A first step to enhance the robot-patient interaction in a Robot Assisted Ultrasound system

Master thesis

Laia Vives Benedicto

Aalborg University
Robotics

Robotics
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
A first step to enhance the robot-patient interaction in a Robot Assisted Ultrasound system

Theme:
Master Thesis

Project Period:
Spring Semester 2024

Project Group:
1056

Participant(s):
Laia Vives Benedicto

Supervisor(s):
Thomas B. Moeslund
Galadrielle Humblot-Renaux

Page Numbers: 78

Date of Completion:
May 31, 2024

Abstract:

The physical discomfort when manoeuvring the transducer in an ultrasound scan device brings Hera, a Robot Assisted Ultrasound system developed by Life Science Robotics to aid in obstetric scans. It consists of a robot arm that holds the transducer while controlled by the sonographer via a joystick. No published approach considered the other side of the human-robot interaction, the robot-patient interaction to enhance the patient's comfort. The project aims to provide a starting point to detect patient movement and provide a robot-patient interaction. The proposed approach consists of developing and testing a Kernelized Correlation Filter that keeps tracking the patient's abdomen under the presence of the robot in view by correcting the tracking box with feature-matching techniques. The approach is on average 9% more accurate than the default tracker when occlusion is present on the frame. It can also re-identify the patient in long tracking sequences after losing the target due to a big occlusion, reaching an accuracy of 84.5% compared to the default 69,4%. Scanning the upper part of the abdomen fails in detecting points in the region of interest. Moreover, textureless clothing leads to wrong matches since the points are detected at the skin-clothes boundary. In spite, the approach can potentially track the abdomen to determine the real-time position of the patient and enhance the interaction with the robot, but further research is needed to detect more robust features and handle occluded areas.

Summary

Ultrasound scans have been used since 1940 to observe internal organs. However, the repetitive movements to manoeuvre the transducer while applying force lead to work-related musculoskeletal disorders. To relieve the physical discomfort, Life Science Robotics developed Hera, a Robot Assisted Ultrasound system for obstetric scans in which a robot arm holds the transducer, controlled by the sonographer while sitting in a comfortable position. Despite robot-assisted systems for ultrasound imaging being under research, no approach considered the other side of the human-robot interaction, the robot-patient interaction to enhance the patient's comfort. Life Science Robotics seeks to improve its system focused on the sonographer-robot interaction by enhancing a sonographer-robot-patient interaction.

This project aims to contribute to the research of enhancing patient comfort when performing obstetric scans with RAU. The focus is on tracking the patient's abdomen partially occluded by the robot using classical computer vision approaches. By answering the question "Is it possible to track a patient's abdomen with the robot occluding the view using classical visual tracking techniques?" the project intends to provide a starting point to detect patient movement. The proposed approach consists of developing and testing a modified version of the Kernelized Correlation Filter to correct the tracking box under the appearance of occlusion. This is done by integrating feature matching when an occlusion is detected and calculating the transformation matrix with respect to a reference frame to update the box position.

Compared to the default performance of the tracker, the proposed method's average accuracy increased by 9% when occlusion was present on the frame. Despite the low average improvement, the approach can re-identify the patient in long tracking sequences after being lost due to a significant occlusion in the abdomen, reaching an accuracy of 84.5% compared to the 69,4% achieved by the default tracker. The accuracy of the track is dependent on the amount of occlusion in the abdomen. An accuracy of around 90% is achieved when the robot scans the lower part of the abdomen but it can decrease to less than 50% when the robot scans the upper part of the abdomen since the algorithm fails in detecting points on the region of interest. Moreover, the textureless skin area of the abdomen re-

lies on the contrast between the clothes and the skin to detect features, leading to wrong matches if the clothes are also textureless. Therefore, the approach shows the potential to accurately track the abdomen with the robot in view to detect patient movement but further research is needed to detect more robust features and handle large occluded areas.

Preface

Aalborg University May 31, 2024

This thesis completes my studies in the Robotics Master's Degree at Aalborg University. The project is in collaboration with Life Science Robotics, with their Robot Assisted Ultrasound system.

The provided link contains the tracking performance videos from three of the validation set videos.

`https://drive.google.com/drive/folders/15RZ1JSoLFsY-CVrVU6JnE6JSwaSW7hEc?usp=sharing`

Laia Vives Benedicto
<lvives22@student.aau.dk>

Abbreviations

AGAST	Adaptive and Generic Accelerated Segment Test
APCE	Average Peak to Correlation Energy
BRISK	Binary Robust Invariant Scalable Keypoints
CenSurE	Center Surrounded Extrema
CIU	Control Interface Unit
CRF	Corner Response Function
CVAT	Computer Vision Annotation Tool
DCF	Discriminative Correlation Filter
DFT	Discrete Fourier Transform
DoG	Difference of Gradients
ECU	External Control Unit
ERS	Edge Response Suppression
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximate Nearest Neighbors
FN	False Negative
FP	False Positive
HOG	Histogram of Oriented Gradients
ICP	Iterative Closest Point
IOU	Intersection Over Union
KCF	Kernelized Correlation Filter
LoG	Laplacian of Gaussian
LSH	Locality-Sensitive Hashing
LSR	Life Science Robotics
NMS	Non-Maximum Suppression
ORB	Oriented FAST and Rotated BRIEF
RAU	Robot Assisted Ultrasound
RGB	Red Green Blue
ROI	Region Of Interest
SIFT	Scale-Invariant Feature Transform

SLAM	Simultaneous Localization and Mapping
TP	True Positive

Contents

1 Introduction **1**

1.1 Problem statement 2

1.2 Report outline 4

2 Problem analysis **5**

2.1 The RAU system 5

2.1.1 What causes the patient movement problem? 7

2.2 Research towards a potential solution 8

2.2.1 Selection of the sensors and datatype 8

2.2.2 Visual tracking approaches 9

3 Related works **11**

4 Theoretical framework **13**

4.1 Discriminative correlation filters 14

4.2 Kernelized correlation filters 15

4.2.1 Histogram of Oriented Gradients 16

4.3 Average Peak to Correlation Energy 17

4.4 Feature matching 19

4.4.1 Feature detectors 19

4.4.2 Feature descriptors 27

4.4.3 Matchers . . . 29
4.5 Affine transformation . . . 30

5 Dataset and performance evaluation criteria 31
5.1 Constraints and assumptions . . . 31
5.1.1 Patient variables . . . 31
5.1.2 Environment variables . . . 32
5.2 Materials . . . 33
5.3 Setup before recording . . . 33
5.3.1 Software adaptation . . . 33
5.3.2 Preparation of the scene . . . 34
5.4 Recordings . . . 35
5.5 Ground truth extraction . . . 35
5.6 Evaluation metrics . . . 38
5.6.1 Metric for bounding box approaches . . . 38
5.6.2 Metrics keypoints approaches . . . 39
5.6.3 Robustness study . . . 39

6 Development of the patient visual tracker 40
6.1 Occlusion handling overview . . . 40
6.2 Evaluation of feature matching approaches . . . 42
6.2.1 Evaluation of feature detectors parameters . . . 42
6.2.2 Evaluation and selection matching method . . . 45
6.3 Adapted solution for the RAU system . . . 53

7 Performance evaluation 55
7.1 Evaluation setup . . . 56
7.2 Evaluation of the default KCF performance . . . 57

7.2.1 Results of test1: No occlusion . . . 58
7.2.2 Results of tests 2,3 and 4: Occlusion . . . 60
7.2.3 Results of test5: Long videos . . . 61
7.2.4 Summary of the default performance . . . 63
7.2.5 Weaknesses of KCF . . . 63
7.3 Evaluation of the proposed solution . . . 64
7.3.1 Summary of the proposed approach performance . . . 64
7.3.2 Test1: No occlusion . . . 65
7.3.3 Test2: Static occlusion . . . 65
7.3.4 Test3: Dynamic occlusion, static patient . . . 66
7.3.5 Test4: Dynamic occlusion, dynamic patient . . . 67
7.3.6 Test5: Long videos . . . 67

8 Final test **69**

9 Discussion **71**
9.1 Limitations of the approach . . . 72
9.2 Future work . . . 72

10 Conclusion **73**

Bibliography **75**

A Test results **79**
A.1 Default KCF testing details . . . 79
A.1.1 Test2: Static occlusion . . . 79
A.1.2 Test3: Dynamic occlusion, static patient . . . 81
A.1.3 Test4: Dynamic occlusion, dynamic patient . . . 83
A.2 Improved performance . . . 85

A.2.1 Test1: No occlusion 85
A.2.2 Test2: Static occlusion 86
A.2.3 Test3: Dynamic occlusion, static patient 87
A.2.4 Test4: Dynamic occlusion, dynamic patient 87
A.2.5 Test5: Long videos 88

B Outfit data **89**

Chapter 1

Introduction

Ultrasound scans have been an imaging diagnostic method since the 1940s. Considering that it is radiation-free and can provide real-time images, ultrasound scans are the least harmful imaging technique to observe internal organs[1]. Despite its common use, ultrasound scans require highly skilled sonographers who apart from properly analysing and identifying the view, have to manoeuvre the probe with repetitive arm movements while applying force, leading to work-related musculoskeletal disorders [2]. To relieve the physical discomfort, Life Science Robotics developed the Robot Assisted Ultrasound (RAU) system, called Hera, presented in Figure 1.1 [3]. In the RAU system, the transducer is held by a robot arm. The sonographer controls that arm using a joystick, sitting in a comfortable position.

Currently, the system has three cameras and it creates a 3D model of the patient. The model is displayed in the user interface for the sonographer to visualize the location of the probe. Furthermore, it is used for the system to automatically follow the curvature of the abdomen when the sonographer moves the probe with the joystick. However, the 3D model is made at the beginning of the scanning session and it does not get updated. Therefore, if the patient moves from its original position, the system needs to be restarted, reducing the available time for the ultrasound imaging. For this reason, this project aims to develop and test a visual tracker that can be used to detect and handle patient movement with the cameras already integrated into the system, to avoid restarting the system when movement is detected.

Figure 1.1: Hera, the Robot Assisted Ultrasound system from Life Science Robotics.

1.1 Problem statement

Robot-assisted ultrasound systems are in under-research development approaches, barely used in clinical practices in the last years [4]. When the focus is on obstetric scans, the sources are quite limited [5, 6]. While solutions have been presented to enhance the comfort of the sonographer, in obstetric or other kinds of ultrasound scans, by encouraging the interaction of the sonographer with the robotic system, no research considered the other side of the human-robot-interaction, the interaction of the robot with the patient to enhance the comfort of the patient [4, 5, 6, 7, 8].

During a conventional ultrasound scan, the sonographer freely moves the transducer

around the abdomen. Sometimes the sonographer may request the patient to change position for a better view of the neonatal or will momentarily stop scanning for the patient's need to relocate to a more comfortable position. In an ideal robotic-assisted ultrasound system the same functionality would be expected despite the use of the robotic arm. Thus, the robot arm should work as an extension of the sonographer's arm.

The Robot Assisted Ultrasound (RAU) system from Life Science Robotics (LSR) currently enhances the sonographer-robot interaction while ensuring the patient's safety. They implement this by providing the sonographer full control of the robot arm with the user interface and the joystick. In other words, the robot does not move unless the sonographer intentionally triggers the movement, getting control of the translation and orientation applied to the probe. Nonetheless, it is assumed that the patient is static in one position. The system does not consider the case the patient needs to move to either facilitate the view of the neonatal or feel more comfortable. A shift in the patient position requires a restart of the system.

LSR seeks to improve the system and enhance a sonographer-robot-patient interaction. To handle the patient's movement, in Figure 1.2 is presented a conceptual pipeline of the missing part of the system to provide a robot-patient interaction. The device should first track in real-time the patient. Then, displacement with respect to a reference frame should be detected in the camera and correlated to the real-world coordinate system in centimetres for example. Finally, a protocol should be activated to adjust the system to the movement while ensuring the patient's safety.

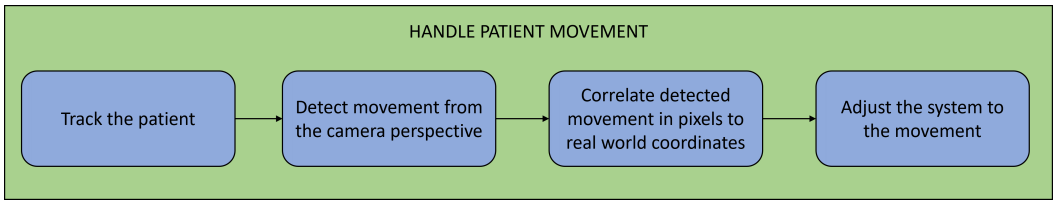


Figure 1.2: Conceptual pipeline to handle patient movement.

This project focuses on the first block in Figure 1.2, tracking the patient, in particular, the abdomen of the patient. With the RGB frames of one camera and classical visual tracking techniques, the project prioritizes studying the accuracy of the approach under the presence of occlusion to ensure the patient's safety. Even though real-time performance is another relevant factor when ensuring safety, a high-speed tracker will be considered when choosing the method but its speed is not analyzed nor prioritized in this project. For all of this, the main research question is:

"Is it possible to track a patient's abdomen with the robot occluding the view using classical visual tracking techniques?"

From this main question, the following research sub-questions arise:

- Does the accuracy get affected when increasing the amount of occlusion in the frame?
- Does the colour and texture of the clothes affect the performance?
- Could the approach be used to estimate the displacement and rotation with respect to a reference frame?

1.2 Report outline

In chapter 2 the RAU system is presented and the problem is explained. This chapter also exposes the main approach selected to develop. In chapter 3 previous attempts to solve the problem are presented. chapter 4 contains the theoretical aspects of the methods used in this project. In chapter 5 the created dataset and the evaluation metrics are exposed. chapter 6 describes the development of the proposed approach. In chapter 7 is presented the performance evaluation of the default tracker and the performance of the proposed improved tracking approach. chapter 8 provides the final test performed on the validation part of the dataset. In chapter 9 the results and the performance are analyzed and discussed considering the observed behavior during the performance. chapter 10 ends the report by concluding the work and giving an answer to the research questions.

Chapter 2

Problem analysis

This chapter provides a more detailed overview of the components and steps to perform an ultrasound scan using Hera and describes the main challenges the system faces when intending to track the patient. Furthermore, it exposes the considerations and decisions made when scoping the project. For instance, the chapter provides the reasons behind focusing on classical methods or using a single camera. At the end of the chapter, the chosen method is presented.

2.1 The RAU system

In Figure 2.1 are presented the components of the RAU system. The current RAU system consists of two main sets of sensors and actuators: the directly controlled by the sonographer marked in orange, and the internally handled by the system marked in green. The sonographer manipulates the user interface, a touchscreen used to initialize and control the scanning session. Furthermore, on the left side, it has the emergency stop button, which is utilized to stop the robot immediately when an unexpected situation occurs. To manoeuvre the robot, the haptic control device contains the joystick to control the translation, rotation and force applied to the probe. Performing an ultrasound scan additionally implies the use of three RGB-D RealSense cameras to create the 3D model of the patient, a Kuka robot arm, and an ultrasound transducer.

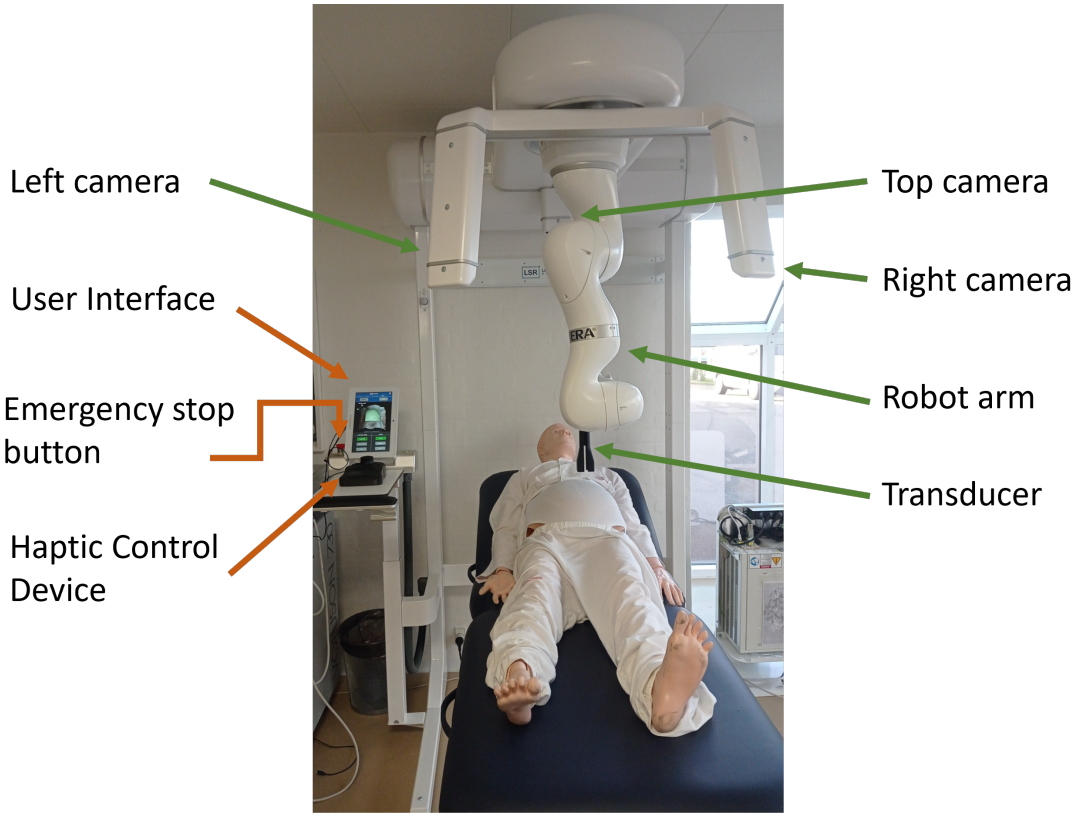


Figure 2.1: Components of the RAU system. In orange are displayed the components that the sonographer manipulates while in green are shown the sensors and actuators implied for the assisted ultrasound scan.

To perform an ultrasound scan, initially, the robot is placed in a safe position, the patient lays on the bed. Following the step process in Figure 2.2, the touchscreen displays the top view of the patient in real-time with the Region of Interest (ROI) overlaid. The sonographer has to correct the patient's position to fit the abdomen inside the region since the robot is only allowed to move within the marked area. In the second step, a 3D model of the patient is created by combining the view of the three cameras. Then, in the third step, the 3D model is displayed on the touchscreen and the sonographer can click on it to select the start position, indicated with a virtual probe. The green indicates the surface that the robot can reach for the scan. In the fourth step, the sonographer uses the touchscreen to move the robot from a **Home** position to the **Start** position. In the last step, the ultrasound scan can be performed by moving the joystick in the haptic control device of six degrees of freedom (translation and rotation). When using the joystick, the sonographer controls the x-y axes, the height is automatically handled by the system, following the curvature of the scanning surface. The touchscreen shows in real-time the probe position as well as available settings to be applied to the transducer. For example, it can control the speed of the robot's movements or the force applied to the abdomen. Haptic feedback is provided

with the haptic control device to give an idea of the amount of pressure applied to the abdomen. At any time, the sonographer can press the **Stop** button to move the robot upwards and pause direct contact with the patient, for example, when more gel needs to be applied. When the scanning session is over, the sonographer has to move the robot back to the **Home** position before the patient can safely step out of the bed.

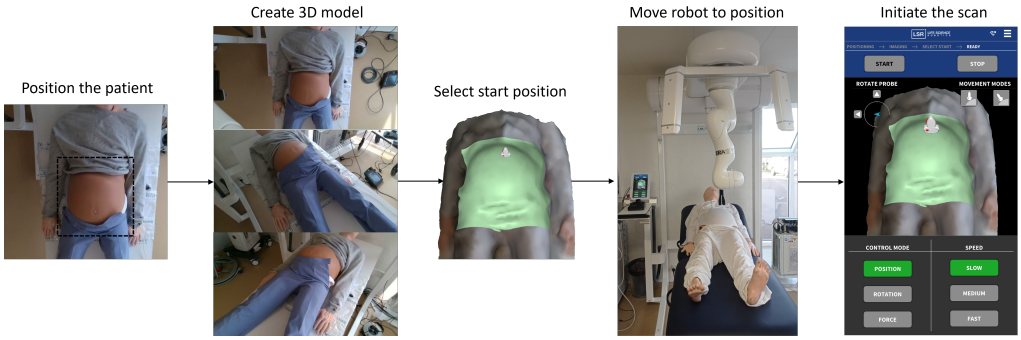


Figure 2.2: Step-by-step process to perform an ultrasound scan with the RAU system.

2.1.1 What causes the patient movement problem?

It is important to notice that one of the reasons for creating the 3D model at the beginning is that the robot is not in the field of view of the cameras. The **Home** position was intentionally chosen to avoid obstructing the view of the cameras. Contrarily, When the robot is performing the ultrasound scan, it is partially occluding the ROI, becoming part of the new scanning area if a new 3D model is taken. Therefore, the 3D model is never updated in the system.

Not updating the model causes two issues. On the one side, the ROI has a predefined constant size according to the robot arm reach, meaning that the entire abdomen might not fit within the ROI in all women's sizes. Given the case that the sonographer wants to use the transducer in an area out of the reach of the robot, the current procedure is to move back to the **Home** position. Then, the process presented in Figure 2.2 must be entirely done again to be able to scan the new desired region. This subtracts significant time from the scheduled scanning session.

On the other side, given the scenario that the patient feels uncomfortable and wants to reposition, the patient could ask the sonographer to press the **Stop** button and release the pressure applied to the abdomen under the patient's query. Nonetheless, if the patient moves, the 3D model will not match the real position, risking possible collisions or harder force applied to undesired parts of the abdomen. If the patient wants to move, the sonographer has to restart the scan by moving the robot to the **Home** position and repeating the scanning steps for safety matters.

What is defined as movement?

In an ideal RAU system, the patient should be allowed to move any part of the body as long as the abdomen stays still. This means that tracking the full human body would potentially trigger movements accepted in the system. The solution should focus on uniquely tracking the abdomen.

Main challenge of the project

The restriction of using the abdomen as the target for the tracker highlights the main challenge of this project, tracking a region that is mostly textureless and partially occluded during the entire ultrasound scan.

2.2 Research towards a potential solution

By selecting the sensors and datatype that will run the tracker as well as researching common visual trackers in the literature, in this section the method developed in this project is chosen.

2.2.1 Selection of the sensors and datatype

When selecting a potential solution for the RAU system, the first decision is on the choice of the cameras and data that will be used to track the patient. The system has three cameras integrated as presented in Figure 2.1. Using one or more of the cameras has its advantages and disadvantages. Multiple-camera approaches are a common approach when accuracy, efficiency, scalability and reliability are required since one camera would be able to see parts of the patient that are occluded in another camera. However, the system becomes complex for the appearance changes and proper data fusion[9]. Despite this, in many cases, multi-camera approaches track the target independently in each camera but handle the correlation between the different tracks [10]. Running a tracker in the three cameras would additionally increase the computational cost. It is considered that using a single camera can set the focus on finding a simpler but more robust approach for the system. In the future, the tracker could be implemented in the remaining cameras to increase the robustness if necessary.

Another relevant factor related to the camera is the data type. Visual tracking can be performed on RGB or depth images, or a combination of both. Considering that tracking of depth images requires the additional process of creating first a point cloud, adding

complexity and computation to the system the RGB image will be used in this project for its simplicity and a rich number of approaches developed throughout the years [11].

