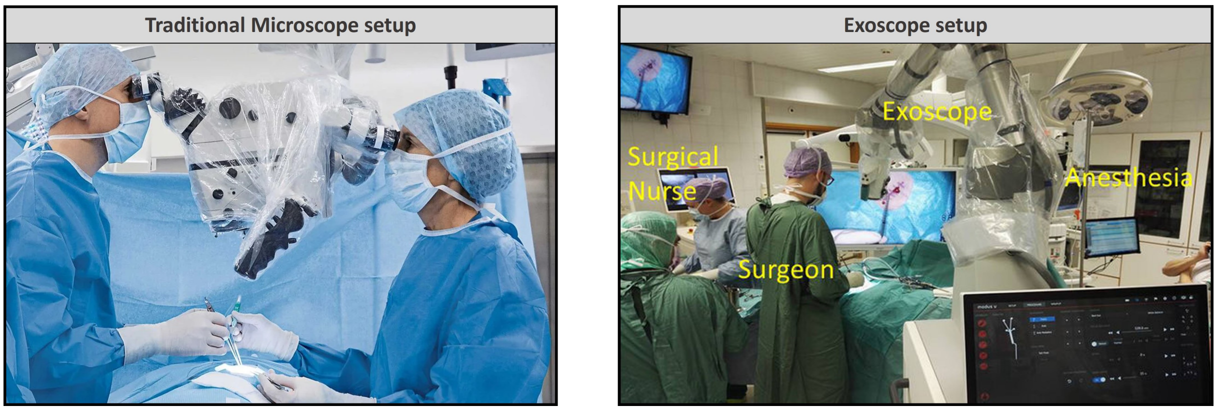

Figure 1.1: Traditional Microscope VS Exoscope setups: on the left-hand side it can be appreciated the traditional setup with the microscope (image taken from Leica Microsystem website) while on the right-hand side the photo of a typical exoscope setup during cranial surgery taken from [1].

The exoscopes have been largely accepted in the operating room and their superiority in terms of ergonomics has been ascertained [2]. However, the need for manual repositioning the exoscope, each time a different point of view of the scene is required, may threaten to

diminish the advantages brought: indeed, the manual movement of the exoscope causes the continuous switching between operation theatre and visualization system leading to interruptions that compromise the smoothness of the surgical procedure and that can possibly lead to longer operation times [3].
Alternatives to the manual repositioning (e.g. manual control units [3], foot-joystick control [4], gaze control [5], voice control [6], head gesture control [7]) have been proposed but none of them seemed to provide optimal operating conditions. Researchers of the Nearlab Laboratory of the Politecnico di Milano have proposed [8] a framework for an autonomous vision-guided camera holder that tracked and followed a selected surgical instrument based on a markerless visual servoing technique. However, the problem of this system is represented by the low processing speed of the Convolutional Neural Network (CNN) used to detect the tool in the images. This issue caused a low responsiveness of the system with respect to tool movements and, as consequence, a slow, inaccurate, and unstable tracking of the surgical instrument.
In this context, visual tracking techniques combined with position predictors could overcome the above presented limitations. Indeed, tracking algorithms are usually characterized by higher processing speed with respect to the object detection methods thanks to the fact that they exploit prior knowledge. These approaches track some points between consecutive frames and so they can use appearance and motion information acquired in the previous frames to speed up their task and make it more efficient. Predictors like the Kalman Filter predictor or the Particle Filter predictor, moreover, can be used to estimate the future position of a moving object exploiting its model and measurements coming from various sensors. The combined action of tracking technique and of position predictors applied to the autonomous exoscope system demonstrates high potential in facilitating the adoption of autonomous exoscopes in neurosurgery. This would allow the surgeon to focus entirely on the surgical operation while the camera repositioning task would be executed in real-time by a robotic holder assistant.

1.2. Aim of the Work

In this thesis work, a novel hybrid tracking module is proposed to solve the issues caused by the use of the CNN and so to allow the autonomous exoscope system to achieve a real-time tool tracking. The hybrid tracking module is composed of an Optical flow tracking module that substitutes the CNN whenever is possible and by a modified Particle Filter predictor designed to estimate the future position of the tracked instrument on the basis of the previous movements.

This thesis work is organized as follows. Chapter 2 is dedicated to the description of the state of the art. First, a brief overview of tool detection and tracking is presented. Then, various visual servoing techniques are discussed. Chapter 3 contains the materials and methods used in this work: it describes how the surgical tool is detected in the images and how this is tracked thank to the new proposed strategies. Then, the techniques used to control the camera holder are discussed. After that, a description of the experimental protocol followed in order to validate the system and to test its effectiveness is presented in Chapter 4. Finally, Chapter 5 illustrates the results of the conducted experiments that are then discussed in section 6. The final considerations on the work as well as the limitations and further research can be found in Chapter 7.

2 | State of the Art

2.1. Object Detection

Object detection is a computer vision task that involves identifying and locating objects of interest within an image or video. It has numerous applications, including autonomous driving, visual servoing, surveillance, facial recognition, augmented reality and more [9]. Among object detection approaches, there are two fundamental categories: marker-based and markerless object detection [10]. Each approach has distinct characteristics and applications.

Marker-based object detection relies on the presence of physical markers or patterns specifically designed to aid in the detection and tracking of objects within a visual scene. These markers are typically artificial and may include passive markers and active markers: to the first category belong for example color and shape-based markers (QR codes, barcodes, fiducial markers, unique patterns) and optical retro-reflective markers (that reflect a non-visible wavelength of light). One example of active markers is represented by infrared markers that emit (and not only reflect) infrared light to facilitate the localization of an object. The main steps involved in marker-based object detection are the following:

1. Marker Identification: The system identifies and locates markers within the scene by recognizing their unique patterns or codes.

2. Pose Estimation: Once markers are detected, their positions and orientations are calculated relative to the camera or sensor.

3. Object Localization: Objects associated with these markers can be accurately located based on the known marker positions.

Markerless object detection, instead, aims to identify and locate objects within an image or video without relying on predefined markers or patterns. This approach relies on the inherent visual features and characteristics of objects in the scene. In this case, the primary steps involved are the following:

1. Feature Extraction: The system extracts distinctive features, such as edges, corners,

textures, or key points, from the image or video frames.

2. Object Recognition: Using machine learning or computer vision algorithms, objects are recognized based on the extracted features, and their positions are estimated.
3. Object Localization: Bounding boxes or masks are placed around the recognized objects to indicate their locations.

Among the markerless objects detection methods, there is one that is based on the kinematic chain of the robot [11][12]. The position estimated with this strategy, however, could not be accurate because of measuring errors in the kinematic chain reconstruction. Markerless object detection is particularly valuable in applications where the use of physical markers or patterns is impractical or undesirable, such as in surgery, where adding markers could create problem of manageability and in sterilization phases [10]. Advances in deep learning have significantly improved the accuracy and robustness of markerless object detection systems.
Deep learning-based objects detection involves the use of neural networks (NN), particularly CNNs, to recognize objects of interest in images or video frames and to estimate their position [13]. The key steps in deep learning-based object detection are:

1. Data Collection and Annotation: A labeled data set is collected, containing images of objects along with corresponding annotations that specify the object's class (e.g. in surgical field: laparoscopic instrument, suction cannula, blood vessel) and the bounding box indicating its location within each image.
2. Model Architecture: A deep neural network architecture is chosen or designed for the task. Common choices include Faster R-CNN, YOLO, SSD, or custom architectures tailored to the specific requirements of object detection.
3. Training: The model is trained on the labeled data set using techniques like back-propagation and gradient descent. During training, the network learns to recognize features and patterns that represent objects.
4. Inference: After training, the model is used for inference on new, unlabeled data. It takes an input image and produces predictions regarding the presence and location of objects.
5. Post-processing: The model's output, which includes bounding boxes and class probabilities for detected objects, may undergo post-processing steps such as non-maximum suppression to refine and filter the detections.

For what specifically concern this thesis work, as reported in [10], most of the aforementioned markerless and marked-based tool detection techniques have been used to detect surgical tool in many research and a lot of studies are in progress.

2.2. Visual Object Tracking

Object tracking is a computer vision task that involves locating and following a specific object within a video sequence or a series of images. It finds applications in various domains, including surveillance, robotics, autonomous vehicles, and more [14].
Object tracking algorithm works as follows:

1. Initialization: The object tracker is initialized by specifying the object to track in the initial frame. This is often manually or automatically (e.g. using an object detection algorithm) by drawing a bounding box around the object.

2. Tracking: The tracker continuously estimates the position and scale of the object in subsequent frames by analyzing different features. Various tracking algorithms can be used for this purpose.

3. Re-detection (optional): To handle challenges like occlusion or tracking failure, some tracking systems incorporate re-detection steps to identify the object in the presence of difficulties.

The object's characteristics that a tracking algorithm can use for its purpose are multiple [15]. Here some examples are reported together with the relative tracking methods:

- Specific features or key points within the object (Feature point tracking, Corner detection-based tracking);
- Color information or color histograms (Color histogram matching, Color-based mean-shift tracking);
- Edge information (Edge detection-based tracking)
- Motion cues or optical flow (Lucas-Kanade optical flow-based tracking, Motion model-based tracking)
- Handcrafted features and mathematical models (Mean-Shift, KLT Tracker, Template Matching)
- Object templates (Template matching, Normalized cross-correlation).

There exist also tracking algorithms that combine multiple tracking techniques for improved performance and robustness (e.g. combining color and motion information).

Tracking algorithms can be classified on the basis of when the procedure is carried out and so they are divided in online tracking methods and offline tracking methods or based on the number of objects that they tracks simultaneously (Single Object vs. Multi-Object Tracking).
Tracking algorithms compared with tool detection methods have several advantages and disadvantages [16] that are listed in the table below:

Advantages	**Disadvantages**
Motion analysis: they provide information about object motion, velocity, and trajectory	Accuracy: tracking accuracy can vary depending on factors like object appearance and motion
Real-time monitoring: they allow for continuous monitoring of objects in video streams	Initialization: they have to be initialized to know which objects/pixels have to follow
Robustness against partial occlusions: they can maintain tracking even when objects are temporarily occluded	Challenging environments: object tracking can be difficult in complex or changing environments
Behavior analysis: useful for understanding and predicting object behavior	Object occlusion: they struggle when objects are heavily occluded.
Object model and training not needed: often these algorithms follow simple points between frames without any knowledge of the complex model of the object. Moreover, they do not need training or data set like neural network tool detection methods	Assumptions: many of these algorithms are based on constancy assumptions (e.g. the intensity of a pixel does not change, shape or color of object remain constant) or on simplifying assumptions (e.g. small movements between frames)
Computationally efficient: using the prior knowledge given by motion and behaviour analysis they can track the object rapidly and without an high computational cost	

Table 2.1: Advantages and disadvantages of tracking algorithms compared to tool detection methods.

The choice between object tracking and tool detection depends on the specific application and the goals of the computer vision system. Some applications may require a combination of both tasks to achieve the desired functionality (sometimes the object detection methods are used to initialize the tracking algorithms and to correct their possible drifts caused to long executions).

As reported in [10], also in the surgical field tracking algorithm are used and are under study.

2.3. Visual Servoing

Vision plays a key role in a robotic system, as, mimicking human sight, it can be used to retrieve geometrical and qualitative information on the workspace where the robot operates, without contact. Such information may be employed for an online path planning or for feedback control [17]. Nowadays several robotic controllers integrate vision systems in order to, for example, facilitate the pick and place task in case of non-constant position or orientation of the object to handle (bin-picking problem) [18] or to catch a flying ball with a robotic arm [19] or to follow an object [1][8].
In visual servoing, the vision sensor (e.g. a 2D camera, a stereo camera, an exoscope, a LIDAR) is used as the main sensor in the loop feedback to compute an appropriate error vector defined between the current pose of the object identified by the camera and the pose of the robot's end-effector.
The vision-based control schemes can be divided into two categories on the basis of the space in which they realize visual servoing:

- **Position-based visual servoing**, when it is realized in operational space;
- **Image-based visual servoing**, when it is realized in the image space.

The main difference lies in the fact that the schemes of the first category use visual measurements to reconstruct the relative 3D pose of the object with respect to a specific reference frame, while the schemes of the second category define an error function in terms of quantities that can be directly measured in an image. For example, with image-based methods if the goal is to track a point that the camera is observing, then the feedback error will be computed directly in the image plane as the Euclidean distance between the center of the image and that point. Then, a control law, which maps this error directly to robot motion, is constructed.
The position-based visual servoing approach is conceptually similar to the operational space control. The main difference is that feedback is based on visual measurements and pose reconstruction. The advantage of this method regards the possibility of acting directly on operational space variables, while the drawback of this approach is that the feedback loop could open due to lack of visual measurements and so instability may occur, if the object exits from the camera field of view.
In the image-space visual servoing approach, the control action is computed on the basis of the error defined as the difference between the desired and actual values of the image

feature parameters. In this case the advantages are that the 3D pose of the object do not need to be reconstructed and that the aforementioned opening of the feedback loop is less probable thanks to the direct control on the image features. However, the non-linear mapping between the image feature parameters and the operational space variables, can cause singular configurations that could bring to instability or saturation of the control action. Moreover, since the end-effector trajectories cannot be easily predicted, collisions with obstacles or joint limits violation may occur.
Another important aspect to take into account when discussing visual servoing is the camera positioning. There exist two main configurations: **eye-in-hand** and **eye-to-hand**.

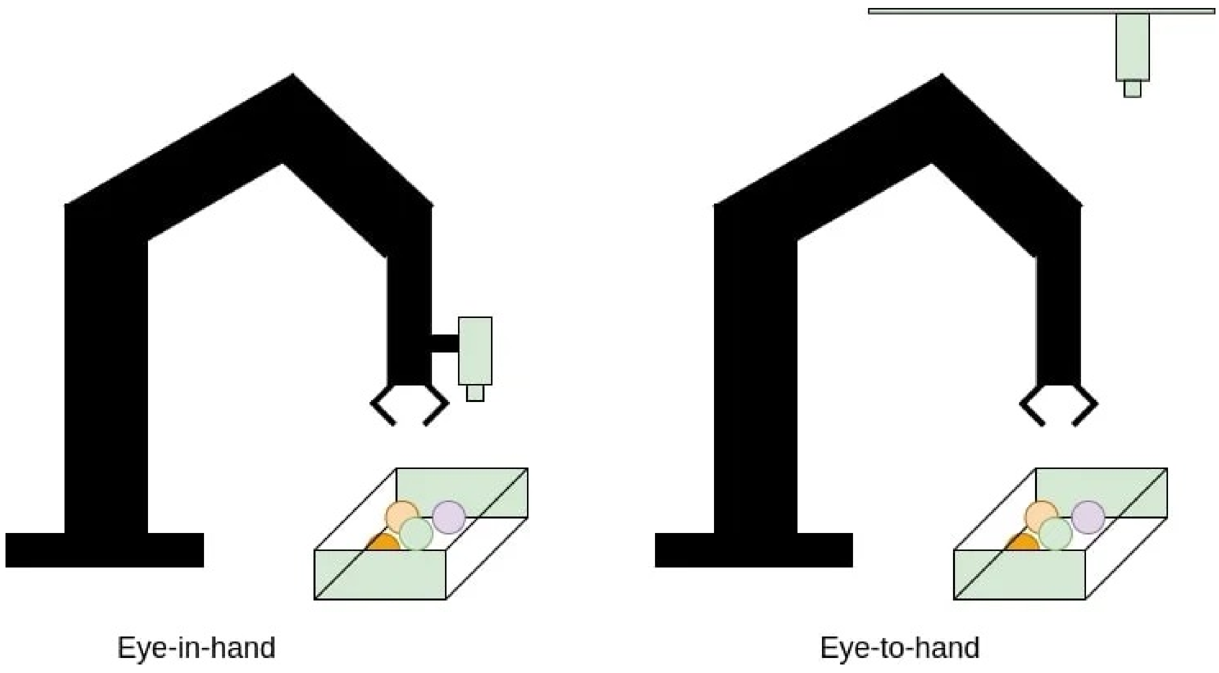

Figure 2.1: Camera configurations: on the left-hand side eye-in-hand setup and on the right-hand side eye-to-hand setup (Figure taken from [20]).

As shown in Figure 2.1, in eye-in-hand setup the camera is attached to the end effector or to the wrist of the robot. This configuration is optimal to observe without occlusions the area in which the end effector operates, but the geometric relationship between the camera and the workspace changes as the manipulator moves and the field of view can change dramatically for even small motions of the robotic arm.
In the eye-to-hand configuration, as represented in Figure 2.1, the camera is mounted in a fixed location so that it can observe the manipulator and its workspace. Differently from the previous setup, in this case the field of view does not change as the manipulator moves and the geometric relationship between the camera and the workspace is fixed and

can be calibrated offline. The drawbacks of this configuration are the possible occlusions of the workspace caused by the robotic arm motion.
Among all the visual servoing setups, two are particularly relevant in the contest of neurosurgery in general and of this thesis work in particular: the Modus V system released in 2017 by Synaptive Medical [1] and the markerless autonomous exoscope system developed by the Nearlab laboratory of the Politecnico di Milano described in [8].
The Modus V system represents the first autonomous camera control in neurosurgery. The system achieves hands free manipulation by means of robotic arm technology based on the Canadarm device. One surgical tool is equipped with passive markers that are recognized by a special stereo camera mounted in eye-to-hand configuration. In the system is present a foot pedal for motion activation, zoom in/out and auto-focus on/off.
The markerless autonomous exoscope presented in [8], instead, mount the exoscope on a 7 degrees of freedom redundant collaborative manipulator (KUKA LWR 4+ released by KUKA Roboter GmbH). It exploits the images streamed by the exoscope and a Yolo V3 Convolutional Neural Network (CNN) to detect, without the use of any type of marker, a specific surgical instrument and then uses visual servoing to align the camera with the tool.
The second setup is thought to be cheaper, to reduce the number of devices needed to guide the exoscope and to overcome the problem of sterilization given by the modification of the surgical instruments with markers. Unfortunately, because of a series of limitations, among which the most relevant is the low processing speed of the system caused by the use of the CNN, this system is not ready to the application in a real scenario.

3 | Materials and Methods

In this Chapter, an overview of the implemented system is presented, starting with a description of the employed hardware and then proceeding to explain the three different modules constituting the system, as shown in Figure 3.1:

- **Tool detection module**: it is employed to recognize a selected surgical instrument.
- **Hybrid tracking module**: it is used to track and predict the future position of the target tool.
- **Visual-servoing controller**: it is responsible for zeroing the error between the desired and actual pose of the robot.

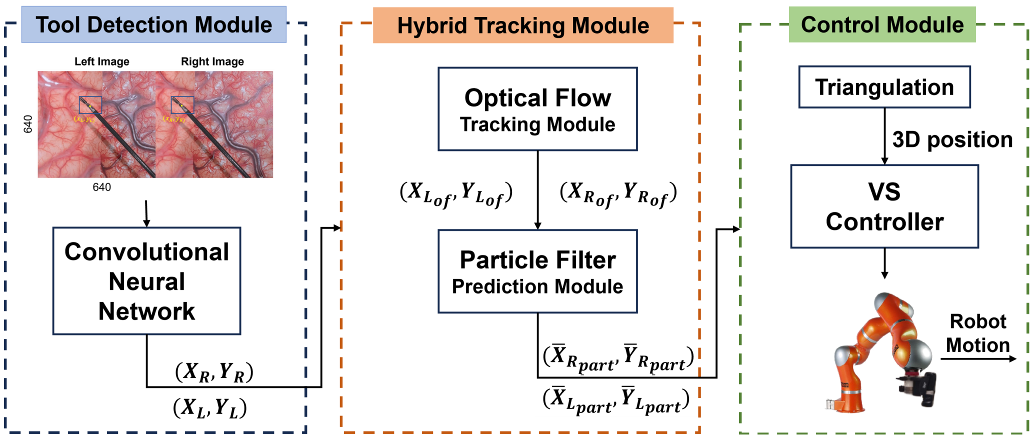
Tool Detection Module
Left Image
Right Image
640
640
Convolutional Neural Network
(X_R, Y_R)
(X_L, Y_L)
Hybrid Tracking Module
Optical Flow Tracking Module
$(X_{L_{of}}, Y_{L_{of}})$
$(X_{R_{of}}, Y_{R_{of}})$
Particle Filter Prediction Module
$(\bar{X}_{R_{part}}, \bar{Y}_{R_{part}})$
$(\bar{X}_{L_{part}}, \bar{Y}_{L_{part}})$
Control Module
Triangulation
3D position
VS Controller
Robot Motion

Figure 3.1: Overall System: images acquired by the stereo camera are sent to a CNN which detects the surgical tool. The 2D position of the tool is sent to a tracking module, then a prediction module estimates the future position of the tool in the image space. Finally, the 3D position of the tool is extracted from the predicted position and is fed to a visual-servoing controller.

3.1. Hardware Overview

The system is composed of a 7 Degrees of Freedom (Dof) manipulator, a 3D camera attached to its end-effector in an eye-in-hand configuration through a 3D printed support and a computer equipped with a 40GB GPU (Nvidia A100).

3.1.1. The Vision Sensor

In this study, the JVC Everio GS-TD1 Full HD 3D Camcorder was used as a vision sensor. The camera streams two images, right and left (one for each lens), that are used for the 3D position reconstruction of a laparoscopic instrument.
The camera is equipped with two 1/4.1" CMOS sensors and Twin HD GT lenses with an intra-axial distance of 35 mm (Figure 3.2) which allows to capture the scene from two slightly different point of views, granting so the 3D capabilities to the camera.

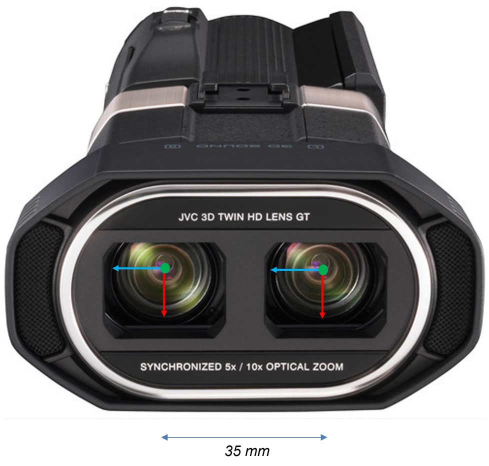

Figure 3.2: Vision sensor: JVC GS-TD1 Full HD 3D Camcorder.

In addition to the 3D capture, it offers full 1920×1080p high-definition capabilities as well as various zoom modes in both formats (3D and 2D) as shown in Table 3.1.

Image sensor	2x 1/4.1" CMOS		
Lens(3D)	F1.2 - 2.28, f= 3.76 mm to 18.8 mm		
Lens(2D)	F1.2 - 2.8, f= 3.76 mm to 37.6 mm		
Intra-axial distance	35 mm		
Image quality (3D)	3D Full HD 1920×1080 ×2		
Image quality (2D)	2D Full HD 1920×1080		
Zoom	Optical: ×5(3D)	Optical: ×10(2D)	Digital: ×200(2D)
Weight	675.85 g		
FPS	30		

Table 3.1: JVC GS-TD1 Full HD 3D Camcorder specifications.

The stereo camera is thus composed of two different 2D cameras and the most adopted representation of a 2D camera is the so-called "pinhole" camera [21].
A pinhole camera is a simple camera without a lens and with a single small aperture (so, it is an ideal model of a camera). Light rays pass through the aperture and project an inverted image on the opposite side of the camera, on the so-called image plane.
Figure 3.3 shows how the **pinhole camera model** works.

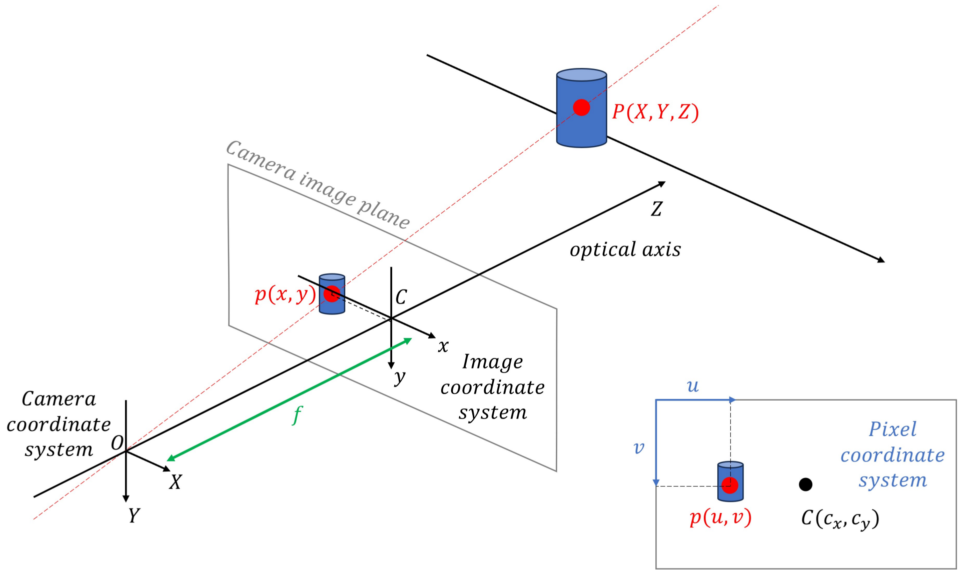

Figure 3.3: Pinhole camera model: the projection center $\boldsymbol{O}$ is the point where all the rays of light passing through the pinhole intersect. $\boldsymbol{P}(X,Y,Z)$ represents a point in the camera coordinates system which is projected on the camera image plane as $\boldsymbol{p}(x,y)$ or in pixel $\boldsymbol{p}(u,v)$. f represents the focal length, the distance of the image plane from $\boldsymbol{O}$ measured on the optical axis, where the camera is punting at. $\boldsymbol{C}$ is the principal point, where the camera image plane and the optical axis intersect.

