POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Biomedical Engineering

Master Degree Thesis

Inferior vena cava tracking in ultrasound videos

Supervisors
Prof. Luca Mesin
Ing. Piero Policastro

Candidate
Giulia Cinicola

Academic Year 2024-2025

Abstract

The inferior vena cava (IVC) is the largest vein in the human body that carries oxygen-poor blood from the lower part of the body to the right atrium of the heart. The objective of this study is to assess and compare tracking algorithms of OpenCV library to determine the most effective approach for tracking IVC in ultrasound (US) videos. Tracking is used to follow the movement of the vein, providing a complete view of the IVC's position, leading to the identification of respiratory movements and the assessment of diameter variations induced by them. The algorithms chosen for tracking are Lucas-Kanade, KCF, MIL, TLD, MOSSE, Median Flow, Mean Shift, Block Matching, Boosting, ORB and CSRT. These algorithms are widely used for pedestrian and vehicle tracking in traffic surveillance and people recognition in crowded environments. Here, they were applied to US videos for the specific task of IVC tracking. The algorithms were tested on 13 US videos showing a longitudinal view of the IVC, with different length and resolution to increase variability. The experiments were conducted on a MacBook Air equipped with an Apple M1 chip featuring an 8-core CPU and 8 GB of RAM. The 11 algorithms selected were implemented using Python, and the tracking results for each frame were compared with the ground-truth regions (GT-ROI). The GT-ROIs for each frame were created by manually selecting points on the upper and lower borders of IVC. Based on the extreme points, ROI was defined as a rectangle, which is used to evaluate each tracker's performance. The algorithms were tested on videos by initiating tracking from the location of the GT-ROI in the first frame of each video analyzed. To assess the trackers' ability to correctly locate the IVC, the Intersection over Union (IoU) and the Euclidean distance between the center of the algorithm's resulting ROI and the GT-ROI were calculated. Two other parameters used to evaluate performance are the percentage of failed frames and the percentage of false positives, which, using a 0.6 IoU threshold, indicate frames where the IVC position is inaccurately identified. For each pair of successive frames, the distance between the centers of the tracked ROIs was compared with the corresponding distance in the GT-ROIs. If the discrepancy exceeds 0.5 cm, the tracking is considered unreliable, as it indicates that the tracker fails to accurately follow the movement of the IVC. The best performing trackers are KCF, Block Matching and Median Flow which demonstrate good execution speed (KCF: FPS = 32.84, Block Matching: FPS = 47.05, Median Flow: FPS= 55.0). Median Flow achieved an IoU value of 0.82 $\pm$ 0.08, KCF obtained an IoU of 0.79 $\pm$ 0.09 while Block Matching has an IoU of 0.80 $\pm$ 0.08. The false positive rate is 1.84 $\pm$ 0.06% for Median Flow, 6.03 $\pm$ 0.11% for KCF and 6.50 $\pm$ 0.10%

for Block Matching. The Euclidean distance is 0.44 ± 0.21 cm for Median Flow, KCF and Block Matching show a value of 0.57 ± 0.30 cm and 0.49 ± 0.23 cm respectively. All three trackers show good ability to track the movement of the IVC in the various frames of the video, with the percentage of unreliable distances ranging from 0.61% for KCF to 1.02% for Block Matching. The difference between the best trackers lies in the percentage of failed frames: Median Flow in some video is unable to detect the position of IVC in any frame. In contrast, KCF and Block Matching show no errors in detection. In conclusion, to ensure reliable tracking of IVC and follow its movements, KCF turns out to be the best tracker.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ultrasound (US) is a safe and generally non-invasive medical imaging and diagnostic tool that uses high-frequency sound waves. It enables the visualization of tissues or organs within the human body and the monitoring of their changes. In this context, tracking allows to follow the specific movement of an area over time, analyzing tissue deformation and stiffness across a series of frames. The set of tracking algorithms applied to ultrasound imaging is valuable for assessing the functionality of dynamic organs, such as the heart, or the stiffness of tissues, which may indicate conditions such as tumors or fibrosis.