Of the three cameras in the RAU system, the top camera will be used since it provides the best view of the position of the patient in the bed. The patient is placed perpendicular to the top camera, perceiving the moment in the xy-axis of the image.

2.2.2 Visual tracking approaches

Visual tracking consists of estimating the position of a target in a sequence of images. Several approaches in the literature can be considered to track the abdomen of the patient [11]. They are mainly based on bounding boxes and points. Point approaches include approaches like joint estimation and motion analysis. Motion analysis consists of estimating flow vectors between two consecutive frames for each pixel in the image. By segmenting the image, in other words grouping the pixels belonging to the abdomen, the patient could be tracked. Moreover, segmentation could also be applied to the robot and create a multi-object tracking system. Joint estimation only takes relevant keypoints in the human body to estimate the motion. The main drawback in point-based estimation is that is not straightforward to evaluate, but the motion would be accurate, detecting translation and rotation movements. Many trackers are based on bounding boxes instead [12]. A box is simple to define and to evaluate since it is defined by two points instead of a group of pixels. However, a single bounding box can only consider translational movement. To detect rotation, a secondary box or additional approaches would be necessary.

A study from 2019 compared the classical and deep-learning-based trackers developed between 2015 and 2019 [13]. The study concluded that Discriminative Correlation Filters (DCF) approaches achieve the best performance. Furthermore, the paper states that Kernelized Correlation Filters (KCF) [14] is the fastest DCF tracker among the 24 studied trackers. However, handcrafted trackers decrease the performance when changes appear in the environment such as occlusions. Neural networks have been integrated into KCF as a baseline to improve its drawbacks [13].

A more recent survey published in 2023 reaffirmed DCF as the most outstanding approach in visual object tracking and relates their popularity to the computationally efficient methodology applied [15]. Within the focal point in handcrafted approaches based on DCF to handle occlusions, Xin Du, et. al [16] proposed the integration of KCF with feature-matching. Their approach uses the default KCF tracker and applies feature matching by detecting points with Oriented FAST and Rotated BRIEF (ORB) to correct the estimated box position when the target is occluded. Feature-matching techniques are the fundamentals of computer vision to detect salient points in two images and find correspondences [11]. It is considered that this approach has the potential to overcome the occlusion caused by the robot arm since at every frame some points will likely be detected within the ab-

domen, for the target on a single independent point rather than the entire target region. Furthermore, the approach corrects the box by calculating the transformation matrix of the matched ORB features, which could be used later to calculate rotational changes. Therefore, the approach keeps the simplicity of bounding box trackers but rotational movements could be calculated with the transformation matrix.

Selected method to track the patient

For the potential successful tracking of the target, while the presence of occlusions, the combination of KCF tracker with feature matching is the approach selected to develop. In comparison to Xin Du, et. al [16] which focused on ORB and tested the performance on the OTB-2015 dataset, the main contribution of this project is the implementation and performance test of the approach in a real-world complex scenario application, the RAU system. Furthermore, common feature-matching approaches in the literature will be evaluated together with ORB to select the most suitable feature detector for the RAU system [17].

Chapter 3

Related works

Previous attempts intended to research an approach to handle the patient movement in the LSR's RAU system. The first shot consisted of updating the 3D model of the patient during the scan and removing the points in the point cloud that belonged to the robot. The occluded region by the robot was reconstructed to achieve an updated 3D model. Movement was detected by studying the point cloud transformation between the reference position and the new position by using the Iterative Closest Point (ICP) algorithm. The approach reported a slow computation time that would potentially lead to a late reaction to the movement. It was reported that the method should run at specific intervals of time rather than in real time. Moreover, the entire patient's body was considered a rigid body [18]. In this report's proposed approach, computing KCF and feature-matching in RGB images can provide a faster tracking performance for the natural high speed of the KCF tracker and the avoidance of additional computational steps such as creating the point cloud. Furthermore, the new approach focuses on uniquely tracking the abdomen, becoming closer to the ideal movement detection.

During the last semester, an internship was done at LSR focused on deepening into the patient movement problem [19]. Different methods were explored to visualize the advantages and disadvantages between them. Within the wide range of possibilities, two main approaches were considered. One of the main approaches was to track the robot by placing a box around its end effector and not consider the pixels in the box when studying patient movement. Even though the box position was highly reliable since the end effector's position and orientation were known and the cameras were calibrated, the remaining robot's joints' positions were not known. Therefore, the box only covered part of the visible robot in the image. Moreover, the box covered most of the ROI pixels to study patient movement. Even though tracking the robot can provide a clear distinction between the patient's movement and the robot's movement, the problem increases in complexity due to the need to track multiple targets and classify the two types of movement that are

constantly overlapping. Furthermore, a different approach than the bounding box, such as instant segmentation, would be required to classify the pixels for each target for more accurate tracking. The newly chosen method could be affected by the robot's movement, introducing false positives, but it narrows down the complexity of the track and it does not require a large dataset to train a classifier that would distinguish the patient from the robot.

The second approach in the internship considered applying a neural network inference model to estimate the patient's joint position from the top camera and study the displacement between a reference frame's joint position and the current frame. The approach focused on analyzing the closest joints to the abdomen (the joints placed on the shoulders and hips) and triggering a boolean metric indicating the detection of a shift. It was observed that the keypoints were visually well estimated in the presence of occlusions but significant fluctuations were present when estimating the same keypoints over time. The lack of datasets containing images similar to the ones captured by the RAU cameras and the black hole in the neural network decision-making process limits the possibility of improving the method's performance. The switch to a classical approach requires less input data to make changes to the output and the internal computed processes can be step-by-step analyzed to improve and adapt to the system.

Chapter 4

Theoretical framework

This chapter presents the theoretical fundamentals of the methods used in this project. It starts by exposing the principles of Discriminative Correlation Filters (DCF) delving into the particularities of Kernelized Correlation Filters (KCF), the baseline approach in this project.

To improve the performance of KCF when the target is occluded, the focal point is on the proposal of Xin Du, et. al. [16], which consists of detecting the occlusion and then applying feature-matching to relocate the target's position and correct the tracking box. Thus, it is based on three main parts:

- Calculate the Average Peak to Correlation Energy (APCE) from the response map to detect the presence of occlusions in the ROI.

- Use Oriented FAST and Rotated BRIEF (ORB) to match the features between two frames.

- Correct the tracking box based on the transformation matrix and the previous frame position.

The mathematical concepts behind each of the three added-on to the tracker are explained in the subsequent sections. As stated in chapter 2, to enhance the accuracy of the tracker in the RAU system, this chapter also introduces the theoretical aspects of common feature detectors and descriptors in the literature that will be evaluated together with ORB to select the approach that best fits the problem [17, 11]. Moreover, the paper does not specify the matching technique used, so in this project, Brute-Force [20] and FLANN [21] basics will

also be introduced. In the last section, the calculation of the affine transformation matrix is also presented to correct the tracking box with respect to the reference frame position.

4.1 Discriminative correlation filters

Discriminative correlation filters (DCF) are based on tracking-by-detection which consists of identifying the target in the frame for updating the bounding box. To detect the target, DCF trains a filter by extracting features from the target which creates a peak response where the target is located. This is achieved by learning the coefficients w of a regulated linear regression model. The goal is to define the function that minimizes the squared error of the target's response $f(x_i)$ with respect to the estimated response y_i as represented in Equation 4.1, such that $x_i * w \approx y_i$ [15, 14]. Being x_i the extracted feature at the pixel position i from the target. When the filter is applied to another image it will output high response values at the location in the image with the closest similitude to the target. The model regularizes the regression by applying the penalty λ.

$$\min_{w} \sum_{i} (f(x_i) - y_i)^2 + \lambda \|w\|^2 \tag{4.1}$$

The coefficients or weights of the correlation filter are calculated as shown in Equation 4.2, where X is a matrix containing the features from the ROI, X^T is the transposed X matrix and y is the matrix of the estimated response. The determinant of $X^T X$ is close to zero when there are correlations between the features, meaning that they are linearly dependent. However, if the determinant is zero the inverse does not exist. Therefore, by adding λI the determinant increases, reducing the coefficients when doing the inverse. Lower coefficients increase the robustness to noise and keep a high response when the target suffers slight modifications [14].

$$w = (X^T X + \lambda I)^{-1} X^T y \tag{4.2}$$

The key aspect of the computational efficiency of DCF relies on the approximation of a circulant matrix. Conceptually, in a circulant matrix at each row, the elements are the same but shifted one position to the right. Assuming that the feature matrix is circulant as contextually shown in Equation 4.3, the Discrete Fourier Transform (DFT) can be applied to calculate the weights in the frequency domain.

$$X = C(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{4.3}$$

When applying the Discrete Fourier Transform (DFT) to a circulant matrix, only the first row of X is taken. The DFT of each element of the first row of X can be placed in the diagonal of a matrix, reducing the computation cost since it is a matrix of zeros except for the elements in the diagonal.

In the Fourier domain, Equation 4.2 is expressed as in Equation 4.4, where $\odot$ represents the element-wise dot product. To get the weights from the frequency domain $\hat{w}$ in the spatial domain w the inverse DFT has to be applied.

$$\hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda} \tag{4.4}$$

4.2 Kernelized correlation filters

The singularity of KCF with respect to the fundamentals of DCF presented in the previous section is that applies a kernel to the correlation model. The core idea is to transform a linear problem into a non-linear problem. Linear correlation filters get more complex when the number of features increases since it is more challenging to fit a hyperplane within the data, creating a bottleneck that raises the computational time. However, higher performance will be achieved with a higher amount of features. For this reason, when solving a non-linear problem, the data is fitted in a higher dimensional surface instead, being more complex to solve but more adaptive to the samples. The advantage of KCF is the use of the "kernel trick" which consists of solving a non-linear problem linearly, handling the complexity but keeping the computational cost down. Applying a kernel to the features is equivalent to mapping the features into a higher dimensional feature space (ϕ) and performing the dot product between all features as presented in Equation 4.5, where $k(x_i, x_j)$ is the dot product of one pair of features, x_i and x_j. A kernel K is the matrix containing the dot product of all pairs of features.

$$k(x_i, x_j) = \phi(x_i) \odot \phi(x_j) \tag{4.5}$$

The kernel K simplifies the computational cost of mapping the features in a new feature space and fits a hyperplane by studying the similarity between them. It was proven that

kernels such as Radial Basis Function kernel keep the circulant properties that define the correlation filter and therefore can be integrated into the approach. The kernelized version of the regression model is presented in the Equation 4.6, where α is the new coefficient to optimize.

$$\alpha = (K + \lambda I)^{-1} y \tag{4.6}$$

When applying the DFT assuming the input data is a circulant matrix, Equation 4.6 is expressed as in the Equation 4.7 where k^{xx} is the first row of the kernel matrix.

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda} \tag{4.7}$$

This chapter presents the internal parts of the Kernelized Correlation Filter (KCF) tracker, the mathematical foundation that connects the target with the bounding box following it. To get in context, a simple conceptual diagram is presented to visualize the workflow. A deeper mathematical explanation is given to comprehend the internal functionalities of the tracker based on its developers [14]. This project uses a Python implementation based on the original paper [22]. Thus, the mathematical foundations are aided by the Python implementation, exposing a detailed connection of the theoretical part with the algorithm.

4.2.1 Histogram of Oriented Gradients

Kernelized Correlation filters was the first tracker in using the Histogram of Oriented Gradients (HOG) [23] features for tracking [15].

HOG studies the changes in the colour intensity in the image. For each colour channel, the algorithm consists of computing the gradients in the x and y directions in each pixel, returning higher values for higher changes in the intensity. Then, the magnitude and direction of the gradients are calculated per pixel computing Equation 4.8 and Equation 4.9, where g_x and g_y are the gradients in the x and y directions.

$$g = \sqrt{g_x^2 + g_y^2} \tag{4.8}$$

$$\theta = \arctan \frac{g_y}{g_x} \tag{4.9}$$

The largest magnitude and its associated angle are kept from the three magnitudes and angles in each pixel, one per channel. The image is divided into 8x8 pixel cells. The histogram is computed inside each cell representing the frequency of each direction between

0 and 180 degrees in bins of 20 degrees, returning the direction with the highest magnitude in each cell. To make the features robust to light changes, a block of 16x16 pixels is slid through the image to normalize the magnitudes by dividing each magnitude by the l2-norm of the block seen in Equation 4.10, where g_k is the magnitude at cell k, within the block size n.

$$|g| = \sqrt{\sum_{k=1}^{n} |g_k|^2} \tag{4.10}$$

The image features are a vector containing the normalized magnitudes of each 8x8 pixel cell.

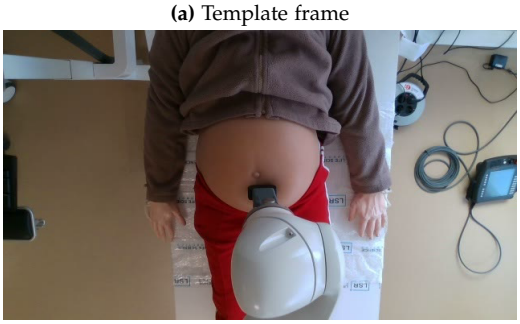
4.3 Average Peak to Correlation Energy

To detect when the target is occluded and use feature-matching instead of the default KCF algorithm it depends on the APCE parameter. The Average Peak to Correlation Energy is defined as in Equation 4.11, where F_{max} and F_{min} are the highest and lowest values in the response map, and $F_{w,h}$ is the response map value at each pixel position of the image, representing w the width and h the height of the pixel position [16].

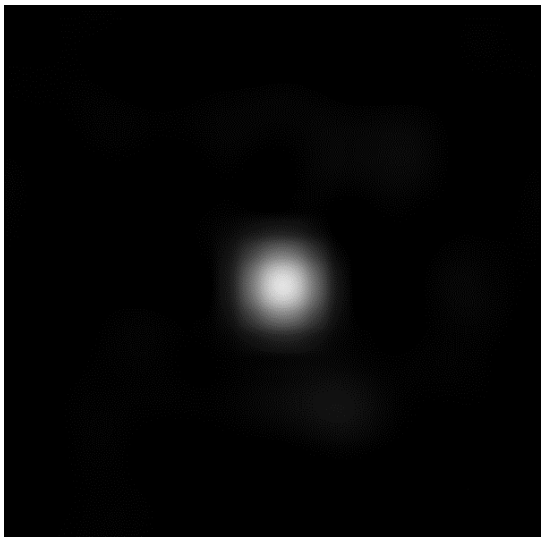
$$\text{APCE} = \frac{(F_{\max} - F_{\min})^2}{\text{mean}\left(\sum_{w,h}(F_{w,h} - F_{\min})^2\right)} \tag{4.11}$$

The numerator computes the absolute difference between the extreme values of the response map while the denominator computes the average absolute difference between each pixel response value and the minimum. Thus, it calculates the ratio of the peak value in the image. For instance, in a scenario where the target does not change, the peak value is high and concentrated in a small region of pixels as seen in Figure 4.1. Most pixels have a response value close to the minimum while few pixels are close or at the peak. As a result, the average response value throughout the image will be low but in relation to the peak will result in a high APCE. Contrarily, when the target does not look the same between two consecutive frames, the peak value is lower and scattered in the image as seen in Figure 4.2. In the example, the robot has moved making visible part of the transducer and a larger part of the skin. In that case, the mean throughout the image will be higher but closer to the peak value, getting a small APCE.

(a) Template frame



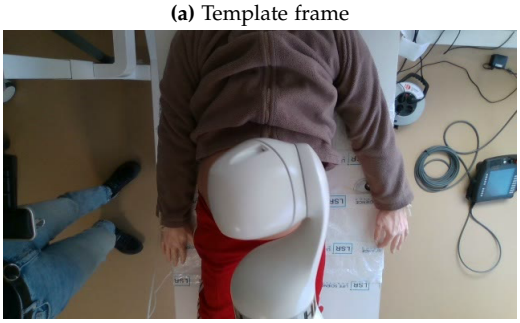
(b) Frame i



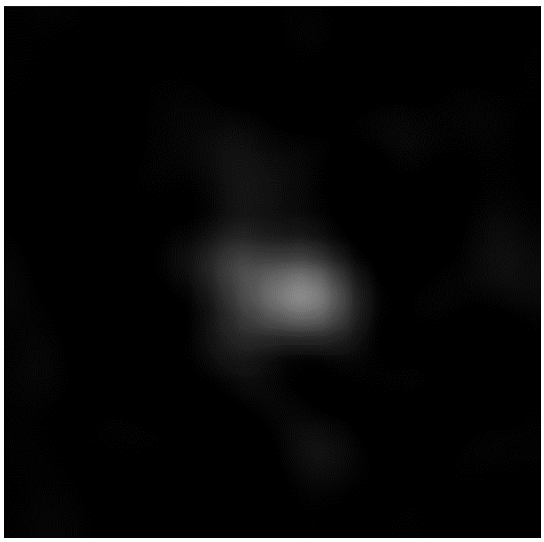
(c) Response map with high APCE

Figure 4.1: Response map example that results in high APCE. **a)** and **b)** show the template and the current frame respectively.

(a) Template frame



(b) Frame i



(c) Response map with low APCE

Figure 4.2: Response map example that results in low APCE. **a)** and **b)** show the template and the current frame respectively.

4.4 Feature matching

Feature matching consists of corresponding salient points in two images. It is a prerequisite step for many applications. Feature matching is used for creating the 3D structure of a scene seen from different points of view. Furthermore, it is used in Simultaneous Localization and Mapping (SLAM) for estimating the relative camera pose or in visual homing for estimating the direction of movement of the robot. Moreover, feature matching can be used to recognize objects in images and estimate their trajectory, being the basis of feature-based tracking [17]. In this project, feature matching will be used to identify the partially occluded abdomen by the robot. The diagram in Figure 4.3 shows the main components of feature matching adapted to the RAU system. On the left side, there is the template which will be the ROI of the first frame containing the abdomen. On the right side, represented is a full frame of the video sequence. The approach consists of detecting and describing points in the template and the frame independently and finding correspondences between all possible pairs, defined as matching [11].

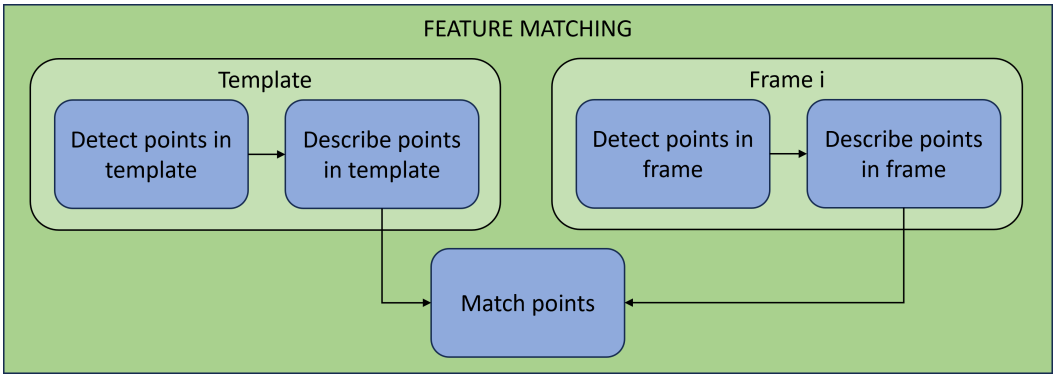


Figure 4.3: Diagram of feature matching, where the template is the ROI and frame$_i$ represents the i_{th} full frame in a video sequence.

4.4.1 Feature detectors

A feature is a distinctive element in an image that can easily be identified in different frames. There are different sizes of features, from keypoints to regions. However, for its simplicity, keypoints are the easiest to detect and define. Furthermore, they are stable in the presence of occlusion or scale and orientation changes. When detecting keypoints, the light intensity changes of a grayscale image is studied. A small patch around the point is taken to determine its robustness. For a feature to be reliable, it has to be consistent in the presence of transformations that the target can experience in an image [11]. These are constrained to translation and rotation in the 2D plane for this project. Common characteristics required for a feature to be reliable are presented below based on [11]:

- **Stability:** For a feature to be robust it has to stand out compared to its nearest pixel neighbours.
- **Repeatibility:** A feature is significant when it is found in a transformed image. The transformation can be in terms of rotation, scale or light intensity. If the same feature is detected in several image transformations it is repeated, meaning it is robust.
- **Scale invariance:** The feature shall still be found when the distance from the camera to the feature or the resolution of the images is different. Performing the same operations at multiple image resolutions and matching the features through the levels is an approach used by feature detectors to ensure scale invariance.
- **Rotation invariance:** When looking for rotation invariance, it is important to consider that a feature that is invariant to rotation could match different features that share the same rotation, leading to false positives. For this reason, to ensure that a feature is unique and rotation invariant, looked for is the dominant orientation of a local patch. This is done by calculating the gradients in each pixel of the patch and computing the average direction.
- **Affine invariance** Apart from the scale and rotation invariance, for a feature to be robust it has to handle affine transformations. Affine transformations include shifts in scale and rotation but also any other geometric transformation that preserves points, lines and planes[11].

Apart from ORB, the chosen approach by Xin Du, et. al. to handle occlusions with KCF[16], many other algorithms have been developed to detect robust keypoints which could adapt better, or not, to the characteristics of the RAU's target. Common detectors in the literature that will be studied together with ORB are Features from Accelerated Segment Test (FAST), Adaptive and Generic Accelerated Segment Test (AGAST), Binary Robust Invariant Scalable Keypoints (BRISK), Scale-Invariant Feature Transform (SIFT) and Center Surrounded Extrema (CenSurE) [11, 17]. However, since many detectors are related to the Harris Corner Detector, a brief introduction is given to comprehend the functionalities of the studied detectors.

Harris corner detector

The Harris Corner detector is the benchmark of the studied detectors in this project. Its basic functionalities are introduced to understand the functionalities of some of the later evaluated methods.

The detector studies the stability of a keypoint by comparing it against itself. The point is slightly slid towards both directions (x and y) to be compared with the original point

and study the degree of similitude. By computing the squared differences between corresponding pixel values, weighted by a kernel function, the Equation 4.12 shows the auto-correlation function [11].

$$EAC(\delta u) = \sum_i w(x_i)[I_0(x_i + \delta u) - I_0(x_i)]^2 \tag{4.12}$$

In the equation, δu represents the displacement in the image patch. $w(x_i)$ is a weight to give more importance to central pixels. $I_0(x_i)$ and $I_0(x_i + \delta u)$ indicate the intensity of the original and shifted patch respectively. The auto-correlation function returns a small value when the patches are similar, indicating that is stable over small shifting, being a potential feature or point of interest. To ensure rotation and illumination invariance it expands the function using Taylor to consider multiple angles of rotation, getting the moment matrix in Equation 4.13. I_x and I_y are the partial derivatives of the light intensity in the x and y directions [11].

$$M = w \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{4.13}$$

At each pixel the response of the moment matrix is studied by computing Equation 4.14. With the eigenvalues of the matrix $\lambda 1$ and $\lambda 2$, $Det(M)$ is defined as the product of the eigenvalues and $Tr(M)$ is defined as the sum of the eigenvalues. k it is a free parameter. When both eigenvalues have similar values and they are large, it indicates that the region is a corner. If one eigenvalue is considerably larger than the other it is an edge, otherwise, it is a flat region [24].

$$R = Det(M) - kTr(M)^2 \tag{4.14}$$

Features from Accelerated Segment Test

FAST is a high-speed intensity-based corner detector. It is a variation of Harris Corner Detector that provides the same accuracy but at a lower computational cost. Its high speed is due to efficiently rejecting points with low-intensity variations [25].