The pinhole camera model relates a three-dimensional scene to the two-dimensional image captured on the camera sensor using similar triangles. Considering this model, the relationship between 2D projection of a point on the image and its 3D coordinates with respect to a world coordinates frame can be represented as follows

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.1}$$

where s is an arbitrary scaling factor accounting for the fact that points along a ray through the camera's pinhole are projected to the same image point, u and v are the 2D coordinates of the point expressed in pixel in the image plane.
f_x and f_y are the ratios (expressed in pixel) between the focal length f (expressed in mm) and the size of the pixel, s_x and s_y ($f_x = f/s_x$, $f_y = f/s_y$). f_x and f_y, together with the coordinates of the principal point (c_x,c_y), constitute the camera intrinsic matrix.
The extrinsic matrix instead is composed of the rotation ($\boldsymbol{r}$) and translation ($\boldsymbol{t}$) of the camera with respect to the world reference frame.
Finally, (X, Y, Z) are the 3D coordinates of the point expressed in the camera reference frame.
For simplicity the intrinsic and extrinsic parameter matrices will be called $\boldsymbol{A}$ and $\boldsymbol{T}$ respectively, so the equation 3.1 becomes:

$$s\boldsymbol{p} = \boldsymbol{ATP} \tag{3.2}$$

Lenses that are used in cameras introduce two types of distortions not considered in the ideal pinhole model: radial and tangential. Radial distortions are caused by lens imperfect shape and causes straight lines in real world to be bulged out in the image. Radial distortions can be represented as follow

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.3}$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.4}$$

The tangential distortions, instead, are caused by the CMOS chip not being perfectly parallel to the lens, making some areas in the image look nearer than expected and are represented as

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.5}$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{3.6}$$

where k_1, k_2, k_3, p_1, p_2 are the five camera distortion parameters.
Since these distortions are not taken into account in the pinhole model, its validity decreases toward the edges of the image where distortions effects increase while it remains reliable around the center. Distortion matrices are thus computed during the calibration procedure and applied to the images captured by the camera to avoid the loss of validity of the model.

A crucial aspect in the visual servoing framework is to have reliable information coming from the camera, since this sensor provides the feedback information to the controller. In order to guarantee reliability, the **camera calibration** process is carried out.
The calibration procedure aims at estimating:

- the intrinsic parameters represented by the matrix $\boldsymbol{A}$ in equation 3.2, expressing geometrical properties of the camera;
- the extrinsic parameters represented by $\boldsymbol{T}$ in 3.2 and expressing the pose of the camera frame with respect to the base frame.

The camera calibration was performed by the researchers of the NearLab laboratory exploiting the Zhang method [22] which is implemented in the OpenCv library. This approach requires extracting some image points observing a pattern from different orientations. The image 2D coordinates (x, y) and the corresponding 3D real world coordinates of those points are the information needed for the calibration. For the system described in this work, a (13×9) chessboard pattern has been used and the points extracted were the intersection of two black squares. A number of 17 images of the calibration pattern were used and the 3D positions of the points in the real world were expressed with respect to the chessboard upper left corner. The algorithm supposes, without loss of generality, that the pattern used is being kept stationary at XY plane meaning that the Z coordinate of the points in the real world is equal to zero. To eliminate the scaling factor ambiguity, the length of the sides of the squares (20 mm, as shown in Figure 3.4) was provided to the algorithm.

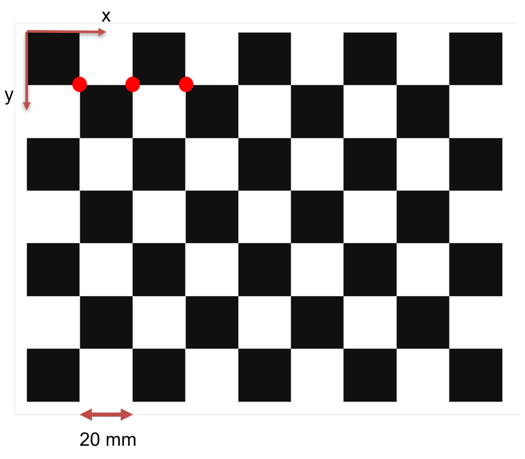

Figure 3.4: Chessboard pattern and examples of extracted points (red dots). Real 3D coordinates are expressed with respect to the upper left corner of the chessboard pattern.

The intrinsic parameters for the right camera resulting from the camera calibration are the following:

$$\boldsymbol{A}_R = \begin{bmatrix} 1062.92 & 0 & 467.91 & 0 \\ 0 & 2124.9 & 541.86 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.7}$$

The extrinsic parameters, expressing the position of the left camera with respect to the right one, were the identity matrix for the rotation and a -34.7 mm translation along the X axis. This result is coherent with the data provided by the stereo camera manufacturer which reports 35 mm lens spacing (intra-axial distance in table 3.1).
The distortion coefficients obtained with the camera calibration are:

$$Dist_{coeff} = \begin{bmatrix} -2.78E-1 & 2.15E-1 & 7.61E-4 & 8.94E-5 & -8.30E-2 \end{bmatrix} \tag{3.8}$$

and have been used to undistort the images before the triangulation procedure explained in Section 3.4.1.

3.1.2. The Camera Holder

The camera holder structure is composed of the 7 DOF robotic manipulator and the 3D printed camera support.
In this work, the stereo camera has been mounted in an eye-in-hand configuration on a KUKA LWR 4+ robotic manipulator, released in 2010 by the German manufacturer of industrial robots KUKA Roboter GmbH. This is a collaborative robot, so it has been purposely designed to directly cooperate with humans inside a defined workspace thanks

to appropriate safety measures dedicated to avoiding injuries to people. This robot has been widely used for research purposes.

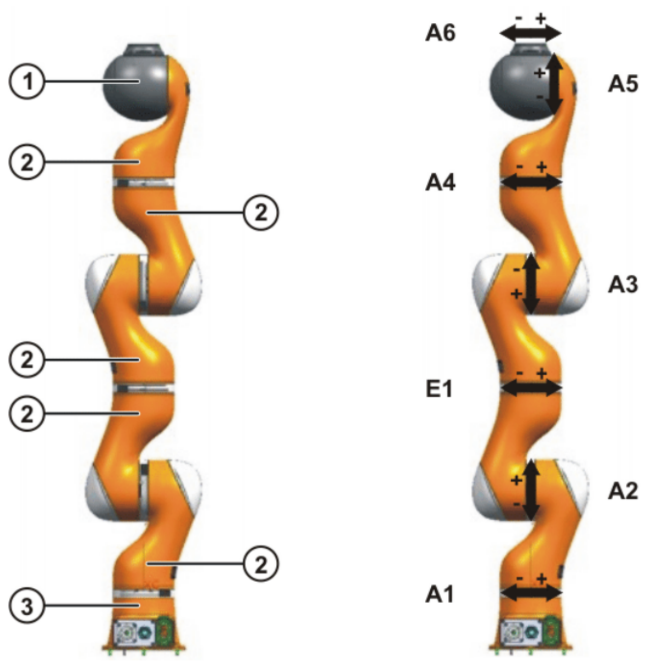

Figure 3.5: Main assemblies and robot axes.

As shown in Figure 3.5, the robot is composed of a 2-axis in-line wrist (1), five joint modules (2), and a base (3) totaling to 7 degrees of freedom. When the joint space dimension is higher than the task space dimension ($7 > 6$), robots are defined "redundant". Redundant manipulators are usually characterized by high dexterity and manoeuvrability, easier obstacle and singularities avoidance, energy efficiency and increased safety. It is a lightweight robot (approx. 16 kg) and it can lift up to 7 kg (rated payload).

In order to secure the 3D camera on the Kuka LwR 4+ robotic manipulator, the NearLab's researchers have modeled a support in Solid Works and they have printed it using the Ultimaker S3 3D printer. To be able to sustain the weight of the camera, the support was printed in ABS with an 80% infill density and triangular infill pattern. The printed camera support is shown in Figure 3.6.

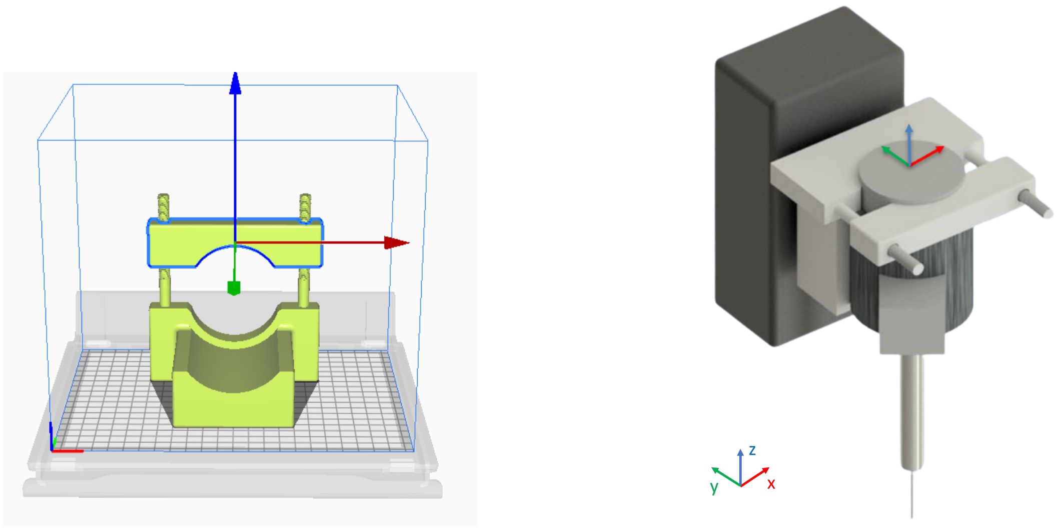

Figure 3.6: 3D model of the camera support in two pieces on the left-hand side and support (light grey), camera (black), end-effector (dark grey) assembly. The center of mass (COM) is computed with respect to the coordinates frame on the upper flange.

Once everything has been printed and assembled, some parameters necessary for the robot control (in particular for the gravity compensation) have been estimated:

- $Mass = 1,713\ kg$
- $Density = 0,842\ g/cm^3$
- $\boldsymbol{P}_{COM} = (-1.071,\ 46.481,\ -57.252)\ mm$

The coordinates of the load center of gravity ($\boldsymbol{P}_{COM}$) must be expressed with respect to the face of the mounting flange on axis A7 as described in the robot manual and illustrated in Figure 3.7.

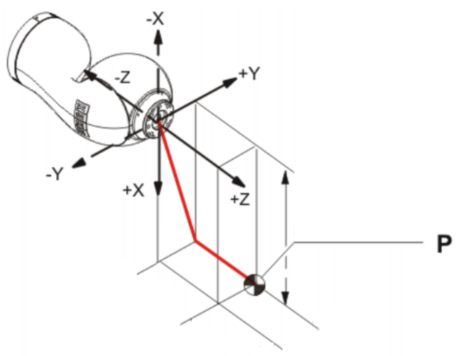

Figure 3.7: Face of the mounting flange on axis A7.

In order to retrieve the transformation $\boldsymbol{T}_C^{EE}$ that relates the end-effector (EE) frame and the camera frame, eye-in-hand calibration [23] was finally performed by the NearLab's researchers. The process is similar to the one carried out to estimate intrinsic and extrinsic camera parameters. Eye-in-hand calibration consisted in retrieving the position in the camera frame of a (13×9) stationary chessboard pattern shown to the camera while recording the position of the EE (computed through the forward kinematic of the manipulator). A number of 8 poses together with the relative image coordinates of the chessboard pattern have been recorded. Knowing that the position of the pattern with respect to the base frame remains unchanged during the motion of the robot, the computation of the position of the camera with respect to the EE is reduced to the solution of an $\boldsymbol{A}\boldsymbol{T}_{\boldsymbol{C}}^{\boldsymbol{EE}} = \boldsymbol{T}_{\boldsymbol{C}}^{\boldsymbol{EE}}\boldsymbol{B}$ equation, where $\boldsymbol{A}$ is the transformation between two different robot EE positions and $\boldsymbol{B}$ is the transformation between the camera positions corresponding to the robot EE positions considered. Using the OpenCV implemented function $calibrateHandEye()$, the camera-EE relation is obtained:

$$\boldsymbol{T}_C^{EE} = \begin{bmatrix} 1 & 0 & 0 & -0.00619 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.213 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

meaning no rotations, $-0.619\ cm$ and $21.3\ cm$ translation along X and Z axes respectively. All the other transformations between different reference frames will be discussed in the control Chapter 3.4.

3.2. Tool Detection Module

The tool detection module has the role of recognizing the tip of the surgical instrument in the images coming from the stereo camera and estimating its position. To achieve this, a CNN YoloV5 (You Only Look Once, version 5) [24] has been employed.
YoloV5 is an advanced object detection framework that employs a single-stage architecture for real-time object detection within images and videos. It is characterized by its streamlined and efficient design, emphasizing accuracy, speed, and ease of use. YoloV5 has been used in this work since the detection of the instrument must be performed in real-time to have a fast repositioning of the camera.

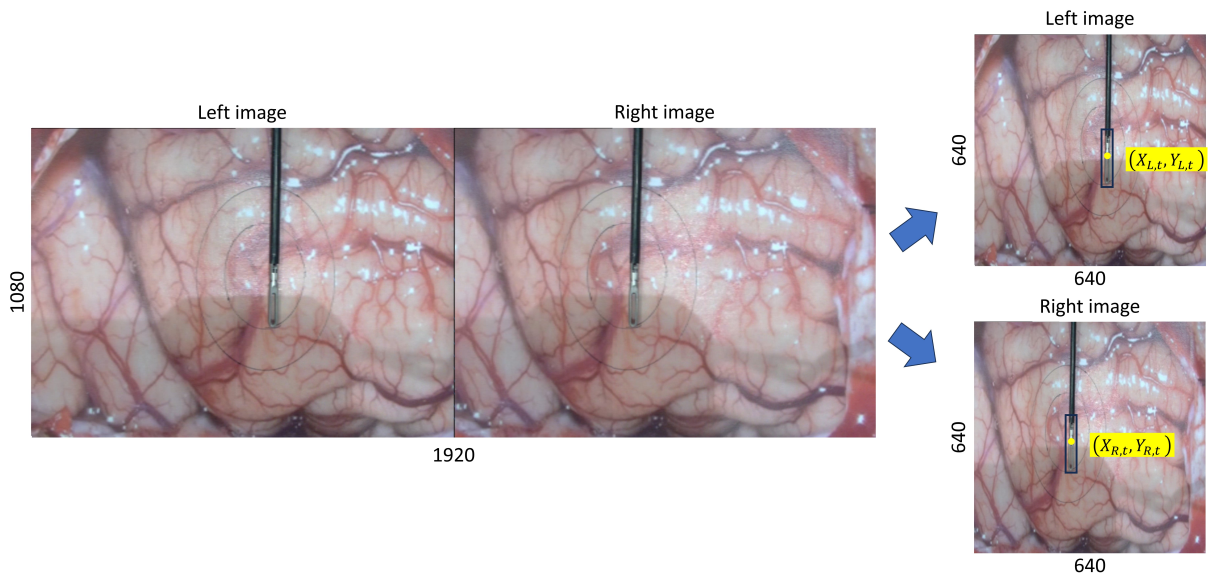

Figure 3.8: Left and right images before (left-hand side) and after (right-hand side) processing them and feeding the NN. $(X_{L,t}, Y_{L,t})$ and $(X_{R,t}, Y_{R,t})$ represent the center coordinates of the bounding box in the left image plane and in the right image plane respectively.

In this work's setup, the two images, right and left, acquired by the stereo camera come together with total a resolution of 1920×1080 pixels (see Figure 3.8). The image is then divided in two with size 960×1080 pixels and the two images obtained are downsampled to 640×640 pixels in order to speed up the inference while maintaining good accuracy. Finally, the downsampled images are sent to the NN.
The output of the NN $[x_1, y_1, x_2, y_2, conf, class_{pred}]$, for each image, is represented by the image coordinates of the upper right corner and lower left corner of the bounding box, the confidence of the prediction and the class predicted i.e. the instrument's tip. After

the identification of the instrument, the bounding box was extrapolated to the original images size and the actual position of the tooltip on the image plane was considered to be as the center of the bounding box, $(X_{R,t}, Y_{R,t})$ for the right image and $(X_{L,t}, Y_{L,t})$ for the left image.

A pre-trained model of the object detection NN on the Common Objects in Context (COCO)[25] data set is made available by the author. Even if this data set does not include images of medical instruments in surgical scenarios, it has been chosen to fine-tune the model since training the detector from scratch would require a very high amount of data and computational power. The fine-tuning was performed with a custom data set created in the NearLab laboratory. This data set is not representative of a clinical scenario, but it has been used to demonstrate this work's method conceptually. An example of this data set can be appreciated in Figure 3.9.

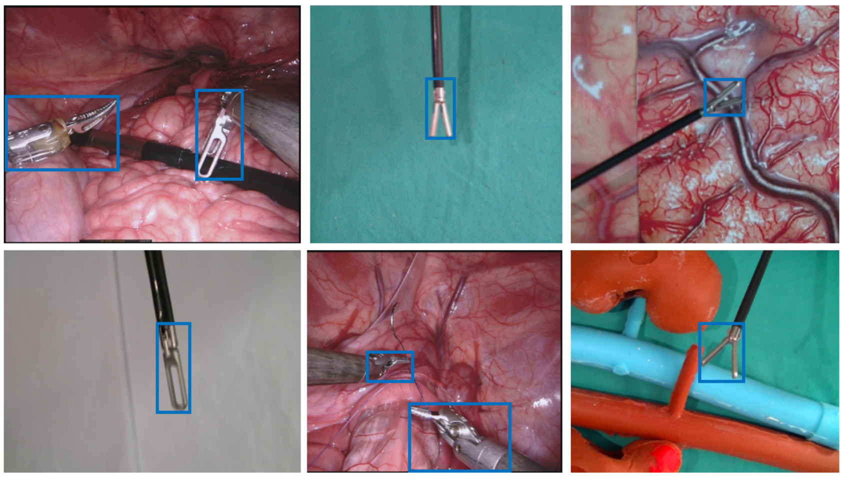

Figure 3.9: Training/validation data set: laboratory recorded images and real endoscopic ones from the EndoVis challenge.

A number of 4100 images of the laparoscopic grasping forceps used in this study were recorded in the laboratory setup and were manually annotated. Different backgrounds were used in order increase the variability and provide robustness in the training procedure. Also, given that the grasping forceps is very similar to the surgical instruments of the Da Vinci surgical robot by Intuitive, another 1800 images were extracted from the 2017 EndoVis challenge[26] for which the bounding boxes were extracted from the segmentations provided. The obtained final data set totals to a number of 5900 images and was split in 5300/600 for the training and validation step.

The training procedure was performed by the NearLab's researchers with a number of

5300 images while data augmentation was performed as rotations, translations, changes in brightness and left-right flips. A mini-batch size of 8 images was used. The hyper-parameter Intersection over Union (IoU) which represents how much the predicted bounding box overlaps with the ground truth was set to 0.5 meaning that a predicted bounding box that overlaps more than 50% with the ground truth is considered a True Positive (TP) while it is considered a False Positive (FP) otherwise. The learning rate was set to 0.001 and the network was trained for a maximum of 400 epochs. Finally, the confidence threshold was set to 0.4 representing the value above which a prediction is considered to be valid.

The CNN worked on every single frame sent by the camera and returned a position estimation at a frequency approximately of 12.5 Hz because of the large computational load of the NN.

3.3. Tracking Module

Since the tool detection module is too slow, because of the delays deriving by the use of the CNN, new approaches have been investigated to face this problem, giving rise to the so-called "Hybrid tracking module".
First of all, different approaches for the tool detection have been investigated to reduce the computational load coming from the CNN (**CNN15** and **CNN30** strategies). After that, a modified Particle Filter has been introduced to predict the future position of the surgical instrument and to consequently reduce the tool detection delay (**PARTICLE** strategy). Then, an efficient tracking module has been considered to substitute, whenever is possible, the CNN (**OPFL** strategy). Finally, a hybrid strategy that involves both the tracking module and the particle filter predictor has been used to reduce as much as possible all the system's delays (**HYBRID** strategy).
The first two strategies were developed by the NearLab laboratory, while the other three strategies are the main contribution of this thesis work. All these methods will be discussed in detail in the following Subsections.

3.3.1. CNN-Based Strategies

To reduce the computational load given by the use of the CNN, the NearLab's researchers changed the activation protocol of the CNN. Firstly, they developed a strategy that uses the CNN only on the right image and exploits the epipolar geometry concept (briefly explained in Appendix A) to determine the position of the instrument in the left image. In Figure 3.10 is shown a simplified version of the workflow of the system that employs the aforementioned strategy.

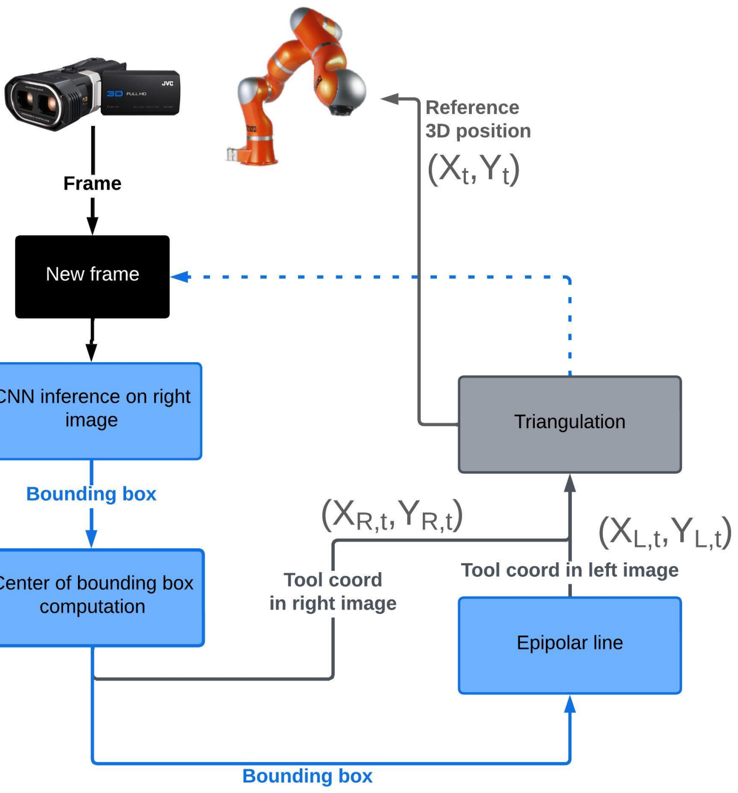

Figure 3.10: The CNN15 strategy (simplified) workflow.

This strategy allows to reach an estimated position publication rate equal to 15 Hz, so it has been named "CNN15".

To further reduce the computation load of the method just described, researchers found a compromise strategy between surgical instrument detection accuracy and speed. They discovered that by making inferences every 2 frames, the system achieves a processing speed close to 30 Hz, which is also the frame rate of the camera. Thanks to this functioning frequency, the strategy that exploits this compromise is called "CNN30".

3.3.2. Particle Filter Strategy

A Particle Filter, also known as a Sequential Monte Carlo method, is a powerful and versatile estimation technique used in probabilistic filtering and state estimation problems [27]. It is particularly effective in scenarios where the underlying system dynamics are nonlinear, non-Gaussian, and may involve significant uncertainties. Particle Filters provide a flexible framework for tracking and estimating the evolving state of a system over time by combining principles from probability theory and Monte Carlo simulations. In a Particle Filter, the state of a dynamic system is represented by a set of particles, where each particle corresponds to a potential state hypothesis. These particles are propagated through time using the system's nonlinear dynamics, and at each time step, they are updated based on new measurements or observations. Weights are assigned to the particles according to their likelihood of explaining the observed data, and resampling is performed to generate a new set of particles that better represent the true state of the system [28].

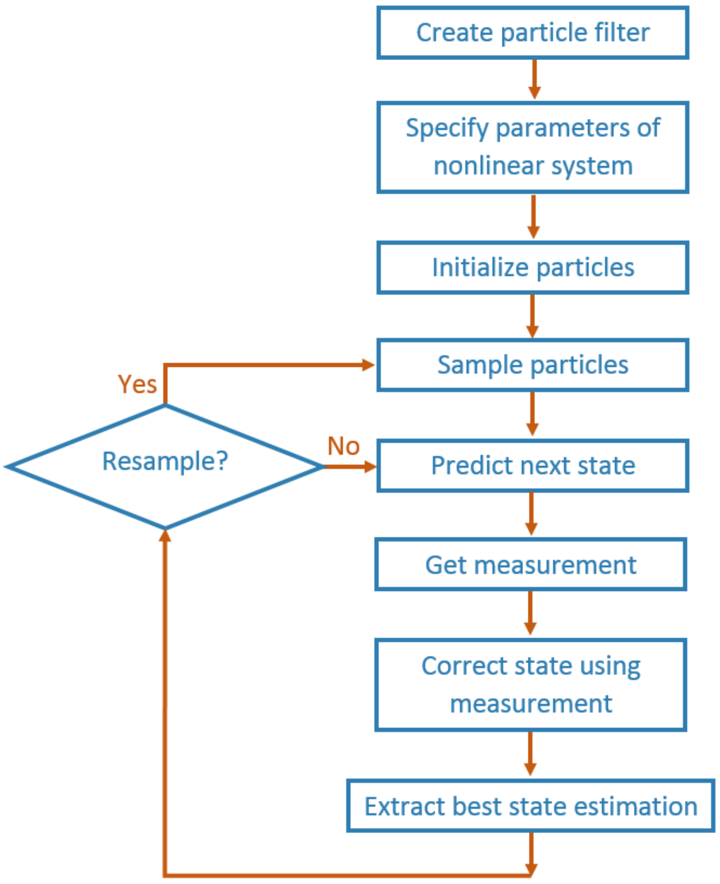

Figure 3.11: The usual Particle filter workflow (MathWorks [29]).