The goal of this thesis project is to analyze tracking algorithms available in the Open Source Computer Vision (OpenCV) library, evaluate their performance on ultrasound videos of the inferior vena cava (IVC), and identify the most effective technique. Specifically, the first chapter describes the physics of ultrasound imaging and its application in the medical field. Additionally, it includes an explanation of the cardiovascular system and its components, with a focus on the geometry and function of the vessel of interest in the study. The second chapter focuses on presenting the materials and methods employed, then tracking algorithms used and metrics adopted to evaluate their performance. It also provides a description of the dataset utilized and pipeline implemented with Python programming language in order to achieve the objective. Finally, the obtained results are presented.

1.1 Ultrasounds

Ultrasound images are diagnostic images generated using ultrasound, which are high-frequency sound waves. Sound waves, which can propagate through various materials such as fluids, soft tissues, and solids, are longitudinal mechanical waves. They can be described in terms of particle displacement or pressure variations [1]. Ultrasound waves are characterized by the following properties:

- **frequency (`f`)**: defined as the number of waves per second, it depends on the sound source. Ultrasound devices operate within a range of 2 MHz to 15 MHz [2].

- **propagation speed (`v`)**: is determined by the medium's density and stiffness. The density (ρ) represents the amount of mass contained in a given volume, denser blood slows the propagation of the wave, in contrast for less density. The stiffness (`K`) reflects the material's resistance to deformation under an applied force, stiffer vessels allow the pressure wave to travel faster because they don't dilate easily. For human soft tissues, the propagation speed is typically approximated as 1540 m/s [3].

- **wavelength (λ)**: represents the distance between two consecutive wave crests or similar points on the wave. It is equal to the ratio of the propagation speed to the frequency [2].

- **acoustic impedance (`z`)**: defined as the product of the medium's density and the propagation speed of the wave, acoustic impedance represents the resistance to the propagation of sound waves. Depending on the variation in acoustic impedance, the tissue produces a stronger or weaker return echo. A smaller impedance difference results in a weak echo, which appears gray on the ultrasound image; conversely, a larger impedance difference between tissues generates a strong echo, displayed as white on the image [1]. Figure 1.1 illustrates a series of interfaces between different tissues, highlighting variations in acoustic impedance. The intensity scale in decibels (dB) shown in the image indicates how the impedance difference between tissues affects the strength of the reflected echo.

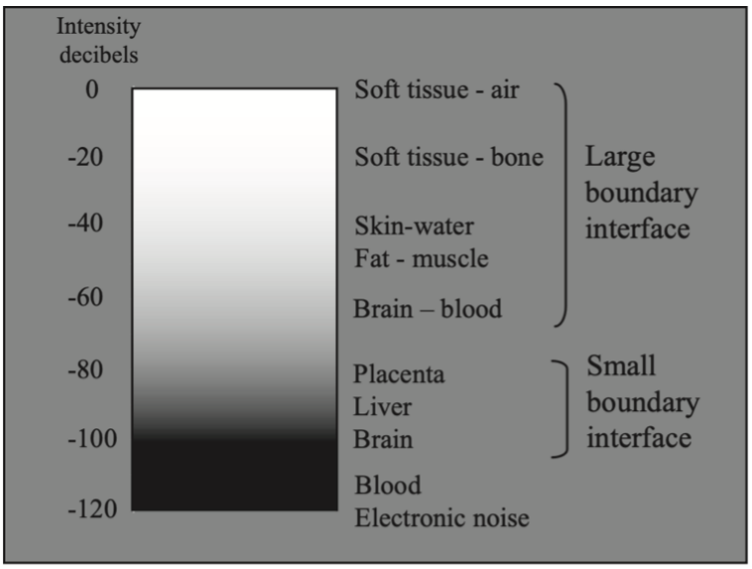

Figure 1.1. Acoustic interfaces according to reflected ultrasound intensity (in decibels) and various tissue types and body interfaces [1].

- **resolution** and **penetration depth**: depend on the ultrasound frequency. The resolution is directly proportional to the frequency, while the penetration depth decreases as the frequency increases [2].

- **attenuation**: is proportional to frequency; an increase in frequency results in a loss of signal power. It is considered to be 0.5 dB/(MHz · cm) for soft tissues [4]. A high-frequency ultrasound beam will experience greater attenuation compared to a low-frequency one. As a result, deeper structures require low-frequency probes, while superficial structures require high-frequency probes [2].