Given an image, FAST computes the intensity gradients at each pixel and discards the gradients that are lower than a threshold. This removes around 80-90 % of the pixels. In each of the remaining points, a digital circle is taken centred in the candidate point as seen in Figure 4.4, which displays the default circle taken by OpenCV [26]. The circle is divided into pairs of points that are connected and pass through the candidate pixel, creating a line. First, the main axes, horizontal and vertical, are computed. In the image, the main axes are highlighted in blue, represented by A and B lines. The intensity differences in

each line are calculated as seen in Equation 4.16, where I represents the pixel intensity in each point, being P the central or candidate point. This is integrated into the Corner Response Function (CRF) in Equation 4.15, where A belongs to $\alpha = 0$ and B is $\alpha = \pi/2$. The CRF is used to determine whether the point is a corner or not by comparing it with a given threshold. If CRF is lower than the threshold the point is discarded, otherwise, the intensity difference in the remaining lines is integrated into CRF. The remaining lines are computed as an interpolation of the main axes calculated as seen in the example in Equation 4.17 for line C. With all computed points NMS is used to keep only the main points [25].

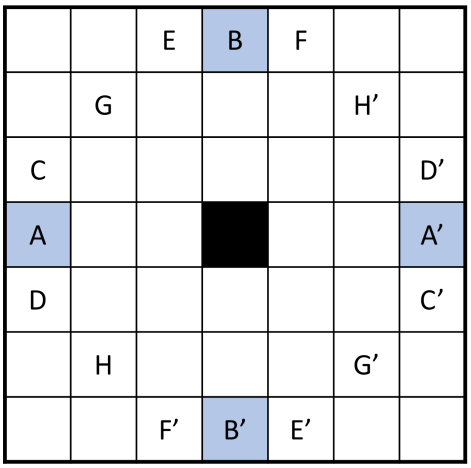


Figure 4.4: Default digital circle when using FAST in OpenCV. The black cell represents the candidate keypoint while the blue cells indicate the two main pairs of points.

$$CRF = \min(r_{\alpha i}, r_{\alpha i+1}, ..., r_{\alpha i+n}) \tag{4.15}$$

$$\begin{aligned} r_A &= (f_A - f_C)^2 + (f_{A'} - f_C)^2 \\ r_B &= (f_B - f_C)^2 + (f_{B'} - f_C)^2 \end{aligned} \tag{4.16}$$

$$\begin{aligned} f_C - f_P &= (f_A - f_P) \cdot \cos\alpha + (f_B - f_P) \cdot \sin\alpha \\ f_{C'} - f_P &= (f_{A'} - f_P) \cdot \cos\alpha + (f_{B'} - f_P) \cdot \sin\alpha \end{aligned} \tag{4.17}$$

Oriented FAST and Rotated BRIEF

ORB uses FAST to detect keypoints but scores them and filters out possible detected edges by computing the response of the moment matrix from Harris Corner detector (Equa-

tion 4.14). It also integrates a feature descriptor which is rotated BRIEF. It was created to achieve real-time performance while keeping the accuracy and being less affected by noise, but it cannot handle well scale changes[27].

Adaptive and Generic Accelerated Segment Test

AGAST is an improvement of FAST which consists of detecting keypoints faster and with an adaptive threshold. It uses the first step of FAST, it computes the image gradients to filter out the majority of the pixels and keep a few candidates. It also takes a digital circle of 16 points around the candidate point to compare the intensities to those neighbours, but it uses an adaptive threshold instead and a decision tree with 6 possible answers for the current state of the point $S_{n\to x}$, depending on the previous state $S'_{n\to x}$(Equation 4.18) [28].

$$S_{n\to x} = \begin{cases} d, & I_{n\to x} < I_n - t \\ \overline{d}, & I_{n\to x} \nsim I_n - t \wedge S'_{n\to x} = u \\ s, & I_{n\to x} \nsim I_n - t \wedge S'_{n\to x} = \overline{b} \\ s, & I_{n\to x} \nsim I_n + t \wedge S'_{n\to x} = \overline{d} \\ \overline{b}, & I_{n\to x} \nsim I_n + t \wedge S'_{n\to x} = u \\ b, & I_{n\to x} > I_n + t \end{cases} \tag{4.18}$$

For each possible response, a probability is assigned to quantify the certainty for the candidate point to be a corner as seen in Equation 4.19. Depending on the probability response it computes a decision tree optimal homogenous (textureless regions) or heterogeneous regions.

$$p_X = \prod_{i=1}^{N} p_i \quad \text{with} \quad p_i = \begin{cases} 1 & \text{if } S_{n\to i} = u \\ p_s & \text{if } S_{n\to i} = s \\ p_{bd} & \text{if } S_{n\to i} = d \text{ or } b \\ p_{bd} + p_s & \text{if } S_{n\to i} = d \text{ or } b \text{ (previous state)} \end{cases} \tag{4.19}$$

The decision tree finishes when the corner criteria is met or is no longer met. The corner criteria states that from the 16 neighbour points, at least 9 classify the candidate as a corner by agreeing on being darker or brighter.

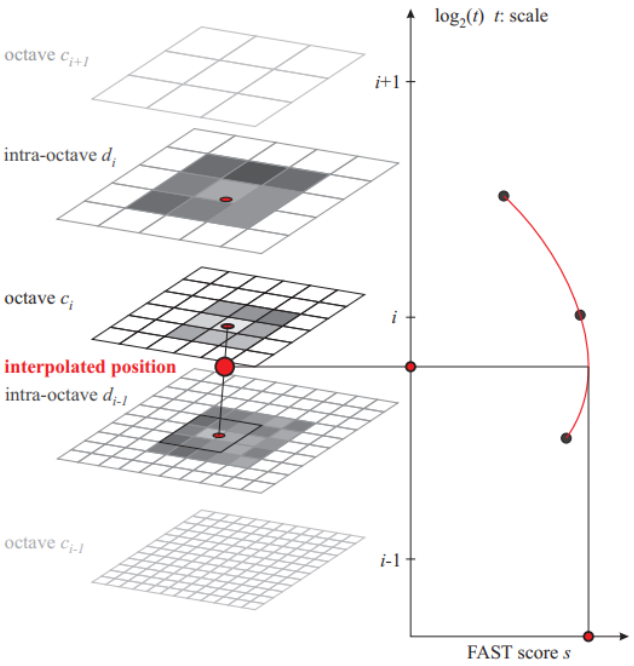


Figure 4.5: Keypoint detection with brisk. On the left are seen the octave and intra-octave layers. On the right side is the parabola fitting to determine the real size of the keypoint. **Source:** [29].

Binary Robust Invariant Scalable Keypoints

This feature detector also integrates its own descriptor. In this section, the detector is introduced. In the feature descriptor section, its descriptor will be explained.

BRISK downsamples the image to create a pyramid of resolutions and detect keyppoints at different scales. The downsampling consists of reducing the size by half per octave. Moreover, between each octave, it creates an intra-octave that downsamples by 1.5 the first time and by half the remaining layers as seen on the left side of Figure 4.5. In each of the layers, FAST is used to detect keypoints. It applies NMS in the 3x3x3 neighbouring pixel space to remove repeated detected corners. Furthermore, it computes a sub-pixel refinement to determine the real scale of the keypoint and how the scale affects the feature. It consists of fitting a parabola within the same keypoint detected in the current level and the two adjacent levels as shown on the right side of Figure 4.5. The vertex of the parabola belongs to the real scale of the feature[29].

Differrence of Gaussians

SIFT is a feature detector and descriptor algorithm. In this section, the focus is on the detection part which consists of computing the Difference of Gaussians (DoG). The description part is left for the feature descriptors section.

Difference of Gradients (DoG) is a gradient-based detector that seeks to robustly detect keypoints by focusing on scale invariance. The algorithm applies Gaussian blur kernels to create a scale space L (Equation 4.20). Given the Gaussian kernel in Equation 4.21, multiple levels of Gaussian blur are achieved by multiplying σ with a constant factor k [30].

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{4.20}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.21}$$

At the last level of the scale space, the image resolution is down-sampled in half, initiating a new octave with twice the initial σ. An octave encloses the multiple scale spaces of the same resolution image. In Figure 4.6 a visual representation of the octaves and the scale space is seen on the left side. The Difference of Gaussian levels is created by subtracting two adjacent scale spaces ($L(x, y, k\sigma) - L(x, y, \sigma)$), getting N-1 levels per octave as represented in the right side of the image, being N the number of levels. For each of the octaves, the algorithm compares every pixel of one DoG level with the 8 closest neighbours in that level and the 18 neighbours in the adjacent levels (9 in the level above and 9 in the level below) as seen in Figure 4.7. With this analysis the minimum and maximum pixels are detected in each octave, being candidate keypoints. A pixel that is detected in multiple levels and octaves is, therefore, a robust point in scale changes [30]. The algorithm also applies Non-Maximum Suppression (NMS) and Edge Response Suppression (ERS) to remove redundancies or unstable keypoints. NMS checks if the neighbours have smaller values, ensuring that it is a local maximum, removing the detected points that do not fulfil the requirement. ERS computes the intensity gradients in the point to detect if it is part of an edge and remove it [31].

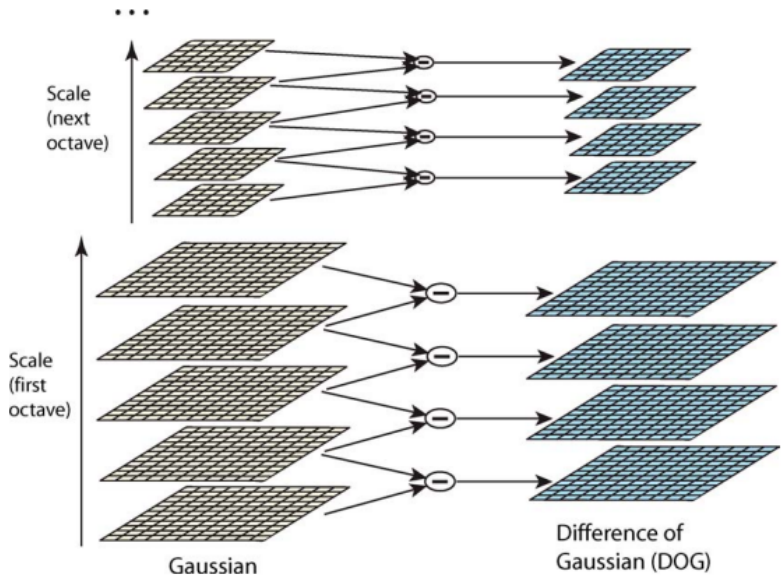


Figure 4.6: Graphical representation of the scale space creation from the original image and the resultant Difference of Gaussian levels. **Source:** [30]

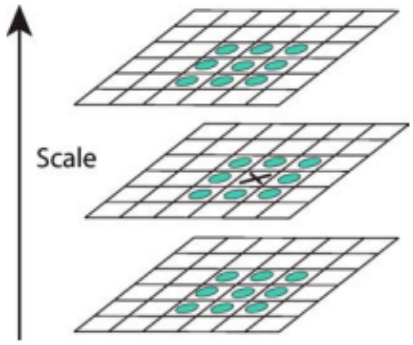


Figure 4.7: Keypoint analysis between the DoG levels. **Source:** [30]

Center Surrounded Extrema

CenSurE is a curvature-based detector. It is based on center-surrounded filters. These filters are equivalent to computing the second derivatives in the image to detect the local maximums and minimums by improving the Laplacian of Gaussian (LoG) filter. LoG applies Gaussian blur at different scales and then computes the second derivative of the

intensity pixels to extract local maximums and minimums. However, this process is computationally expensive. CenSurE detector simplifies it using a bi-level filter with values -1 and 1. It consists of a kernel where the -1 and 1 values are placed equivalently to LoG filter, becoming more efficient. A 3x3x3 neighbourhood in a pyramid of Gaussian blurred images is taken to detect potential corners. To keep only the most relevant corners it also runs the Harris response moment matrix [32].

4.4.2 Feature descriptors

Once the keypoints of an image have been detected, their appearance has to be described to be uniquely identified in other images. Even if the detector gets just single points, a small patch is used to analyse the appearance of its surroundings. The descriptors robustly transform the relevant local information into a high-dimensional vector.

The descriptor is created following three steps. First, the low-level information, which comes from the intensity and gradient values of the patch, is extracted. Then the patch is divided into smaller groups of pixels and pooling techniques are applied to extract the relevant information in each part. Finally, the results after pooling are normalized and introduced into a vector.

Rotated Binary Robust Independent Elementary Features

In ORB, the detected keypoints using FAST are described with the binary descriptor rotated Binary Robust Independent Elementary Features (rBRIEF). BRIEF by default is not invariant to rotation. The descriptor takes a patch around the image and applies Gaussian noise. Random points spread in the patch are selected to compare their intensity values and create an array of binary responses, being 1 if the first point has a higher intensity than the second, and 0 the other way around. BRIEF has the characteristic that the variance in the bin array is high while the mean is close to 0,5. High variance means that the response at different keypoints will be different, getting a unique binary where the mean between 0 and 1 is about 0.5. Rotated BRIEF takes the main orientation of the keypoint and rotates the tested pixel locations, getting a different response that does not keep the high variance and mean of 0.5. Therefore, rBRIEF uses a learning algorithm such as PCA that selects the best points to be evaluated that will not be affected for the rotation [27].

Binary Robust Invariant Scalable Keypoints descriptor

BRISK creates a pattern of a pair of concentric circles around the keypoint, increasing in size when getting away from the keypoint as represented in Figure 4.8. The blue circles

select the pixels in the patch that will be analyzed to compute the descriptor. The red circle denotes the σ of the Gaussian blur kernel applied to the circle. For each pair of circles is calculated the intensity gradient between two adjacent circles. The intensity gradient at two circles at a further distance is also computed to estimate the dominant orientation of the patch. The BRISK descriptor is constructed by rotating the circular pattern in the dominant direction and comparing the intensity gradients between two points. If the intensity of the first point is higher than the second is annotated as 1, otherwise it is 0 [29].

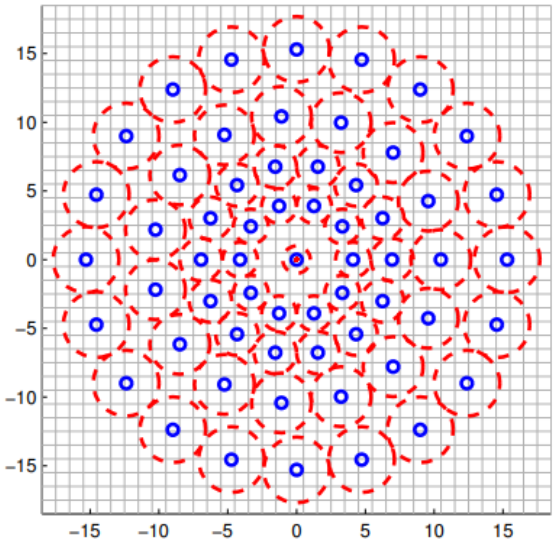


Figure 4.8: Concentric circles representation to select the keypoints that will create the descriptor. **Source:** [29].

Scale- Invariant Feature Transform

In the previous section, it was seen that SIFT detects keypoints that are robust to scale changes. When creating the descriptor, SIFT focuses on the orientation of the patch around the keypoint. A 16x16 patch is taken to calculate the main orientation and magnitude of each pixel in the patch. In Figure 4.9 an example is shown on the left side for an 8x8 patch instead. A Gaussian weight is applied on top to give higher importance to the pixels closer to the centre of the patch (the detected keypoint). Afterwards, the patch is divided into 4x4 non-overlapping quadrants (2x2 non-overlapping quadrants on the right side of the Figure 4.9) and HOG is calculated. Each quadrant becomes an eight-bin histogram of angles, creating a descriptor of 128 angles due to the 4x4 quadrants times 8 bins each [30].

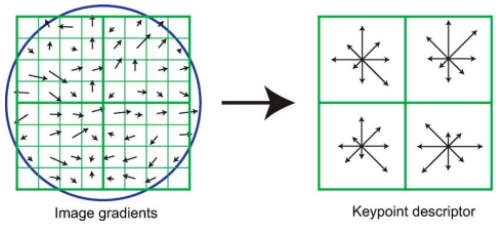


Figure 4.9: Example of the creation of the SIFT descriptor by taking an 8x8 patch around the keypoint and computing a descriptor of 2x2. In this example, the descriptor would have 32 angles. **Source:** [30].

4.4.3 Matchers

Matching is the process of finding the same features in two different images. Common approaches in OpenCV are the Brute-Force and the Fast Library for Approximate Nearest Neighbors (FLANN) matcher [26].

Brute-Force

Brute-Force is known for its simplicity and high accuracy in finding matches. For each keypoint in the first image, it systematically computes the Euclidean distance to every keypoint in the second image, looking to find the nearest neighbour. The point in the second image with the lowest distance becomes a potential match. The two keypoints will be matched if the distance between them is lower than an internal threshold [11, 20].

Fast Library for Approximate Nearest Neighbors

Fast Library for Approximate Nearest Neighbors (FLANN) is a library available in OpenCV that provides different algorithms to match features faster for high-dimensional data based on the nearest neighbour approach. FLANN speeds up the process by finding points that are approximately similar. It guesses what points fit well together even if they are not an exact match. One of the main algorithms is the use of kd-trees. It consists of creating a tree structure where in each node, similar features are put together, for instance, similar intensity values. the descriptors are split into the 5 highest variance components. The algorithm starts the search in the node where the source keypoint would be grouped, and the user specifies the number of nodes that should be explored to find the nearest match [21].

Another method is to use k-means tree. It takes K descriptor points as centres and clusters the remaining points to the closest centre. A tree is created by splitting the cluster into smaller ones. To match keypoints, the branch with the smallest distance to the source

point is used to search for the nearest neighbour [21].

When the descriptors are binary such as in ORB or BRISK, in OpenCV is recommended to use Locality-Sensitive Hashing (LSH) [26]. LSH consists of applying hash functions to the descriptors. These functions create a code that defines the descriptor. The code is similar when two descriptors are similar. Each descriptor is put into a container which is accessible by the code, forming a table of containers. Feature matching is performed by creating the hash code in the source keypoint and looking through the table to match the same or most similar code [33].

4.5 Affine transformation

An affine transformation is a linear 6 degrees of freedom transformation followed by a translation that can be computed from at least three-point correspondences. When calculating the transformation, the affine transformation keeps the parallel lines, the ratio of the line length, and the ratio of the area. Therefore, the transformation between two sets of points depends on the initial orientation. Still, their position on the plane provides a stable transformation when the points are concentrated in a region of the image [34]. In Equation 4.22, the transformation matrix between two sets of points is presented, where t_x and t_y is the translation applied and the $a_{i,j}$ elements represent a combination of rotation and scaling transformation.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.22}$$

Chapter 5

Dataset and performance evaluation criteria

The RAU system is a product in development for a unique application. The data required for development and testing are not available from existing datasets. For this reason, a dataset has been created. The dataset is an essential part of the development and testing of this project since on the one hand, it will aid in anticipating failures, making corrections and improving the solution. On the other hand, it will be used to evaluate the performance of the proposed algorithm.

5.1 Constraints and assumptions

To create the dataset, the variables that affect daily scans have been considered. They were classified into patient variables, referring to all the factors in the patient itself that change from person to person, and environmental variables, being the factors in the field of view of the cameras that can change and are not related to the patient.

5.1.1 Patient variables

The RAU system was designed to perform antenatal scans. The patients will be pregnant women, however, the following aspects will differ:

- Skin tone
- Height

- Body shape
- Size of the abdomen
- Clothes colour and texture
- Additional components and circumstances such as jewellery, tattoos, malformations or diseases.

Since the product is still in development, for safety matters, the patients composing the dataset were not pregnant women, it was a pregnant doll used for the company to develop and test. Consequently, it is assumed that the skin tone, height and body shape are constant and that additional components or circumstances are not present. In summary, on the patient side, the dataset will be made from the same subject considering different clothes colours and textures.

5.1.2 Environment variables

The following factors can change during the day while performing different scans:

- Light changes
- Shadows
- Distance of the bed to the camera
- External people such as the sonograph or attendants can also be in the frame.

Light changes and shadows could be considered the same factor since both change pixel intensities over time. However, in this case, light changes refer to the source of illumination, so given the case where light is constant, human movements can produce different shadows throughout the scan.

It is important to consider that the product is expected to be placed in a clinic room with constant lighting. Nevertheless, for the recording of the dataset, the system was located at the LSR office next to a window, thus, the dataset will have light changes due to the sun's movement. On the environment side, it is assumed that the bed will always be at the same distance to the camera since only 2D movements are considered and that there will be a maximum of one external person in the view to control the system and the doll.

5.2 Materials

For the recording of the dataset, it was needed the RAU system presented in Figure 2.1 with the three cameras calibrated, the doll, a variety of clothes, and a rope to apply movements to the doll without a person being on the frames.

5.3 Setup before recording

This section presents the development to set up and record the dataset. Overall, the processes can be divided into the software adaptation to be able to record videos while the RAU system was running and the preparation of the scene.

5.3.1 Software adaptation

The current RAU system is composed of two units: The External Control Unit (ECU) and the Control Interface Unit (CIU). The ECU is the main unit which controls the communication and operations between all inputs and outputs of the system to ensure integration between all system components. The CIU controls the user interface, displaying different views depending on the sonographer's inputs and the robot's behaviour. Both units are composed of ROS nodes mainly programmed with C++. In the ECU it was integrated an additional ROS node programmed in Python to record data.

Following the diagram in Figure 5.1, the ECU starts the cameras when the user has pressed the "Start" button in the interface. Each camera frame is published through a CompressedImage message. At every iteration, the ECU sends a new frame per camera while handling all the other processes until the user presses the "Stop" button. Furthermore, an extra button was added to the interface called "AutoMove". When "AutoMove" is enabled, the haptic device is stopped and commands are automatically sent to the robot to move from its current position towards the left at the speed the user had set. The user has to press the "AutoMove" button again to stop the automatic motion and use the haptic device to control the robot's motion. This additional feature facilitated moving the doll while the robot was also moving. The node created to record the dataset "Dataset Node" listens to the CompressedImage messages and stores each frame into a video file created with VideoWriter from OpenCV [26].