The usual particle filter algorithm (Figure 3.11), after the initialization phase and the first sampling, acts as follows at each time instant:

1. **Predict next state**: based on a specified state transition function, particles evolve to estimate the next state.
2. **Get measurements**: collect the measurements from the sensors.
3. **Correct state using measurements**: measurements are used to adjust the predicted state and correct the estimate.
4. **Extract best state estimation**: the best state estimate is automatically extracted based on the weights of each particle.
5. **Decide if is necessary to resample or not**: once the state has changed enough, resample the particles based on the newest estimate. If resampling is not triggered, the same particles are used for the next estimation. If resampling is activated, particles are resampled using different methods and policies [30].

This method is used in very different fields (robotics, economics, statistics, etc.) and in this work has been used just as a predictor of a system difficult to model. In fact, the purpose is to predict future movements of a surgical tool without any sensor placed on the instrument or on the surgeon hand, and without any clue about the task the tool is used for in that moment.
Particle filter has been preferred to the more common Kalman filter because the latter is based on the model Gaussianity assumption and since our model cannot be considered Gaussian, the Particle filter, which employs simulation methods to generate state and innovation estimates, was more suitable.
Considering this, a modified Particle filter has been developed and its workflow is described below.
Firstly, the weight of every single particle is initialized with the following formula:

$$w_{R,i} = \frac{1}{N} = w_{L,i} \tag{3.10}$$

where $w_{R,i}$ and $w_{L,i}$ indicate the weight of the i^{th} particle related to the right and the left image respectively, and N the total number of particles used.
Every time a new tool position estimation, $(X_{R,t}, Y_{R,t})(k)$ from the CNN or $(X_{L,t}, Y_{L,t})(k)$ from the epipolar geometry, is available, the Particle filter is used to predict the future position of the tool and it works as follows (considering for simplicity only the tool position in the right image $(X_{R,t}, Y_{R,t})(k)$ as input, since for the left image all happens in exactly the same way):

1. **Speed and direction estimation**: the speed and the direction of the instrument movement are computed. To estimate the velocity, the discrete derivative between two consecutive positions is used:

$$speed_{R,t}(k) = \frac{\sqrt{\Delta^2_{X,R}(k) + \Delta^2_{Y,R}(k)}}{\Delta_t(k)} \tag{3.11}$$

where $speed_{R,t}(k)$ is the speed of the tool at the k^{th} instant, and $\Delta_{X,R}$, $\Delta_{Y,R}$, Δ_t are the variation of the tool x coordinate, tool y coordinate and time t, respectively, between two consecutive position received by the tool detection module at instant k and $k-1$. These variations are defined as follows:

$$\Delta_{X,R}(k) = X_{R,t}(k) - X_{R,t}(k-1)$$

$$\Delta_{Y,R}(k) = Y_{R,t}(k) - Y_{R,t}(k-1)$$

$$\Delta_t(k) = t(k) - t(k-1)$$

in which $X_{R,t}(k)$ and $Y_{R,t}(k)$ are the tool coordinates at the k^{th} instant in the right image, while $X_{R,t}(k-1)$ and $Y_{R,t}(k-1)$ are the same coordinates but at the instant $k-1$. $t(k)$ and $t(k-1)$ represent the time at which the k^{th} and the $(k-1)^{th}$ positions have been received from the NN.
Since the speed estimated through discrete derivative results very noisy, it has to be filtered through a low-pass Finite Impulse Response filter of the first order:

$$v_{R,t}(k) = speed_{R,t}(k)(1-\alpha) + v_{R,t}(k-1) \cdot \alpha \tag{3.12}$$

where $v_{R,t}(k)$ is the filtered tool velocity and $\alpha = 0.9$ is a parameter that has been fine-tuned experimentally.
The function $atan2$ is used instead to determine the direction of motion:

$$h_{R,t}(k) = atan2(\Delta_{Y,R}(k), \Delta_{X,R}(k)) \tag{3.13}$$

2. **Particles' heading randomization**: once the direction of motion of the tool $h_{R,t}$ is estimated the heading of each particle is distributed normally around that direction:

$$\boldsymbol{h}_{particles,R}(k) \approx \ Norm(h_{R,t}(k), 0.1) \tag{3.14}$$

where $\boldsymbol{h}_{particles,R}(k)$ is the vector of all the $h_{R,i}(k)$ headings of the particles.

3. **Prediction of future particles' position**: as discussed above, the modelling of the surgical tool moving in the space was complicated. To overcome this problem the concept of "odometry" has been used. Odometry is a position estimation technique widely used in mobile robots' applications and it consists in the estimation of the variations in a robot's pose (translation and orientation) over time using the measurements coming from different sensors [31].
In this work the only sensor available to detect the tool position and movements is represented by the camera fixed to the robot. The velocity ($v_{R,t}$) and direction ($h_{R,t}$) of movements estimated above, a predefined prediction horizon dt, and odometry have been used to predict the future particles' position ($X_{R,i}, Y_{R,i}$).
In particular the Runge-Kutta odometry is exploited when the absolute value of the heading variation $|\Delta_{h,R}(k)| = |h_{R,t}(k) - h_{R,t}(k-1)|$ is under a certain threshold:
$$X_{R,i}(k) = X_{R,t}(k) + v_{R,t}(k) \cdot dt \cdot \cos\left(h_{R,i}(k) + \frac{\Delta_{h,R}(k) \cdot dt}{2}\right) \tag{3.15}$$
$$Y_{R,i}(k) = Y_{R,t}(k) + v_{R,t}(k) \cdot dt \cdot \sin\left(h_{R,i}(k) + \frac{\Delta_{h,R}(k) \cdot dt}{2}\right) \tag{3.16}$$
When $|\Delta_{h,R}(k)|$ is over the threshold, the exact odometry is used:
$$X_{R,i}(k) = x_{R,t}(k) + \frac{v_{R,t}(k) \cdot (\sin(h_{R,i}(k) + \Delta_{h,R}(k) \cdot dt) - \sin(h_{R,i}(k)))}{\Delta_{h,R}(k)} \tag{3.17}$$
$$Y_{R,i}(k) = y_{R,t}(k) - \frac{vR,t(k) \cdot (\cos(h_{R,i}(k) + \Delta_{h,R}(k) \cdot dt) - \cos(h_{R,i}(k)))}{\Delta_{h,R}(k)} \tag{3.18}$$
The prediction horizon dt has been fine-tuned on the real system.

4. **Prediction of future position of the tool**: the future position of the tool $(\hat{X}_{R,t}, \hat{Y}_{R,t})(k)$ is estimated taking a weighted average of the particles' position computed above:
$$\hat{X}_{R,t}(k) = \frac{\sum_{i=1}^{N} w_{R,i}(k-1) \cdot X_{R,i}(k)}{\sum_{i=1}^{N} w_{R,i}(k-1)} \tag{3.19}$$
$$\hat{Y}_{R,t}(k) = \frac{\sum_{i=1}^{N} w_{R,i}(k-1) \cdot Y_{R,i}(k)}{\sum_{i=1}^{N} w_{R,i}(k-1)} \tag{3.20}$$

5. **Particles' weight update**: the weight of the particles is updated taking into account the Euclidean distance between the predicted particle position and both the actual tool position and a potential future position of the tool. This additional term was intended to assign more weight to the direction of the motion being tracked.

From the equation point of view:

$$w_{R,i}(k) = \frac{\max\left(\boldsymbol{distance}_R\right)(k) - \ distance_{R,i}(k)}{\sum_{i=1}^{N}\left(\max\left(\boldsymbol{distance}_R\right)(k) - \ distance_{R,i}(k)\right)} \tag{3.21}$$

where $\boldsymbol{distance}_R(k)$ is the vector of all the $distance_{R,i}(k)$ that derive from the equation 3.22 in which all the value are referred to the k^{th} instant, $\Delta X_{R,t,i} = X_{R,i} - X_{R,t}$, and $\Delta Y_{R,t,i} = Y_{R,i} - Y_{R,t}$:

$$distance_{R,i} = \sqrt{\Delta X^2_{R,t,i} + \Delta Y^2_{R,t,i}} + \sqrt{\Delta X^2_{future} + \Delta Y^2_{future}} \tag{3.22}$$

where $\Delta X_{future} = \Delta X_{R,t,i} - v_{R,t} \cdot dt \cdot \cos(h_{R,t})$ and $\Delta Y_{future} = \Delta Y_{R,t,i} - v_{R,t} \cdot dt \cdot \sin(h_{R,t})$ represent the difference between the current position and the potential future position.

The algorithm of the modified Particle filter above described can be appreciated also in Figure 3.12

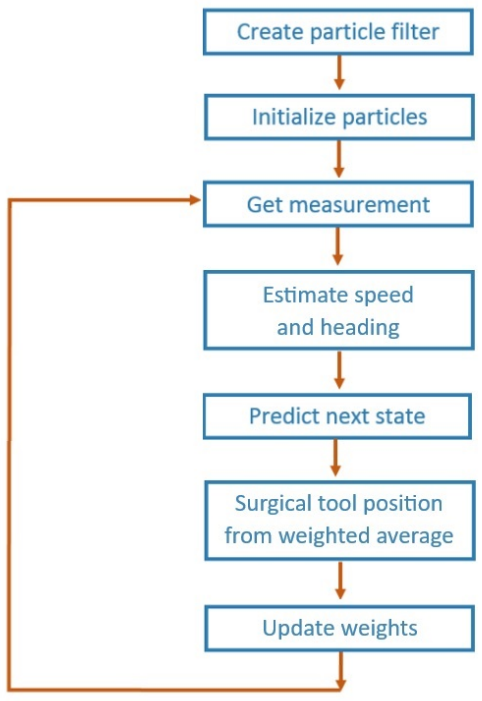

Figure 3.12: The modified Particle filter workflow presented in this work.

The modified Particle filter described above has been integrated in the CNN30-based system as shown in Figure 3.13.

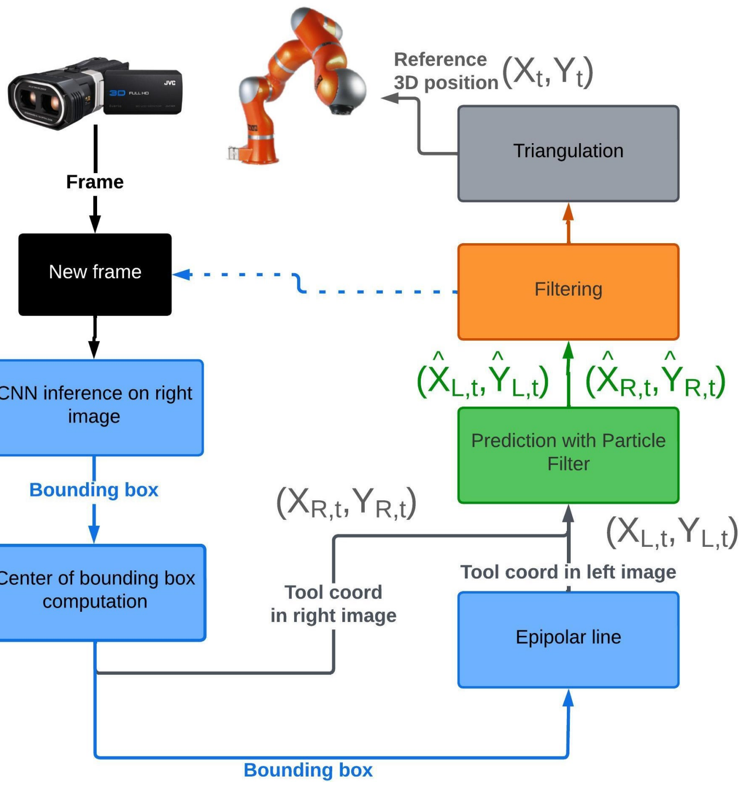
Reference 3D position (X_t, Y_t)
Triangulation
Frame
New frame
Filtering
$(\hat{X}_{L,t}, \hat{Y}_{L,t})$
$(\hat{X}_{R,t}, \hat{Y}_{R,t})$
CNN inference on right image
Prediction with Particle Filter
Bounding box
$(X_{R,t}, Y_{R,t})$
$(X_{L,t}, Y_{L,t})$
Tool coord in left image
Center of bounding box computation
Tool coord in right image
Epipolar line
Bounding box

Figure 3.13: The Particle filter strategy (simplified) workflow.

The estimated future tool coordinates obtained by this technique are a bit noisy, so they have to been filtered. To do so, a low-pass Finite Impulse Response filter of the first order is employed:

$$\tilde{X}_{R,t}(k) = \hat{X}_{R,t}(k)(1-\beta) + \tilde{X}_{R,t}(k-1) \cdot \beta \tag{3.23}$$

$$\tilde{Y}_{R,t}(k) = \hat{Y}_{R,t}(k)(1-\beta) + \tilde{Y}_{R,t}(k-1) \cdot \beta \tag{3.24}$$

where $\tilde{X}_{R,t}(k)$ and $\tilde{Y}_{R,t}(k)$ are the filtered coordinates of the tool in the right image at the k^{th} instant and $\beta = 0.7$ is a parameter that has been fine-tuned experimentally.
Once the tool future coordinates in the two images, $(\tilde{X}_{R,t}, \tilde{Y}_{R,t})$ and $(\tilde{X}_{L,t}, \tilde{Y}_{L,t})$, have been estimated these are used to triangulate the tool 3D position $\boldsymbol{P}_t^C = (X_t, Y_t, Z_t)$ in the camera frame and then, through the necessary transformations, in the base frame, (as described in the Chapter 3.4.1).

3.3.3. Optical Flow Strategy

This strategy exploits the computational efficiency of the objects tracking modules to follow one or more points in the image between two consecutive frames. [14]
In fact, usually tracking algorithms are faster than detection algorithms because when these methods are tracking an object that was detected in the previous frame, they know a lot about the appearance of the object. These algorithms also know the location in the previous frame and the direction and speed of its motion. So, in the next frame, tracking modules can use all this information to predict the location of the object in the next frame and do a small search around the expected location of the object to accurately locate the object. A good tracking algorithm will use all information it has about the object up to that point while a detection algorithm always starts from scratch.
There are many different techniques to track an object in the images and in this work, to track the surgical tool, the Optical flow-based method has been used. This choice has been made because Optical flow-based methods offer several advantages in the context of video object tracking, including:

- **Motion Estimation**: Optical flow inherently captures the motion information between consecutive frames. This enables trackers to accurately estimate the displacement of objects even in challenging scenarios with complex motion patterns.

- **No Need for Object Models**: Unlike appearance-based methods that rely on object models and templates, Optical flow methods do not require prior knowledge of the object's appearance. This makes them robust in situations where the object's appearance changes due to illumination variations or occlusions.

- **Low-Level Features**: Optical flow operates on pixel-level information, which means it can capture fine-grained motion details. This can be advantageous when tracking objects with subtle motion cues or when dealing with partial occlusions.

- **Real-time Processing**: Optical flow algorithms can often be implemented efficiently, making them suitable for real-time applications where timely tracking updates are crucial.

- **Robustness to illumination and object appearance changes**: Optical flow methods are less affected by lighting changes compared to appearance-based methods. This is because they focus on tracking the motion of pixels rather than relying heavily on color or texture information. In addition, since Optical flow is primarily concerned with motion estimation, it can track objects with similar motion patterns even when their appearances differ significantly.

- **Simple Assumptions**: Optical flow methods are based on the assumption of brightness constancy, which simplifies the tracking process. This makes them applicable to a wide range of scenarios without overly complex model requirements.

The Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of an object or camera [32]. Optical flow is based on the assumption that neighboring pixels in successive frames of a video sequence move together with similar velocities, preserving their spatial relationships. In other words, if an object in an image is moving, the pixels belonging to that object should exhibit consistent movement across frames.
To compute Optical flow, algorithms analyze the changes in pixel intensity values between consecutive frames. The displacement of pixels or image patches is estimated based on how their intensity values shift.
Various methods can be used for this purpose and in this thesis the so-called "Differential method" of Lukas-Kanade technique has been exploited [33]. This method utilizes the brightness constancy assumption, stating that the intensity of a pixel remains unchanged between frames.
In particular, the Lucas-Kanade method assumes small motion between frames and minimizes the difference between the intensity values of corresponding pixels. The basic formulation involves the partial derivatives of image intensity with respect to time and spatial coordinates.
Considering the intensity of a pixel $\boldsymbol{I}(x, y, t)$ in one frame and the same pixel in the following frame, after Δt seconds, because of the assumption that intensity does not change, the following equation is obtained:

$$\boldsymbol{I}(x, y, t) = \boldsymbol{I}(x + dx, y + dy, t + \Delta t) \tag{3.25}$$

where I is the intensity of a pixel, x and y are the coordinates of the pixel in the image at time t, while dx and dy represent how much the coordinates of the pixel have changed in Δt seconds. Using the Taylor series approximation on the right-hand side, removing common terms, and dividing by Δt, the Optical flow equation can be obtained:

$$f_x \cdot u + f_y \cdot v + f_t = 0 \tag{3.26}$$

In this equation $f_x = \dfrac{\delta f}{\delta x}$ and $f_y = \dfrac{\delta f}{\delta y}$ are image gradients. Similarly, f_t is the gradient along time. $u = \dfrac{\delta x}{\delta t}$ and $v = \dfrac{\delta y}{\delta t}$ are unknown and are computed using the Lukas-Kanade method.

Because of the assumption that all the neighboring pixels will have similar motion, Lucas-Kanade method takes a 3×3 patch around the points to follow, so all the 9 points have the same motion. The gradients f_x,f_y,f_t can be found for these 9 points. So now the problem becomes solving 9 equations with two unknown variables which is over-determined. The solution is obtained with the least square fit method:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \tag{3.27}$$

While Optical flow-based methods offer many advantages, it is important to note that they also have limitations. They can struggle in cases of occlusions, abrupt motion changes, or scenarios where the brightness constancy assumption is violated. Furthermore, handling large displacements and scale changes might pose challenges.
It is also common for tracking algorithms to accumulate errors and the bounding box tracking the object slowly drifts away from the object it is tracking. To fix these problems, a detection algorithm is run every so often. Therefore, while designing an efficient system usually object detection is run on every n^{th} frames while the tracking algorithm is employed in the $n-1$ frames in between.

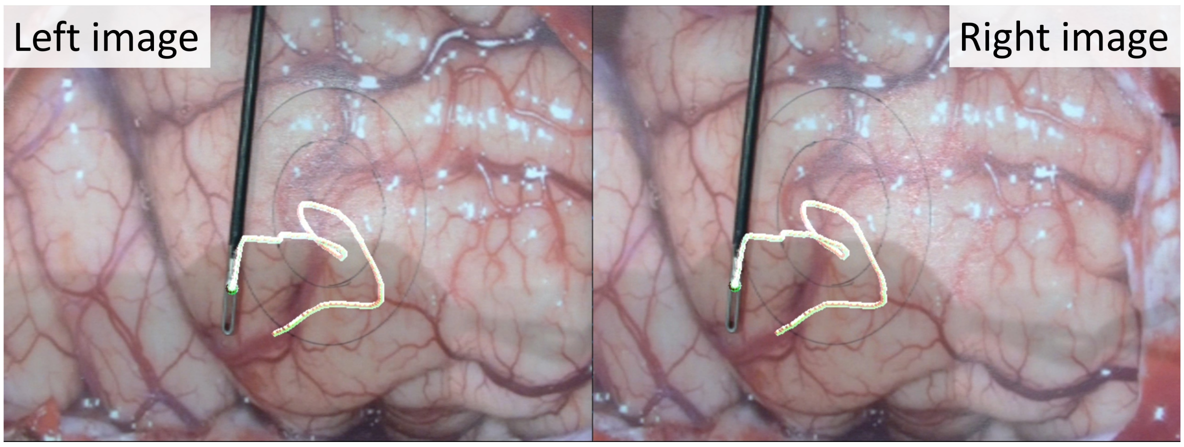

Figure 3.14: The Optical flow tracking applied to the surgical scenario: the white tracks are the paths followed by the tool from the last tool detection via CNN to the frame in the image. The paths are composed of multiple positions tracked by the Optical Flow tracking algorithm.

This work proposes an Optical flow strategy that is based on the Lukas-Kanade method with pyramids offered by OpenCV [34] and that works as explained above but includes some extra "safety measures".
In particular, pyramids are used to solve the "large displacements" problem. In fact, they are employed to accommodate not only small movements of the tracked points but also larger ones since they allow for image scaling, enabling the system to perceive significant movements as if they were smaller, contributing to improved tracking performance.
An example of the tracking of the optical flow applied to this work's scenario can be appreciated in Figure 3.14.

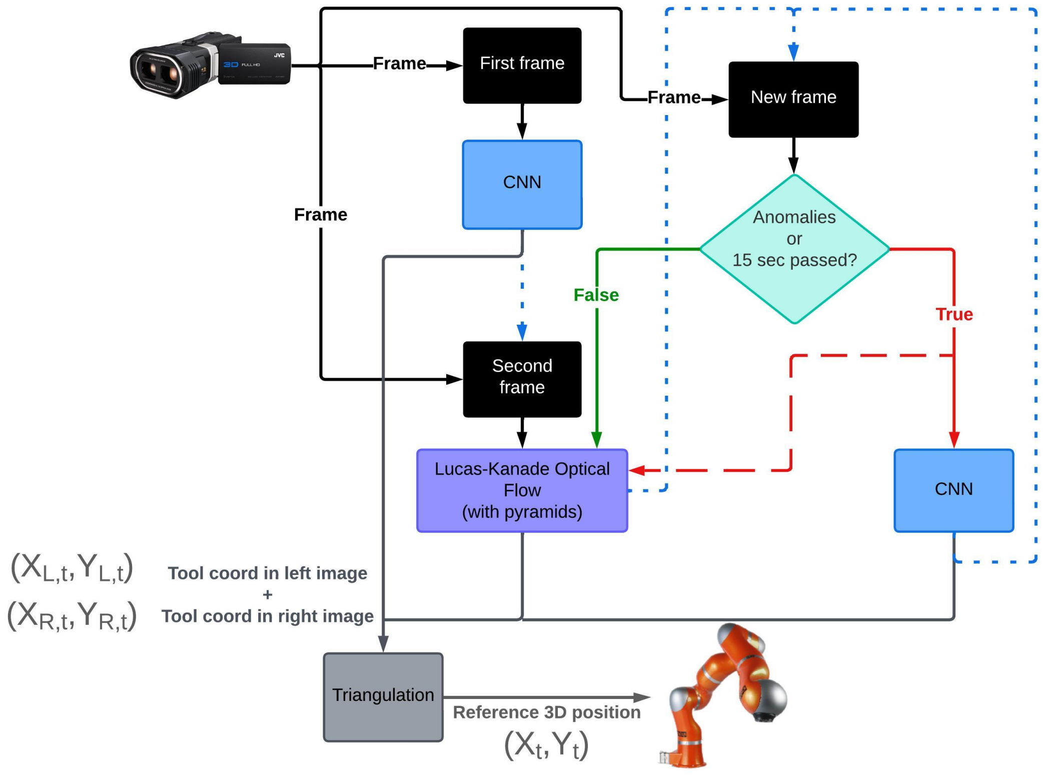

Figure 3.15: The Optical flow strategy (simplified) workflow.

The workflow of the proposed strategy is illustrated in Figure 3.15 and described in detail below. In this strategy, the same approach is applied to the left and right images coming from the camera (epipolar line is not used), on the contrary to what has been seen in the first three strategies. Knowing this, during the explanation of the OPFL approach, only what happens on the right image will be considered, since on the left image the algorithm works in the exact same way:

- **On the first frame** that comes from the camera an object detection is made using the CNN. The result of the tool detection is represented by the tool coordinates in the image space $(X_{R,t}, Y_{R,t})$ which are used for two purposes: these are the co-ordinates used to determine the instrument position $\boldsymbol{P}_t^C = (X_t, Y_t, Z_t)$ in 3D and they are the starting point for the Optical flow functioning. In fact, this point in the image space and other 8 surrounding points to be followed are passed to the tracking algorithm (Figure 3.16). 9 points are thus used to track the tool on the following frame. The choice of using 9 points has been made to be more robust with respect to partial occlusions or small changes in the pixel intensities.
  The first frame is then converted to gray scale and saved since it is needed for the Optical flow algorithm.

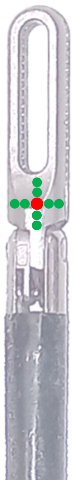
Figure 3.16: The 9 points tracked by the Optical flow (in red the point found by the CNN, in green the surrounding points).