1.1.1 Ultrasound generation principle

Ultrasound machines generate US waves and receive the reflected echoes. The ultrasound beam originates from a transducer placed in contact with the skin, which converts electrical energy into high-frequency sound waves. Short bursts of ultrasound pulses are sent into the patient, each lasting about 1 ms, and several thousand pulses are emitted per second [1]. When these waves encounter variations in acoustic impedance between different materials, part of the energy is reflected back to the transducer, while the rest is transmitted to the second tissue. The amplitude of the reflected and transmitted waves depends on the impedance variation.
A gel is used between the transducer and the patient's skin to eliminate air [3], since at the interface between tissue and air, the impedance difference is high, and the wave energy cannot penetrate further.

Transducers consist of disk-shaped crystals, known as piezoelectric elements, with diameters of a few millimeters and thicknesses ranging from 1.8 mm for a 1 MHz transducer to 0.18 mm for a 10 MHz transducer [1]. These crystals utilize the piezoelectric effect to generate sound waves when an electric field is applied, and conversely, to produce an electric field when impacted by a sound wave. By analyzing the propagation speed and the return time of each echo, the distance between the transducer and the tissue boundary is calculated by the scanner. These calculated distances are then used to construct two-dimensional images of tissues and organs.

Ultrasound probes consist of a single piezoelectric element, but most modern transducers are made up of arrays containing dozens or even hundreds of small piezoelectric elements. Various types of probes exist, as shown in Figure 1.2.

The linear array is characterized by small piezoelectric elements arranged in a straight line. Typically, it consists of 256 elements, each measuring 1 mm x 10 mm [3]. This configuration produces high-resolution rectangular images and is used for superficial areas of the body.

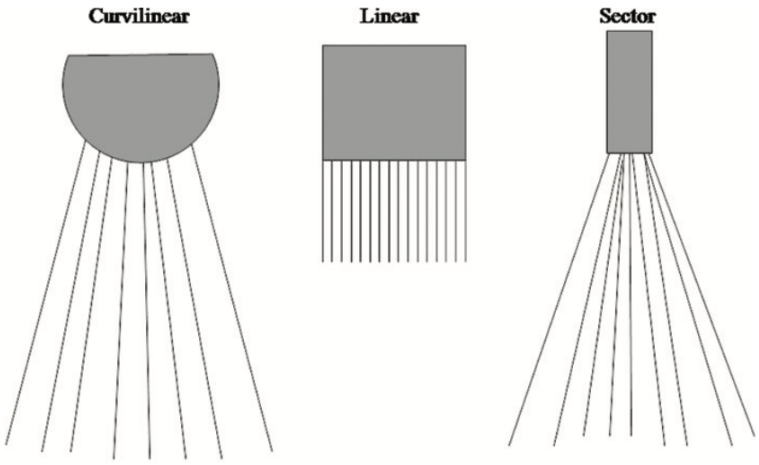

Figure 1.2. Types of arrays made of piezoelectric elements, used to emit ultrasounds and receive reflected echoes [2].

In the convex or curvilinear array, the piezoelectric crystals are arranged in a curved line. This configuration generates fan-shaped images, which broaden the field of view. The convex array requires fewer elements than the linear array to achieve a wider field of view, especially with increasing penetration depth [2]. It is commonly used for examining deep structures.
The sector or phased array is characterized by a smaller transducer area, which is useful for analyzing heart and cardiac structures. This configuration makes imaging of the thoracic structures between the ribs possible; the ultrasound passes through the intercostal spaces and diverges, increasing the depth field [3].

1.1.2 Ultrasound imaging

There are several main ultrasound imaging modes, defined based on the type of ultrasound emission. Continuous emission produces Doppler-mode images, while pulsed emission generates B-mode (Brightness mode) images. The M-mode (Motion mode) is based on a single line of ultrasound emitted continuously along a direction.

The B-mode (on the top of Figure 1.3) [3] is based on echoes generated by the reflection of ultrasonic waves at tissue boundaries and internal irregularities. Each echo is displayed as a bright dot, positioned in the image at the relative location of the structure that generated the echo within the body. Dot's brightness depends on the amplitude of the echo: structures that reflect a larger amount of ultrasound appear brighter than those that reflect a smaller percentage.