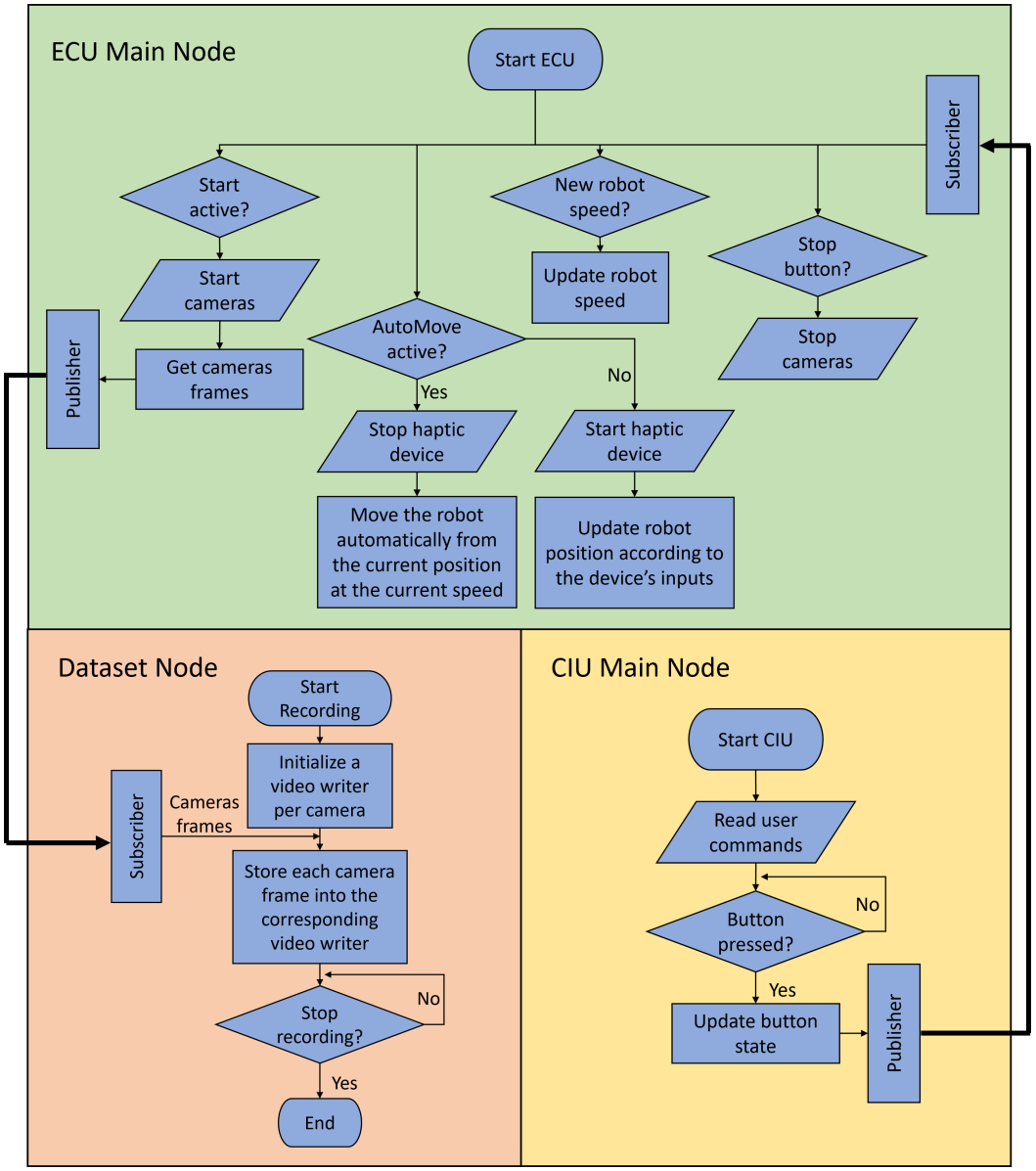


Figure 5.1: Diagram of the main software components to record the dataset.

5.3.2 Preparation of the scene

To simulate the doll as a different patient in each video, four tops and four bottoms were bought at a second-hand store. They all had different colours, textures, patterns and sizes to have a variety of data. Together with the clothes the doll wears by default, there were a total of 25 possible combinations. The doll was initially dressed and placed on the bed,

always at the same distance from the camera. A rope was attached to each of the wrists to be able to move it while not being on the frame. The doll was moved at different speeds pushing and pulling from the feet and by using the ropes.

5.4 Recordings

A total of 54 videos were recorded from each camera with a total of 22 different clothes combinations. Even though the project focuses on a single-camera approach, the dataset provides equivalent recordings of each of the cameras that could be used in future work, avoiding the need to record a new entire dataset.

The recorded videos have three categories. First, the robot moves but the doll stays still. Then the "AutoMove" button was pressed to move the doll while the robot moved. Finally, the "AutoMove" button was pressed to stop the robot and move only the doll. It is important to note that some recordings were done independently, so there is a short video per category. But others were done subsequently, getting a single long video for all three scenarios. Moreover, the robot was sent to the **Home** position and some videos of doll movement without obstacles were recorded to study the influence of occlusion in the algorithm performance.

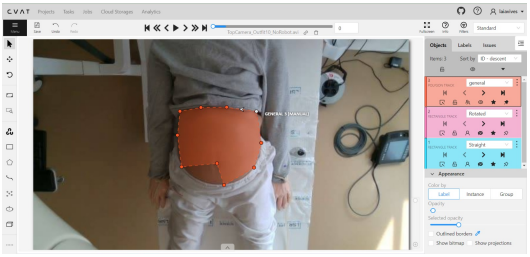
The dataset was divided into development data and verification data. This means that to develop the proposed solution, about 80% of the recording videos were used. The remaining 20% of the videos were stored and only used for final test purposes. The selected verification data are videos that present a singularity compared to the whole dataset. For example, the doll does a specific pose throughout the video, the ropes are not used or someone external appears in the frame to relocate the doll due to an error.

5.5 Ground truth extraction

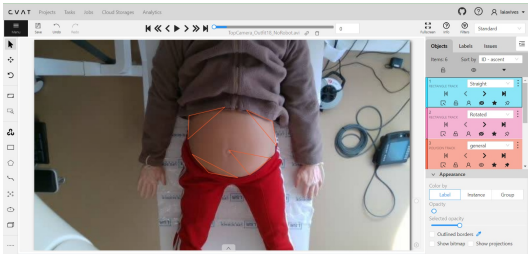
The creation of the dataset provides a closer result to the real-world performance. However, there is the absence of an existing baseline information that can be used as a ground truth. Baseline annotations play an important role since they will help determine the accuracy and robustness of the different algorithms. To address this issue, the Computer Vision Annotation Tool (CVAT) [35] was employed to manually create a ground truth. CVAT creates an XML file of all the annotations in the video that can later be loaded and interpreted in Python. Furthermore, it provides tools to enhance the accuracy of the annotations such as indicating the size in pixels or the degrees of rotation of a rectangle.

Since the proposed method is based on bounding boxes and keypoints, boxes and points were annotated using CVAT, see the examples in Figure 5.2. Considering that it is not

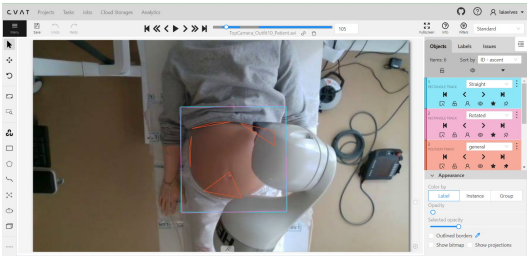
efficient to manually annotate the position of every pixel in the image, ten points within the region of interest were chosen, looking for potential features (Figure 5.2a). However, it was challenging to keep the accuracy when each point had to be placed independently due to rotational movements. For this reason, in the remaining videos, four triangles were used to store their three vertices, getting 12 points instead (Figure 5.2b). The use of triangles helped increase the accuracy of the annotations due to the focus on smaller regions rather than the entire abdomen. In the case of occlusion, it was aimed to guess the position of the occluded points as seen in Figure 5.2c. For the bounding box approaches, a box of 180x180 pixels was used to enclose the abdomen. Since single-bounding box approaches cannot consider rotation, the annotations were done with a box always with the same orientation, and a box that rotated according to the patient rotation to evaluate the rotational effect on the performance Figure 5.2d.



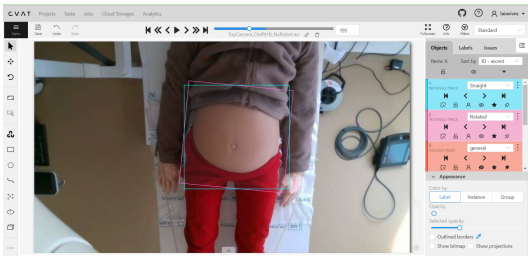
(a) 10 point selection ground truth example.



(b) 12 point selection ground truth example.



(c) Occlusion scenario ground truth example.



(d) Bounding boxes ground truth example.

Figure 5.2: Examples of use of CVAT to annotate the ground truth.

Annotated videos with keypoints

Since the annotation of single points is a tedious task and the guess of potential detected features will affect the performance results since the detectors could not detect that specific point, the videos where the robot and the patient move simultaneously, classified as AutoMove, were not considered. The reason is that feature matching does not depend on a time variable, it finds correspondences between two frames. Therefore, the amount of occlusion caused by moving the robot is equivalent to the occlusion caused by moving

the patient or the occlusion caused by moving both. Furthermore, the videos where the patient was static and the robot moved were also not annotated. In this particular scenario is more reliable to take as reference points the first frame detected points since it is known that their location will not move.

For the remaining scenarios, annotations were made every 5 seconds to only two outfits of the development set, presented in Figure 5.3 and Figure 5.4. Seen in the images is a scenario where the clothes have similar colours to the background, which can be challenging to distinguish the patient, and a scenario where the clothes have high colour contrast with the background. This selection can provide details on how the clothes can affect the performance of the algorithms. Overall, the keypoints annotated ground truth is composed of four videos: patient movement without occlusion and patient movement with static occlusion for each clothing.

Figure 5.3: Outfit with low colour contrast between the clothes and the background.

Figure 5.4: Outfit with high colour contrast between the clothes and the background.

Annotated videos with boxes

For the bounding box approach all videos in both, the development and the verification set, were annotated. In the short videos of the development set, 10 annotations were made every 5 seconds (every 75 frames) starting on the first frame. Thus, the videos were annotated until the second 45, unless the video was shorter than 45 seconds, the annotations were done every 5 seconds until the end of the video. For the long videos of the development set, the annotations were split through the three parts of the video, robot movement, Automove and patient movement. At roughly the beginning, the middle and the end of the video, between 5 and 10 annotations were made every 5 seconds. The number of annotations was dependent on the length of each of the scenarios.

The bounding boxes were also annotated in the verification set. In these videos, the annotations were made every second, every 5 seconds or every 10 seconds depending on the length of the video. The length of the video can vary from 30 seconds to 4 minutes.

5.6 Evaluation metrics

To fairly compare the applied approaches to the dataset different metrics have been used depending on the approach under evaluation, which can mainly be divided into bounding boxes and keypoint approaches. The chosen metrics seek to study the accuracy of the tracking, stability and robustness to occlusion of all algorithms.

5.6.1 Metric for bounding box approaches

The Intersection over Union (IoU) is a widely used metric to quantify the accuracy of the tracking algorithm [36]. Here R_t^G is the ground truth region at frame t and R_t^P is the area of the predicted box at frame t, the IoU measures the overlap between both regions with respect to the union of the two boxes. Since the regions are composed of pixels, the IoU can be calculated by the Equation 5.1. The true positive (TP) parameter represents the pixels matching the ground truth region. The false positive (FP) parameter is defined by the pixels within the predicted box that are not part of the overlap area. The false negative (FN) parameter represents the pixels part of the ground truth that are not in the overlap. A tracking algorithm will be accurate when the majority of the pixels correspond to the TP parameters, giving an IoU value close to 1 [37]. Examples of the region overlap and its parameters are illustrated in Figure 5.5.

$$IoU = \frac{|R_t^G \cap R_t^P|}{|R_t^G \cup R_t^P|} = \frac{TP}{TP + FN + FP} \tag{5.1}$$

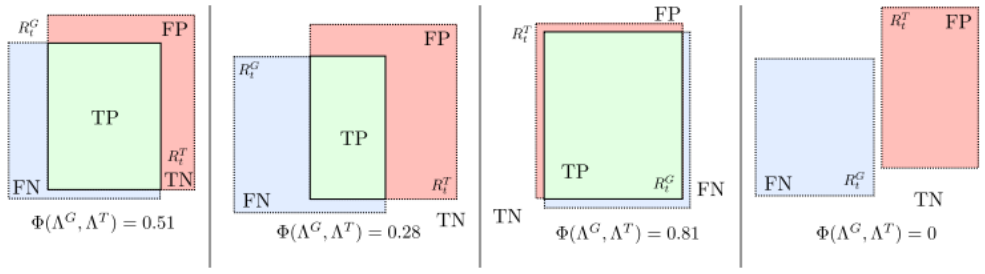


Figure 5.5: Examples of overlap between the ground truth and the predicted box. Source: [37]

5.6.2 Metrics keypoints approaches

When working with keypoints, the main metric used to measure the accuracy is the Euclidean distance [16, 38]. The Euclidean distance is a metric to measure the accuracy of an algorithm, which is calculated as in Equation 5.2,

$$d(pixels) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5.2}$$

where x_1 and y_1 represent the ground truth keypoint position, while x_2 and y_2 represent the predicted keypoint position by the algorithm. A small distance, d indicates that the ground truth and estimated points are similar, implying a high-accuracy approach.

Apart from the Euclidean distance, the repeatability and the percentage of correct points/-matches were also considered to study the precision.

On the one hand, repeatability refers to the number of frames where the same keypoint is detected when there is no occlusion [38], defined as in Equation 5.3. On the other hand, the percentage of correct points/matches refers to the number of detected points/matches that correspond to the ground truth as described in Equation 5.4 [31]. In this project, if the Euclidian distance between the ground truth and the detected point is higher than 2 pixels it will be considered a wrong match as Steffen Gauglitz et. al. [31] did in their study.

$$\%\ \text{repeated points} = \frac{\text{number of detected points in all frames}}{\text{number of detected points in the reference frame}} * 100 \tag{5.3}$$

$$\%\ \text{correct points/matches} = \frac{\text{number of correct points/matches}}{\text{total matches}} * 100 \tag{5.4}$$

5.6.3 Robustness study

The previously presented metrics were evaluated in different challenging scenarios to measure the robustness of the algorithms in gradually challenging situations. For example, the first level considers patient movement with no robot occlusion. In this scenario, the algorithms should ideally result in high accuracy and precision performances. In the second level, a static occlusion scenario is presented by having the robot in a static position of the frame. The next level would consider a dynamic occlusion by having the robot move on the frame.

Chapter 6

Development of the patient visual tracker

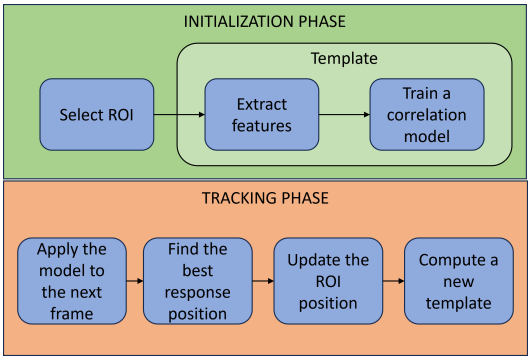
Kernelized correlation filter (KCF) trackers stand out for their high-speed computation while reaching accurate tracking performance. The use of the frequency domain and the integration of the 'kernel trick' explained in chapter 4, efficiently reduce the computational cost, making KCF suitable for real-time tracking. Nonetheless, the performance drops when occlusions appear in the view. Since the template model updates at every frame to incorporate appearance changes of the target over time, when the target is progressively occluded, the model gradually contains more features belonging to the occlusion, starting to track the occlusion instead. This chapter presents the development of a KCF tracker that integrates occlusion handling to improve its performance by integrating feature matching techniques [16]. An overview of the incorporation of the occlusion handling approach into the default KCF tracker pipeline is first introduced. To enhance the accuracy of the performance in the RAU system, the feature-matching methods presented in chapter 4 will be evaluated in the dataset videos to select the most suitable detector, descriptor and matcher for the RAU system. The chapter concludes with the adapted proposed solution developed in this project, considering the modifications and discrepancies with the reference approach [16].

6.1 Occlusion handling overview

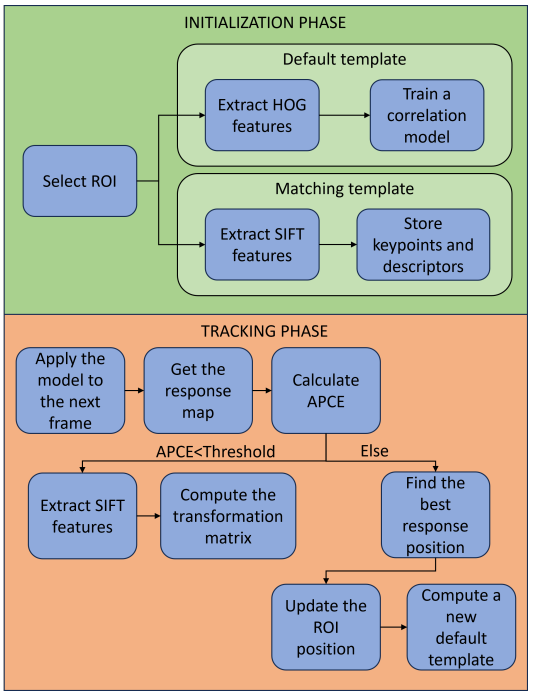
In Figure 6.1 the integration of the occlusion handling approach in the default KCF tracking pipeline can be observed. The tracking consists of two main parts: the initialization phase, highlighted in green, and the tracking phase, highlighted in red. By default (Figure 6.1a), in the initialization phase, the target to follow is selected by enclosing it within a bounding box. The image pixels within the bounding box are known as the Region Of Interest (ROI). In this project, the target will be the abdomen of the patient. A template

is created by extracting HOG features from the ROI and training a correlation filter that provides a high response at the target's position. The tracking phase consists of looping detecting the target in a new frame and updating the position of the ROI. The tracking is achieved by applying the trained filter to the next frame to find the location in the image with the highest response and update the box to that position. A new template is created by extracting features of the updated position to update changes in the target's appearance.

To handle the presence of occlusions (Figure 6.1b), the approach consists of computing a feature detector in the initialization phase to identify salient points inside the ROI and store them as a matching template. During the tracking phase, the resultant response map when applying the trained filter to a new image is used to calculate the Average Peak to Correlation Energy (APCE) parameter and detect the presence of occlusions. If occlusions are detected, features will be extracted from the new frame. Matching will be performed between the matching template and the new frame. The transformation matrix will be used to map the reference box into the new frame. The bounding box of the tracker will be overwritten to update the template model.



(a) Default KCF workflow



(b) Integration of feature-matching with SIFT into the KCF workflow

Figure 6.1: Comparative workflow diagrams between the original KCF approach [14] and the proposed method to increase its robustness to occlsuions [16].

6.2 Evaluation of feature matching approaches

6.2.1 Evaluation of feature detectors parameters

When using a keypoint detector in OpenCV, a set of parameters can be customized to adapt the detector to the system. The goal of this test is to find the most suitable parameters for the RAU system when detecting a region containing the abdomen. The detectors that do not detect a minimum of three keypoints will be discarded since three keypoints are the minimum to compute the transformation matrix between two frames.

Inspired by Andres Marmol et. al study [31], AGAST, BRISK, CenSurE, FAST, ORB, and SIFT feature detectors were computed using OpenCV. Andres's study focuses on knee arthroscopy images, which challenge smooth and unstructured images. This condition is also presented when looking for features in the abdomen since it is also a smooth and unstructured region. Therefore, the parameters used in the knee approach could apply to the RAU system. These parameters were compared to the default parameters used by OpenCV and self-tuned parameters. when self-tunning parameters it was looked to detect more keypoints by dropping the restrictions caused by contrast thresholds, for example. In Table 6.1 are shown the changed parameters in the paper and the self-tunning with respect to the default values in each method.

Method	Parameters	Default	Paper	Custom
AGAST	threshold	10	10	8
BRISK	thresh	30	17	10
CenSurE	maxSize	45	45	30
	responseThreshold	30	5	5
	lineThresholdProjected	0	10	5
	lineThresholdBinarized	8	8	8
	suppressNonmaxSize	5	6	5
FAST	threshold	10	12	8
ORB	scaleFactor	1.2	1.3	1.3
	edgeThreshold	31	30	15
	fastThreshold	20	16	10
SIFT	nOctaveLayers	3	3	2
	contrastThreshold	0.04	0.01	0.01
	edgeThreshold	10	10	5
	sigma	1.6	1.6	1.2

Table 6.1: Used parameters for each of the methods. The parameters that are not specified are the default ones from OpenCV.

To evaluate the different parameters and methods a single Python code was made. First,

the region of interest was manually selected. In all cases, it consisted of a 180x180 pixels region containing the abdomen and a bit of the surroundings as seen in Figure 6.10. Then, a loop was created where each of the detectors was computed by applying the different parameters and storing the number of detected keypoints in an Excel file. Once the three parameters had been computed, the repeatability metric was calculated for every pair, including the result in the Excel file. Five videos were tested on frames 1, 150 and 450. The frames are equivalent to the first frame, at the second 10 and the second 30 of the video. In Figure 6.3, the first frame of the five tested videos is presented. As seen in the images, the doll was wearing different clothes to study the overall performance instead of basing the results on a unique case scenario. It is important to mention that for this evaluation there was no occlusion caused by the robot. The test focuses on detecting keypoints on the ROI, if there is occlusion it is expected to detect fewer keypoints belonging to the patient.

Figure 6.2: Example of the 180x180 region

(a) First frame of video1

(b) First frame of video2

(c) First frame of video3

(d) First frame of video4

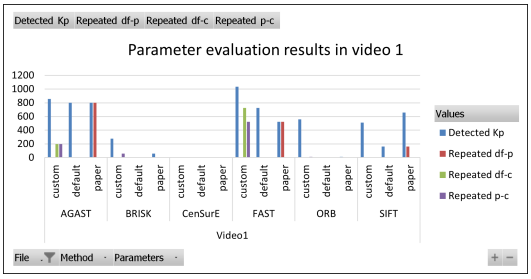
(e) First frame of video5

Figure 6.3: First frame of the videos used to evaluate the AGAST, BRISK, CenSurE, FAST, ORB and SIFT detectors.

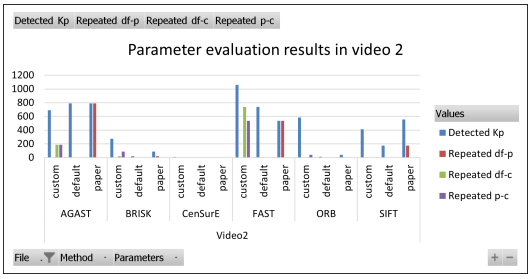
To evaluate this test the number of keypoints and the repeatability metric were considered. With the number of keypoints it can be determined if the detector has enough keypoints to compute the transformation matrix. The ratio of repeatability between two sets of parameters gives information about the percentage of detected points that are common when using different parameters. In feature detection, it matters the number of features but it also relevant the stability and reliability of those features. Detecting the same points across multiple parameters ensures the robustness of the chosen parameters.

In Figure 6.4 the results are presented as a graphical representation. For each video, the sum of the detected keypoints and the sum of the repeated points per pair of parameters of the three tested frames is shown according to the tested parameters and method. In Table 6.2 is seen the average detected keypoints and average repeatability in %. In all tested videos, AGAST, FAST and SIFT detect the largest amount of keypoints. However, the repeatability percentage in AGAST and SIFT when using the custom parameters is low, at 30,3% and 0,0% respectively, which drops the reliability in detecting robust features since the repeatability between the default parameters and the paper is at 100,0%. Contrarily, BRISK, FAST and ORB have a high percentage of repeatability when using the custom parameters while detecting more points. Regarding CenSurE, it can be observed that with any of the three tested parameters, the average number of detected keypoints reaches 4, so it is discarded from a possible detector to perform feature matching.