- **On the second frame** the Optical flow tracking is used for the first time. It requires the current frame (the second one) converted to gray scale, the previous gray frame, the vector of points to follow, and some parameters (e.g. the number of pyramids to use) that have been fine-tuned experimentally to grant a good tracking in different scenarios of tool speed and image background.
  The Optical flow function offered by OpenCV returns a vector with the points it has been able to track between the two consecutive frames. Taking the mean value of the x coordinates and of the y coordinates between the aforementioned points, the tool coordinates in the image space $(X_{R,t}, Y_{R,t})$ are obtained.
  Once the tool coordinates are retrieved on both images, the algorithm presented in this thesis work checks for inconsistencies in the estimated positions and if it finds

them, it generates a different type of errors. The anomalies detected and the error returned are presented in Table 3.2 (here, for completeness, the errors on the right and left images are reported).

Inconsistency	**Error on**
Zero points (over 9) tracked by the Optical flow function on the Right/Left image	Right/Left image
Tool detected in an area too close to borders of Right/Left image	Right/Left image
The distance between the previous and the current positions in the Right/Left image is too large	Right/Left image
The position in the Right/Left image has not changed in the last 60 frames	Right/Left image
Inconsistency between Right and Left positions: error triggered if the distances between the x coordinates ($X_{R,t}$ and $X_{L,t}$) or between the y coordinates ($Y_{R,t}$ and $Y_{L,t}$) becomes too large	Both images

Table 3.2: Inconsistencies detected and relative errors generated: the first four inconsistencies generate error only on the specific image in which they happened, while the last inconsistency causes an error on both the images at the same time.

Depending on the presence of one or more errors and on which kind of errors have been activated, the algorithm will use or not the position estimated on a specific image to triangulate the tool in 3D space $\boldsymbol{P}_t^C = (X_t, Y_t, Z_t)$ and it decides if activate the tool detection through CNN on the following frame. In particular, (considering just the Right image case), if the "Error on Right image" is not present, the position estimated on this image $(X_{R,t}, Y_{R,t})$ is used for the triangulation and it will be saved together with 8 surrounding points for the functioning of the Optical flow on the following frame (including also the current frame in gray scale). If the "Error on Right image" has been generated, the estimated point $(X_{R,t}, Y_{R,t})$ is discarded, the previous tool position (estimated using the previous frame) is used to triangulate the tool position and on the following frame, the CNN tool detection is employed.
If an error is present on both images (like in the case of "Inconsistency between Right and Left position"), for safety reasons, no position will be sent to the robot as desired position.
Table 3.3 resumes all the possible cases and the algorithm behavior at the k^{th} instant.

Error case	**Positions used at instant k for triangulation**		**Method used at instant $k+1$ for position estimation**	
	Right	**Left**	**Right**	**Left**
None	$(X_{R,t}, Y_{R,t})(k)$	$(X_{L,t}, Y_{L,t})(k)$	OPFL	OPFL
Only Right	$(X_{R,t}, Y_{R,t})(k-1)$	$(X_{L,t}, Y_{L,t})(k)$	CNN	OPFL
Only Left	$(X_{R,t}, Y_{R,t})(k)$	$(X_{L,t}, Y_{L,t})(k-1)$	OPFL	CNN
Both	No command sent to the robot		CNN	CNN

Table 3.3: Table of error cases and relative algorithm behavior.

- **Every time a new frame comes** the algorithm checks if in the previous iteration (on the previous frame) of the Optical flow one or more errors occurred and if the number of the frame arrived is a multiple of 450. If at least one of these two conditions holds, the tool detection is triggered. As seen before, if one error occurs on one image, the CNN will be used on that image, and in the case of a frame multiple of 450, object detection is performed on both images. This periodic use of the CNN is done approximately every 15 seconds and it has the aim to guarantee that the Optical flow continues to follow always the same area of the tool.
  Once determined which strategy to use to estimate the tool position in the current frame, all works as described for the first frame in the case of CNN or for the second frame in the case of Optical flow.

The sporadic use of the CNN on both the images that happens in this strategy does not affect the system's performance. In fact, the OPFL strategy allows to reach a frequency of publication of the position estimation equal to 30 Hz. In this case, moreover, the bottleneck of the publication frequency is not represented by the computer vision strategy but by the hardware of the experimental setup. Indeed, during some tests, it has been proven that this strategy can reach a position estimation frequency close to 200 Hz, but since the frame rate of the 3D camera is equal to 30 Hz, so is the position publication frequency.

3.3.4. Hybrid Strategy

The last strategy developed is called "Hybrid" because it employs both the Optical flow and the modified Particle filter. The aim of this strategy is to exploit the advantages deriving from the two methods: the tool's coordinates are estimated efficiently through the tracking module and a prediction of the future position is made to face all the unavoidable delays.
The flowchart of this strategy is showed in a simplified way in Figure 3.17.

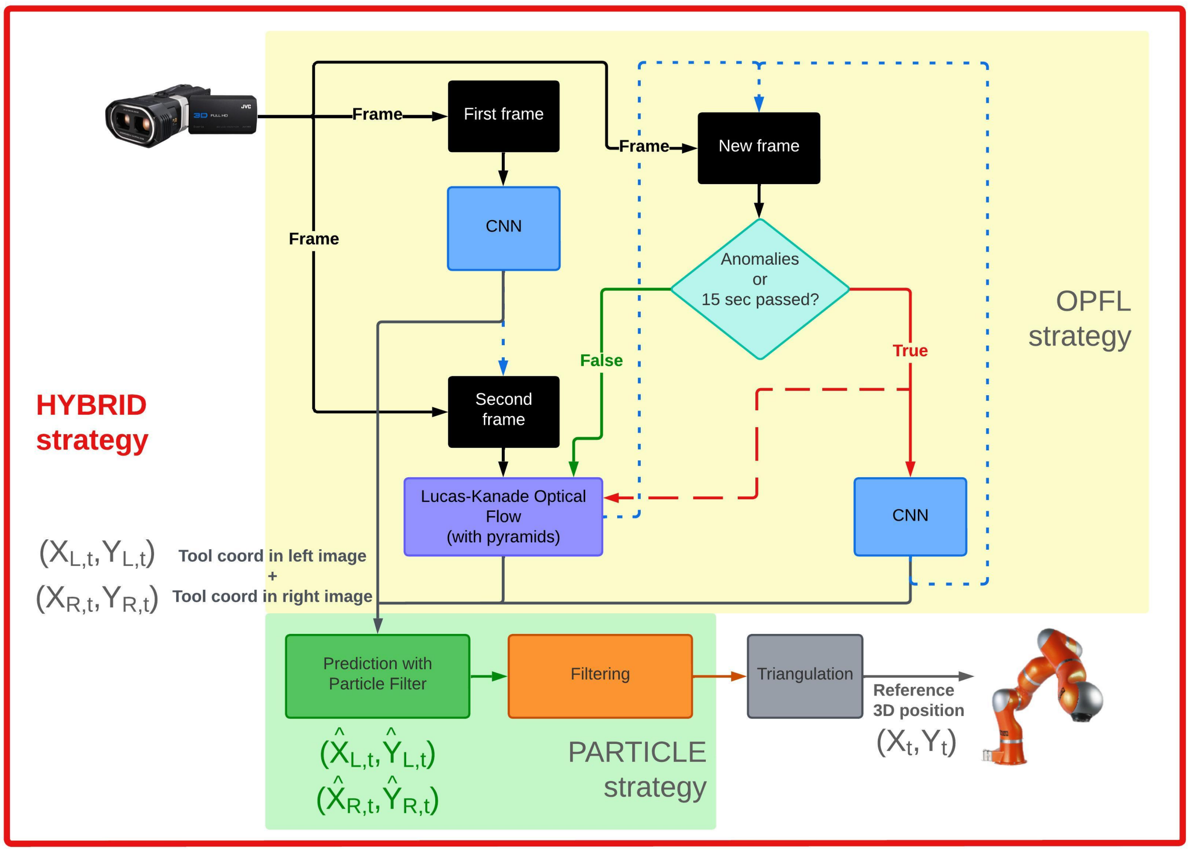
Frame
First frame
Frame
New frame
CNN
Frame
Anomalies or 15 sec passed?
OPFL strategy
False
True
HYBRID strategy
Second frame
Lucas-Kanade Optical Flow (with pyramids)
CNN
$(X_{L,t}, Y_{L,t})$ Tool coord in left image
+
$(X_{R,t}, Y_{R,t})$ Tool coord in right image
Prediction with Particle Filter
Filtering
Triangulation
Reference 3D position
(X_t, Y_t)
$(\hat{X}_{L,t}, \hat{Y}_{L,t})$
$(\hat{X}_{R,t}, \hat{Y}_{R,t})$
PARTICLE strategy

Figure 3.17: The Hybrid strategy (simplified) workflow.

The main structure of the algorithm is the same as the Optical flow. There is a tool detection via CNN on the first frame, every 450 frames, and each time is needed because of an inconsistency in the position estimation or tracking that fails. Then, in all the other cases, the Optical flow tracks the tool between two consecutive frames. As before, the estimated position and the frame in gray scale are saved for the next functioning of the tracking module. As explained in the previous Section, since most of the Hybrid strategy is composed of the OPFL strategy, also in this case, the same approach is applied to the left and right images coming from the camera. What changes in the Hybrid strategy with

respect to the Optical flow approach is the presence of the modified Particle filter after the estimation of the tool position.
Once the positions $(X_{R,t}, Y_{R,t})$ and $(X_{L,t}, Y_{L,t})$ have been retrieved through the tool detection or via the tracking module, they are passed to the Particle filter algorithm to predict the future tool positions in the two images $(\hat{X}_{R,t}, \hat{Y}_{R,t})$ and $(\hat{X}_{L,t}, \hat{Y}_{L,t})$.
Finally, the above-mentioned future tool positions, after being filtered as described in Section (3.3.2), are used to triangulate the tool in the 3D space. The obtained position $\boldsymbol{P}_t^C = (X_t, Y_t, Z_t)$ is therefore taken as a reference position for the visual servoing (after the transformation from the camera reference frame to the base frame, as described in the Chapter 3.4.1).
Also in this case, the frequency of publication of the tool positions can reach up to 30 Hz and the bottleneck is represented by the hardware of the experimental setup.

A table summarizing the main characteristics of the different strategies used in the tool detection and in the hybrid tracking modules is reported.

	CNN15	**CNN30**	**PARTICLE**	**OPFL**	**HYBRID**
Frames used	100%	50%	50%	100%	100%
Functioning frequency	15 Hz	$\approx$ 30 Hz	$\approx$ 30 Hz	30 Hz	30 Hz
Tool position estimation mainly through (right image)	CNN	CNN	CNN	OPFL	OPFL
Tool position estimation mainly through (left image)	Epipolar line	Epipolar line	Epipolar line	OPFL	OPFL
Particle filter prediction	No	No	Yes	No	Yes
Position low-pass filtering	No	No	Yes	No	Yes

Table 3.4: Tool position estimation strategies compared.

3.4. The Control Module

The role of this module is to actually make the camera move autonomously. To achieve this, the camera holder (the robotic manipulator) has to be moved by a position-based visual servoing controller in a fast and accurate way.
Before controlling the robot, the position of the tool, $\boldsymbol{P}_{R,t} = (X_{R,t}, Y_{R,t})$ and $\boldsymbol{P}_{L,t} = (X_{L,t}, Y_{L,t})$ obtained from the tool detection module or the hybrid tracking module, is triangulated to get a point in the 3D space. After the triangulation, the 3D position is transformed from the camera reference frame to the reference frame of the base of the robot and finally, it is used as a reference position for the visual servoing controller.
Since surgical exoscope movements can be divided into translational and rotational, two strategies have been employed to move the robot's end-effector in the Cartesian space:

- Position control: the camera moves in the space keeping the orientation fixed and a constant vertical distance from the tool.
- Orientation control: the camera rotates around a predetermined fixed point exploiting the concept of the "Remote Center of Motion".

The chosen regulator is based on the so-called "Resolved velocity controller" strategy.

3.4.1. Tool Position Triangulation and Transformation

Once the positions of the surgical instrument have been identified in the two images, $\boldsymbol{P}_{R,t} = (X_{R,t}, Y_{R,t})$ and $\boldsymbol{P}_{L,t} = (X_{L,t}, Y_{L,t})$, the next crucial step for the visual servoing problem is the 3D reconstruction of the position of the tool with respect to one of the two cameras. Given that during the calibration procedure the extrinsic parameters of the left camera were computed with respect to the right image, the right one was used as a reference also in this situation.
The reconstruction of the position was done by triangulation as shown in Figure 3.18.

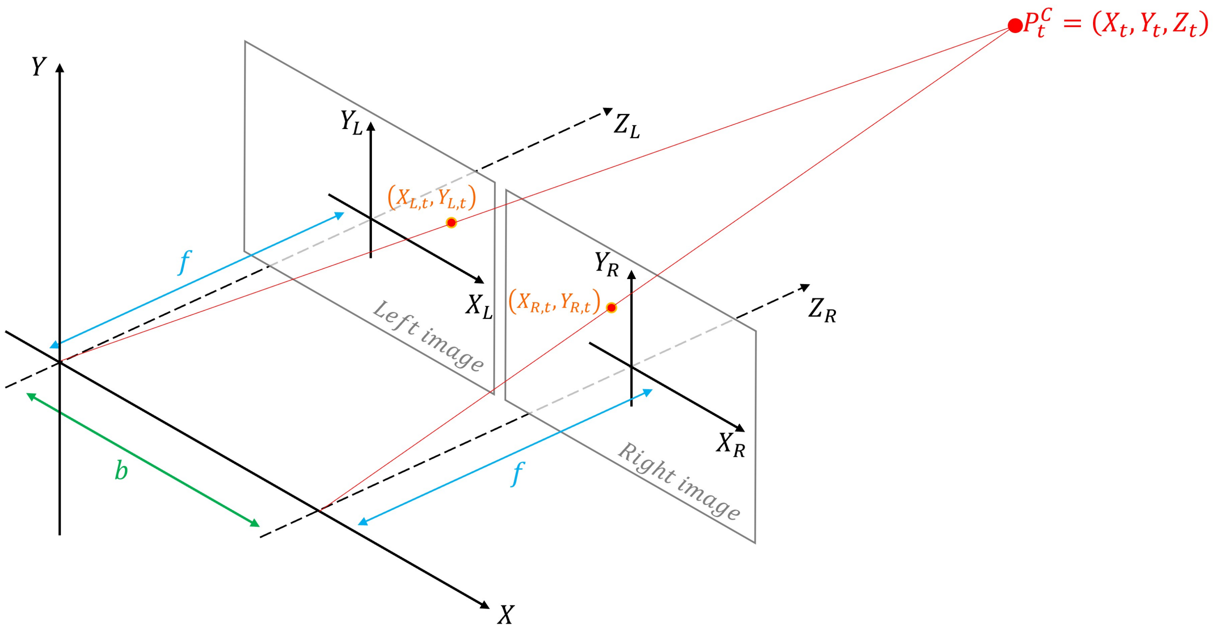

Figure 3.18: Triangulation procedure: to reconstruct the 3D position of the tool in the camera frame, $\boldsymbol{P}_t^C((X_t, Y_t, Z_t)$, the positions of the tool estimated with respect to the two images frames, $(X_{R,t}, Y_{R,t})$ and $(X_{L,t}, Y_{L,t})$, are used. b is the so-called "Baseline" and is the distance between the optical centers of the two camera sensors. f is the focal length. b and f are two fundamental parameters for the position triangulation.

The first step consists in removing the distortions from the points that represent the tool using the distortion coefficients computed during the calibration procedure. This is done in order to eliminate the effects introduced by the lenses of the camera. In this way, the relationship between the 2D coordinates in the image plane and 3D coordinates in the camera reference frame can be considered linear. Thanks to this property and to the (3×4) projection matrices, $\boldsymbol{M}_R$ and $\boldsymbol{M}_L$ (comprising both the intrinsic and extrinsic parameters of the two cameras), is possible to reconstruct the 3D coordinates $\boldsymbol{P}_t^C$ of a point by triangulation.

Given that $\boldsymbol{P}_{R,t}$ and $\boldsymbol{P}_{L,t}$ are the 2D coordinates in the image plane of the same point $\boldsymbol{P}_t^C$ in the 3D world, the following relationships can be written

$$\boldsymbol{P}_{R,t} = \boldsymbol{M}_R \boldsymbol{P}_t^C \tag{3.28}$$

$$\boldsymbol{P}_{L,t} = \boldsymbol{M}_L \boldsymbol{P}_t^C \tag{3.29}$$

Since the measurements of $\boldsymbol{P}_{R,t}$ and $\boldsymbol{P}_{L,t}$ are usually noisy and generally no point $\boldsymbol{P}_t^C$ will satisfy both equations, the best fit must be found.
By computing the cross-product between $\boldsymbol{P}_{R,t}$ and $\boldsymbol{M}_R \boldsymbol{P}_t^C$ (that have the same direction) the homogeneous scale factor s is eliminated and the following is obtained:

$$\begin{bmatrix} X_{R,t} \\ Y_{R,t} \\ 1 \end{bmatrix} \times \begin{bmatrix} m_{R1}^T \boldsymbol{P}_t^C \\ m_{R2}^T \boldsymbol{P}_t^C \\ m_{R3}^T \boldsymbol{P}_t^C \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.30}$$

where m_{Ri}^T represents the i^{th} row of the projection matrix for the right camera.
In this way, for the image point of the right camera $\boldsymbol{P}_{R,t}$, the following equations can be written

$$Y_{R,t}(m_{R3}^T \boldsymbol{P}_t^C) - m_{R2}^T \boldsymbol{P}_t^C = 0 \tag{3.31}$$

$$X_{R,t}(m_{R3}^T \boldsymbol{P}_t^C) - m_{R1}^T \boldsymbol{P}_t^C = 0 \tag{3.32}$$

$$X_{R,t}(m_{R2}^T \boldsymbol{P}_t^C) - Y_{R,t}(m_{R1}^T \boldsymbol{P}_t^C) = 0 \tag{3.33}$$

After that the same relations for the left image have been computed, by putting together the linearly independent equations written for $\boldsymbol{P}_{R,t}$ and $\boldsymbol{P}_{L,t}$, the following expression is obtained:

$$\boldsymbol{A} \boldsymbol{P}_t^C = 0 \tag{3.34}$$

where

$$\boldsymbol{A} = \begin{bmatrix} Y_{R,t} m_{R3}^T - m_{R2}^T \\ X_{R,t} m_{R3}^T - m_{R1}^T \\ Y_{L,t} m_{L3}^T - m_{L2}^T \\ X_{L,t} m_{L3}^T - m_{L1}^T \end{bmatrix} \tag{3.35}$$

Solving for $\boldsymbol{A}$ by means of singular value decomposition(SVD), the 3D coordinates $\boldsymbol{P}_t^C$ is reconstructed.

Once the surgical instrument has been triangulated in the 3D space and the position $\boldsymbol{P}_t^C = (X_t, Y_t, Z_t)$ has been obtained, several reference frame transformations have to take place before computing the feedback error needed by the regulator.
They are described in Figure 3.19:

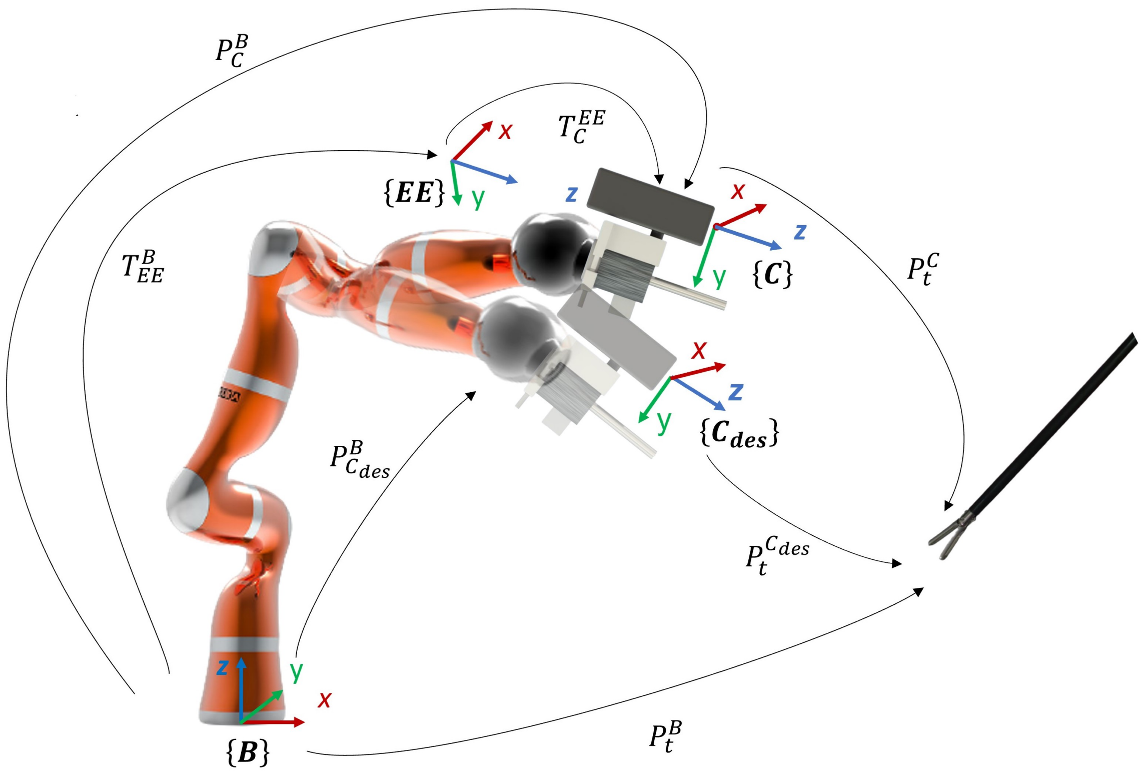

Figure 3.19: Transformations between different reference frames: $\boldsymbol{\{B\}}$ is the reference frame of the robot base, $\boldsymbol{\{EE\}}$ is the reference frame of the end-effector, $\boldsymbol{\{C\}}$ and $\boldsymbol{\{C_{des}\}}$ are the references frames associated to the actual and desired camera poses, respectively. $\boldsymbol{T}_{EE}^B$ and $\boldsymbol{T}_C^{EE}$ are the transformation between the base frame and the EE frame and between the EE frame and the camera frame. $\boldsymbol{P}_C^B$ and $\boldsymbol{P}_{C_{des}}^B$ are the actual and desired poses of the camera frame with respect to the base frame. $\boldsymbol{P}_t^C$, $\boldsymbol{P}_t^{C_{des}}$, and $\boldsymbol{P}_t^B$ are the positions of the tool in the $\boldsymbol{\{C\}}$, $\boldsymbol{\{C_{des}\}}$ and $\boldsymbol{\{B\}}$ frame respectively.

By using the kinematic chain of the manipulator $\boldsymbol{T}_{EE}^B$ together with the transformation obtained by the eye-in-hand calibration procedure $\boldsymbol{T}_C^{EE}$, the estimated object position $\boldsymbol{P}_t^C$ can be expressed with respect to the manipulator's base reference frame:

$$\boldsymbol{P}_t^B = \boldsymbol{T}_{EE}^B \cdot \boldsymbol{T}_C^{EE} \cdot \boldsymbol{P}_t^C \tag{3.36}$$

The elements of the homogeneous transformation $\boldsymbol{P}_t^{C_{des}}$ (between the tool position and the camera desired pose) will be discussed in the Subsections 3.4.2 and 3.4.3 on the basis of the camera movements strategy used.
Exploiting $\boldsymbol{P}_t^{C_{des}}$ and $\boldsymbol{P}_t^{B}$, the new pose of the camera with respect the base frame can be computed:

$$\boldsymbol{P}_{C_{des}}^{B} = \boldsymbol{P}_t^{B}(\boldsymbol{P}_t^{C_{des}})^{-1} \tag{3.37}$$

while the actual pose of the camera, $\boldsymbol{P}_C^B$, can be derived from:

$$\boldsymbol{T}_C^B = \boldsymbol{T}_{EE}^B \cdot \boldsymbol{T}_C^{EE} \tag{3.38}$$

3.4.2. Position Control

The position control error is given by:

$$\boldsymbol{e}_{pos} = \boldsymbol{P}_{C_{des}}^{B} - \boldsymbol{P}_{C}^{B} \tag{3.39}$$

In this case, the homogeneous transformation between the tool position and desired camera pose is the following:

$$\boldsymbol{P}_{t}^{C_{des}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.40}$$

since the goal is to keep the instrument close to the center of the camera image with a distance d on the Z coordinate (without modifying the initial orientation of the camera). This error, $\boldsymbol{e}_{pos}$, was then fed into a Resolved velocity controller which calculates the desired joint velocities as explained later on the controller Subsection.

3.4.3. Orientation Control

The concept of the Remote Center of Motion (RCM) is a fundamental principle in robotics and biomechanics that plays a crucial role in designing and controlling robotic systems, particularly in minimally invasive surgical procedures. The RCM, also known as the Instantaneous Center of Rotation ($\boldsymbol{ICR}$) or Pivot Point, is a point within a mechanism or robotic system where the motion of a tool or end-effector is equivalent to a rotational movement around a stationary point. This is particularly relevant in applications where precise and controlled movement is required within confined spaces, for example when surgical instruments are used in laparoscopic or robotic-assisted procedures [35] [36].
The RCM has numerous applications beyond medicine, including robotics, kinematics, and mechanical engineering. Its significance lies in its ability to facilitate safe and accurate manipulation of tools within constrained environments while minimizing tissue damage and enhancing the dexterity of the robotic system.
In this work the idea behind the RCM is exploited in the Orientation control. This movement strategy includes an initial positioning phase, and the general functioning steps are the following:

1. While the initial positioning is running, the surgeon is asked to move the surgical tool approximately in the center of the area in which the surgical operation will happen. In this phase, the camera moves in the position control mode and the purpose is to establish the $\boldsymbol{ICR}$ position in such a way that the camera offers an optimal field of view without tilting too much. This is a very important step for the overall orientation control operative success because the performance of this strategy is highly affected by the $\boldsymbol{ICR}$ position.