The B-mode system requires precise information on the target's distance from the transducer, as well as the position and orientation of the ultrasound beam. After the emission of an ultrasound pulse, the transducer operates in receive mode. Echoes from structures closer to the transducer return faster, resulting in brighter and more visible points on the image. Echoes from greater depths take longer. The distance of each echo from the transducer is correlated with its depth. This process, known as pulse-echo sequence, generates a single B-mode line. A complete ultrasound image is made up of 100 or more B-mode lines, each obtained from the sequential processing of reflected echoes [3].

The M-mode (on the bottom of Figure 1.3) is used in conjunction with the B-mode to show the real-time movement of structures, such as heart walls and valves. It provides a temporal image that shows how a given structure moves over time. In the M-mode image, the vertical axis represents depth, that is the distance between the body position and the echo recording point, while the horizontal axis defines time [5].

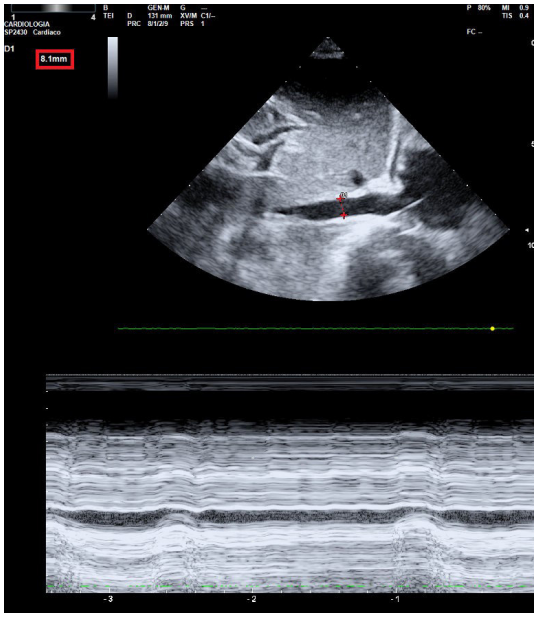

Figure 1.3. On the top of the image the B-mode acquisition and on the bottom of the image the M-mode acquisition.

Doppler imaging leverages the Doppler effect [6] to measure the motion of fluids within the body, capturing their direction, velocity, and flow rate. The Color Doppler mode (Figure 1.4), through the phase shift of the ultrasound, is used to define the direction of fluid movement, thus approaching or moving away from the transducer. Different colors are used (typically red and blue), the color hue is defined by the frequency shift of the waves [7]. The result is a velocity map: velocity is measured by repeated measurements, 8 to 16 lines of data are acquired in one direction and the velocity along it is calculated.

The process is repeated for different directions [4].
Power Doppler mode, attaching different color intensities, is used to define the magnitude of fluid flow: brighter areas represent higher flow rates [8].

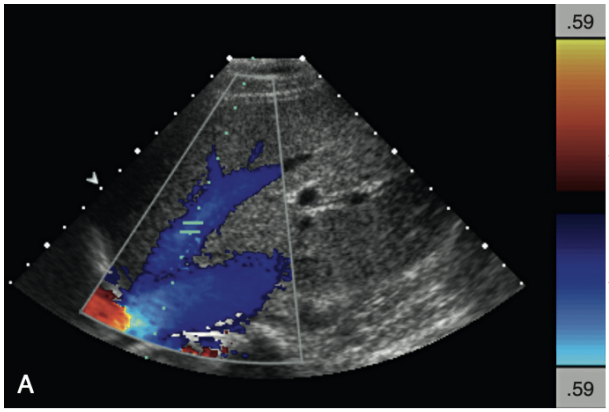

Figure 1.4. Color Doppler acquisition.

1.2 Cardiovascular system

The cardiovascular system is a complex system composed of organs and vessels responsible for controlling the speed and the amount of blood transported throughout the body, providing oxygen and nutrients to the tissues and removing waste products. It allows the maintenance of homeostasis and the proper functioning of organs. It includes the heart, arteries, veins, and capillaries.

1.2.1 Heart

The heart is an organ located in the thoracic cavity, positioned between the lungs. It extends downward and slightly to the left, between the second and fifth intercostal spaces, as shown in the Figure 1.5. In an adult male, the heart weighs approximately 350 grams [9] and pumps an average of 5 liters of blood per minute [10].