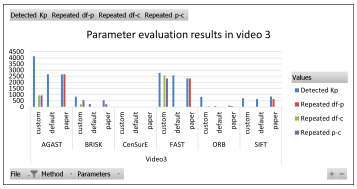
As a result, AGAST with the default parameters, BRISK with the custom parameters, FAST with custom parameters, ORB with custom parameters and SIFT with the paper's parameters will be used in the next test to study the feature matching performance.



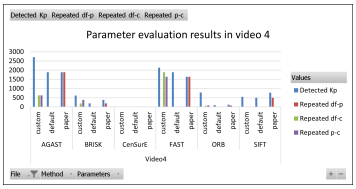
(a) Results of video 1



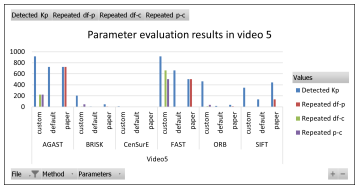
(b) Results of video 1



(c) Results of video 1



(d) Results of video 1



(e) video5

Figure 6.4: Results of the evaluation of the different parameters applied to SIFT, ORB, AGAST, FAST, CenSurE and BRISK detectors in five different videos

Table 6.2: Average test1 results per method

Method	**Average detected keypoints**			**Repeatability average [%]**		
	Default	Paper	Custom	Default-Paper	Default-Custom	Paper-Custom
AGAST	457.5	457.5	620.7	100.0	30.3	30.3
BRISK	30.1	75.6	147.5	86.4	86.4	99.9
CenSurE	0	0.2	2.9	0.0	0.0	10.0
FAST	438.9	368.5	528.9	100.0	100.0	100.0
ORB	13.6	21.9	214.7	66.8	54.4	80.9
SIFT	107.8	219.1	168.9	100.0	0.0	0.0

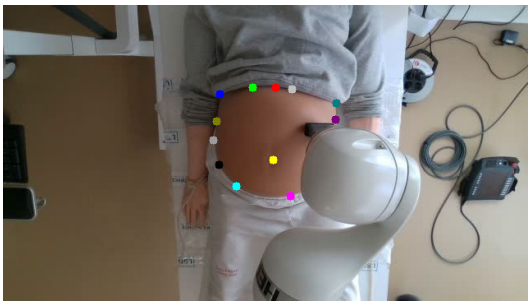
6.2.2 Evaluation and selection matching method

To evaluate and compare the performance of the introduced feature-matching approaches, a total of three tests were performed. The tests seek to study the performance of each of the components in feature matching. Therefore, the first test focuses on evaluating the parameters that define the detectors. The second test evaluates the performance of each detector with the selected parameters. The third test analyzes the matching performance of the different feature detectors and descriptors using the brute-force versus the Flann matcher in the three levels of complexity.

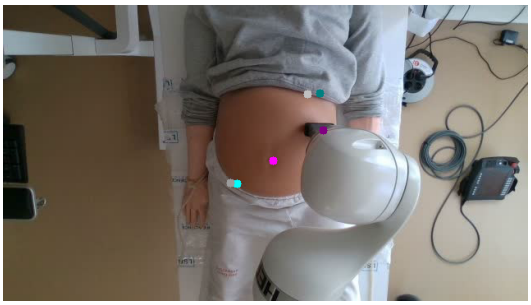
Test2: Evaluation of the feature matching performance with respect to annotated ground truth

With the remaining detectors, it was evaluated the matching performance in the videos with annotations made with CVAT. First of all, the detected keypoints were described. In the case of BRISK, ORB and SIFT, were used the descriptors integrated in the detectors. For AGAST and FAST, in which OpenCV does not integrate descriptors, were described with BRISK and ORB since they have the same data type. Therefore, the matching performance was analyzed for AGAST described with BRISK and AGAST described with ORB. The equivalent was done for the FAST detector.

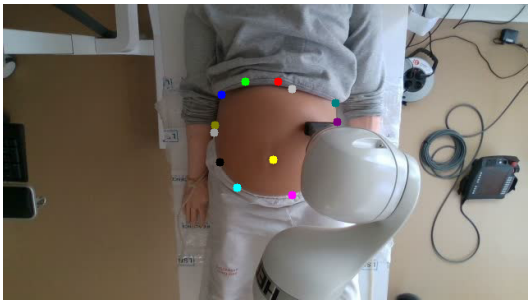
Frames 1, 150 and 450 from the four annotated videos were taken to analyze the performance. In the first frame, the detectors were run in the ROI alike Figure 6.10. The Euclidean distance was computed between each detected point and the ground truth points to find the best representation of the ground truth between all candidates. The detected points that were not selected as ground truth were discarded. In Figure 6.5 are presented examples of the selected points to be matched according to the smallest Euclidean distance to the ground truth. The manually annotated ground truth keypoints are shown in Figure 6.5a. Twelve points were manually selected by guessing where could be relevant features. It was expected that not all of them would be detected as it happened in Figure 6.5b, where some of them were not detected, or two ground truth points were assigned to the same detected keypoint, such as yellow and pink points.



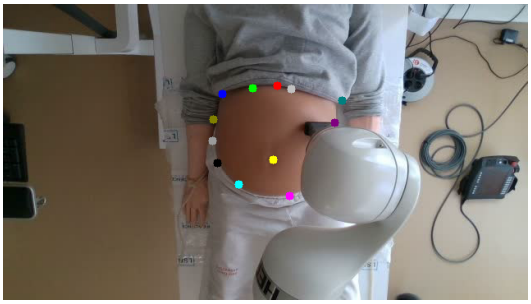
(a) Annotated ground truth



(b) Best ground truth representation for AGAST+ORB



(c) Best ground truth representation for FAST+BRISK

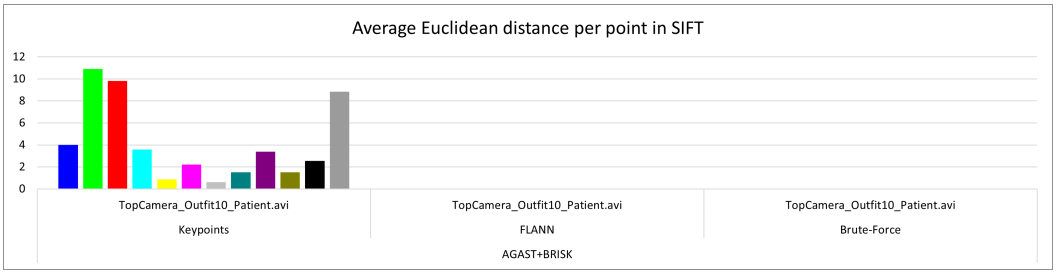


(d) Best ground truth representation for SIFT

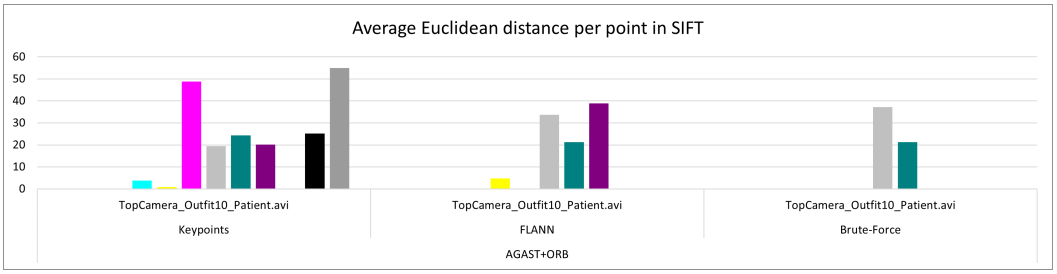
Figure 6.5: Examples of the selected keypoints to match in frames 150 and 450 according to the ground truth.

In frames 150 and 450 the detectors and descriptors were run in the entire image. Subsequently, Brute-Force and FLANN were computed to match the first frame selected keypoints to the detected points in frames 150 and 450. The Euclidean distance was calculated between every matched point in frames 150 and 450 to the annotated ground truth of the same frames. In Figure 6.6, the Euclidean distance between the annotated points and the detected or matched points for one video file with static occlusion is presented. The left group represents the points in the first frame. The middle and right groups represent the average matched points in frames 150 and 450 when using Brute-Force and when using FLANN respectively. Each colour in the graphics belongs to a point, according to Figure 6.5a. It is important to mention that on the one hand, a low Euclidean distance in the first frame but a high distance in the matches imply a wrong match since the point was properly detected in the first frame, but was not close to the annotated ground truth in the matched frames. On the other hand, a high Euclidean distance in the first frame and a high distance in the matched frames do not directly reflect a wrong match. It can be that none detected points were placed in the annotated location, so the closest one was taken and that one was correctly matched in the subsequent frames. For this reason, visual inspection was also applied to verify the results by storing the frames of all matches. If the Euclidean distance is 0 (with no bars visible), it means that there were no detected points. It does not mean that it was perfect.

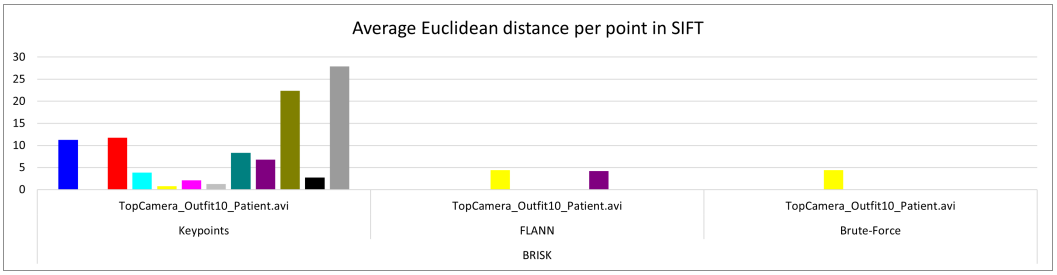
From the graphics is seen what keypoints were matched in frames 150 and/or 450 and how accurate was the match. It is highlighted that for this video no match was found for AGAST+BRISK and FAST+BRISK. Furthermore, it can be seen that in ORB (Figure 6.6d) the Euclidean distances in the first frame are relatively small. However, there is a match with an Euclidean distance higher than 100 pixels, which belongs to a wrong match. Therefore, in this video, ORB also did not have any match. Contrary, all the detected and matched points using SIFT had an Euclidean distance of a maximum of 10 pixels with respect to their ground truth. Particularly, the first point (blue, (0,0,255) in RGB) was matched and had an Euclidean distance of less than 2 pixels for either Brute-Force and FLANN Figure 6.6g.



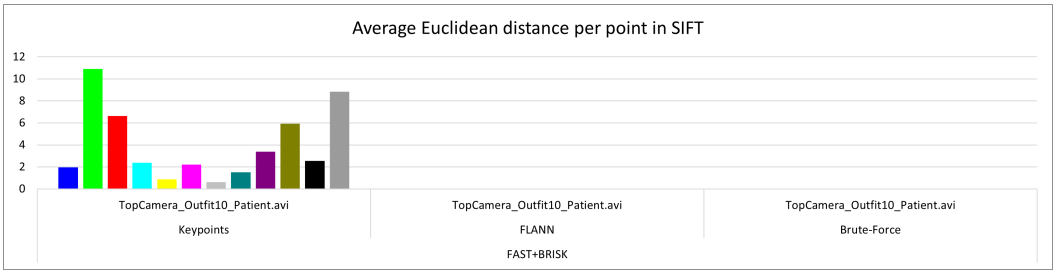
(a) Average Euclidean distance per point for AGAST+BRISK.



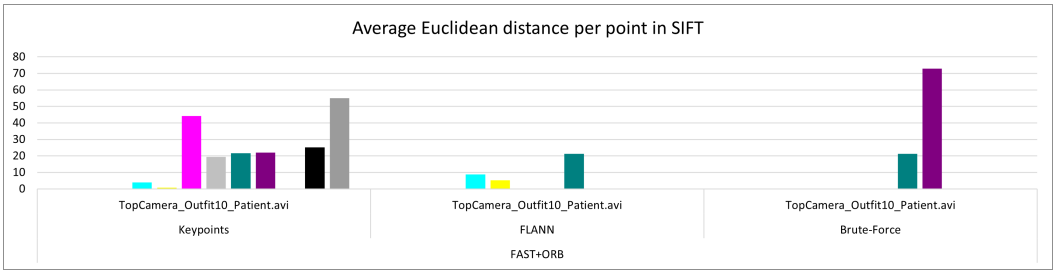
(b) Average Euclidean distance per point for AGAST+ORB



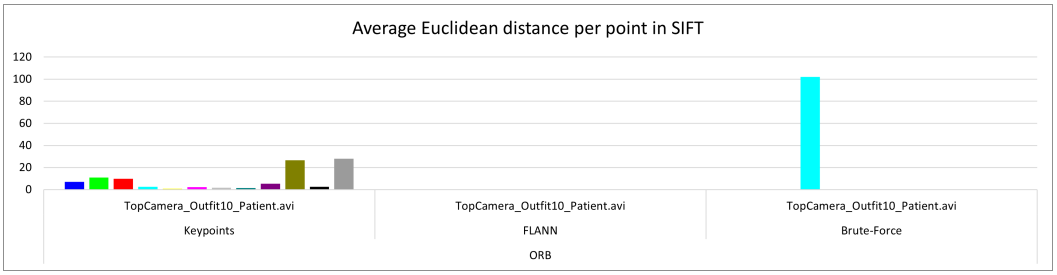
(c) Average Euclidean distance per point for BRISK



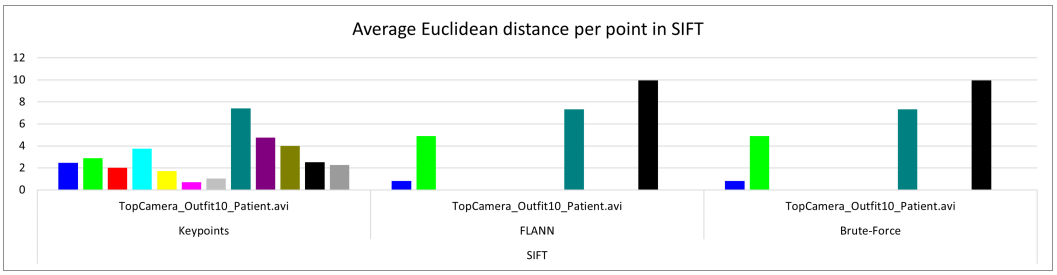
(d) Average Euclidean distance per point for FAST+BRISK.



(e) Average Euclidean distance per point for FAST+ORB.



(f) Average Euclidean distance per point for ORB



(g) Average Euclidean distance per point for SIFT

Figure 6.6: Average Euclidean distance per detected point for each tested method when using Brute-Force and FLANN. Results of one of the four evaluated videos.

Since the points are not directly comparable throughout the evaluated videos due to being

placed in different positions or belonging to different patients, the percentage of correct matches and the percentage of repeatability metrics were used to evaluate the performance. Taking into account the human error when annotating the detected points in the first frame, as well as verification via visual inspection, it is considered a wrong match when the Euclidian distance is higher than 30 pixels. Thus, if from twelve selected points, two are matched in frame 150 and one has an Euclidean distance higher than 30 pixels, the method in that frame will score 50% of correct matches. The repeatability in this case is considered a boolean metric. To highlight the methods that found the same keypoint in frame 150 and frame 450, for each point, the method scored 100 if the point was correctly matched in frame 150 and frame 450, otherwise, it scored 0. In Table 6.3 are presented the average results of each tested method. From the results, AGAST+BRISK and FAST+BRISK are discarded since any point got matched. It is also observed that ORB got the lowest percentage of correct matches with 66.7% for Brute-Force and 62.5% for FLANN, but higher repeatability than BRISK, which has a 100 % correct match but any point was repeated.

Table 6.3: Average test2 results per method

Method	**Average of Good Matches [%]**		**Repeatability Average [%]**	
	Brute-Force	FLANN	Brute-Force	FLANN
AGAST+BRISK	0.0	0.0	0.0	0.0
AGAST+ORB	80.0	66.7	2.1	2.1
BRISK	100.0	100.0	0.0	0.0
FAST+BRISK	0.0	0.0	0.0	0.0
FAST+ORB	83.3	91.7	2.1	4.2
ORB	66.7	62.5	2.1	2.1
SIFT	91.1	80.0	2.1	2.1

Overall, a high percentage of correct matches is required to get an accurate transformation matrix. The repeatability metric is also relevant since it indicates that a point is stable and robust throughout the frames, giving higher reliability when computing the transformation matrix, but it can be affected by occlusions. For all of this, AGAST+BRISK, FAST+BRISK and ORB were discarded. The remaining methods moved to the last test to select the best feature-matching approach for the RAU system. However, it is remarkable that in this test FAST+ORB achieved the best performance in terms of percentage of correct matches and repeatability.

Test3: Evaluation of the feature matching performance in dynamic occlusion scenarios

Given the ambiguity of the results in test 2 due to the human error when annotating the ground truth and the restriction of matching a specific set of points, it was necessary

to conduct a last test to choose the feature-matching approach. The test was performed in dynamic occlusion scenarios where the patient had a static position. Therefore, the first frame detected keypoints were the ground truth. Five videos from the dataset were selected for the test as seen in Figure 6.7. For each video, the ROI was selected in the first frame and a mask was applied on top of the robot to only consider points belonging to the patient. The previous test was performed but this time the detectors were not constrained to detect specific keypoints.

(a) First frame of video1

(b) First frame of video2

(c) First frame of video3

(d) First frame of video4

(e) First frame of video5

Figure 6.7: First frame of the videos used to evaluate the AGAST+ORB, BRISK, FAST+ORB and SIFT detectors and descriptors with Brute-Force and FLANN matchers.

To evaluate the performance of the feature matching approaches, the average number of matches, the average Euclidean distance between each matched point and the ground truth, the percentage of correct matches and the repeatability were considered. As explained earlier, to be able to compute the transformation matrix at least for matches is necessary. As it is observed in Figure 6.8, AGAST+ORB and BRISK when using the Brute-Force matcher are below 4 matches, not being suited for this solution. Furthermore, it is noticeable that SIFT gets an average of about 24 matches more than the other approaches.

In Table 6.4 are presented further details of the matching performance. Ideally, the Euclidean distance shall be zero if a point is matching the ground truth. In this case, FAST+ORB when using Brute-Force presents the worse accuracy with an error of 1.24 pixels. Omitting the discarded methods for not reaching the minimum amount of matches, AGAST+ORB and BRISK with FLANN achieve the highest accuracy with 0.77 pixels of average error. However, AGAST+ORB with FLANN has the lowest percentage of good

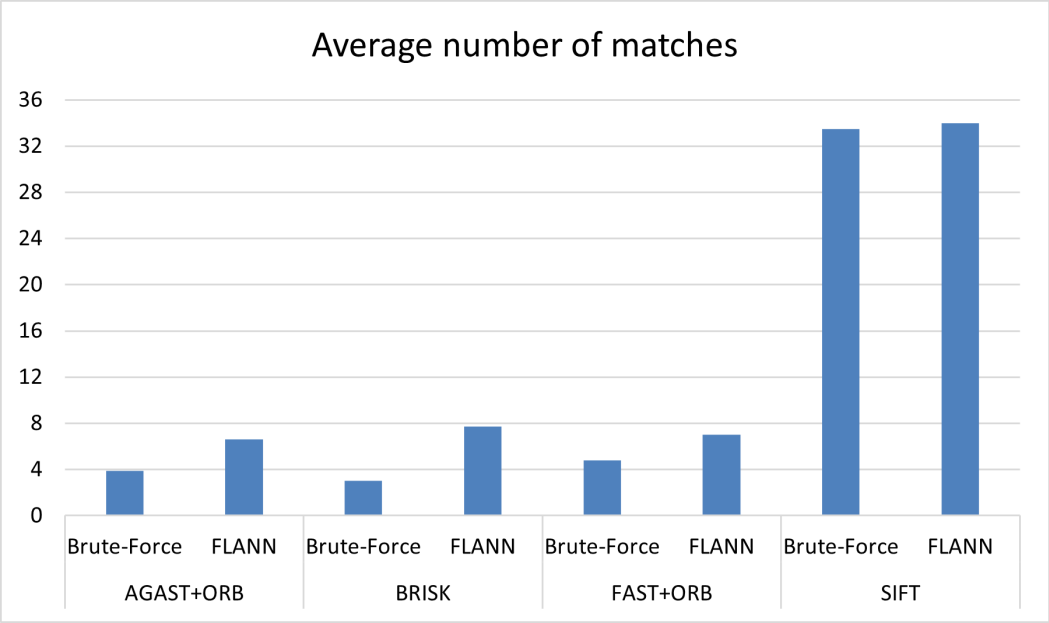


Figure 6.8: Average number of matches per method.

matches at 92.68 % and BRISK has the lowest repeatability percentage of 9.1%. According to the Euclidean distance error, the average of good matches and the average of repeatability, SIFT with FLANN matcher is the method that has a good balance between accuracy, percentage of good matches and repeatability. Moreover, analyzing the results for each video individually, SIFT with either of the matches was the only method that got more than four matches in all frames and videos, which correlates with the outstanding average of matches seen in Figure 6.8. To summarise, SIFT with Brute-Force was the feature-matching approach chosen to correct the KCF box.

Table 6.4: Average test3 results per method

Method	**Euclidean distance [pixels]**		**Average of Good Matches [%]**		**Repeatability Average [%]**	
	Brute-Force	FLANN	Brute-Force	FLANN	Brute-Force	FLANN
AGAST+ORB	0.72	0.77	100.0	92.68	5.7	27.88
BRISK	0.56	0.77	100.0	98.4	2.6	9.1
FAST+ORB	1.24	0.95	97.9	98.6	7.7	16.4
SIFT	0.83	0.86	98.2	98.2	16.2	11.7

6.3 Adapted solution for the RAU system

The pipeline presented in Figure 6.1b has been updated by incorporating SIFT as a detector and descriptor method, and Brute Force is used to match features as seen in Figure 6.9.

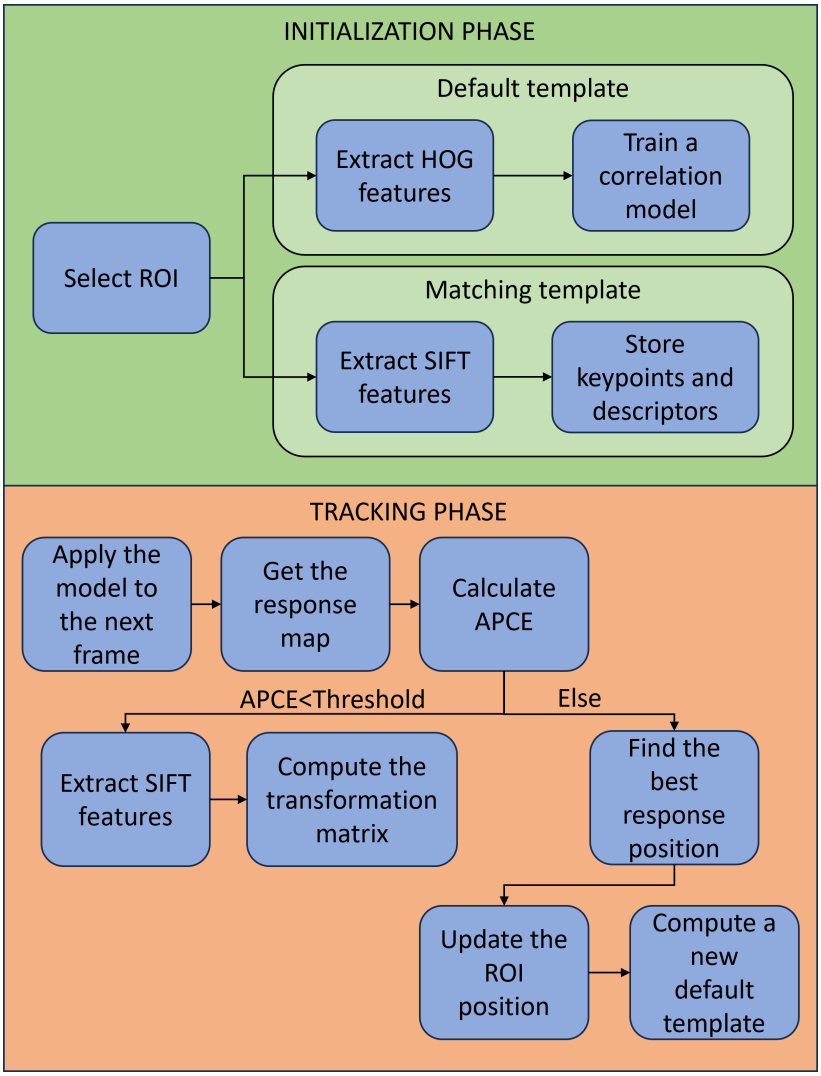


Figure 6.9: Diagram of the occlusion handling approach integrated into the default KCF pipeline.