2. Once the tool stops at the center of the working space and the controller detects a null error between the camera position and the tool position, the $\boldsymbol{ICR}$ is defined, the position control mode is disengaged, and the orientation control modality starts to work.

3. When the orientation control is working, the position of the target object, $\boldsymbol{P}_t^B$, is used to compute the desired orientation of the EE, which rotates around the $\boldsymbol{ICR}$, located in the middle of the tool as shown in Figure 3.20.

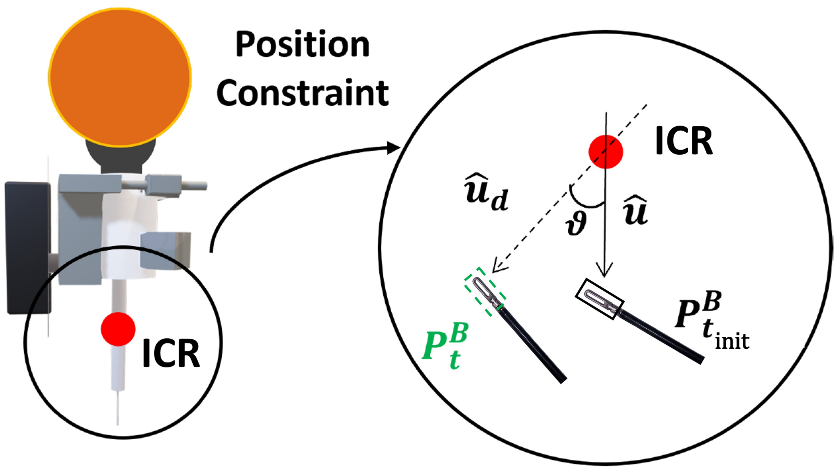

Figure 3.20: Position constraint representation. The robot rotates around ICR by an angle θ calculated as the angle between the initial position $\boldsymbol{P}^B_{t_{init}}$ and the desired position $\boldsymbol{P}^B_t$.

To control the orientation of the robot, firstly, the rotation angle between the initial tool direction $\hat{\boldsymbol{u}} = \boldsymbol{P}^B_{t_{init}} - \boldsymbol{ICR}$ and the desired tool direction $\hat{\boldsymbol{u}}_{\boldsymbol{d}} = \boldsymbol{P}^B_t - \boldsymbol{ICR}$ is defined as:

$$\theta = \arctan^{-1} \frac{||\hat{\boldsymbol{u}}_{\boldsymbol{d}} \times \hat{\boldsymbol{u}}||}{\hat{\boldsymbol{u}}_{\boldsymbol{d}} \cdot \hat{\boldsymbol{u}}} \tag{3.41}$$

The vector describing the rotation from $\hat{\boldsymbol{u}}$ to $\hat{\boldsymbol{u}}_{\boldsymbol{d}}$ in Figure 3.20 is computed as:

$$\boldsymbol{u} = \frac{\hat{\boldsymbol{u}}_{\boldsymbol{d}} \cdot \hat{\boldsymbol{u}}}{||\hat{\boldsymbol{u}}_{\boldsymbol{d}} \times \hat{\boldsymbol{u}}||} \tag{3.42}$$

Finally, the desired orientation was calculated as described in [36]:

$$\boldsymbol{R}^B_{C_{des}} = (I + \boldsymbol{\Lambda} \cdot \sin(\theta) + 2 \cdot \boldsymbol{\Lambda}^2 \cdot \sin 2(\frac{\theta}{2}))\boldsymbol{R}^B_C \tag{3.43}$$

where $\boldsymbol{\Lambda}$ is the rotation matrix given by u, and $\boldsymbol{R}^B_C$ is the actual orientation of the EE computed from the kinematic chain of the robot. From $\boldsymbol{R}^B_{Cdes}$, the orientation error is computed using quaternion notation as:

$$\boldsymbol{e}_{rot} = \boldsymbol{q}_{rotC} \cdot \boldsymbol{q}^{-1}_{rotC_{des}} \tag{3.44}$$

where $\boldsymbol{q}_{rotC}$ and $\boldsymbol{q}^{-1}_{rotC_{des}}$ are the current and desired quaternions. This error, $\boldsymbol{e}_{rot}$ is then sent to the Resolved velocity controller to calculate desired joints' velocities as in the position control mode.

For a matter of optimization and constancy of the field of view, it has been chosen to keep the yaw angle constant and to vary the roll and pitch angles. To understand which are the used roll, pitch and yaw angles, please refer to Figure 3.21.

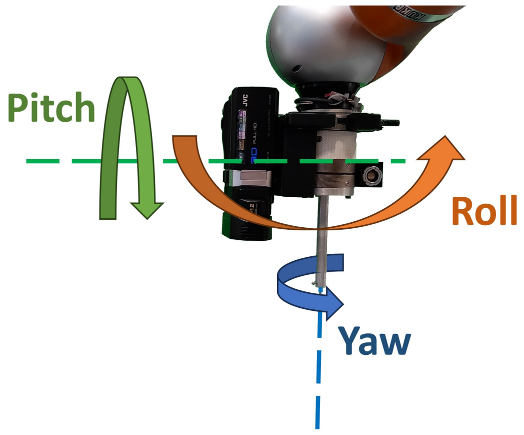

Figure 3.21: The roll, pitch and yaw angles used.

3.4.4. Resolved Velocity Controller

A position-based visual servoing control approach has been implemented for this study. Usually position-based visual servoing control is realized using a PD controller with gravity compensation, whose control law has the following expression:

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}^{\#}_{A_W}(\boldsymbol{q})(\boldsymbol{K}_P \cdot \boldsymbol{e} - \boldsymbol{K}_D \cdot \boldsymbol{J}_A(\boldsymbol{q}) \cdot \dot{\boldsymbol{q}}) \tag{3.45}$$

where $\boldsymbol{u}$ is the control input, $\boldsymbol{q}$ is the vector of joints' position, $\dot{\boldsymbol{q}}$ is the vector of joints' velocity, and $\boldsymbol{g}(\boldsymbol{q})$ is the gravity compensation term. $\boldsymbol{J}^{\#}_{A_W}(\boldsymbol{q})$ is the pseudo-inverse of the analytical Jacobian matrix, $\boldsymbol{e}$ is the error vector and $\boldsymbol{J}_A(\boldsymbol{q})$ is the analytical Jacobian of the manipulator in the operational space. Finally, $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$ are the proportional and derivative gains respectively.
The pseudo-inverse has to be employed when the robot is redundant and so infinite solutions exist to the inverse kinematics problem [37]. In particular, in this work, the weighted pseudo-inverse of the analytical Jacobian has been used:

$$\boldsymbol{J}^{\#}_{A_W}(\boldsymbol{q}) = \boldsymbol{W}^{-1}\ \boldsymbol{J}^T_A(\boldsymbol{q})\ (\boldsymbol{J}_A(\boldsymbol{q})\ \boldsymbol{W}^{-1}\ \boldsymbol{J}^T_A(\boldsymbol{q}))^{-1} \tag{3.46}$$

where $\boldsymbol{W}$ is the matrix of the weights of the joints' velocities and $\boldsymbol{J}_A$ is the analytical Jacobian.

In this system, like in many other similar setups, the information deriving from visual measurements is computed at a frequency lower or equal to the camera frame rate. This quantity is at least one order of magnitude lower than the typical frequencies used for motion control of robot manipulators [17]. As a consequence, in the digital implementation of the typical control law (3.45), to preserve the stability of the closed-loop system, the control gains must be set to values much lower than those typically used for motion control. Therefore, the performance of the closed-loop system in terms of speed of convergence and disturbance rejection capability turns out to be poor.
Neglecting the effects of the tracking errors deriving from manipulator dynamics and disturbances, the controlled manipulator can be considered as an ideal positioning device.
This allows to assume that the manipulator is equipped with a high-gain motion controller in the joint space and that the desired and actual joints' positions and velocities are approximately equal, so $\boldsymbol{q} \approx \boldsymbol{q}_{des}$ and $\dot{\boldsymbol{q}} \approx \dot{\boldsymbol{q}}_{des}$.
These assumptions have been verified on the real robot, which results to be very accurate and fast in reaching the desired joints' positions and speeds.
Visual servoing can be thus achieved by computing the trajectory $\dot{\boldsymbol{q}}_{des}(t)$ on the basis of

visual measurements, so that the operational space tracking error $\boldsymbol{e}$ goes asymptotically to zero.
The joints' trajectory $\boldsymbol{q}_{des}(t)$ can be computed integrating the joints' velocity profile $\dot{\boldsymbol{q}}_{des}(t)$, that is obtained as follows:

$$\dot{\boldsymbol{q}}_{des}(t) = \boldsymbol{J}^{\#}_{A_W}(\boldsymbol{q})\boldsymbol{K}\boldsymbol{e} \tag{3.47}$$

in which $\boldsymbol{J}^{\#}_{A_W}$ is the analytical Jacobian pseudo-inverse, $\boldsymbol{K}$ is the matrix of the controller gains, and $\boldsymbol{e}$ is the vector of feedback errors.
The study of the stability for this type of controller can be performed with the so-called "Lyapunov Method". The Lyapunov function V can be chosen as follows:

$$\boldsymbol{V} = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} \tag{3.48}$$

$\boldsymbol{V}$ is of course greater than zero. The time derivative of the Lyapunov function $\dot{\boldsymbol{V}}$ can be obtained as:

$$\dot{\boldsymbol{V}} = \boldsymbol{e}^T\dot{\boldsymbol{e}} \tag{3.49}$$

Knowing that the following equation holds:

$$\dot{\boldsymbol{e}} = -\boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.50}$$

It is straightforward that equation 3.47 becomes:

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = 0 \tag{3.51}$$

And exploiting the relation 3.51, the equation 3.49 becomes:

$$\dot{\boldsymbol{V}} = -\boldsymbol{e}^T\boldsymbol{K}\boldsymbol{e} \tag{3.52}$$

that is lower than zero if and only if $\boldsymbol{K}$ is positive definite. So, thanks to the Lyapunov Method, since $\boldsymbol{V} > 0$ and $\dot{\boldsymbol{V}} < 0$, with $\boldsymbol{K}$ positive definite, the equilibrium $\boldsymbol{e} = 0$ is proven to be globally asymptotically stable. Moreover, for a positive definite matrix $\boldsymbol{K}$, equality 3.51 implies that the operational space error tends to zero asymptotically with a convergence of exponential type and speed depending on the eigenvalues of matrix $\boldsymbol{K}$; the larger the eigenvalues, the faster the convergence.
The controller described above is called Resolved Velocity Controller, and in this thesis it has been implemented as shown in Figure 3.22.

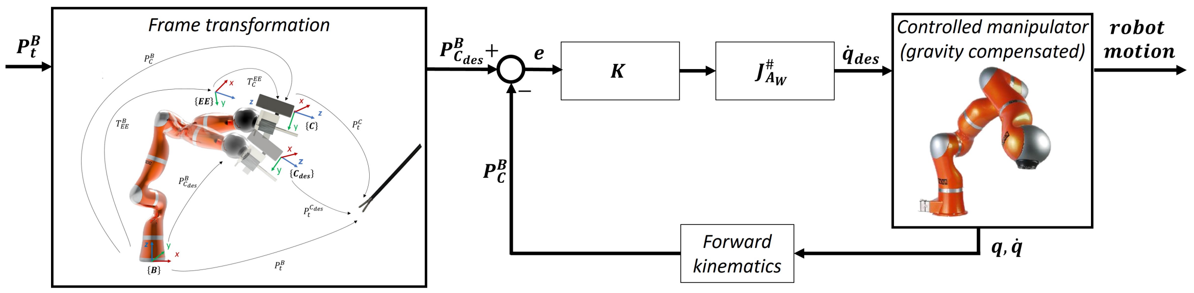

Figure 3.22: Control scheme of the overall system: the estimated position of the tool in the base frame, $\boldsymbol{P}_t^B$, is processed to generate the desired position of the camera, $\boldsymbol{P}_{C_{des}}^B$ (according to the type of control: position or orientation). The actual position of the camera, $\boldsymbol{P}_C^B$, is derived from the forward kinematics calculated thanks to the joints' positions and velocities measurements given by the robot's sensors. Then, the feedback error $\boldsymbol{e}$ is computed through $\boldsymbol{P}_{C_{des}}^B$ and $\boldsymbol{P}_C^B$ and it is fed into a resolved velocity controller. This controller is composed of the positive-definite gain matrix $\boldsymbol{K}$, by the weighted pseudo-inverse matrix of the analytical Jacobian and by the controlled manipulator.

The value of $\boldsymbol{K}$ has been fine-tuned experimentally in order to guarantee rapidity and smoothness of the movements at the same time. This process has taken place after the evaluation of the optimal strategy of the hybrid tracking module, and has determined the choice of the following parameters:

Errors on	Position control gains	Orientation control gains
$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$	$\boldsymbol{K}_{pos} = \begin{bmatrix} 4.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$	$\boldsymbol{K}_{pos} = 0.0 \cdot I(3)$
$\begin{bmatrix} roll \\ pitch \\ yaw \end{bmatrix}$	$\boldsymbol{K}_{rot} = 1.0 \cdot I(3)$	$\boldsymbol{K}_{rot} = 0.7 \cdot I(3)$

Table 3.5: Resolved velocity controller parameters.

As can be noticed from Table 3.5, different parameters are needed for the two motion

control strategies because errors have to be weighted differently depending on the approach used. In particular, when the robot is in position control mode the robot has to vary especially the X and Y camera positions, while the Z position does not have to change that much and not very quickly. The orientation errors should be equal to zero with this strategy but if they are not, they are rapidly corrected.
In orientation control strategy if the errors on X, Y or Z would be considered, the camera would not rotate around a fixed point but would try to follow the surgical instrument as in position control mode, so the position errors have not to be taken into account. The errors on the angles, in this case, are bigger than in the previous strategy, so, to guarantee smoothness and stability, the gains have to be smaller.

To control the robotic manipulator, the ROS (Robotic Operating System) middleware has been employed. The robot and the ROS network communicate through the "Fast Robot Interface" (FRI) of KUKA that allows to have a fast (up to 1 kHz) and reliable communication channel. The control frequency used in this thesis' setup is 200 Hz.

4 | Experimental Protocol

This chapter describes how the proposed system was evaluated.

First, the different tool tracking strategies were tested to assess which one guarantees the best performance in surgical instrument tracking using the position-controlled autonomous exoscope.

Then, the tuning of the position and orientation controllers was carried out, and the same experiment was performed, exploiting the best strategy of the previous testing phase, to evaluate the improvements achieved in the robot control.

Finally, the overall system was validated through a User Study to assess whether the proposed system was able to overcome the limitations of the camera control strategies currently used in neurosurgery by comparing traditional (manual) and proposed (automatic) modes of camera movement.

4.1. System Characterization

4.1.1. Strategies Validation

In this evaluation phase, the strategies developed by the Nearlab laboratory (CNN15 and CNN30) and the ones introduced in this thesis work (PARTICLE, OPFL, and HYBRID) were compared to assess which offered the best performance in tracking the surgical instrument.
The test involved the use of a second 7 DoF robotic manipulator (KUKA LBR iiwa produced by KUKA Roboter GmbH) to move the surgical tool along a predefined 2D trajectory. This second robotic arm was used to guarantee a high degree of repeatability across the experiments, since it allowed to move the tool along the same trajectory and with the same velocity profile in all the tests ensuring their comparability.
How the experimental setup is structured can be appreciated in Figure 4.1.

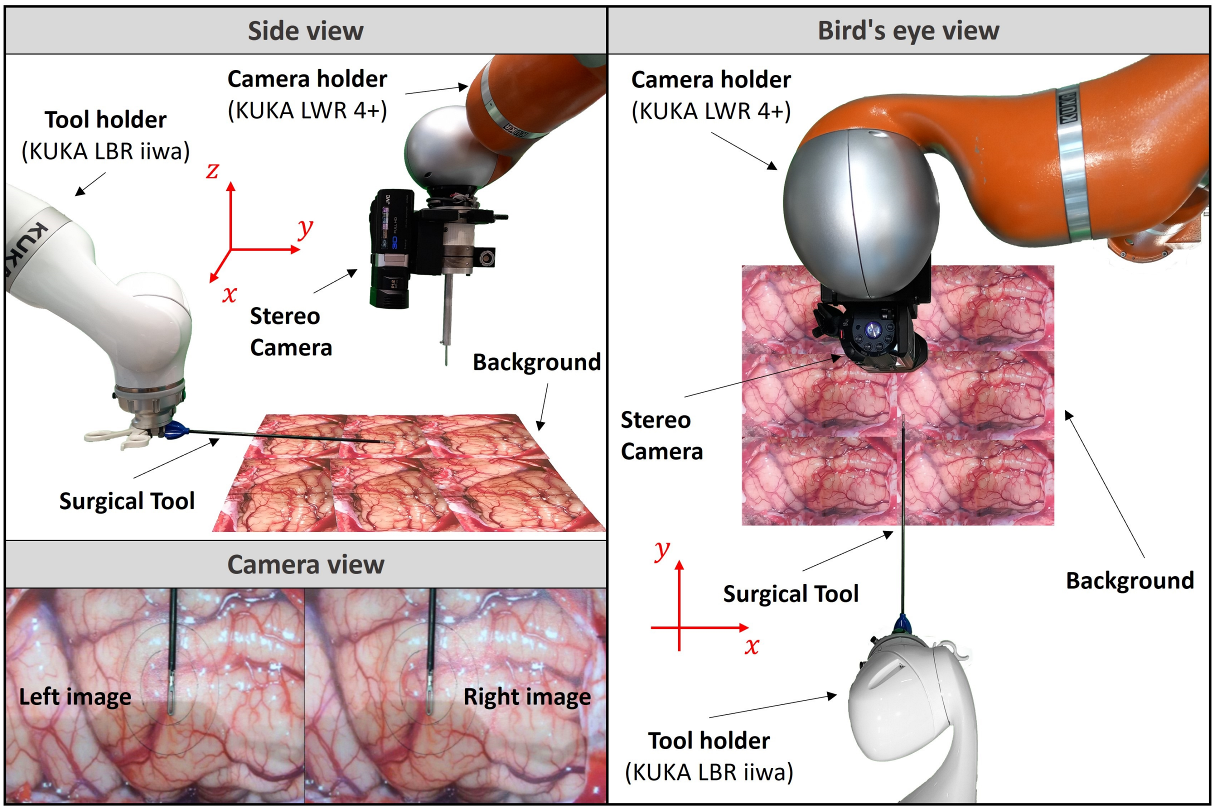

Figure 4.1: Experimental setup from different point of view. The KUKA LWR 4+ robotic arm moves the stereo camera trying to track the surgical tool attached to the EE of the KUKA LBR iiwa robot. The background under the instrument can be changed (green background and realistic background are available). The camera view shows what the camera sees during the experiments.

To test the robustness of the various strategies, the system was tested in 4 different scenarios considering two different tool speeds and two different backgrounds.
The two speeds were chosen on the basis of a study carried out with neurosurgeons about the typical speeds reached during brain surgeries: the system was thus tested with the tool moving at about 2.5 cm/s (low speed) and about 4 cm/s (high speed).
The two backgrounds were:

- A green background, which should facilitate the tool detection;
- A realistic background representing a portion of the brain during a surgical operation, which mimics the real scenario of the use of the exoscope and that complicates the tool detection.

The second redundant manipulator was programmed with the KUKA's proprietary software KUKA Sunrise Workbench © to move the surgical tool with the two different velocities following the trajectory described below and in Figure 4.2:

1. The first movement was represented by a circular trajectory with a 7 cm diameter traveled in counterclockwise direction starting from the lowest point;
2. Then a movement of 5 cm in the x positive direction was made (referring to the red axis drawn in Figure 4.1 and Figure 4.2);
3. It followed a 10 cm linear movement in the x negative direction;
4. The tool then went back to the initial position with a 5 cm linear movement in the x positive direction;
5. It followed a linear motion of 7 cm in the y positive direction;
6. The last movement, before concluding the test, was represented by a half circular trajectory with a 7 cm diameter traveled in clockwise direction toward the test's initial position;

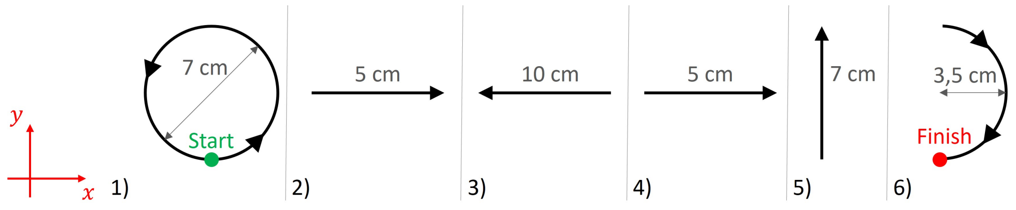

Figure 4.2: Trajectory followed by the surgical tool during the evaluation of the tool tracking strategies.

All the tests were repeated 5 times and, during this experimental phase, only the Position control mode was used to move the camera (so the orientation of the camera was constant). Since the controller was not already tuned and for safety reasons, due to the presence of instabilities, during these experiments the proportional gains of the controller were set equal to one, resulting in $\boldsymbol{K}_{pos} = \boldsymbol{I}_3$ and $\boldsymbol{K}_{rot} = \boldsymbol{I}_3$.

To summarize, five strategies were tested in this evaluation performing five repetitions for each of the four scenarios.

The performance indexes evaluated for these tests were the Tracking Error and the Center Error explained below.
The Tracking Error, TE_{xy}, was defined as the xy distance between the position of the tool and the position of the camera and it was computed as follows:

$$TE_{xy} = ||\boldsymbol{P}_C^B - \boldsymbol{P}_{tool}^B|| = \sqrt{(X_C^B - X_t^B)^2 + (Y_C^B - Y_t^B)^2}\ [mm] \tag{4.1}$$

where $\boldsymbol{P}_C^B(X_C^B, Y_C^B)$ and $\boldsymbol{P}_{tool}^B(X_t^B, Y_t^B)$ are the positions of the camera and of the tool respectively expressed in the base frame.
The Center Error, CE_{xy}, was defined as the distance between the tool position in the camera frame, $\boldsymbol{P}_{tool}^C(X_t^C, Y_t^C)$, and the center of the image (C_x, C_y):

$$CE_{xy} = ||X_t^C - C_x, Y_t^C - C_y|| = \sqrt{(X_t^C - C_x)^2 + (Y_t^C - C_y)^2}\ [mm] \tag{4.2}$$

4.1.2. Control Module Validation

After the validation of the detection strategy, the system controllers were fine-tuned. To evaluate the effectiveness of the fine-tuning, the system was tested again with an experiment in both the control modality: position and orientation.

Position Control

In the Position control mode, the experiments performed were exactly the same as the strategies evaluation described in Section 4.1.1. The only difference was that, in this case, just the best tracking strategy was used with the tuned controller.

Orientation Control Validation

The tests of the system in the Orientation control mode, in principle, worked similarly to the experiment performed to evaluate the strategies and the fine-tuning of the Position control mode. The differences in the tests performed on this control modality were represented by:

- **A smaller trajectory traveled by the tool**: since the Orientation control mode was designed to be applied to scenarios where the movements are small, the considered trajectory was smaller. In particular, the trajectory followed by the tool was equal in terms of sequence of motions to the one described in Section 4.1.1, but different in dimensions, as described in Figure 4.3.

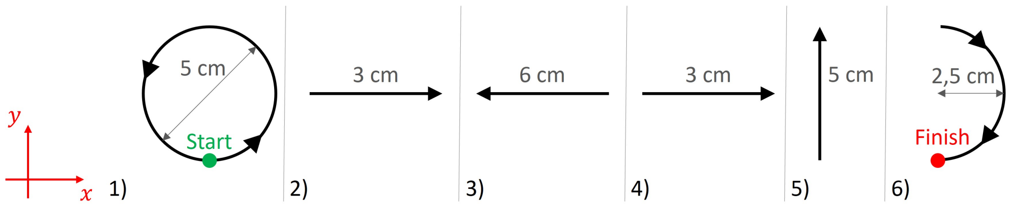

Figure 4.3: Trajectory followed by the surgical tool during the evaluation of the orientation controller tuning.

- **The metrics considered**: since the camera in this control mode rotated around a fixed point, the Tracking Error TE_{xy} considered in the Position control mode was no longer suitable to evaluate the performance of the system. So, the performance indexes evaluated in this case were: the Center Error CE_{xy} (computed as described in equation 4.2 of the previous Section) and two new Tracking Errors, TE_{roll} and TE_{pitch}. TE_{roll} was defined as the distance between the current and desired roll angle of the camera and it was obtained as follows:

$$TE_{roll} = \min(|roll_{cam} - roll_{des}|\,,\; 2\pi \; - |roll_{cam} - roll_{des}|)\;[deg] \tag{4.3}$$

  where $roll_{cam}$ is the actual roll angle of the camera, while $roll_{des}$ is the desired roll angle of the camera. TE_{pitch} was computed ins the same way of TE_{roll} but considering the pitch angle:

$$TE_{pitch} = \min(|pitch_{cam} - pitch_{des}|\,,\; 2\pi \; - |pitch_{cam} - pitch_{des}|)\;[deg] \tag{4.4}$$

  where $pitch_{cam}$ is the actual pitch angle of the camera, while $pitch_{des}$ is the desired pitch angle of the camera.