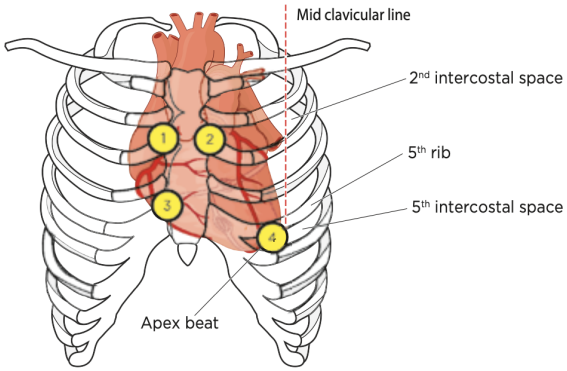

Figure 1.5. Placement of the heart between the second and fifth intercostal spaces, in the thoracic cavity. Valves: 1 = aortic valve; 2 = pulmonary valve; 3 tricuspid valve; 4 = mitral valve [9].

Anatomically, the heart is divided into four chambers [10]: 2 atria and 2 ventricles, separated by heart valves that ensure unidirectional blood flow. Specifically, the tricuspid valve separates the right atrium from the right ventricle whereas left atrium and left ventricle are separated by the mitral valve or bicuspid valve . The atria are the upper chambers of the heart, which receive incoming blood, while the ventricles are the lower chambers responsible for pumping blood. At the base of the large vessels emerging from the ventricles, there are the pulmonary valve, located between the right ventricle and the pulmonary trunk, and the aortic valve, located between the left ventricle and the aorta. An internal view of the heart is shown in the Figure 1.6.

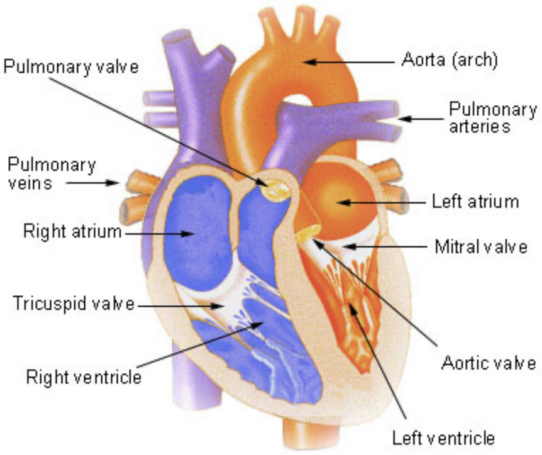

Figure 1.6. Internal view fo heart [10].

The cardiovascular system is organized into two main circuits: the pulmonary circulation, in which oxygenation of blood from the right ventricle to the lungs takes place. The second circuit is called the systemic circulation, where blood, thanks to the pumping of the left ventricle, reaches the tissues through capillaries. After delivering oxygen and nutrients, the de-oxygenated blood returns to the heart.

1.2.2 Arteries

Arteries are blood vessels that carry blood to the tissues and organs of the body from the heart, playing a crucial role in delivering oxygen and nutrients to the tissues and maintaining adequate blood flow. The major arteries in the body are shown in the Figure 1.7.

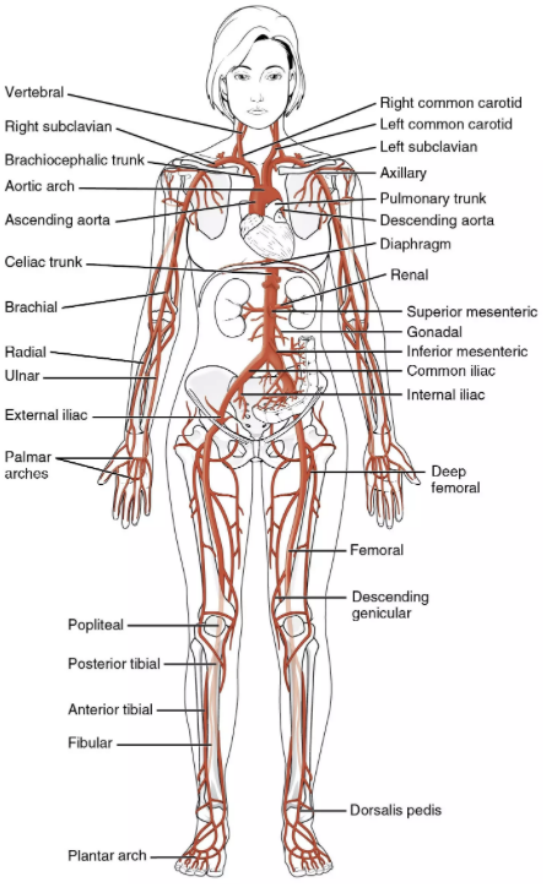

Figure 1.7. Main arteries of the human body [11].