Since the abdomen appearance is mostly textureless, the selected ROI includes part of the surroundings to enhance the detection of features. The visible contrast between the shirt and the abdomen and between the trousers and the abdomen can aid in detecting more reliable features to match. For consistency, in this project, the ROI is a box of 180x180 pixels centred on the abdomen as seen in Figure 6.10.

Figure 6.10: Example of the 180x180 region

The proposed solution is programmed in Python taking a handcrafted implementation of KCF as a baseline [22]. The tracker has three parameters to initialize, the types of features extracted which can be the grayscale pixel value or three-channel HoG, the fixed window bool which always has the same box size if true, and the multi-scale bool which adapts to scale changes in the target if true. The authors of KCF highlighted its robustness and high accuracy for the use of HoG, thus HoG features were chosen [14]. Since in a real scan is not expected the patient to move significantly from its reference position, even though the top camera in the RAU system is a bit tilted, the fixed window bool is set to True. Therefore, the multi-scale parameter is false.

Feature matching is integrated using the libraries from OpenCV [26]. When feature matching is required to correct the box, the affine transformation will apply rotation changes in the bounding box points. Since the KCF tracker does not consider rotated boxes, the centre of the corrected box is calculated to redefine the corrected box without any rotation applied.

The developed algorithm assumes the displacement between two consecutive frames is small. Therefore, if when correcting the box, the difference between the previous box centre and the current centre is higher than 50 pixels, the correction is discarded. It takes the position of the previous frame.

Chapter 7

Performance evaluation

The proposed approach, a tracker to follow the patient's movements while the robot is carrying out an ultrasound scan, is evaluated in this chapter. To fairly analyze the improvement of the proposed approach under the presence of occlusion, the default KCF tracker algorithm [22] was first evaluated. The evaluation provides an overview of the default tracker performance and a study of its main drawbacks when used in the RAU system. With the baseline performance of the tracker, the testing was repeated for the proposed solution.

The test setup was common for both approaches and consisted of five gradually more complex tests, using various different outfits. The outfits are presented in Appendix B. For each test, the Intersection over Union (IoU) metric was used to provide the percentage of similarity between the returned box by the tracker and the ground truth box. All videos from the development set were used, divided into the five tests according to their content. The first test evaluated the performance when there was no occlusion, the robot was not in the view and the patient moved. It consisted of 8 videos between 30 and 45 seconds. The second test studied the impact of a static occlusion. The robot was present in the view in a static position and the patient moved. It contained 9 videos of 45 seconds. In the third test, the impact of having a dynamic occlusion was studied while the target was static, so the patient was static but the robot moved. This test had 10 videos of 45 seconds as well. In the fourth test, the dynamic occlusion was also analyzed when both, the patient and the robot moved. There were 11 videos of 45 seconds. The last test focused on studying the performance in longer videos. Five videos between 2.40 minutes and 4 minutes were used with the robot always on view and with a mix of robot and patient movements. For further details on the ground truth annotations or the content of the videos, it is recommended to read chapter 5.

7.1 Evaluation setup

For the performance evaluation, a Python script was developed to compute the IoU and store the results while running the tracker for later analysis. At the beginning of the script was introduced the video path and the XML file associated with the annotations. Following the flowchart in Figure 7.1, when executing the code the XML file is read in the first place to store in a dictionary, *straight_rectangle*, the top left and right bottom vertices of the annotated rectangle for each of the frames. In another dictionary, *rotated_rectangle*, the corresponding vertices and the angle of the rotated rectangle per frame are stored. Each frame had a straight and rotated box but only ten of them were correctly placed. For this reason, after getting the annotations per frame, the video is initialized and the frame numbers with correct annotations are specified. While the video has frames, each of them is read. If it is the first frame, the ROI is selected by providing the *straight_rectangle* box associated with the first frame. Therefore, the first frame was not evaluated since it was used to initialize the tracker.

The tracker contains three initialization parameters, the type of features extracted, the use of a fix box and the use of a multi-scale bounding box. Both approaches use HOG features. However, the default tracker is analyzed triggering the multi-scale bounding box besides the proposed solution which is evaluated by setting the fixed-scale box. It was considered that even though the top camera is a bit tilted, the size of the abdomen will not significantly change, so having the multi-scale parameter active can help visualize drawbacks and comprehend the behaviour of the algorithm. In the proposed solution, the box size is fixed to smooth the integration of correcting the bounding box since the corrected box is constrained to a fixed size.

After the first frame, it is checked whether the current frame is in the list to evaluate or not. In case it is in the list, the IoU is calculated between the estimated box and the $straight\_rectangle_i$, being i the current frame. The IoU is also calculated between the estimated box and the $rotated\_rectangle_i$. The video file name, the frame number, the IoU respect to the straight rectangle and the IoU respect to the rotated rectangle were appended to an Excel file. After running the code in all videos, a unique Excel file was generated to analyze the results.

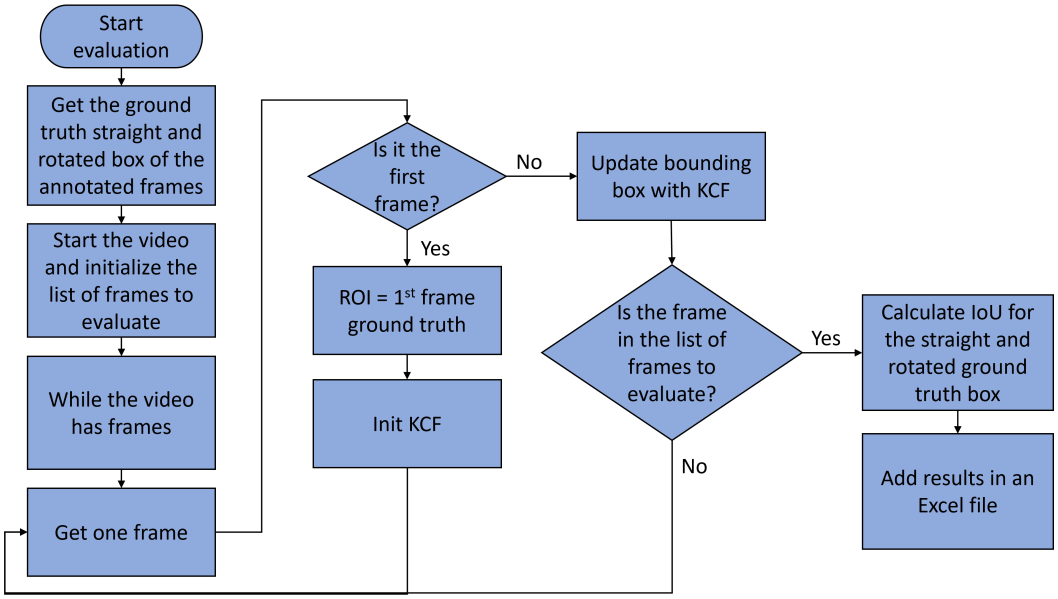


Figure 7.1: Flowchart of the developed Python script to evaluate the default performance of KCF tracker. The script takes the annotations from the XML file and runs the KCF tracker. A list is provided with the frames that shall be evaluated. While running the tracker, the IoU for the straight box and the rotated box are calculated for each frame under evaluation and the results are stored in an Excel file.

The authors of KCF used the position of the centre of the box instead to evaluate the performance. They considered a correct track when the centre was within 20 pixels. They argued that the box overlap penalizes the tracker when the size changes [14]. In this project, it was considered that the IoU is a more suitable metric for bounding box approaches. Considering that the ground truth box has a size of 180x180 pixels, taking the 20 pixels threshold as a 20-pixel shift in one direction, it would be equivalent to a minimum IoU of 80% to be considered a good track.

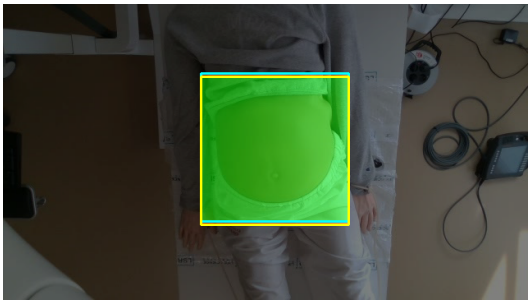
7.2 Evaluation of the default KCF performance

This evaluation of the default performance seeks to analyze and comprehend the behaviour of the tracker. In this section, the performance results of each of the five tests are analyzed. Due to the equivalent performance, the results of tests 2, 3 and 4 are presented together to avoid redundant explanations. However, the results of each test individually are provided in section A.1. Therefore, the following sections analyzed the performance when there was no occlusion when the robot was present in the view either static or dynamically and for the set of long videos. At the end of the section, a summary of the performance in all tests is provided and the main drawbacks of the tracker are highlighted.

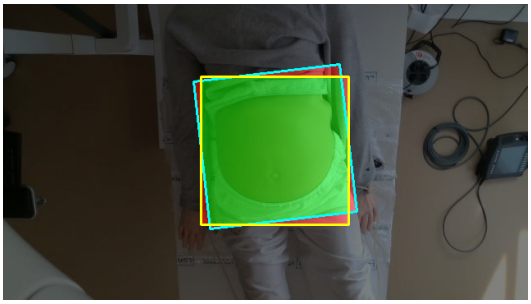
7.2.1 Results of test1: No occlusion

In the first test, all the videos in the dataset that do not have the robot in view were used to study the default performance of the tracker. In this case, there were a total of 8 videos between 30 a 45 seconds with one frame annotated every 5 seconds, every 75 frames. Considering that the first frame was not evaluated, between 5 and 9 frames were evaluated per video. In the end, a total of 54 evaluated frames were evaluated in this test. Since there is no occlusion, the abdomen shall be visible throughout the video, therefore a high accuracy is expected. However, the tracker uses HoG to extract features which contain gradient and orientation information making the features rotation invariant. Thus, if the patient rotates it could lose track even though the robot is not in the view.

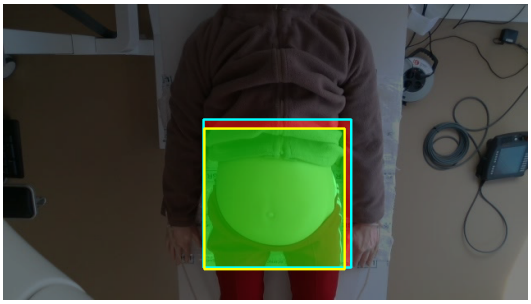
In Figure 7.2 are presented examples of the IoU metric in remarkable situations seen in the videos. In Figure 7.2a and Figure 7.2b the same frame of the same video is presented when taking the straight rectangle versus the rotated rectangle as a ground truth. Considering that the tracker was not designed to rotate the bounding box, it was expected that it would not overlap perfectly, it achieved an IoU of 89.0%. However, it is remarkable that the IoU concerning the straight box is 95.7%, which also indicates that even though HoG is not rotation invariant it can handle some rotational movements on the bed. Highlighted in Figure 7.2c is the fact that the tracker decreased the size of the box. This example shows that the multi-scale parameter is working properly and as explained in the previous section, it can be seen that the abdomen's size does not significantly change. Apart from human error, it should also be considered that some errors will also be due to the tracker adapting to the size. Nonetheless, visual inspection is always applied to the analysis to help determine to what extent a low IoU is due to human error or tracking failures. In Figure 7.2d, an example is presented of many of the evaluated frames. As it was expected, the tracker follows the target with a high IoU of 96.8% in this particular example.



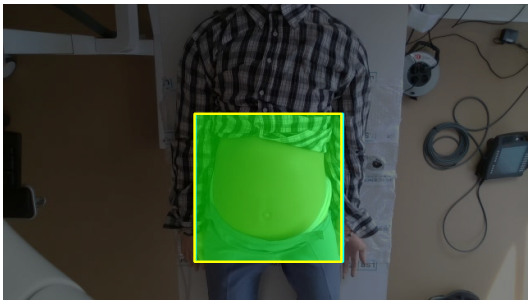
(a) IoU metric example with the straight ground truth rectangle where the patient rotated (Outfit 10).



(b) IoU metric example with the rotated ground truth rectangle where the patient rotated (Outfit 10).



(c) IoU metric example when the patient significantly moved downwards (Outfit 18).



(d) IoU metric example of the performance of the tracker in many other frames (Outfit 12).

Figure 7.2: Examples of the IoU metric in three different tested videos. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles). **a)** and **b)** show the same frame of the same video but **a)** presents the IoU metric with respect to the rotated box and **b)** represents the IoU metric with respect to the straight box. In **c)** it is observable that the tracking box decreased in size, while in **d)** a high overlap between the estimated and the ground truth boxes is observed.

Overall, as was expected the performance for the straight box is better than the rotated box. In the straight box IoU column of Table 7.1, it is seen that the performance is not lower than 90.9 %. The 9.1% of error could be attributed to human error when annotating the videos and the size changes of the box aforementioned. Furthermore, the accuracy is higher than the minimum 80% considered as a good track. As a result, the default KCF does not present weaknesses when there is no occlusion.

Table 7.1: Average test1 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 10	7	95.0	93.5
Outfit 12	5	95.1	95.6
Outfit 14	9	90.9	85.7
Outfit 15	8	93.9	92.5
Outfit 16	5	95.8	95.5
Outfit 18	6	92.1	91.9
Outfit 19	7	96.8	96.5
Outfit 20	7	92.7	92.6
Total	54	93.8	94.0

7.2.2 Results of tests 2,3 and 4: Occlusion

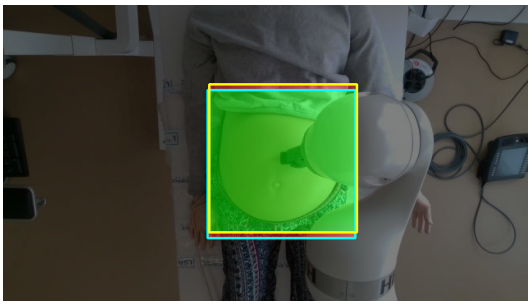
In these tests, the robot was in view considering three different scenarios. In test two the robot was static while in tests 3 and 4 it was dynamic. The tracker's performance in those scenarios was equivalent. It is important to consider that the initial robot's position was different for each of the videos and that the robot was already in view in the first frame. With the robot already in the first frame, the template filter will contain a part of the robot that can affect negatively when looking for the highest correlation in the next frame. Furthermore, the tracker slightly updates the template at every frame. This is done by interpolating the very first reference frame's features with the new frame's features, adapting to appearance changes. For that reason, it is expected that the performance will depend on the percentage of visible robot versus visible abdomen in the ROI. If the abdomen is more visible is expected to track towards the correct target, otherwise, it could track the robot instead. It is also important to mention that the videos have a variety of movements, in some of them the patient moves a lot while in others the patient does small shifts. The speed of movement and direction are also influential in the performance since they will affect the appearance of the abdomen between two consecutive frames. In Table 7.2 the number of videos and frames evaluated in each test as well as the average results are presented. As it is observed, the average accuracy is still higher than 80% which would classify the test as a good tracking.

Table 7.2: Caption

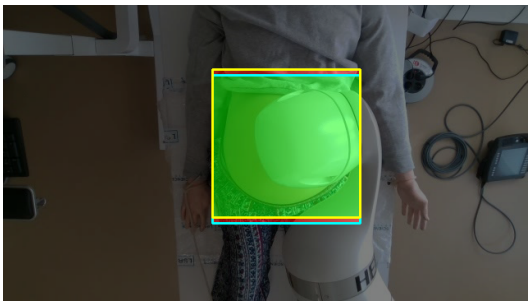
Test	**N videos**	**N evaluated frames**	**Mean IoU [%]**	
			Straight Box	Rotated Box
Test 2	9	77	85.2	85.7
Test 3	9	81	86.2	86.1
Test 4	11	108	84.3	84.1

Despite the good performance on average, the introduction of the robot in the frame presented significant changes in the tracking. The Outfit13 video of test 4 has been chosen to visualize the performance chronologically. In Figure 7.3 it can be seen that the robot gradually occludes the abdomen in Figure 7.3a and Figure 7.3b but the performance did not decrease. A possible reason could be that the robot does not translate, it rotates instead during those 5 seconds of motion. Probably the most reliable features in Figure 7.3a are located in the probe and the division between the abdomen skin and the clothes, due to the hard contrast with the skin. However, since the probe does not move, possible features on the left side of the abdomen were robust enough to keep the high response while slowly integrating the robot's features in the template. Despite this, there is a moment where the abdomen is barely visible, probably getting the highest response towards the robot which is also part of the template. In Figure 7.3c and Figure 7.3d the robot moves to make visible

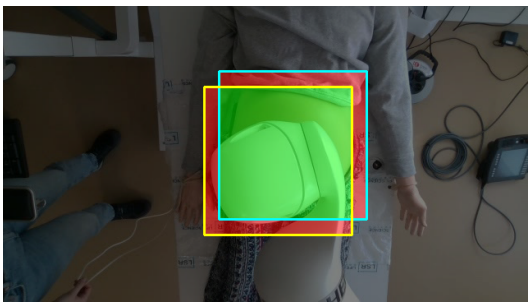
the abdomen again but it is seen that the tracker starts following the robot instead, dropping the accuracy to 18% at the second 45 as it was expected that could happen when the robot covers most of the abdomen. Even though the template gets slightly updated every new frame, being the new target the robot's end effector it is not expected that it will get back to track the abdomen since the robot is always in the view.



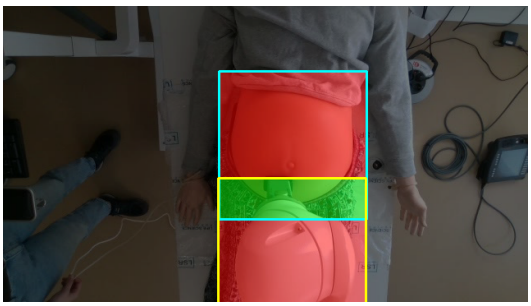
(a) IoU metric at second 25.



(b) IoU metric at second 30.



(c) IoU metric at second 35.



(d) IoU metric at second 45.

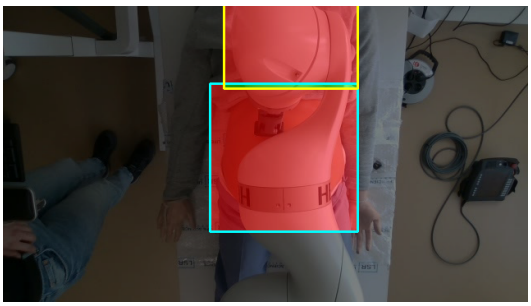
Figure 7.3: Example of the IoU metric in one video (Outfit 13) where the robot and the patient move chronologically. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles).

7.2.3 Results of test5: Long videos

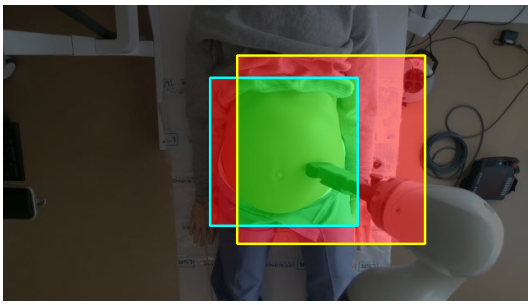
In the results of tests 2,3 and 4, an example was shown where the tracker started following the robot. This test seeks to visualize how the tracker behaves after losing the patient and starts tracking the robot. It is expected that the target will not be recovered since the robot is always on the view. Five videos were tested with a length between 2.40 and 4 minutes. The videos are structured such that in the first part the robot and the patient move simultaneously, then the robot moves but the patient is still and finally the patient moves but the robot stays still. Between 5 and 10 frames were extracted for each of the cases getting samples at the beginning, middle and end of the video. Sometimes the video length of one particular type was not long enough to get 10 frames, so the largest amount

of frames was taken, being 5 the minimum. In this test, 117 frames were evaluated.

In Figure 7.4, a relevant behaviour of the tracker was observed in the video Outfit12. The tracker lost the target and started following the robot in Figure 7.4a, but part of the patient is back in the tracking box sometime later in Figure 7.4b. This could indicate that the tracker can recover the tracker, however, of how the code is structured, it does not store multiple templates. What makes KCF seem that it recovered the track are two factors. First, in the RAU system, the robot always operates on top of the abdomen. Even if KCF starts tracking the robot, part of the bounding box will match the ground truth since they are overlapping. The second factor is that the appearance of the robot also changes from the camera perspective. As soon as the robot's appearance changes, the tracker will explore larger box sizes to try to find the best correlation. Probably the box will grow a little to adapt to the changes in appearance and part of the patient will be back on the ROI. Features of the patient will be considered again which will slowly lead to Figure 7.4b. In the RAU system, it could be considered that the track can be recovered since depending on the robot's movements it could slowly focus again on the abdomen and lose the robot. However, if the robot had freedom of movement and moved out of the abdomen area, the appearance changes would probably set the focus on another area in the image, making it evident that lost the track.



(a) IoU metric at the last frame of the second set of samples.



(b) IoU metric at the first frame of the last set of samples

Figure 7.4: Example of the IoU metric in video Outfit12 4 minutes long. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles).

In Table 7.3 the results are shown. In this test, it can be observed that the performance considerably decreased from an average of around 85% in tests 2,3 and 4 to 69.4% in the case of the straight ground truth rectangle, classifying the test as a wrong track. Even though the performance accuracy is highly dependent on the robot's movement, the results ensure that KCF cannot handle occlusions.

Table 7.3: Average test4 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 12	29	40.0	40.1
Outfit 16	20	75.7	76.1
Outfit 17	27	90.7	90.7
Outfit 18	19	69.7	70.0
Outfit 19	22	74.6	73.2
Total	117	69.4	69.2

7.2.4 Summary of the default performance

After presenting the results for each test, this section summarises the overall performance. In Table 7.4 it is observed that the accuracy decreased when the robot was part of the view (tests 2 to 5) compared to the accuracy when there was no robot (test1).

Table 7.4: Average accuracy per test in the default tracker. The total amount of videos, frames and the average IoU per test are presented

Test	**N videos**	**N evaluated frames**	**Mean IoU [%]**	
			Straight Box	Rotated Box
Test 1	8	54	93.8	94.0
Test 2	9	77	85.2	85.7
Test 3	9	81	86.2	86.1
Test 4	11	108	84.3	84.1
Test 5	5	117	69.4	69.2

7.2.5 Weaknesses of KCF

The default testing was done to analyze the accuracy of the performance of the tracker but also to comprehend its behaviour in different scenarios. In this section, the weaknesses of the tracker are stated to improve them with the proposed solution.