Also with the Orientation control mode the best tool tracking strategy was used during the tests that involved four scenarios (given by two instrument speeds and two backgrounds as described in 4.1.1)

4.2. User Study

A User Study was conducted to check if the developed system, compared to the traditional exoscope control mode, was effective in reducing the users' workload during the execution of a task. To mimic the surgical environment, 12 users were requested to perform a bimanual task using two surgical instruments and moving the stereo camera in three different modalities:

- **Manual**: the users had to move manually the camera in the XY plane trying to keep the two tools at the image center. The users had to move the tools towards the desired direction, then leave one of the two tools (usually the cannula) and use the free hand to move the robot. The robot that is controlled through a Cartesian Impedance regulator (the orientation and the Z coordinate are kept constant);
- **Auto - Translation**: the autonomous exoscope system is controlled with the Position control mode (described in Chapter 3.4.2) and the users decide when to track the tool with the camera by pressing a pedal;
- **Auto - Rotation**: the autonomous camera is controlled with the Orientation control mode (described in Chapter 3.4.3) and the users decide when to track the tool with the system by pressing a pedal.

All users had to perform the tests using all the strategies and they had to repeat the experiments three times for each mode.
To avoid that the eventual presence of a learning curve could affect the results of this testing phase, the order of the strategies proposed to the users was chosen from the set of permutations of the strategies. For example, the first user followed this order: $t1, t2, t3, r1, r2, r3, m1, m2, m3$ where the numbers indicate the repetitions, t stands for Translation, r for Rotation, and m for Manual. While, for instance, the second user followed this order: $r1, r2, r3, m1, m2, m3, t1, t2, t3$.
Before the start of each User Study, users signed an informed consent document that described the tests they had to perform and how the recorded data were stored. After that, they were instructed on how the system works and they had some time to familiarize themselves with each camera control mode.

The experimental setup of the User Study can be appreciated in the Figure 4.4.

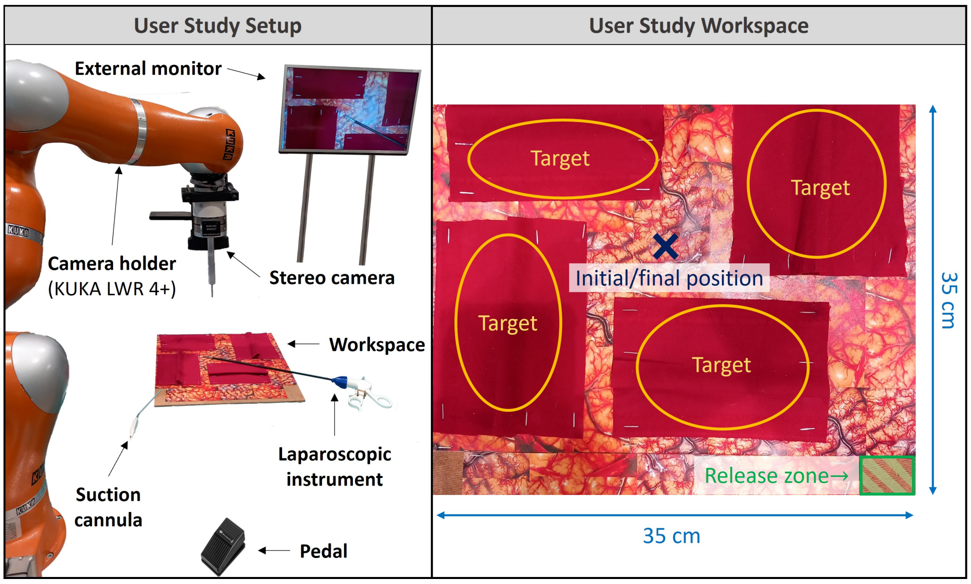

Figure 4.4: User Study Setup (left-hand side) including everything the users were requested to employ during the User Study: the robot to move the camera, the stereo camera to observe the scene, the external monitor to look at camera images, the workspace where to accomplish the task, two surgical tools and a pedal, to decide when to move the robot in the automatic control modes. User Study Workspace (right-hand side): it was divided into 4 different targets, a release zone, and the initial/final position of the surgical instrument.

The workspace, shown in Figure 4.4, was composed of a wood board to which many images of a human brain were glued. On the top of these photos, four different targets were fastened. Each target was made by two pieces of tissue partially attached to the structure below. The two tissues were in part overlapped to allow to hide the object which was represented by a rubber gasket. The workspace included also an initial and final position for the surgical instruments and a release zone for the rubber ring.

A bimanual pick and place of a hidden object was the task considered in this study to mimic a surgical scenario in which the surgeon has to use both the hands to carry out the surgical operation. This specific task was also chosen to test the system on a big surface and to force the users to use both the hands.
The tests, after that the User Study supervisor hid the rubber ring under one of the targets (while the user was not looking at the workspace), were structured as follows:

1. **Starting position**: the users start the test with the tools in the initial position;
2. **Searching for the hidden ring**: the users look for the object under various targets until they find it.

3. **Pick and Place**: once the rubber ring is grasped, the users bring it to the release zone and place it there;
4. **Final position**: the users move the tools towards the final position and when they are in place, the test is concluded. Since initial and final position coincide, a new test, after the reset of the workspace, can be performed.

The phases described above are shown in Figure 4.5.

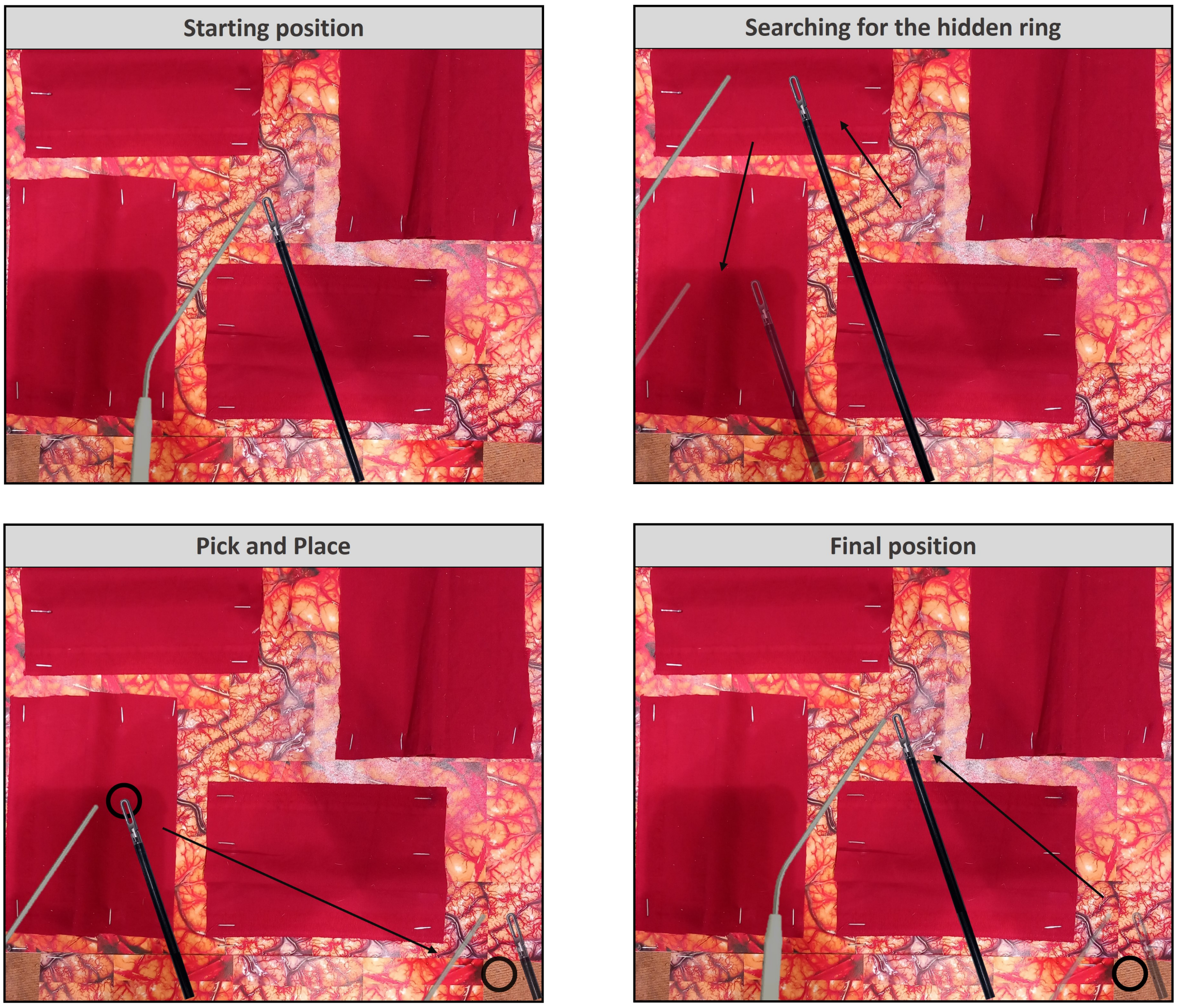

Figure 4.5: Phases of the User Study tests.

In Figure 4.6 is shown what happened when the users arrived on a target to look for the rubber ring under it, in case of unsuccessful and successful research.

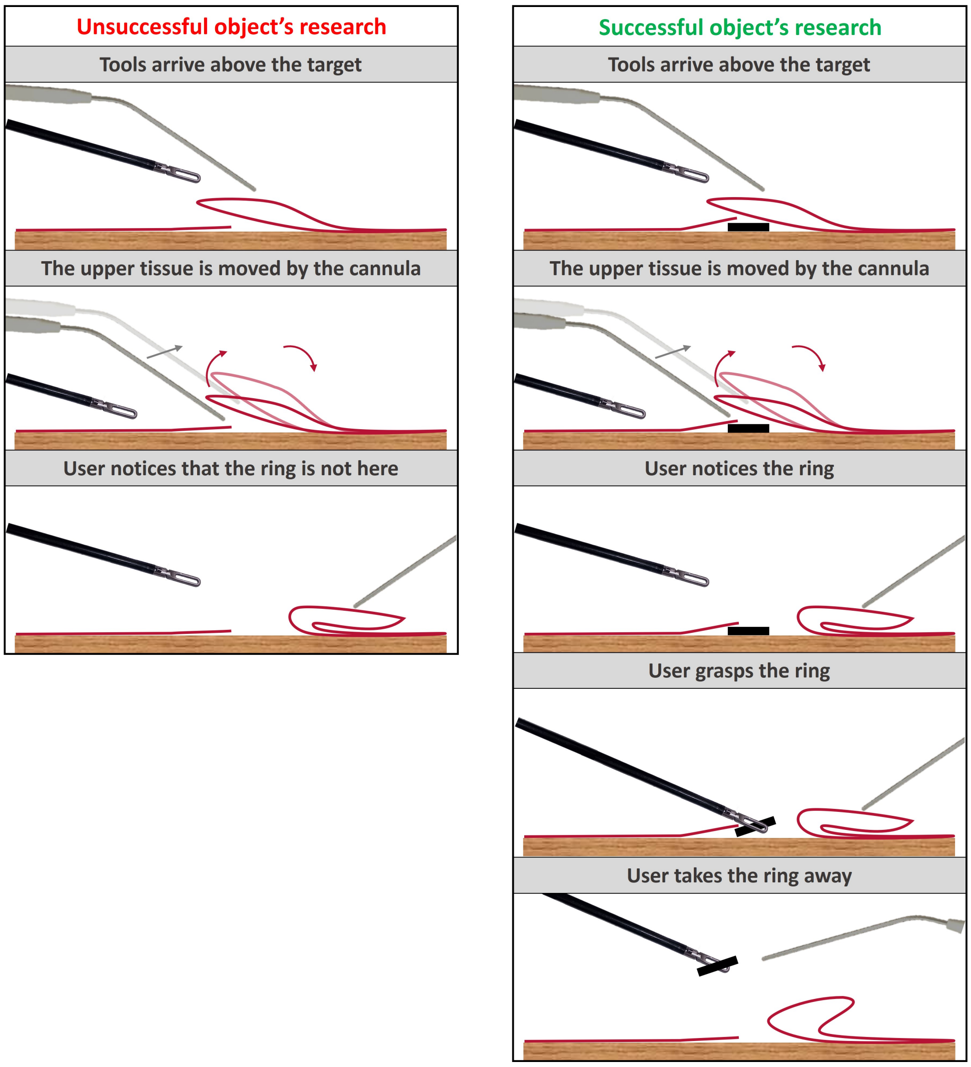
Unsuccessful object's research
Tools arrive above the target
The upper tissue is moved by the cannula
User notices that the ring is not here
Successful object's research
Tools arrive above the target
The upper tissue is moved by the cannula
User notices the ring
User grasps the ring
User takes the ring away

Figure 4.6: Phases of the object's research. On the left-hand side is shown what happens if the research fails: arrival on the target, movement of the upper tissue, discovery of the absence of the ring. On the right-hand side is shown what happens when the ring is under the target: arrival on the target, movement of the upper tissue, discovery of the ring, ring grasping and removal.

The order of targets to visit was chosen by each user.
During the execution of the test, the users were requested to look only at the external monitor that streamed the camera images and not directly to the scene. They were also asked to try to keep always both tools at the center of the image. This last request in the manual control mode was accomplished by following a precise sequence of actions in case long movements were necessary: the users had to move the tools towards the desired direction, then leave one of the two tools (usually the cannula) and use the free hand to move the robot. Finally, they had to grasp again the released tool. These actions had to

be repeated until the desired position was reached.

Different metrics were recorded and analyzed to evaluate the User Study:

- A score accounting for the distance of the laparoscopic tool from the camera center;
- The duration of the task;
- The path length travelled by the tool between specific areas of the workspace;
- A NASA Task Load Index (NASA-TLX) survey for each control strategy;
- A questionnaire.

The score was obtained as described in the following algorithm:

Algorithm 4.1 Score Computation Algorithm $(X_t, Y_t, IS\_OUT\_FOV, score_{curr})$.

```
1: $d = \sqrt{(X_t - X_c)^2 + (Y_t - Y_c)^2}$
2: if $IS\_OUT\_FOV$ then
3:     $score_{curr} = score_{curr} - 5$
4: else if $d > 0.05$ then
5:     $score_{curr} = score_{curr} - 1$
6: end if
7: return $score_{curr}$
```

where (X_t, Y_t) are the coordinates (in meters) of the surgical tool deriving from the tool detection or tracking module. IS_OUT_FOV is a Boolean variable that indicates whether the surgical tool cannot be detected in the images by the CNN or the tracking algorithm because it is occluded or out of the field of view of the camera. $score_{curr}$ is the score of the test, computed up to the current iteration. d is the Euclidean distance between (X_t, Y_t) and the camera center, (X_c, Y_c).

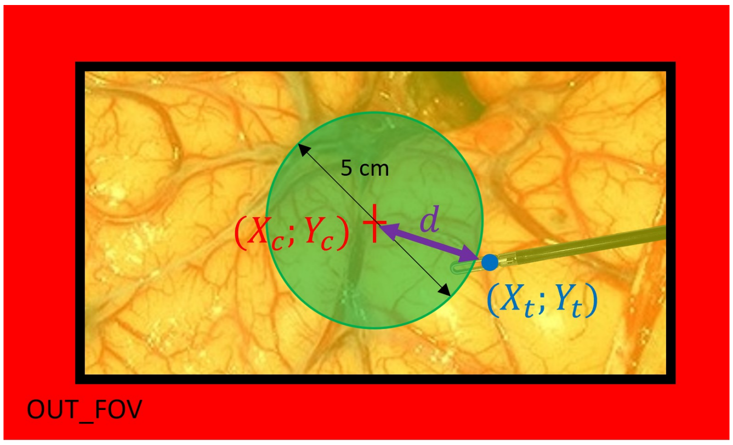

Figure 4.7: User Study Score Areas: in the green zone zero points are assigned, in the yellow zone (all the field of view except for the 5 cm circumference around the camera center) -1 point was assigned, in the red area, out of the field of view of the camera, -5 points are assigned. (X_t, Y_t) is the tool position, while (X_c, Y_c) is the camera center. d is the distance between (X_t, Y_t) and (X_c, Y_c).

So, a penalty of 5 points was assigned if the tool was outside the field of view of the camera, 1 point was subtracted if the tool was more far than 5 cm from (X_c, Y_c) and nothing happened if the tool was inside the 5 cm circumference centered in (X_c, Y_c).
The obtained score was then normalized with respect to the number of attempts needed by the user to find the rubber ring:

$$score\_attempts_r = \frac{score_r}{attempts_r} \tag{4.5}$$

where the subscript r indicates the specific repetition of a test. $score\_attempts_r$ is the score of a specific test normalized with respect to the number of attempts of that test, $attempts_r$. The score of the test before the normalization is represented by $score_r$.
Then, the different $score\_attempts_r$ were normalized again to obtain a value between 0 and 100, considering the worst score observed in the tests:

$$score\_norm_r = -100 \cdot \frac{score\_attempts_r - \min(\boldsymbol{score\_attempts})}{\min(\boldsymbol{score\_attempts})} \tag{4.6}$$

where $score\_norm_r$ is the score of the single tests used to analyze the performance of the control modes and $\boldsymbol{score\_attempts}$ is the vector containing all the scores normalized with respect to the attempts computed with equation 4.5. A score equal to zero represented the worst test, while a score equal to 100 indicated a perfect experiment.

For what concerns the duration and the path length measured during the tool motions, Figure 4.8 describes which movements were considered in the computations.

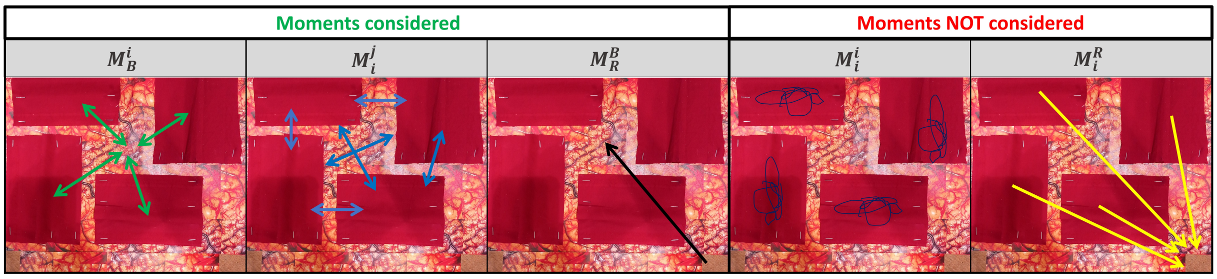

Figure 4.8: Movements considered and not during the evaluation of the duration and the length of the tool motions. The movements considered were: M_B^i, motions between the initial position and one of the i targets, M_i^j, motions between two different targets i and j, and M_R^B, movement from the release zone to the initial/final position. The movements not considered were M_i^i, representing the motions done on the same target while looking for the hidden ring, and M_i^R, which indicate the movements from the target under which the ring was found to the release zone.

The movements M_i^i were not considered since highly dependent on the user's agility in looking for the object and in grasping it, while motions M_i^R were not taken into account because too much dependent on the distance of the last target (the one under which the rubber ring was hidden) from the release zone.

The duration was thus computed as follows:

$$duration_r = \frac{t_r(M_B^i) + \sum t_r(M_i^j)}{attempts_r} + t_r(M_R^B) \tag{4.7}$$

where $duration_r$ is the duration corresponding to the r^{th} repetition of a test. $t_r(M)$ is the time needed to perform a specific movement M in the test identified by r. $attempts_r$ are the number of targets visited before discovering the ring.

Similarly, the path length was computed as shown below:

$$path\_length_r = \frac{l_r(M_B^i) + \sum l_r(M_i^j)}{attempts_r} + l_r(M_R^B) \tag{4.8}$$

where $path\_length_r$ is the path length of the test during the r^{th} repetition of a test. $l_r(M)$ is the length of the path that the tool has travelled during a movement M in the test identified by r.

The length of the path $l_r(M)$ in particular was measured as the distance between two consecutive positions of the tool:

$$l_r(M) = \sum_k \sqrt{\left(X_{R,t}(k) - X_{R,t}(k-1)\right)^2 + \left(Y_{R,t}(k) - Y_{R,t}(k-1)\right)^2} \tag{4.9}$$

in which the ks are the instants past during the execution of the movement M, the position of the tool in the right image at the k^{th} instant is represented by $(X_{R,t}(k), Y_{R,t}(k))$ and $(X_{R,t}(k-1), Y_{R,t}(k-1))$ is the position of the tool in the right image at the $(k-1)^{th}$ instant.

The NASA-TLX is a multi-dimensional scale designed to obtain workload estimates from one or more operators while they are performing a task or immediately afterwards [38]. It requires the user to give a score from 0 to 100 (with a step of 5) to six different aspects related to the task described in table 4.1.

Title	**Description**
Mental Demand	How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical Demand	How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal Demand	How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
Performance	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Table 4.1: NASA-TLX aspects analyzed.

For each camera control mode one NASA-TLX was compiled by the user just after the execution of the third repetition.

Finally, a questionnaire was proposed to investigate more deeply the users' preferences. It was compiled by users after the very last test and was composed of the questions and the possible answers listed below:

Question	Which camera movement modality facilitates the most the task accomplishment?
Possible answers	• Automatic translational • Automatic rotational • Manual • They are similar from this point of view
Question	Which movement modality presents the best field of view to perform the task?
Possible answers	• Automatic translational • Automatic rotational • Manual • They are similar from this point of view
Question	Which automatic modality is more suitable to accomplish the task?
Possible answers	• Automatic translational • Automatic rotational • They are similar from this point of view
Question	Is there a camera movement mode that makes you feel unsafe? (because of unexpected movements or other reasons)
Possible answers	• Automatic translational • Automatic rotational • Manual • No

Table 4.2: Questionnaire: questions and relative possible answers

5 | Results

In this chapter, the results obtained from the validation experiments and the User Study described in Chapter 4 are presented.

5.1. System Characterization

In the System Characterization phase the proposed tool tracking strategies were tested. After the best one was established and exploited for the fine-tuning of the controllers, the performance of two control modes (position and orientation) were evaluated. The results of all these tests are reported below.

5.1.1. Strategies Validation

The comparison between the two instrument tracking strategies developed by the Nearlab (CNN15 and CNN30) and the three new methods introduced by this thesis work (PARTICLE, OPFL and HYBRID) brought the results presented hereafter.
For what concerns the metrics analyzed, Figure 5.1 represent the boxplots of the Tracking Errors TE_{xy} of the different strategies divided by background (BG) and tool speed.

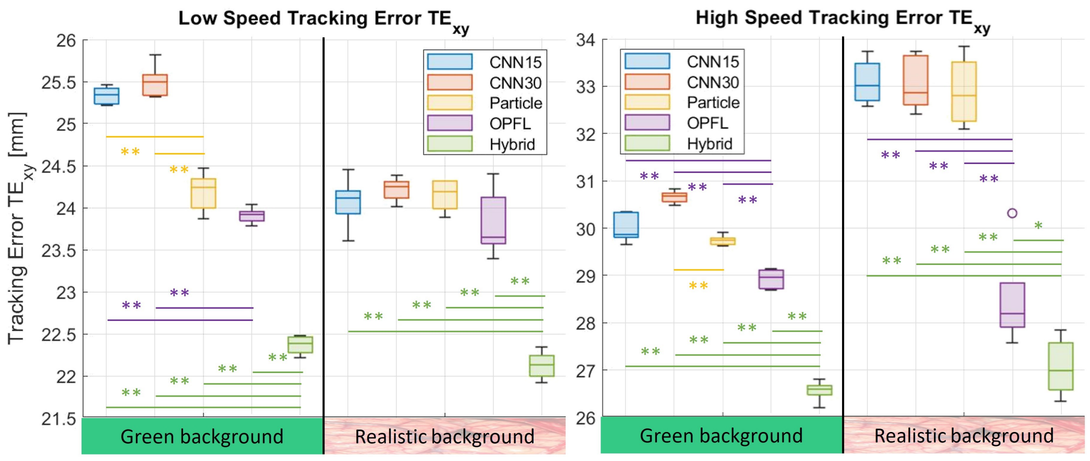

Figure 5.1: Boxplots of the TE_{xy} divided by speeds and backgrounds. (Wilcoxon signed-rank test: *, p < 0.05; **, p < 0.01; ***, p < 0.001).