Designed to withstand the high pressure generated by the heart, arteries are characterized by thick, elastic walls that allow for continuous blood flow between heart contractions. The pulmonary arteries carry oxygen-poor blood to the lungs, while the systemic arteries carry oxygen-rich blood from the left ventricle to the tissues of the body [10].

The wall of an artery consists of three layers [12]:

- **tunica intima**: simple squamous epithelium covered by endothelium, which is in direct contact with the blood. The endothelium allows for regulation of the muscular tone of the arterial wall, influencing smooth muscle constriction and relaxation, and contributes to increasing blood pressure. This layer is connected to the connective tissue by the basal membrane.

- **tunica media**: the thickest layer that contains smooth muscle cells and elastic fibers. Because of this layer, arteries are able, through dilation and constriction, to regulate blood flow and maintain arterial pressure. In larger arteries, the external elastic membrane separates the tunica media from the tunica externa.

- **tunica externa**: a connective tissue sheath containing groups of smooth muscle fibers and collagen fibers. It provides strength and support to the arterial wall.

The arteries closest to the heart are called elastic arteries. These vessels, with diameters greater than 10 mm, are thicker, and all three tunics contain a high percentage of elastic fibers. As arteries move further from the heart, the percentage of elastic fibers in the tunica media decreases, and the amount of muscle increases. Arteries of this type, with diameters ranging from 0.1 to 10 mm, are called muscular arteries [12].
Arteries branch into smaller vessels called arterioles, which are critical in regulating blood flow through the vessels via constriction and dilation mechanisms. Arterioles are not as elastic as larger arteries, they are stiffer because they are subjected to lower pressure [13].

The Figure 1.8 shows the three types of arteries and their wall structures.

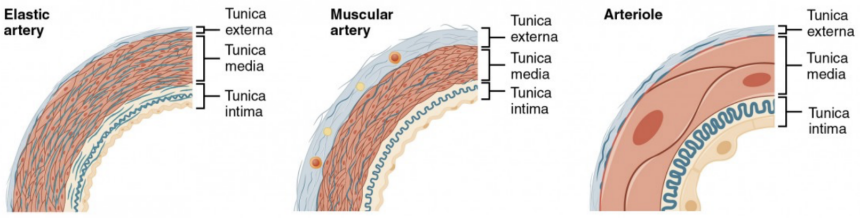

Figure 1.8. Representation of elastic artery, muscular artery and arteriole. Arteriole diameter is measured in micrometers while that of elastic and muscular artery is measured in millimeters [12].

1.2.3 Capillaries

Capillaries are the smallest blood vessels in the human body, with a diameter ranging from 5 to 10 micrometers [12]. Distributed in an intricate and branching network, they ensure that every cell in the body receives the necessary oxygen and nutrients. In certain areas, such as the heart, liver, kidneys, and skeletal muscles, capillaries form particularly extensive networks due to the high demand for oxygen and nutrients in these tissues. In contrast, the skin has a lower density of capillaries [11].
Capillaries serve as a bridge between veins and arteries (Figure 1.9) and are composed of a single layer of endothelial cells that allow the passage of oxygen and nutrients into the tissues, as well as the transport of metabolic waste from the tissues to the blood.

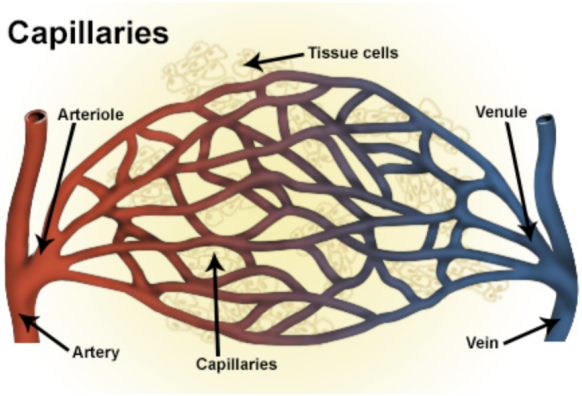

Figure 1.9. Capillaries representation, connection between arterioles and venules [10].