Multi-scale parameter

This parameter is an advantage to improve the accuracy of the track since the camera is slightly tilted, so the distance to the belly is not the same in all positions in the bed. However, when the appearance in the ROI changes, the box size tends to adapt to those

changes, increasing or decreasing in size, to contain the pixels with the highest response which are probably not concentrated in one specific point, leading to the loss of the target. During the ultrasound scan, the patient's appearance will change from the camera view due to the robot. Using a fixed box size could improve the performance in this scenario.

Occlusion handling

The default algorithm cannot handle large occlusions. The created template model keeps updating every frame which makes the occlusions slowly become part of the model when they gradually increase the appearance in the view. The main proposed solution relies on the use of feature-matching techniques to re-identify the patient and correct the tracking box.

7.3 Evaluation of the proposed solution

In the following sections, the results of the new algorithm are presented and deeply analyzed by repeating the testing done with the default tracker. Apart from the IoU metric to quantify the track's accuracy, the default tracker's results were considered to analyze the potential improvement. For further details on the evaluation setup, reading section 7.1 is recommended. The evaluation starts by providing a summary of the performance in all tests. Detailed analysis is provided afterwards for each of the tests.

7.3.1 Summary of the proposed approach performance

The average IoU for the straight box of the default tracker and the potential improvement for each test is presented in Table 7.5. Overall, the proposed solution increased the accuracy of the tracker when the robot was in view (tests 2-5), specifically in test 5 the performance was significantly improved. However, the accuracy dropped when there was no occlusion (test1). The decrease in accuracy in test1 is given for the use of feature matching since the videos did not have the robot in view.

Feature matching is used when the APCE of the current frame is 5% lower than the mean of the previous tracked frames. The threshold was chosen via trial and error, focused on restricting the tracker from following the robot. However, the restrictive condition triggers feature matching in the videos where there are no occlusions. The changes in the APCE values in videos with no occlusion could be due to noise in the videos and rotational movements since HOG is not a rotation-invariant feature.

The elements that can drop the accuracy when using feature matching rely on the number and robustness of detected keypoints, the matching performance and the calculation of the transformation matrix. Further details from the testing are needed to determine possible

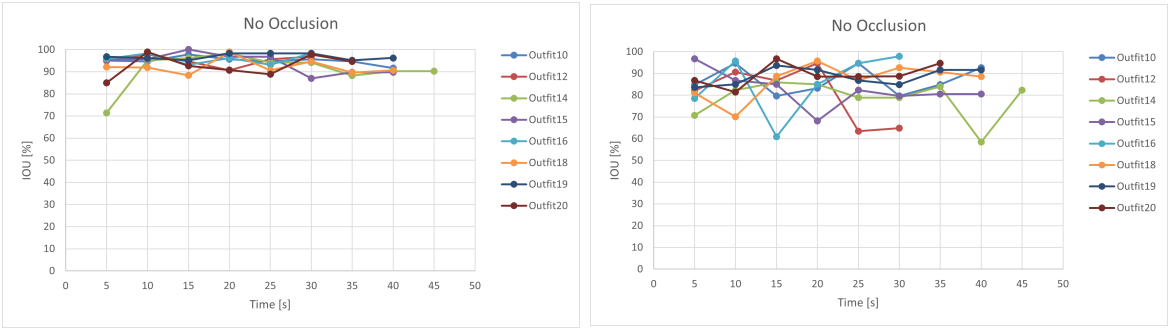
causes of the accuracy performance, presented in the following sections.

Table 7.5: Average IoU for the straight box of the default tracker versus the proposed approach in each test

Test	N videos	N evaluated frames	Mean IoU [%]	
			Default approach	Proposed solution
Test 1	8	60	93.8	84.8
Test 2	9	77	85.2	87.4
Test 3	9	81	86.2	97.4
Test 4	11	108	84.3	87.5
Test 5	5	116	69.4	84.5

7.3.2 Test1: No occlusion

As presented in the summary, the accuracy of this test decreased compared to the default tracker. In Figure 7.5 the performance of each video in the test over time is presented. As it is observed, in general, all of them achieve a lower accuracy, however, the videos with outfits 12,14 and 16 contain the lowest accuracy, dropping to around 60%. Since the outfits in 12,14,16 do not seem to follow a pattern, the reduction in performance could be attributed to not detecting robust enough keypoints or a large amount of keypoint to calculate a reliable transformation.



(a) Performance of the default KFC tracker

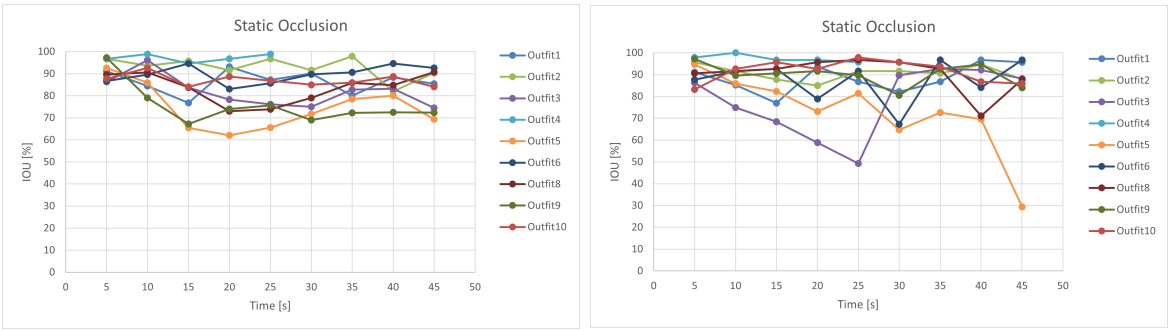
(b) Performance of the proposed solution

Figure 7.5: Comparison of the performance between the proposed solution and the default tracker. The accuracy of each evaluated frame (highlighted with a point) for all tested videos in test1 is presented.

7.3.3 Test2: Static occlusion

The accuracy performance seen in Figure 7.6 shows a similar performance to the default tracker. However, it is noticeable that the video with Outfit 5 gradually drops its accuracy over time reaching 30%. This indicates that the tracker lost the target. The reason for the

failure could rely on the clothes used. In this video, the doll was dressed in a texture-less purple blouse and white trousers that could potentially lead to miss matches. Particularly it is known that in 3 of the 9 evaluated points, feature matching was used and classified as a wrong match. The wrong match is a bool that is true when the distance between the previous centre of the box and the current one exceeds 50 pixels. This brings the hypothesis that the clothes affect the performance by dropping the number and reliability of the detected keypoints.



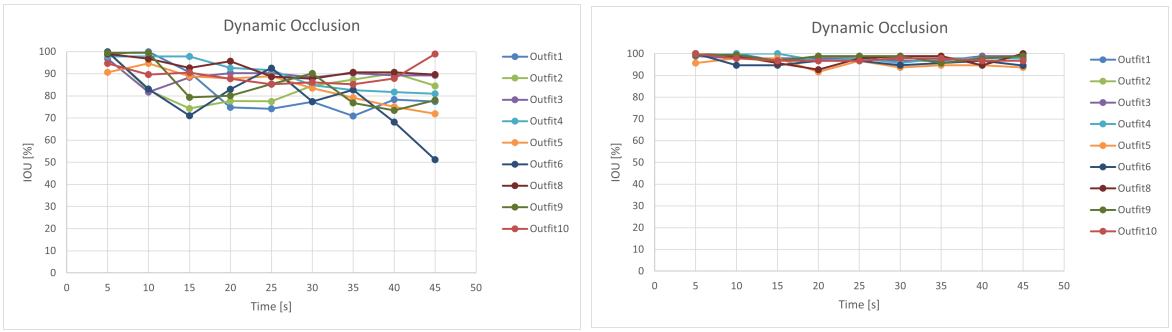
(a) Performance of the default KFC tracker

(b) Performance of the proposed solution

Figure 7.6: Comparison of the performance between the proposed solution and the default tracker. The accuracy of each evaluated frame (highlighted with a point) for all tested videos in test 2 is presented.

7.3.4 Test3: Dynamic occlusion, static patient

The third test, in a more challenging scenario than test 2 since the robot moves increasing the probability of the tracker to start tracking the robot instead, got an outstanding performance in all tested videos as observed in Figure 7.7. The success of this test could be attributed to having the robot mostly moving on the bottom part of the abdomen. Despite this, the frames where the robot was considerably occluding the abdomen did also not present a decline in the accuracy.



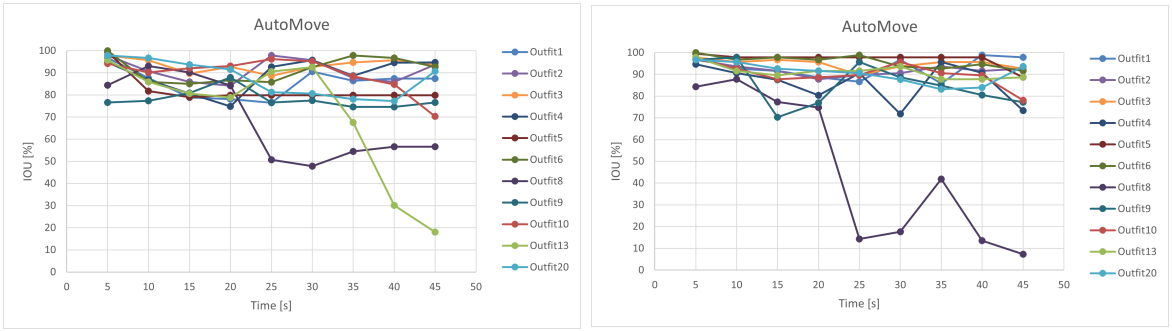
(a) Performance of the default KFC tracker

(b) Performance of the proposed solution

Figure 7.7: Comparison of the performance between the proposed solution and the default tracker. The accuracy of each evaluated frame (highlighted with a point) for all tested videos in test 3 is presented.

7.3.5 Test4: Dynamic occlusion, dynamic patient

The results presented in Figure 7.8 show that the proposed approach was able to correct the tracking box in outfit 13 but achieved an unacceptable performance in outfit 8. In Outfit 8 the doll was wearing the purple blouse which already showed a drop in accuracy in previous frames. In this case, 5 of the 9 evaluated keypoints triggered the wrong match bool when computing feature matching. It suggests that the contrast between the skin and the blouse is not high enough to get reliable features due to its texture-less textile.



(a) Performance of the default KFC tracker

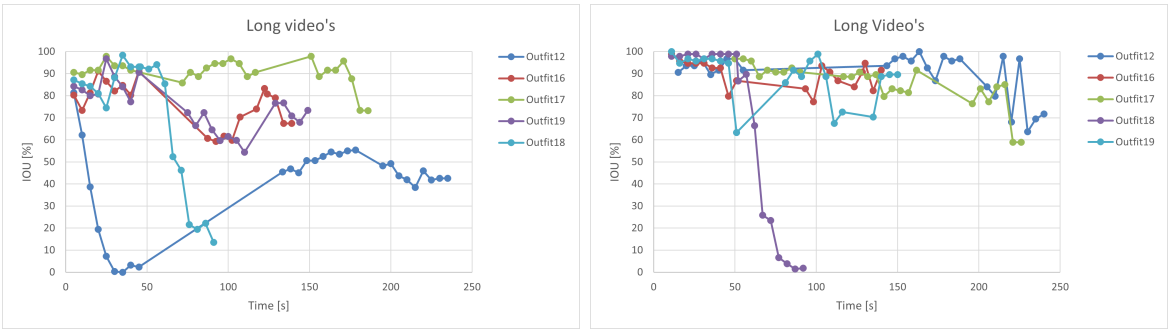
(b) Performance of the proposed solution

Figure 7.8: Comparison of the performance between the proposed solution and the default tracker. The accuracy of each evaluated frame (highlighted with a point) for all tested videos in test4 is presented.

7.3.6 Test5: Long videos

The long videos indicated a significant improvement in the accuracy of the tracker. It is observed that the tracking in the video Outfit 12 significantly improved but it could

not improve the Outfit 18 video. Apart from the clothes, the main difference between both videos is that in Outfit 18 the robot completely occludes the abdomen at second 67, drastically dropping the performance from there. At second 67 it could be that no points were detected. However, the algorithm did not correct the box when the robot made visible the abdomen again. This indicates that the issue might be that the feature detector performance is affected by the clothes since it is not able to detect robust features classifying them in the wrong matches.



(a) Performance of the default KFC tracker

(b) Performance of the proposed solution

Figure 7.9: Comparison of the performance between the proposed solution and the default tracker. The accuracy of each evaluated frame (highlighted with a point) for all tested videos in test5 is presented.

Chapter 8

Final test

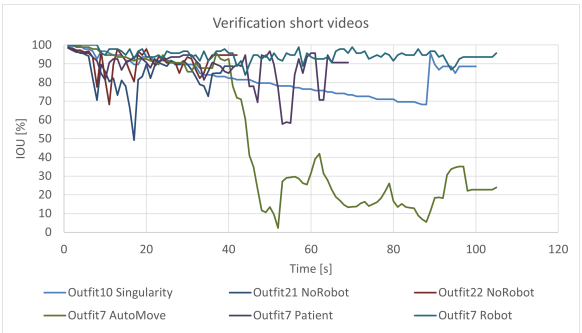
A final test was conducted by evaluating the performance in the verification set of the dataset. The verification set contains ground truth annotations every 1, 5 or 10 seconds in the entire video. It is expected to see more details of the performance to identify the main drawbacks of the proposed approach. Apart from calculating the accuracy of the track with the IoU metric, it is provided the number of frames that used feature matching to correct the box and the number of frames that were classified as a wrong match. It is important to notice that the count of the frames that used feature matching and the frames that feature matching was triggered as a wrong match was made at every frame, considering the frames that were not annotated as well. In Table 8.1 the results are presented. It is seen that the accuracy of the model is above 80 % except for one of the videos, outfit 7 which has the largest amount of wrong matches.

The results indicate that the approach successfully corrects the bounding box numerous times in each video. This suggests that the matching approach and the calculation of the transformation matrix are well computed. Therefore, the issues rely on the detector approach. It is noticeable that the clothes and the position of the robot significantly affect the performance since a high IoU is not directly related to a lower use of feature matching. Therefore, the clothes affect the detection of reliable keypoints since again, the worst performance is attributed to an outfit where the top part was the purple blouse.

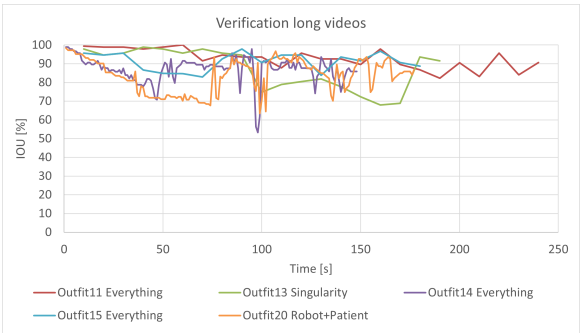
In Figure 8.1 a visual representation of the tracking performance is seen. In the images is observed that the performance can drop throughout a video but the proposed algorithm can recover and continue the track.

Table 8.1: Validation set results. The number of times the box was corrected and the wrong matches is provided. Notice that if the match was wrong the box was not corrected. The frame interval indicates the gap between evaluated frames. The IoU is provided for the straight and rotated box

Video	**N box corrections**	**N wrong matches**	**Frame interval**	**Mean IoU [%]**	
				Straight Box	Rotated Box
Outfit7 AutoMove	19	62	15	50.4	49.6
Outfit7 Patient	61	24	15	88.5	88.5
Outfit7 Robot	147	5	15	94.8	94.3
Outfit10 Singularity	23	0	15	83.3	84.1
Outfit11 Everything	222	2	150	92.8	92.6
Outfit13 Singularity	105	0	150	87.2	87.3
Outfit14 Everything	111	0	15	86.9	86.6
Outfit15 Everything	231	0	150	91.1	91.0
Outfit20 Robot+Patient	107	4	15	83.8	83.5
Outfit21 NoRobot	36	0	15	84.9	85.0
Outfit22 NoRobot	54	0	15	91.5	91.3



(a) Performance of the short videos



(b) Performance of the long videos

Figure 8.1: Final test performance results

Chapter 9

Discussion

This chapter focuses on discussing the results of the proposed approach, its limitations and future work based on the results analyzed in chapter 7 and chapter 8.

The integration of feature matching into KCF tracker improved the general performance of the default KCF tracker when the robot was in the view by 8%. However, the performance in the videos with no occlusion decreased by 9%. This indicates that feature matching was triggered when testing the videos with no occlusion. Even though the APCE parameter could be fine-tuned to avoid triggering feature matching when there are no occlusions, it is required that its performance is robust and consistent to provide reliable tracks. The main drawback of the proposed solution relies on the point detector, SIFT. SIFT was the approach selected due to detecting four times more points in comparison to other considered approaches such as ORB. However, in some videos, the points are wrongly matched. This could be due to detect points in the entire frame. Most of the detected points will fall out of the ROI. Applying feature detection in a slightly bigger region around the estimated ROI could increase the number of detected points. Nonetheless, the detection in cases where the clothes are textureless such as the purple blouse could potentiate the wrong matches instead. Thus, for the RAU system more robust feature detectors are needed to be invariant to clothes. For instance, HOG features could be used instead since the performance kept stable throughout the videos without occlusions with an average IoU of 93.4%.

In contrast, in the videos with occlusion, the proposed approach showed a non-consistent improvement. Most of the tested videos reached an IoU above 80% which classifies the video as a successful track. However, in some other videos, the accuracy dropped below 30%. It was also observed that inconsistency is given within the same video. For example, in the verification set, the performance can fluctuate between 50 and 100 % in the same video. On the one side, this indicates that the system can recover after losing track of the patient. On the other side, it would constantly detect movement triggering false positives,

not ensuring the safety of the patient. These fluctuations are probably given by the gradual increase and decrease of the robot's presence in the ROI. If the robot occludes most of the abdomen there is no room for detection. The cause of the fluctuations is probably due to the requirement of detecting at least three points to run feature matching, otherwise, the box is updated with the default tracker. If in consecutive frames less than three points are detected, KCF tracker will start tracking the robot as seen in the performance evaluation of the default tracker. The algorithm could be constrained so if feature matching is required but not enough points are detected, the previous box position could be taken instead.

9.1 Limitations of the approach

The algorithm also presents limitations if intending to be integrated into the RAU system. First of all, the ROI has a fixed size and is set manually. An approach should be considered to adapt the size of the ROI to the size of the patient's abdomen since it will not be constant. Furthermore, the current output of the track is the pixel position of the bounding box. In order to detect movement the displacement between two frames should be considered. The displacement will represent translation in the x and y direction but it will not consider the patient's rotation. For this reason, the transformation matrix could be used to extract the rotation between the set of detected points. With this, the first block of the patient movement handling pipeline would be fulfilled.

9.2 Future work

This research delved into solely tracking the abdomen of the patient in the RAU system. No previous research focused on the limited area of movement, the abdomen. The current proposed solution showed potential in tracking the abdomen when it is occluded by the robot using classical approaches. The algorithm was able to correct the box position but further research is needed to keep the accuracy approximately constant during the tracking. Future research should focus on studying a more robust method to detect features to reduce the number of wrong matches and be consistent with different clothes textures and colours. Moreover, the computational cost should be considered to be implemented as a real-time tracker in the RAU system.

Chapter 10

Conclusion

This project seeks to contribute to the research of enhancing patient comfort when performing obstetric scans with RAU. The focus is on tracking the patient's abdomen partially occluded by the robot. with the focus on classical computer vision approaches it is intended to provide a starting point to detect patient movement by answering the question: "Is it possible to track a patient's abdomen with the robot occluding the view using classical visual tracking techniques?".

The proposed solution consists of integrating feature matching into KCF tracker to improve the tracker's performance in the presence of occlusion. Compared to the default KCF tracker, the method achieved an average accuracy of 9% higher than the default KCF tracker when there were occlusions. However, the accuracy decreased by 8 % in videos without occlusions. Despite the low average improvement, the approach can re-identify the patient, reaching an accuracy of 84.5% in long videos compared to the 69,4% achieved by the default tracker.

The accuracy of the track is affected by the amount of occlusion. An accuracy of around 90% is achieved when the robot moves on the lower part of the abdomen but it can drop up to less than 50% when the robot scans the upper part of the abdomen, due to the failure of the feature detector since it does not detect enough points to update the box. This originates a fluctuating response in the same video that also is affected by the clothes of the patient. Since the abdomen is a textureless region, the clothes play an important role in detecting points in the contrast part with the skin, leading to wrong matches when no texture is found even though having a distinctive colour to the skin tone.

Overall, the approach shows the potential to accurately track the abdomen with the robot in view to detect patient movement. The translation of the patient could be calculated with the centre of the tracking box. The rotation cannot directly be extracted but since the method integrates the transformation matrix between two sets of points, the rotation could

derive from there. Further research is needed to detect more robust features and handle large occlusions to reduce wrong matches and keep the accuracy stable.

Bibliography

[1] M. Rahim Sobhani et al. "Portable low cost ultrasound imaging system". In: *2016 IEEE International Ultrasonics Symposium (IUS)*. 2016, pp. 1–4. DOI: 10.1109/ULTSYM.2016.7728837.

[2] Gill Harrison and Allison Harris. "Work-related musculoskeletal disorders in ultrasound: Can you reduce risk?" eng. In: *Ultrasound* 23.4 (2015), pp. 224–230. ISSN: 1742-271X.

[3] Life Science Robotics. *Life Science Robotics home page*. https://www.lifescience-robotics.com/.

[4] Qinghua Huang, Jiakang Zhou, and ZhiJun Li. "Review of robot-assisted medical ultrasound imaging systems: Technology and clinical applications". In: *Neurocomputing* 559 (2023), p. 126790. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.126790. URL: https://www.sciencedirect.com/science/article/pii/S092523122300913X.

[5] A. Shyam et al. "Immersive Virtual Reality Platform for Robot-Assisted Antenatal Ultrasound Scanning". English. Autonomous capability;Health care professionals;Health services;Immersive virtual reality;Maternal healths;Stream-based;Teleoperation systems;Ultrasound examination;Ultrasound scanning;Ultrasound system; 2023. URL: http://dx.doi.org/10.48550/arXiv.2309.03725.

[6] Shuangyi Wang et al. "Robotic-Assisted Ultrasound for Fetal Imaging: Evolution from Single-Arm to Dual-Arm System". In: *Towards Autonomous Robotic Systems*. Ed. by Kaspar Althoefer, Jelizaveta Konstantinova, and Ketao Zhang. Cham: Springer International Publishing, 2019, pp. 27–38. ISBN: 978-3-030-25332-5.

[7] Maria Bamaarouf et al. "Development of a Robotic Ultrasound System to Assist Ultrasound Examination of Pregnant Women". In: *IEEE Transactions on Medical Robotics and Bionics* (2024), pp. 1–1. DOI: 10.1109/TMRB.2024.3387047.

[8] Yongqing Fu et al. "Robot-Assisted Teleoperation Ultrasound System Based on Fusion of Augmented Reality and Predictive Force". In: *IEEE Transactions on Industrial Electronics* 70.7 (2023), pp. 7449–7456. DOI: 10.1109/TIE.2022.3201322.