In the boxplots above the HYBRID strategy is the one that presents always the lowest tracking errors. Also the OPFL has very good performance, but the absence of the advantages given by the Particle filter prediction is evident.
All the strategies were compared in pairs through a Wilcoxon signed-rank test for matched samples (with statistical significance established at a threshold of $p < 0.05$). The HYBRID strategy was always statistically different from all the other strategies, as can be noticed in the figure.
The TE_{xy} means and standard deviations of the different tests are reported in the following table:

$\boldsymbol{TE_{xy}}$	**Mean & Standard deviation [mm]**			
Strategy	**Low speed tests**		**High speed tests**	
	Green BG	Realistic BG	Green BG	Realistic BG
CNN15	25.33 ± 0.10	24.07 ± 0.27	30.01 ± 0.28	33.09 ± 0.43
CNN30	25.50 ± 0.18	24.22 ± 0.13	30.66 ± 0.12	33.06 ± 0.52
PARTICLE	24.18 ± 0.21	24.14 ± 0.17	29.74 ± 0.09	32.89 ± 0.65
OPFL	23.91 ± 0.08	23.82 ± 0.35	28.92 ± 0.18	28.49 ± 0.95
HYBRID	$\mathbf{22.36 \pm 0.10}$	$\mathbf{22.12 \pm 0.15}$	$\mathbf{26.59 \pm 0.20}$	$\mathbf{27.06 \pm 0.55}$

Table 5.1: Strategies validation tests: TE_{xy} means and standard deviations.

The second performance index analyzed was the Center Error CE_{xy}. The boxplots of the tests are reported in Figure 5.2.

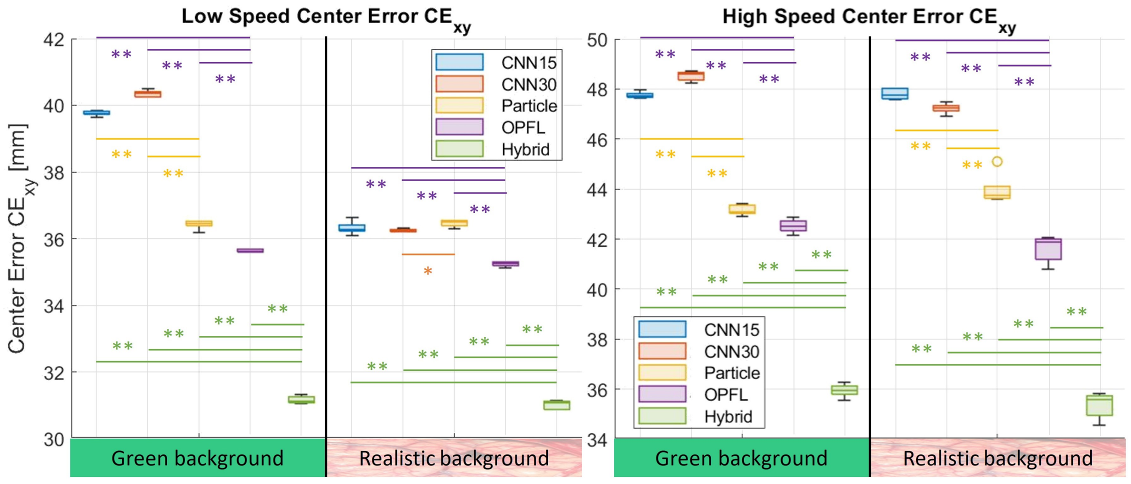

Figure 5.2: Boxplots of the CE_{xy} divided by speeds and backgrounds. (Wilcoxon signed-rank test: *, p < 0.05; **, p < 0.01; ***, p < 0.001).

Also in the boxplots of the Center Error, the HYBRID strategy is the one that presents always the lowest errors. In general, the relationships between the various strategies are similar considering the two performance indexes.
The strategies were compared again in pairs through a Wilcoxon signed-rank test for matched samples. The OPFL and the HYBRID strategy were always statistically different from all the other strategies and between themselves, as can be noticed in the figures.
The CE_{xy} means and standard deviations of the different tests are reported in the following table:

$\boldsymbol{CE_{xy}}$	**Mean & Standard deviation [mm]**			
Strategy	**Low speed tests**		**High speed tests**	
	Green BG	Realistic BG	Green BG	Realistic BG
CNN15	39.76 ± 0.07	36.32 ± 0.18	47.75 ± 0.11	47.80 ± 0.20
CNN30	40.35 ± 0.09	36.24 ± 0.04	48.52 ± 0.17	47.23 ± 0.19
PARTICLE	36.42 ± 0.12	36.46 ± 0.10	43.16 ± 0.19	43.97 ± 0.57
OPFL	35.63 ± 0.04	35.24 ± 0.07	42.52 ± 0.24	41.60 ± 0.48
HYBRID	$\mathbf{31.16 \pm 0.10}$	$\mathbf{31.01 \pm 0.11}$	$\mathbf{35.94 \pm 0.24}$	$\mathbf{35.35 \pm 0.47}$

Table 5.2: Strategies validation tests: CE_{xy} means and standard deviations.

When the tool moves at high speed, instead, the strategies that estimate the position of the tool mainly with the CNN (CNN15, CNN30 and PARTICLE) failed. This can be noticed in particular in Figures 5.3 and 5.4 where it is evident that these strategies, around the second 12 of the tests, presented peaks representing instabilities. Indeed, in these cases, because of the low computational speed of these methods, the tool exited from the field of view of the camera and the robot started to diverge. If the reachable workspace of the camera holder was not limited for safety reasons, the peaks would have been larger. These behaviours were present in both the backgrounds scenarios, but they were more evident with the realistic background, since in this case it was more difficult for the CNN to detect the tool.

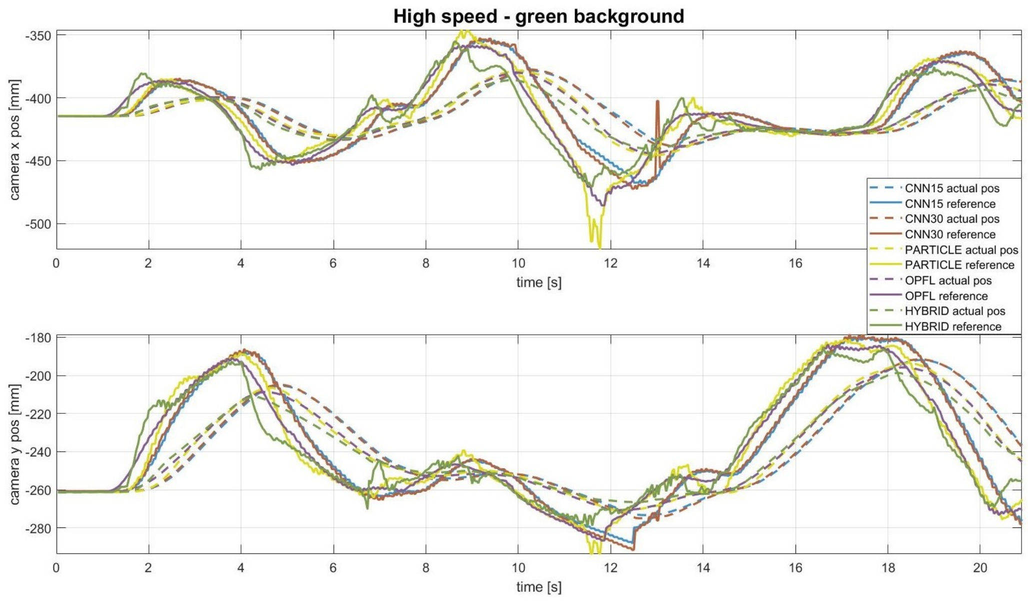

Figure 5.3: Strategies validation tests with high speed (4 cm/s) and green background.

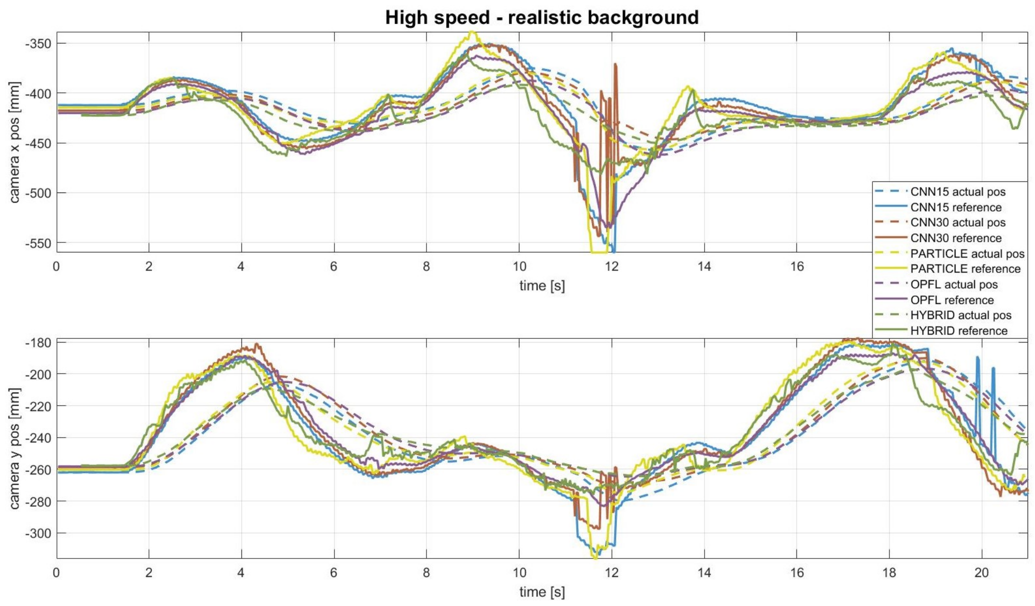

Figure 5.4: Strategies validation tests with high speed (4 cm/s) and realistic background.

5.1.2. Control Module Validation

The position and the orientation controllers were then fine-tuned exploiting the HYBRID strategy and the results of this process are reported in the following Sections.

Position Control

The fine-tuning of the controllers for the Position control mode resulted in the choice of the gains reported in the second column of table 3.5. In position control mode, exploiting the HYBRID strategy and the tuned gains, the same experiments with the two robots were carried out.
In Figure 5.5 are compared the plots of the test with the realistic background and the tool moving at high speed before and after the fine-tuning.

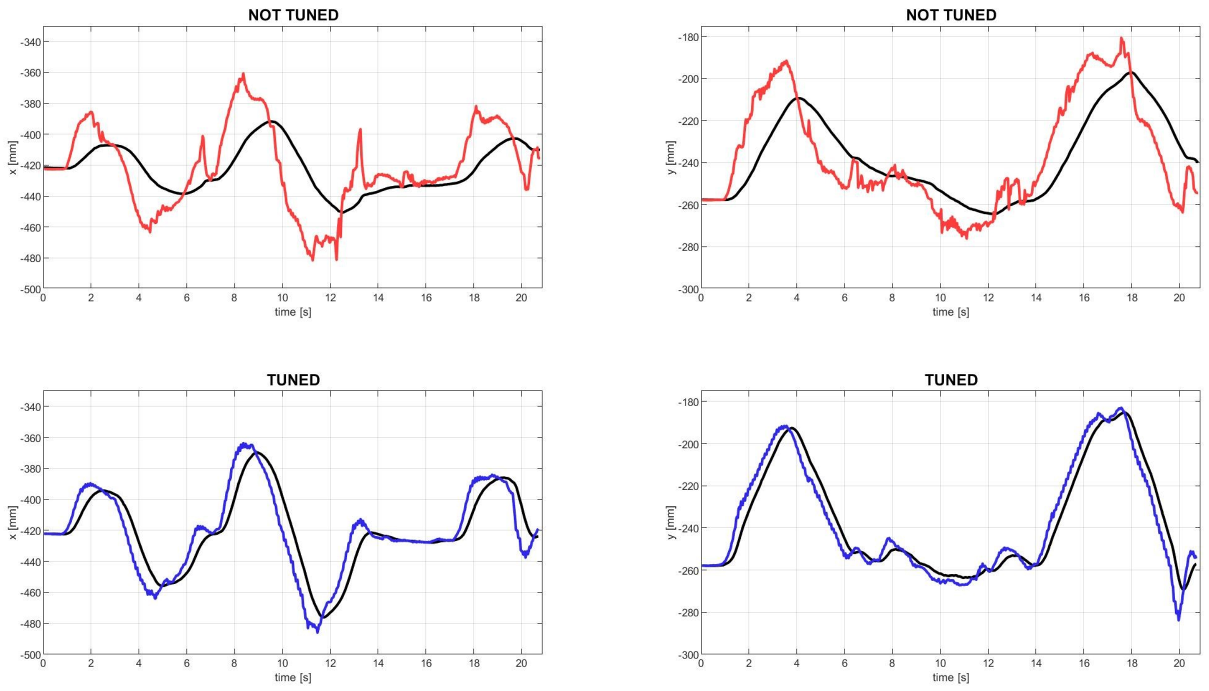

Figure 5.5: Position control validation tests with high speed (4 cm/s) and realistic background before (above) and after (below) tuning. On the left-hand side there are the plots of the x coordinate, while on the right-hand side the plots of the y coordinate are shown. The colored lines represent the actual motions while the black ones the reference trajectory.

The tables below report a comparative analysis of the two evaluated metrics before and after the fine-tuning of the controller.

$\boldsymbol{TE_{xy}}$	**Mean & Standard deviation [mm]**			
Controller	**Low speed tests**		**High speed tests**	
	Green BG	Realistic BG	Green BG	Realistic BG
NOT TUNED	22.36 ± 0.10	22.12 ± 0.15	26.59 ± 0.20	27.06 ± 0.55
TUNED	$\mathbf{10.00 \pm 0.15}$	$\mathbf{9.84 \pm 0.08}$	$\mathbf{12.92 \pm 0.31}$	$\mathbf{13.11 \pm 0.39}$

Table 5.3: Position control validation tests: TE_{xy} means and standard deviations.

$\boldsymbol{CE_{xy}}$	**Mean & Standard deviation [mm]**			
Strategy	**Low speed tests**		**High speed tests**	
	Green BG	Realistic BG	Green BG	Realistic BG
NOT TUNED	31.16 ± 0.10	31.01 ± 0.11	35.94 ± 0.24	35.35 ± 0.47
TUNED	$\mathbf{16.35 \pm 0.15}$	$\mathbf{16.14 \pm 0.14}$	$\mathbf{20.96 \pm 0.19}$	$\mathbf{20.80 \pm 0.16}$

Table 5.4: Position control validation tests: CE_{xy} means and standard deviations.

Tables 5.3 and 5.4 highlight how much the two metrics were reduced after the fine-tuning.

Orientation Control Validation

In the third column of table 3.5 are reported the gains deriving from the fine-tuning of the Orientation controller.
The validation of the Orientation control mode was performed evaluating the performance indexes described in Chapter 4.1.2.
The results of this validation phase are shown in the following table:

	Mean & Standard deviation			
Metric	**Low speed tests**		**High speed tests**	
	Green BG	Realistic BG	Green BG	Realistic BG
$\boldsymbol{TE_{roll}}$ [deg]	4.56 ± 0.03	4.29 ± 0.06	6.00 ± 0.07	5.68 ± 0.08
$\boldsymbol{TE_{pitch}}$ [deg]	4.79 ± 0.03	4.63 ± 0.08	5.78 ± 0.07	5.65 ± 0.08
$\boldsymbol{CE_{xy}}$ [mm]	24.56 ± 0.18	22.41 ± 0.39	30.05 ± 0.31	27.62 ± 0.43

Table 5.5: Orientation control validation tests: means and standard deviations of the various performance indexes.

5.2. User Study

In the overall system validation, 12 users were requested to perform a bimanual task with the system controlled in three different camera control modes and to compile a NASA-TLX and a custom questionnaire about their experiments. During the tests, three performance metrics (duration, score, and path length) were recorded. Since the user had to repeat each experiment three times, the first analysis of the data was aimed at investigating the eventual presence of a learning curve between repetitions.
In Figure 5.6, 5.7, and 5.8 are reported the various performance indexes divided by repetitions and camera control mode.

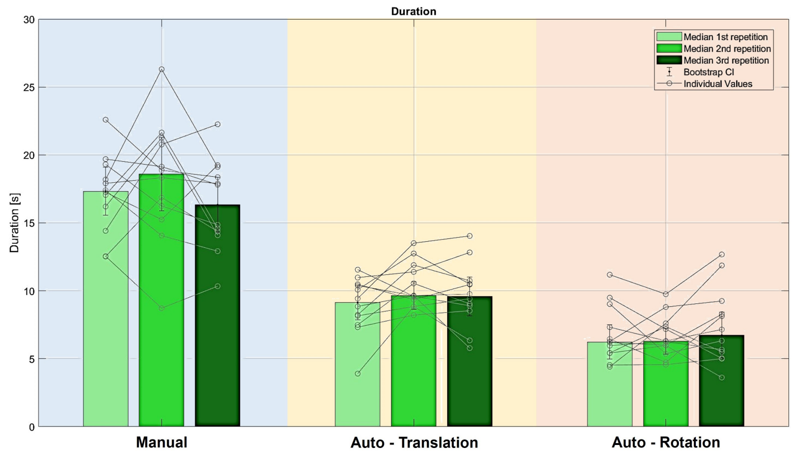

Figure 5.6: Barplots of the Duration performance index divided by repetitions and camera control modes.

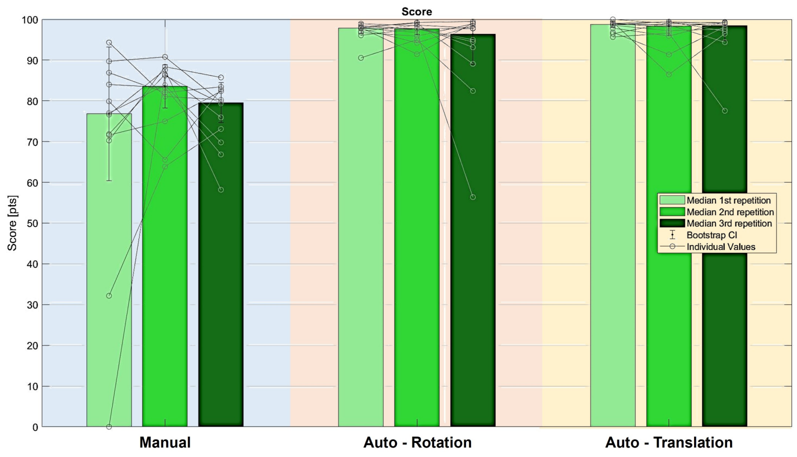

Figure 5.7: Barplots of the Score performance index divided by repetitions and camera control modes.

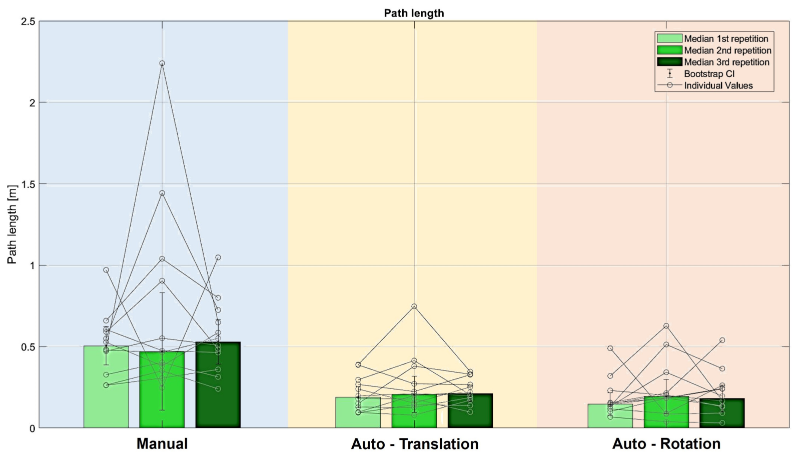

Figure 5.8: Barplots of the Path length performance index divided by repetitions and camera control modes.

As can be noticed from the plots above, there was not a learning curve between repetitions and so, the average of the performance indexes of the three repetitions of the tests executed by the same user with the same camera control mode, was computed.
Since the sample size is small, a normality test would not be reliable and thus a non-parametric test was considered to analyze data. The test chosen to compare the performance of the three camera control modes is the Kruskal - Wallis test (with statistical significance established at a threshold of $p < 0.05$). If a statistical significance was present a post-hoc Dunn-Sidak test was then performed to discover which couples of strategies were statistically different. From these tests and the post-hoc Dunn-Sidak tests, it can be noticed that, for all the performance indexes analyzed, the automatic control modes were always statistically different from the manual mode, as reported in Figure 5.9. Between the two automatic modes, instead, there was not a statistical difference.

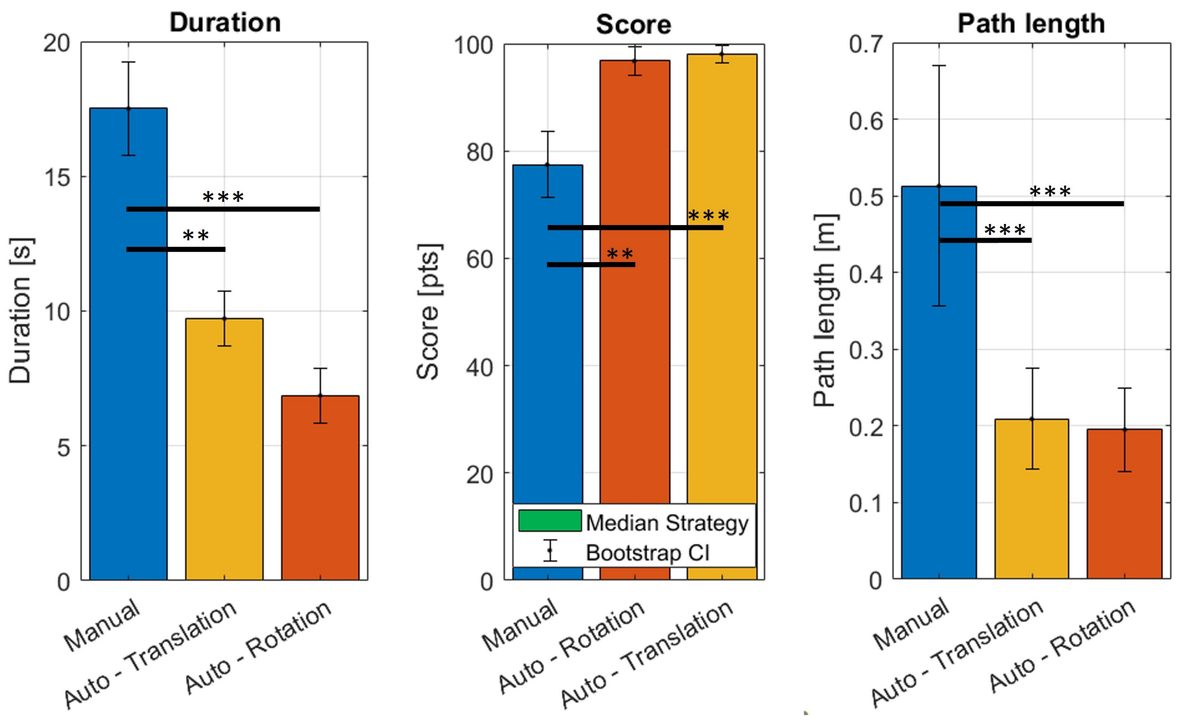

Figure 5.9: Barplots of the three performance indexes divided by camera control strategy. (Kruskal- Wallis test and Dunn-Sidak test: *, $p < 0.05$; **, $p < 0.01$; ***, $p < 0.001$).

In table 5.6 the medians and the standard deviations of the data reported in the figure above are shown. It can be noticed that the Automatic - Translational control mode presents the highest score while Automatic - Rotational modality registered the lowest duration and the lowest path length.

	Median & Standard deviation		
Camera control mode	**Duration** [s]	**Score** [pts]	**Path length** [m]
Manual	17.51 ± 2.71	77.46 ± 9.69	0.51 ± 0.25
Automatic - Translational	9.72 ± 1.58	$\mathbf{98.06 \pm 2.61}$	0.21 ± 0.10
Automatic - Rotational	$\mathbf{6.87 \pm 1.62}$	96.76 ± 4.17	$\mathbf{0.20 \pm 0.08}$

Table 5.6: Orientation control validation tests: means and standard deviations of the various performance indexes.

The results of the NASA-TLX survey are reported in Figure 5.10.

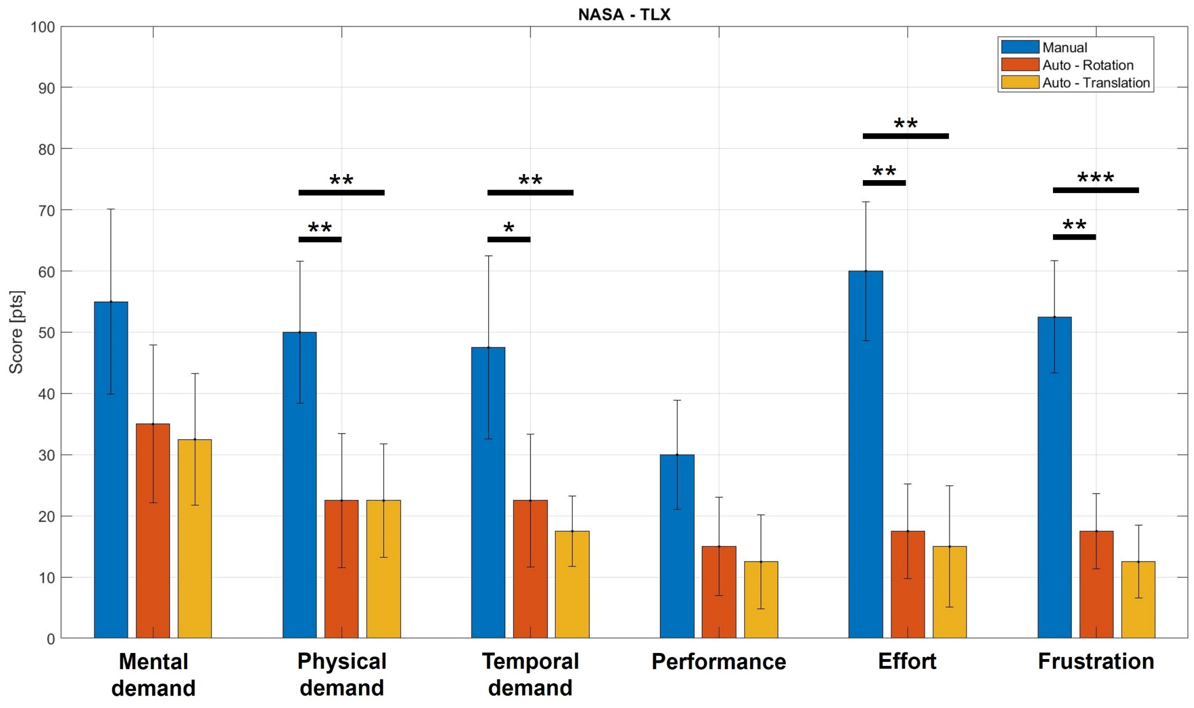

Figure 5.10: Barplots with the results of the NASA-TLX survey. (Kruskal- Wallis test and Dunn-Sidak test: *, p < 0.05; **, p < 0.01; ***, p < 0.001).