As shown in the Figure 1.10, capillaries can be of three main types [11]:

- **continuous**: a continuous layer of endothelial cells with some spaces between them. The basal membrane is uninterrupted. Only small molecules, such as water and gases, can pass through them.

- **fenestrated**: these capillaries have small openings or fenestrations in the endothelial cells. They allow the passage of small molecules, such as hormones and nutrients.

- **sinusoidal**: between the endothelial cells, there are wide intercellular gaps, and the basal membrane is incomplete. The passage of whole cells, such as red and white blood cells, is permitted.

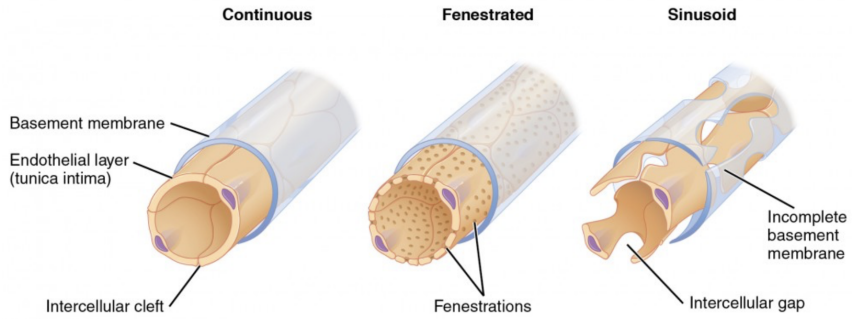

Figure 1.10. Three main types of capillaries: continuous, fenestrated and sinusoidal [12].

1.2.4 Veins

Veins are blood vessels responsible for transporting blood from the body's tissues to the heart. The main veins are shown in the Figure 1.11. A distinctive feature of veins, particularly those in the lower limbs, is the presence of unidirectional valves, which are folds of the tunica intima. These valves prevent blood from flowing backward and help counteract the force of gravity, ensuring blood flows toward the heart [13].

Oxygenated blood from the lungs to the left atrium of the heart comes through the pulmonary veins. There are four pulmonary veins: two from the right lung and two from the left lung. The systemic veins transport de-oxygenated blood from the body's tissues to the right atrium of the heart. The main systemic veins include the superior vena cava, which collects blood from the upper part of the body and the inferior vena cava, which collects blood from the lower part of the body [10].

The walls of veins are composed of the same three layers as arteries, as described in the Section 1.2.2. However, there are some significant differences [12]:

- **tunica intima**: in veins, it is smoother and contains fewer smooth muscle cells and elastic fibers compared to that of arteries, which appears more undulated.
- **tunica media**: it is much thinner in veins than in arteries, reflecting the lower pressure to which veins are subjected.
- **tunica externa**: it is thicker in veins to provide structural support.

Overall, the walls of veins are thinner than those of arteries because the blood pressure in veins is lower. However, veins have a greater capacity to carry larger volumes of blood (approximately 70% of total volume [14]).