[9] Temitope Ibrahim Amosa et al. "Multi-camera multi-object tracking: A review of current trends and future advances". eng. In: *Neurocomputing (Amsterdam)* 552 (2023), pp. 126558–. ISSN: 0925-2312.

[10] Rabah Iguernaissi et al. "People tracking in multi-camera systems: a review". eng. In: *Multimedia tools and applications* 78.8 (2019), pp. 10773–10793. ISSN: 1380-7501.

[11] Richard Szeliski. *Computer Vision Algorithms and Applications*. eng. 2nd ed. 2022. Texts in Computer Science. Cham: Springer International Publishing, 2022. ISBN: 9783030343729.

[12] Arnold W. M. Smeulders et al. "Visual Tracking: An Experimental Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1442–1468. DOI: 10.1109/TPAMI.2013.230.

[13] Mustansar Fiaz et al. "Handcrafted and Deep Trackers: A Review of Recent Object Tracking Approaches". In: *CoRR* abs/1812.07368 (2018). arXiv: 1812.07368. URL: http://arxiv.org/abs/1812.07368.

[14] João F. Henriques et al. "High-Speed Tracking with Kernelized Correlation Filters". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (2015), pp. 583–596. DOI: 10.1109/TPAMI.2014.2345390.

[15] Sajid Javed et al. "Visual Object Tracking With Discriminative Filters and Siamese Networks: A Survey and Outlook". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2023), pp. 6552–6574. DOI: 10.1109/TPAMI.2022.3212594.

[16] Xin Du et al. "An Anti-occlusion Object Tracking Algorithm Using KCF and ORB Feature Detector". In: *2023 5th International Conference on Robotics and Computer Vision (ICRCV)*. 2023, pp. 1–6. DOI: 10.1109/ICRCV59470.2023.10329036.

[17] Jiayi Ma et al. "Image Matching from Handcrafted to Deep Features: A Survey". In: *International Journal of Computer Vision* 129.1 (2021), pp. 23–79. ISSN: 1573-1405. DOI: 10.1007/s11263-020-01359-2. URL: https://doi.org/10.1007/s11263-020-01359-2.

[18] Mathias Klæstrup Mikkelsen. *Aspects of Robot Assisted Ultrasound*. eng. 2023.

[19] Laia Vives Benedicto. *Real-Time Patient Movement Detection for Enhanced Accuracy in Robotic Assisted Ultrasound System at Life Science Robotics*. eng. 2024.

[20] Amila Jakubović and Jasmin Velagić. "Image Feature Matching and Object Detection Using Brute-Force Matchers". In: *2018 International Symposium ELMAR*. 2018, pp. 83–86. DOI: 10.23919/ELMAR.2018.8534641.

[21] Marius Muja and David G. Lowe. "Scalable Nearest Neighbor Algorithms for High Dimensional Data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2227–2240. DOI: 10.1109/TPAMI.2014.2321376.

[22] LCorleone. *KCF_py3*. https://github.com/LCorleone/KCF_py3. 218.

[23] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[24] Christopher G. Harris and M. J. Stephens. "A Combined Corner and Edge Detector". In: *Alvey Vision Conference*. 1988. URL: https://api.semanticscholar.org/CorpusID:1694378.

[25] Miroslav Trajković and Mark Hedley. "Fast corner detection". In: *Image and Vision Computing* 16.2 (1998), pp. 75–87. ISSN: 0262-8856. DOI: https://doi.org/10.1016/S0262-8856(97)00056-5. URL: https://www.sciencedirect.com/science/article/pii/S0262885697000565.

[26] Open Source Computer Vision. *OpenCV-Python Tutorials*. https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html. Online, accessed 15/02/2024.

[27] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[28] Elmar Mair et al. "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test". eng. In: *Computer Vision – ECCV 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 183–196. ISBN: 9783642155512.

[29] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *2011 International Conference on Computer Vision*. 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[30] David G. Lowe. "Distinctive image features from scale-invariant keypoints". eng. In: *International journal of computer vision* 60.2 (2004), pp. 91–110. ISSN: 0920-5691.

[31] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking". eng. In: *International journal of computer vision* 94.3 (2011), pp. 335–360. ISSN: 0920-5691.

[32] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 102–115. ISBN: 978-3-540-88693-8.

[33] Alexandr Andoni and Piotr Indyk. "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. 2006, pp. 459–468. DOI: 10.1109/FOCS.2006.49.

[34] Richard. Hartley and Andrew. Zisserman. *Multiple view geometry in computer vision.* eng. 2. ed. Cambridge: Cambridge University Press, 2003. ISBN: 9780521540513.

[35] Computer Vision Annotation Tool. *Computer Vision Annotation Tool*. https://www.cvat.ai/.

[36] Tran Thien Dat Nguyen et al. “How Trustworthy are Performance Evaluations for Basic Vision Tasks?” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.7 (2023), pp. 8538–8552. DOI: 10.1109/TPAMI.2022.3227571.

[37] Luka Čehovin, Aleš Leonardis, and Matej Kristan. “Visual Object Tracking Performance Measures Revisited”. In: *IEEE Transactions on Image Processing* 25.3 (2016), pp. 1261–1274. DOI: 10.1109/TIP.2016.2520370.

[38] Shaharyar Ahmed Khan Tareen and Rana Hammad Raza. “Potential of SIFT, SURF, KAZE, AKAZE, ORB, BRISK, AGAST, and 7 More Algorithms for Matching Extremely Variant Image Pairs”. In: *2023 4th International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. 2023, pp. 1–6. DOI: 10.1109/iCoMET57998.2023.10099250.

Appendix A

Test results

A.1 Default KCF testing details

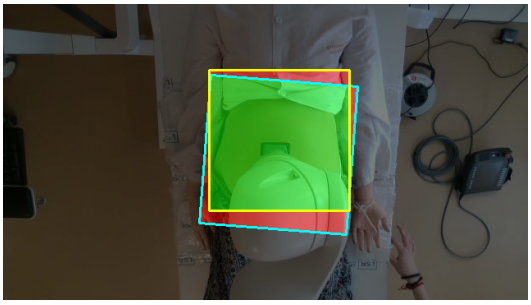
A.1.1 Test2: Static occlusion

In this second test, the robot is in the view but in a static position. The robot's position is different for each of the videos. It evaluated the effect of the occlusion when the robot was present. In this test and the following ones, the robot was already in view in the first frame. Therefore, the template filter will contain a part of the robot that can affect negatively when looking for the highest correlation in the next frame. Furthermore, the tracker updates the template at every frame, so the performance will depend on the percentage of visible robot versus visible abdomen in the ROI. If the abdomen is more visible is expected to track towards the right target, otherwise, it could track the robot instead. It is also important to mention that the videos have a variety of movements, in some of them the patient moves a lot while in others the patient does small shifts. The speed of movement and direction are also influential in the performance since they will affect the visibility of the abdomen between two consecutive frames. Nine videos were evaluated in this test, 45 seconds long each except one of them that was 26 seconds, getting a total of 77 evaluated frames.

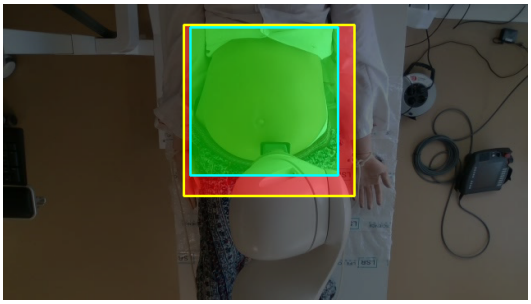
In Figure A.1, the chronological performance in one video is shown. Figure A.1a indicates the first frame of the video where it is observed that initially, the robot was partially covering the lower part of the abdomen. First, some rotation and downward translation were applied to the patient. It is seen in the performance that the tracker is centred on the robot's end effector. Considering that in the first frame, most of the salient features were probably in the robot's end effector and the division between the abdomen skin and the shirt since the HoG studies the gradients in the three colour channels. Thus, it makes sense that the highest correlation was found in the upper part of the abdomen

together with the robot. In Figure A.1c, the patient moved upwards and that originated a response of increasing the size of the bounding box. Here is shown again an answer to the previous hypothesis. Since the skin is textureless for reliable features, it was probably interpreted that the target grew in size due to the increase in pixel distance between the robot's end effector and the shirt. Finally, the patient moved downwards. In this case, the pixel distance between the hypothetical main features is lower, so the bounding box size decreases.

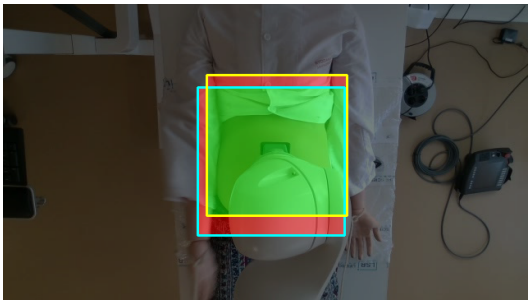
(a) First frame of the evaluated outfit3.



(b) IoU metric after applying some rotation and translation.



(c) IoU metric after moving the patient upwards



(d) IoU metric after moving the patient downwards.

Figure A.1: Example of the IoU metric in one of the tested videos chronologically. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles). **a)** shows the first frame patient position. **b)**, **c)**, **d)** presents the patient movement chronologically at second 20,30 and 45.

In the example, it has been seen that the robot has a clear influence on the performance. The hard contrast of the black tool with the skin probably makes the end effector one of the main reliable features of the ROI. In Table A.1 are presented the results for each of the videos as well as the overall performance. Although some of them kept a high accuracy, in comparison to the first test the performance has decreased to 85.2% for the straight box ground truth. According to the threshold of a minimum of 80% IoU to be considered a good track, the test succeeded for 7 of the 9 tested videos. It is noticeable that in this case the performance compared to the rotated rectangle is higher. After the

analysis in Figure A.1, the high accuracy achieved in outfit 2 and 4 can greatly be attributed to the fact that the patient did not considerably moved, whether in outfit 5 and 9 larger displacements probably occurred, getting performances similar to Figure A.1c.

Table A.1: Average test2 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

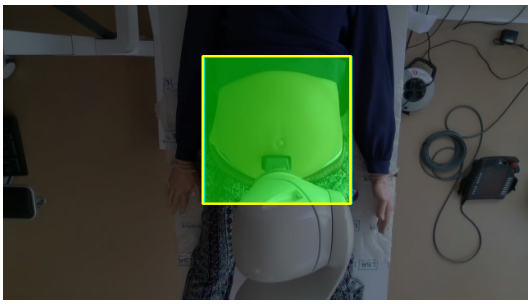
Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1	9	86.3	89.5
Outfit 2	9	92.9	92.6
Outfit 3	9	80.3	83.0
Outfit 4	5	97.2	96.4
Outfit 5	9	74.6	74.3
Outfit 6	9	89.7	91.9
Outfit 8	9	83.5	82.4
Outfit 9	9	75.5	74.4
Outfit 10	9	87.1	96.9
Total	77	85.2	85.7

A.1.2 Test3: Dynamic occlusion, static patient

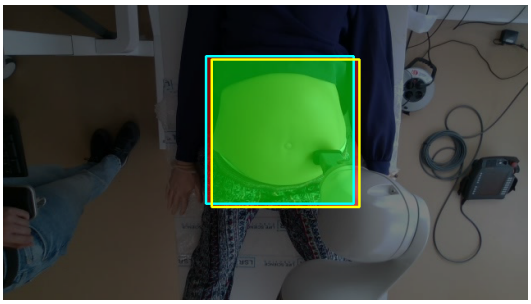
The third test consists of moving the robot with the joystick while the patient is static at one position. Considering that the robot is visible in the first frame, it is expected that the tracker will follow its movements after seeing the influence of the end effector's presence in the ROI in the previous test. It will be analyzed if an increase in occlusion by the robot covering larger parts of the belly decreases the accuracy when tracking the abdomen. For this test 9 videos of 45 seconds were used, having a total of 81 evaluated frames per video. In all the videos the robot is initially placed with the transducer on top of the belly button (this belongs to the first frame). Afterwards, the robot is moved around the abdomen. Each video focuses the movement in particular areas. For example, in some videos the robot only moves on the lower part of the abdomen, some of them focus on the right side, or some others focus on the top side of the abdomen.

In Figure A.2 are shown two selected videos to analyze the performance. In the first example (Figure A.2a and Figure A.2b), the robot moves on the lowest part of the belly. In this scenario it can be seen that the robot movement does not negatively affect the performance. Comparing the visibility of the abdomen between both images it is noticeable that probably the majority of the pixels lie on the patient, on the division between the shirt and the skin, and the division between the skin and the trousers most likely. Probably

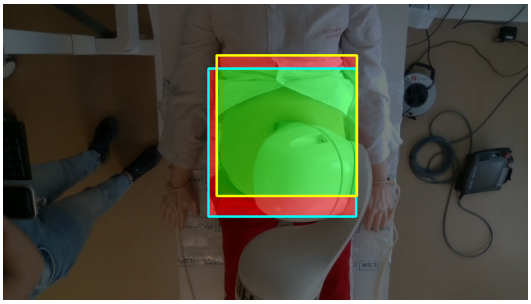
some of the features are also on the belly button. When moving the robot, all those features are still visible. Even though the ones attributed to the robot moved, the majority of them kept in place, attributing the highest correlation between the previous and the new frame. Contrary, in the second example (Figure A.2c and Figure A.2d) the robot moved towards the upper part of the abdomen, covering most of it. In this example it is seen that the tracker tends to track the transducer instead. Considering that the template of KCF is updated at every frame, progressively the inclusion of the robot in the ROI will add features attributed to the robot into the template and remove some that were attributed to the patient. Therefore, the highest correlation is found where the robot is located.



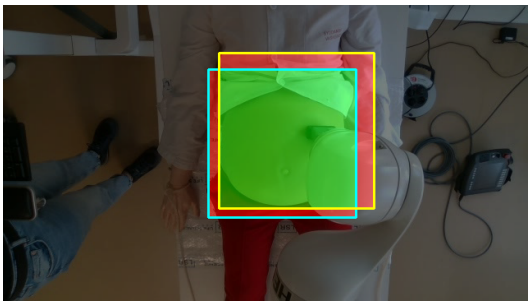
(a) IoU metric when the robot is at the lower part of the abdomen (Outfit8).



(b) IoU metric after a displacement in the lower part of the belly (Outfit8).



(c) IoU metric when the robot moves upwards (Outfit1).



(d) IoU metric after moving the robot towards the right on the top part of the belly (Outfit1).

Figure A.2: Example of the IoU metric in two tested videos where the robot moves. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles). **a)** and **b)** show the influence when the robot moves on the lower part of the abdomen. **c)** and **d)** present the robot influence when working on the higher part of the abdomen

The results for all tested videos are shown in Table A.2. It can be observed that the total average accuracy is higher than in the previous test. Eight of the nine tested videos outperformed the minimum threshold of 80% which was not expected. Having the starting position in the lower part of the belly might have helped with the performance since the

first template had a lot of features belonging to the patient. If the start position was in the upper part instead, the result could have been a lot different. In any case, the tracker shall track the patient with high accuracy regardless of the robot movements and that has clearly seen affected in the presented examples.

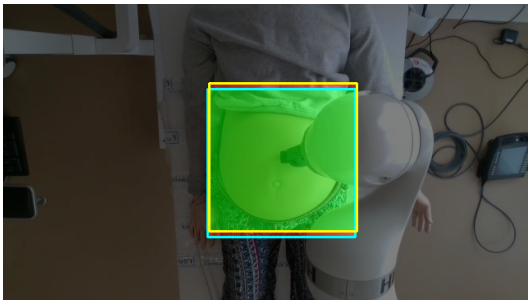
Table A.2: Average test3 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1	9	82.5	82.8
Outfit 2	9	84.3	83.9
Outfit 3	9	89.4	89.5
Outfit 4	9	89.8	90.0
Outfit 5	9	84.5	84.8
Outfit 6	9	78.8	79.4
Outfit 8	9	92.4	91.8
Outfit 9	9	84.7	83.1
Outfit 10	9	89.5	89.4
Total	81	86.2	86.1

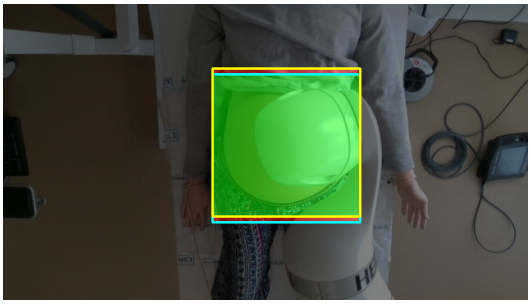
A.1.3 Test4: Dynamic occlusion, dynamic patient

In this fourth test the complexity increases by having movement on the patient and the robot. The tracker should be able to distinguish the two moving parts and track only the patient. Twelve videos of 45 seconds were part of this test, evaluating a total of 108 frames.

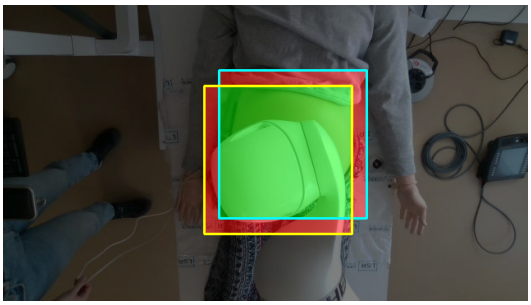
The Outfit13 video has been chosen to visualize the performance chronologically. In Figure A.3 it can be seen that the robot gradually occludes the abdomen in Figure A.3a and Figure A.3b. There is a moment when the abdomen is barely visible, therefore the template is made of features belonging to the robot. In Figure A.3c and Figure A.3d the robot moves to make visible the abdomen again but it is seen that the tracker starts following the robot instead, dropping the accuracy to 18% at the second 45. Even though the template gets updated with every new frame, being the new target the robot's end effector it is not expected that will get back to track the abdomen since the robot is always in the view.



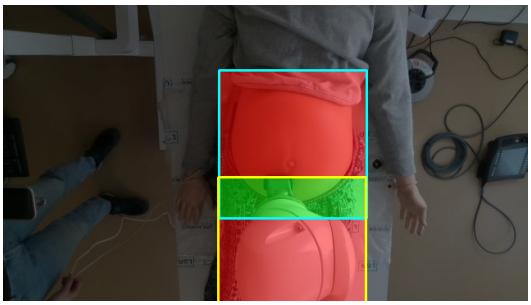
(a) IoU metric at second 25.



(b) IoU metric at second 30.



(c) IoU metric at second 35.



(d) IoU metric at second 45.

Figure A.3: Example of the IoU metric in one video (Outfit 13) where the robot and the patient move chronologically. The cyan and yellow rectangles are the ground truth and estimated rectangles respectively. In green is displayed the overlap region or true positive part (TP), and in red is displayed the union region. (Since the green is on top it cannot be appreciated that the red takes the area of both rectangles).

In Table A.3 is presented the mean per video of the IoU metric. Overall, the average performance is higher than 80% but the accuracy of some videos decreased by around 10%.

Table A.3: Average test4 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1	9	85.5	85.1
Outfit 2	9	91.0	90.8
Outfit 3	9	93.5	93.1
Outfit 4	9	89.2	89.0
Outfit 5	9	82.2	81.6
Outfit 6	9	91.5	91.1
Outfit 8	9	68.7	68.2
Outfit 9	9	78.1	78.3
Outfit 10	9	89.4	89.3
Outfit 13	9	71.1	71.4
Outfit 20	9	87.5	87.4
Total	108	84.3	84.1

A.2 Improved performance

A.2.1 Test1: No occlusion

In this test, it was seen in the default KCF that the abdomen was tracked for all tested videos with an average accuracy of 93.8% and 94.0% when compared to the straight and rotated rectangle respectively. It is expected that the performance will remain the same since the tracker was already able to handle this scenario. A total of 54 frames spread over 8 videos of 30-40 seconds were evaluated in this test.

Table A.4: Average test1 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 10	8	86.8	86.6
Outfit 12	6	80.4	80.4
Outfit 14	9	78.5	78.0
Outfit 15	8	82.5	81.9
Outfit 16	6	85.4	85.2
Outfit 18 *	8	86.8	86.7
Outfit 19	8	88.6	88.7
Outfit 20	7	89.4	88.9
Total	60	84.8	84.6

A.2.2 Test2: Static occlusion

Table A.5: Average test2 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1 *	9	88.3	92.1
Outfit 2	9	90.7	90.7
Outfit 3 *	9	77.8	91.9
Outfit 4	5	97.4	96.6
Outfit 5	9	72.6	72.5
Outfit 6	9	87.5	88.2
Outfit 8	9	90.5	90.3
Outfit 9	9	90.0	89.7
Outfit 10	9	91.5	91.6
Total	77	87.4	89.3

A.2.3 Test3: Dynamic occlusion, static patient

Table A.6: Average test3 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1	9	98.1	98.4
Outfit 2	9	98.2	97.3
Outfit 3	9	97.9	97.3
Outfit 4	9	98.2	97.7
Outfit 5	9	95.1	95.0
Outfit 6	9	96.1	95.1
Outfit 8	9	97.5	97.3
Outfit 9	9	98.3	94.5
Outfit 10	9	97.5	97.7
Total	81	97.4	96.7

A.2.4 Test4: Dynamic occlusion, dynamic patient

Table A.7: Average test4 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 1	9	93.0	91.7
Outfit 2	9	92.0	92.2
Outfit 3	9	94.8	94.9
Outfit 4	9	86.1	86.2
Outfit 5	9	97.0	95.2
Outfit 6	9	95.9	96.2
Outfit 8	9	46.6	46.6
Outfit 9 *	9	85.4	85.3
Outfit 10 *	9	90.0	90.3
Outfit 13	9	91.1	90.9
Outfit 20 *	9	90.6	90.6
Total	108	87.5	87.3

A.2.5 Test5: Long videos

Table A.8: Average test4 results per video, where is indicated the video, the number of evaluated frames and the mean IoU for the video using the straight and the rotated ground truth box. The last row presents the total number of frames with the average IoU values considering all videos together.

Video	**N evaluated frames**	**Mean IoU [%]**	
		Straight Box	Rotated Box
Outfit 12	26	90.0	90.2
Outfit 16	20	90.1	89.8
Outfit 17	27	85.9	86.3
Outfit 18	19	67.7	67.1
Outfit 19	22	88.7	86.8
Total	116	84.5	84.1

Appendix B

Outfit data

This appendix presents all the outfits used during the project. Figure B.1 presents all the 22 outfits analysed

(a) Outfit 1

(b) Outfit 2

(c) Outfit 3

(d) Outfit 4

(e) Outfit 5

(f) Outfit 7

(g) Outfit 7

(h) Outfit 8

(i) Outfit 9

(j) Outfit 10

(k) Outfit 11

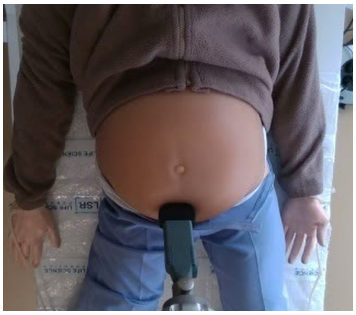
(l) Outfit 12

(m) Outfit 13

(n) Outfit 14

(o) Outfit 15

(p) Outfit 16



(q) Outfit 17

(r) Outfit 18

(s) Outfit 19

(t) Outfit 20

(u) Outfit 21

(v) Outfit 22

Figure B.1: All outfits used during the project