Kruskal- Wallis and Dunn-Sidak tests were performed also on the scores corresponding to the NASA-TLX answers. These tests highlighted a statistical difference between the automatic modalities and the manual one in Physical Demand, Temporal Demand, Effort, and Frustration categories. Even if not statistically different, also the scores of the Mental Demand and Performance fields resulted better with the automatic strategies.
For what concerns the questionnaire, the answers are reported in Figure 5.11.

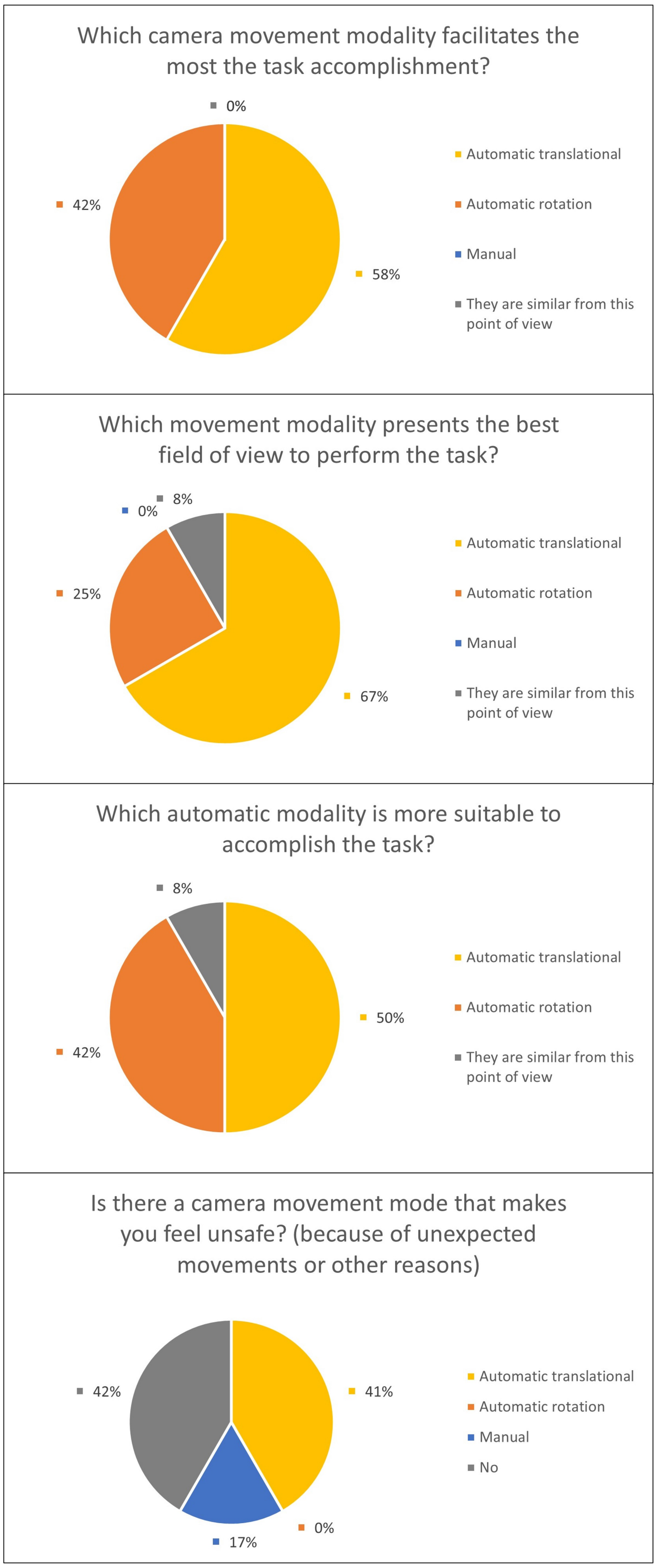

Figure 5.11: Answers to the User Study questionnaire.

6 | Discussion

The results reported in the previous Chapter are discussed below.

For what concerns the Strategies Validation, the proposed methods demonstrated to be more reactive and accurate in tracking the surgical tool with respect to the pre-existing ones (CNN15 and CNN30). In particular the employment of the Optical flow tracking module allowed to reduce considerably the computational load, resulting in an increased processing speed that in turn permitted to send to the robot a reference position more frequently. This explains why the OPFL and HYBRID strategies performed better than the other ones. The PARTICLE strategy, even if showed better performances with respect to CNN15 and CNN30 strategies, was affected by the low processing speed of the CNN and this resulted in larger Tracking and Center Errors than OPFL and HYBRID methods. Moreover, when the Optical flow-based strategies (OPFL and HYBRID) were employed, no instabilities occurred, demonstrating that these methods are more robust than CNN-based ones against different scenarios. In general, the HYBRID strategy showed to be the most suitable for the tool tracking task compared to the other methods, indeed the combined action of the Optical flow and of the Particle Filter predictor allowed to obtain the lowest Tracking Error and the lowest Center Error and to follow the tool in a fast, robust and accurate way,

The fine-tuning of the controllers proved to be effective. As a matter of fact, by comparing the Tracking Errors and the Center Errors before and after the tuning, reported in tables 5.3 and 5.4, and by looking at Figure 5.5, the improvements are evident. The Tracking Error was more than halved in all the scenarios, while the Center Error after the tuning was approximately 40% lower than before. The performance indexes result satisfactory for the Position control mode ($\approx$ 1 cm for the Tracking Error and $\approx$ 1.8 cm for the Center Error) and acceptable for the Orientation control mode ($\approx 5^\circ$ for the Tracking Error on the roll and pitch angles and $\approx$ 2.6 cm for the Center Error). Therefore, after the tuning, the camera moved faster and in a more accurate way than before.

Finally, the User Study demonstrated the preeminence of the Automatic camera control modalities with respect to the traditional way to move the camera. Indeed, the automatic translational and the automatic rotational mode obtained the best values from the performance indexes point of view, resulting always statistically different from the manual mode. In particular, automatic modes allowed to obtain the lowest values of the "duration" metric meaning that if these strategies are used, surgical operation could last less, aspect that is fundamental when the workload and the fatigue of the surgeon are considered. Moreover, the automatic control modalities, permitted to achieve the best results in the "score" field, showing that these modes allow to keep the tool at the image center more effectively with respect to the manual control and so, to reduce the risk of undesired contacts of the surgical instruments with the surrounding tissues while they are out of the camera field of view. Finally, automatic strategies showed the lowest values also for the "path length" index, indicating that they require smaller movements of the tool to track. These results were confirmed also by the users perceptions collected in the NASA-TLX, that also underlined other advantages of the employment of the automatic control modalities. The automatic modes (in particular the translational one) requested the lowest physical and temporal demands that is very important if the real scenario of a surgical operation is considered. Low physical demand means that the surgical operations, that usually last many hours, will be less tiring and thank to the lower temporal demand, they could also be shorter. Moreover, in the NASA-TLX the automatic modes resulted to be less effort-requesting and less frustrating to use, that are other very important aspects. Finally the questionnaire highlighted that users think that the automatic strategies are more effective than the manual mode in facilitating the task accomplishment. In the users' opinion the automatic translational control mode offers the best field of view and between the two automatic strategies there is not a significant difference in terms of suitability of their employment in the task to execute. The last question of the survey revealed that some user felt unsafe with the manual mode (17%) and with the automatic translational control mode (41%) because of unexpected movements or other reasons, while the remaining felt safe (42%) with the system. The fact that some user felt unsafe with the automatic translational camera control mode can be explained by the nature of this strategy. Indeed this method moves automatically the robot on larger distances with respect to the automatic rotational control modality and, if the surgical tool to follow is close to the user, the robot moves next to the user (with all the safety measures that collaborative robots are equipped with) and this can make people feel unsafe.

7 | Conclusions

7.1. Conclusions

Exoscopes represent a promising visual solution in the neurosurgical field aimed at offering an improved field of view and ergonomics to the surgeons compared with traditional surgical microscopes. However, the need for manual repositioning the exoscope, each time a different point of view of the scene is required, may threaten to diminish the advantages brought: indeed the manual movement of the exoscope causes the continuous switching between operation theatre and visualization system leading to interruptions that compromise the smoothness of the surgical procedure and that can possibly lead to longer operation times.

The researchers of the Nearlab laboratory developed a solution to the aforementioned problem that consists in fastening an exoscope on the end effector of a redundant robotic manipulator and using the images coming from the camera to detect a specific surgical tool and to follow it. This visual servoing-based system worked but was impacted by the low processing speed of the CNN used to recognize the instrument.

In this thesis work, a novel hybrid tracking module is proposed to solve the issues caused by the use of the CNN and to allow the system to achieve real-time tool tracking. The hybrid tracking module is composed of an Optical flow tracking module that substitutes the CNN whenever possible and a Particle Filter predictor designed to estimate the future position of the tracked instrument on the basis of the previous movements. Both the components of that module were firstly validated singularly and then together to demonstrate that the proposed strategies improve the system performance and that the combined action of the Optical flow and of the Particle Filter allows to obtain the best tool tracking. Once the validation of the strategies was performed, the Position controller and the Orientation controller were fine-tuned to make the system faster and more accurate in tracking the surgical instrument. This process was effective and enabled the autonomous camera system to follow the tool in real time. Finally, a User Study was organized to investigate whether the proposed system was effective in reducing the users' workload during the execution of a task compared to the traditional exoscope control mode (manual control). From the User Study emerged that the automatic control modes developed to move the camera (translational and rotational) allow to reduce the duration of the task and the tool path length needed to accomplish it, and they permit keeping the instrument at the center of the image better than the manual control. Moreover, from the responses collected for the NASA-TLX survey, it can be concluded that users perceived the task completed by exploiting the automatic control as less physically and temporally demanding, stressful, and in need of effort. Finally, a questionnaire highlighted that automatic camera control modalities, with respect to manual control, facilitated task accomplishment more and offered the best field of view during the test execution in the user's opinion.

7.2. Limitations and further research

The hybrid tracking module proposed in this thesis work demonstrated potential for improving the performances of the pre-existing system by increasing the functioning speed and reducing the tracking errors during its employment. It could facilitate the adoption of autonomous exoscopes in neurosurgery, but, in its present form, a series of limitations are still present:

- The data set used for training is not representative of a clinical scenario, so to translate the system into a real application a more robust training would be required. This should be done using a data set composed of real clinical images comprising also the specific artifacts present in neurosurgery.
- The system in general has not been tested on a realistic scenario. To solve this, it should be tested by surgeons in a scenario that comprise the use of a real exoscope equipped with different zoom capabilities, the variation of the lighting conditions, the employment of multiple tools and of a workspace that better mimic the one in which the surgeon has to actually operate.
- To decide when is right to move the system in automatic modes, it is now necessary that the user press a pedal. Without this pedal, because of occlusions or movements of the surgical tool outside the workspace of the camera holder (and so, out of the field of view of the camera), the system could diverge and instabilities could occur since the tool detection and tracking module fail.
- The system is not able, at the moment, to avoid obstacles nor to establish which is the best trajectory to follow to not hinder the surgical operation.

In future research, therefore, the system should be equipped with an intelligent control system that decides autonomously when is the right moment to make the camera holder move. The autonomous exoscope should be also empowered with a trajectory planner to establish how to move every single link of the redundant robot to avoid collisions with the environment or hinder to the surgeons, maybe exploiting the Artificial Potentials method.

Bibliography

[1] S. Muhammad, Martin Lehecka, and Mika Niemelä. Preliminary experience with a digital robotic exoscope in cranial and spinal surgery: a review of the synaptive modus v system. *Acta Neurochirurgica*, 161:2175 – 2180, 2019.

[2] Adam N. Mamelak, Doniel Drazin, Ali Shirzadi, Keith L. Black, and George Berci. Infratentorial supracerebellar resection of a pineal tumor using a high definition video exoscope (vitom®). *Journal of Clinical Neuroscience*, 19(2):306–309, 2012.

[3] Zefferino Rossini, Andrea Cardia, Davide Milani, Giovanni Battista Lasio, Maurizio Fornari, and Vincenzo D'Angelo. Vitom 3d: Preliminary experience in cranial surgery. *World Neurosurgery*, 107:663–668, 2017.

[4] Judith Rösler, Stefan Georgiev, Anna L. Roethe, Denny Chakkalakal, Güliz Acker, Nora F. Dengler, Vincent Prinz, Nils Hecht, Katharina Faust, Ulf Schneider, Simon Bayerl, Marcus Czabanka, Martin Misch, Julia Onken, Peter Vajkoczy, and Thomas Picht. Clinical implementation of a 3d4k-exoscope (orbeye) in microneurosurgery. *Neurosurgical Review*, 45:627–635, 2022.

[5] Arun Sivananthan, Alexandros Kogkas, Ben Glover, Ara Darzi, George Mylonas, and Nisha Patel. A novel gaze-controlled flexible robotized endoscope; preliminary trial and report. *Surgical Endoscopy*, 35:1–10, 08 2021.

[6] Masahiro Takahashi, Masanori Takahashi, N. Nishinari, H. Matsuya, T. Tosha, Y. Minagawa, O. Shimooki, and Tadashi Abe. Clinical evaluation of complete solo surgery with the "viky®" robotic laparoscope manipulator. *Surgical Endoscopy*, 31:981–986, 2016.

[7] Anto Abramovic, Matthias Demetz, Aleksandrs Krigers, Marlies Bauer, Sara Lener, Daniel Pinggera, Johannes Kerschbaumer, Sebastian Hartmann, Helga Fritsch, Claudius Thomé, and Christian F. Freyschlag. Surgeon's comfort: The ergonomics of a robotic exoscope using a head-mounted display. *Brain and Spine*, 2:100855, 2022.

[8] Elisa Iovene, Alessandro Casella, Alice Valeria Iordache, Junling Fu, Federico Pessina, Marco Riva, Giancarlo Ferrigno, and Elena De Momi. Towards exoscope automa-

tion in neurosurgery: A markerless visual-servoing approach. *IEEE Transactions on Medical Robotics and Bionics*, 5, 05 2023.

[9] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276, 2023.

[10] David Bouget, Max Allan, Danail Stoyanov, and Pierre Jannin. Vision-based and marker-less surgical tool detection and tracking: a review of the literature. *Medical Image Analysis*, 35:633–654, 2017.

[11] Tommaso Da Col, Andrea Mariani, Anton Deguet, Arianna Menciassi, Peter Kazanzides, and Elena De Momi. Scan: System for camera autonomous navigation in robotic-assisted surgery. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2996–3002, 2020.

[12] Tommaso Da Col, Guido Caccianiga, Michele Catellani, Andrea Mariani, Matteo Ferro, Giovanni Cordima, Elena De Momi, Giancarlo Ferrigno, and Ottavio de Cobelli. Automating endoscope motion in robotic surgery: A usability study on da vinci-assisted ex vivo neobladder reconstruction. *Frontiers in Robotics and AI*, 8, 2021.

[13] Vipul Sharma and Roohie Naaz Mir. A comprehensive and systematic look up into deep learning based object detection techniques: A review. *Computer Science Review*, 38:100301, 2020.

[14] Ahmad Ali, Abdul Jalil, Jianwei Niu, Xiaoke Zhao, Saima Rathore, Javed Ahmed, and Muhammad Aksam Iftikhar. Visual object tracking – classical and contemporary approaches. *Frontiers of Computer Science*, 10(1):167–188, February 2016.

[15] Litong Fan, Zhongli Wang, Baigen Cail, Chuanqi Tao, Zhiyi Zhang, Yinling Wang, Shanwen Li, Fengtian Huang, Shuangfu Fu, and Feng Zhang. A survey on multiple object tracking algorithm. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 1855–1862, 2016.

[16] Jingxuan Hao, Yimin Zhou, Guoshan Zhang, Qin Lv, and Qingtian Wu. A review of target tracking algorithm based on uav. In *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pages 328–333, 2018.

[17] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics - Modelling, Planning and Control*, pages 447–448. Springer-Verlag London Limited, 2009.

[18] Ching-Chang Wong, Chi-Yi Tsai, Ren-Jie Chen, Shao-Yu Chien, Yi-He Yang, Shang-

Wen Wong, and Chun-An Yeh. Generic development of bin pick-and-place system based on robot operating system. *IEEE Access*, 10:65257–65270, 2022.

[19] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger. Off-the-shelf vision for a robotic ball catcher. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 3, pages 1623–1629 vol.3, 2001.

[20] Akshay Kumar. An overview of visual servoing for robot manipulators. Technical report, www.control.com, 2020.

[21] Juho Kannala, Janne Heikkila, and Sami S. Brandt. Geometric camera calibration. In Benjamin W. Wah, editor, *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–20. John Wiley & Sons, Inc., 2008.

[22] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[23] Radu Horaud and Fadi Dornaika. Hand-eye calibration. *I. J. Robotic Res.*, 14:195–210, 06 1995.

[24] Ultralytics. Yolov5 by ultralytics. `https://github.com/ultralytics/yolov5`. Accessed on January 1, 2021.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[26] Max Allan, Alex Shvets, Thomas Kurmann, Zichen Zhang, Rahul Duggal, Yun-Hsuan Su, Nicola Rieke, Iro Laina, Niveditha Kalavakonda, Sebastian Bodenstedt, Luis Herrera, Wenqi Li, Vladimir Iglovikov, Huoling Luo, Jian Yang, Danail Stoyanov, Lena Maier-Hein, Stefanie Speidel, and Mahdi Azizian. 2017 robotic instrument segmentation challenge, 2019.

[27] P.M. Djuric, J.H. Kotecha, Jianqui Zhang, Yufei Huang, T. Ghirmai, M.F. Bugallo, and J. Miguez. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, 2003.

[28] Arnaud Doucet, S.J. Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10, 04 2003.

[29] `https://it.mathworks.com/help/robotics/ug/particle-filter-workflow.html`.

[30] Tiancheng Li, Miodrag Bolic, and Petar M. Djuric. Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3):70–86, 2015.

[31] Mohammad O. A. Aqel, Mohammad Hamiruce Marhaban, M. Iqbal Saripan, and Napsiah Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5, 2016.

[32] `https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html`.

[33] John Barron, David Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 02 1994.

[34] Gary Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[35] Jacob Rosen, Jeffrey D Brown, Michael Barreca, Sherif Hanafy, and Blake Hannaford. Performance characteristics of the 'da vinci' surgical manipulator. In *IEEE International Conference on Robotics and Automation*, pages 612–617. IEEE, 2000.

[36] J. Sandoval, H. Su, P. Vieyres, G. Poisson, G. Ferrigno, and E. De Momi. Collaborative framework for robot-assisted minimally invasive surgery using a 7-dof anthropomorphic robot. *Robotics and Autonomous Systems*, 106:95–106, 2018.

[37] C. A. Klein and C. H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:245–250, 1983.

[38] S. G. Hart. Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9):904–908, 2006.

[39] J. Gedge, M. Gong, and Y.-H. Yang. Refractive epipolar geometry for underwater stereo matching. In *Canadian Conference on Computer and Robot Vision*, pages 146–152, 2011.

A | Epipolar Geometry

Epipolar geometry [39] is a fundamental concept in computer vision and stereo imaging that describes the relationship between corresponding points in two images taken from different viewpoints. It provides a geometric framework for understanding the 3D structure of a scene and the relative pose of cameras observing that scene.

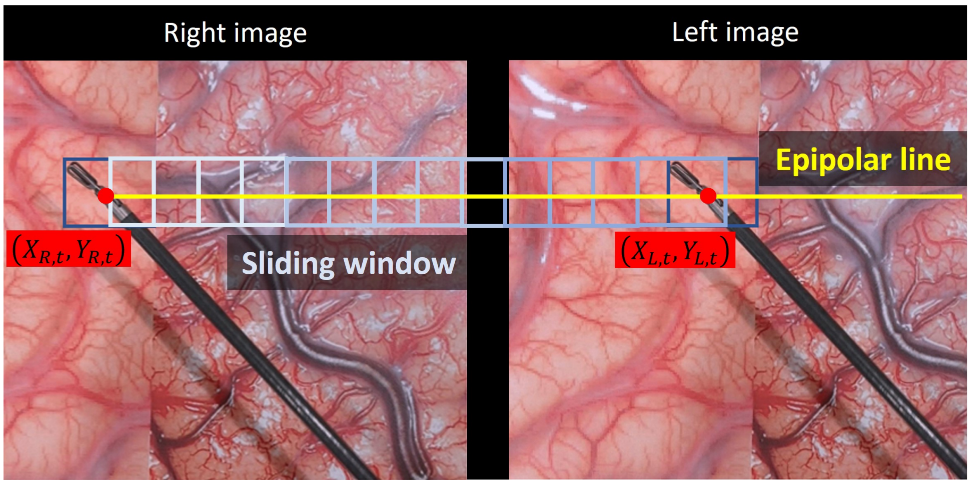

Figure A.1: After that the CNN has estimated the tool position in the right image $\boldsymbol{P}_{R,t} = (X_{R,t}, Y_{R,t})$, The epipolar line is then found and a window is slid along it to find the corresponding point in the left image, $\boldsymbol{P}_{L,t} = (X_{L,t}, Y_{L,t})$.

Considering F (the fundamental matrix computed during the camera calibration phase), $x = \begin{bmatrix} X_{L,t} & Y_{L,t} \end{bmatrix}^T$, and $x' = \begin{bmatrix} X_{R,t} & Y_{R,t} \end{bmatrix}^T$, then:

$$x^T F x' = 0 \tag{A.1}$$

is the relation between corresponding points ($\boldsymbol{P}_{R,t}$ and $\boldsymbol{P}_{L,t}$) in the two images.
The epipolar line l is found as follows:

$$l = F x' \tag{A.2}$$

The window size is defined on the basis of the right image bounding box dimension. Exploiting the OpenCV package "cv2", the cross-correlation between the contents of the bounding box and of the sliding window is computed and, finally, the tool position in the left image is estimated as shown in Figure A.1.

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
BG	Background
CNN	Convolutional Neural Network
CNN15	CNN-based strategy with publication frequency equal to 15 Hz
CNN30	CNN-based strategy with publication frequency equal to 30 Hz
COCO	Common Objects in Context
COM	Center of Mass
DoF	Degrees of Freedom
EE	End Effector
FOV	Field of View
FP	False Positive
FPS	frames per second
GPU	Graphical Processing Unit
HD	High Definition
ICR	Instantaneous Center of Rotation
IoU	Intersection over Union
NN	Neural Network
OPFL	Optical Flow tracking strategy
RCM	Remote Center of Motion
SVD	Singular Value Decomposition
TLX	Task Load Index
TP	True Positive
YOLO	You Only Look Once

List of Figures

List of Tables

Acknowledgements

Concludendo questo tortuoso percorso caratterizzato da qualche intoppo ma anche pieno di soddisfazioni, vorrei ringraziare tutti coloro che mi hanno supportato e mi hanno dato la forza per portarlo a termine.
In primo luogo, vorrei ringraziare la mia Famiglia e Tiziana, che mi hanno dimostrato ogni giorno il loro affetto, credendo in me, standomi sempre vicini e cercando di rendere più semplici, felici e spensierati questi anni.
Vorrei poi ringraziare la Professoressa Elena De Momi che mi ha accolto nel suo gruppo e mi ha dato consigli utilissimi per la stesura di questa tesi. Vorrei esprimere poi la mia gratitudine a Elisa Iovene per il costante e preziosissimo aiuto durante tutte le fasi di realizzazione di questa tesi. Inoltre, vorrei ringraziare tutti i dottorandi e gli studenti del Nearlab, in particolare Junling Fu, che sono stati sempre molto disponibili nei momenti di difficoltà e che mi hanno fatto apprezzare ancora di più questo ultimo anno, grazie ad attimi di spensieratezza e numerose pause caffè.
Vorrei ringraziare anche i miei amici e i miei compagni di università che mi hanno accompagnato durante questo percorso, allietandolo. In particolare, vorrei menzionare Fabio, per tutti i lavori di gruppo fatti insieme e per il reciproco supporto che ci scambiati.
Infine, vorrei ringraziare ancora tutti coloro che in questi anni mi hanno sopportato e che mi sono stati accanto nonostante tutti i "Mi dispiace ma non posso, devo studiare" e gli sbalzi d'umore derivanti da esami o progetti.
Grazie, senza il vostro supporto forse non sarei riuscito a concludere questo percorso. Spero di riuscire a restituirvi ciò che voi avete dato a me.