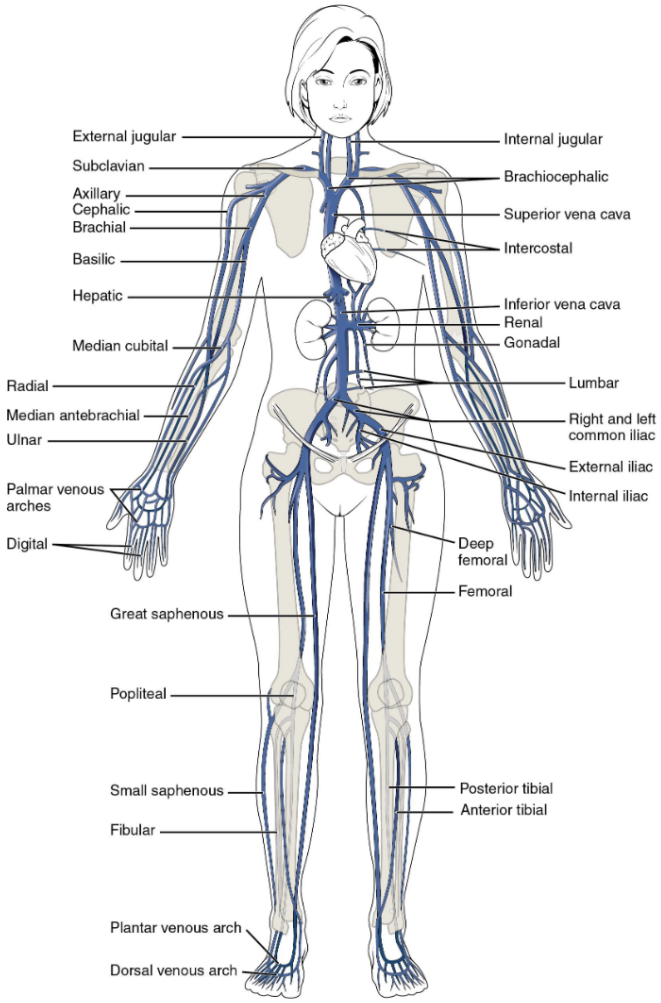

Figure 1.11. Main veins of the human body [11].

Veins with a diameter between 8 and 100 micrometers are called venules. Blood flows from the capillaries into the venules and then into progressively larger veins. The walls of venules are thinner than those of veins and vary depending on their diameter. They are composed of endothelium, a thin layer of smooth muscle cells and elastic fibers and a layer of connective tissue fibers [12].

The Figure 1.12 illustrates the structure of veins and venules.

Figure 1.12. Vein wall structure, on the right, and venule, on the left [12].

1.2.5 Inferior Vena Cava

The IVC, illustrated in Figure 1.13, is the largest blood vessel in the human body, that carries oxygen-poor blood from the lower part of the body to the heart. The inferior vena cava begins at the level of the fourth and fifth lumbar vertebrae, where the common iliac veins merge. Each of the common iliac veins is formed by the joining of the internal and external iliac veins. The internal iliac veins collect de-oxygenated blood from the pelvis, genital region, gluteal area, and the upper thigh, while the external iliac veins drain de-oxygenated blood from the lower limbs and abdominal wall [15]. The IVC consists of four segments: hepatic, suprarenal, renal and infra-renal [14], and is connected to the superior vena cava via lumbar veins, spinal venous plexuses and the azygos vein.

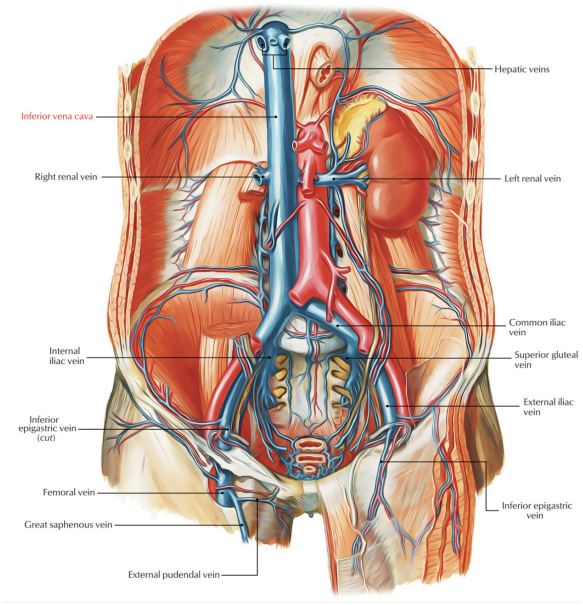

Figure 1.13. Position of the IVC, shown in blue.

The IVC is typically about 22 cm long and has a diameter range from 1.8 cm to 3.2 cm [14]. Larger or smaller diameters indicate pathological conditions such as hypovolaemia or constrictive pericarditis in the former case and heart failure or liver cirrhosis in the latter.

Expiration and inspiration cause movement of the IVC due to changes in intrathoracic and abdominal pressure. During inspiration, the thoracic cavity expands and the diaphragm lowers. As a result, the intrathoracic pressure decreases while the intra-abdominal pressure increases. The IVC undergoes a reduction in diameter and shifts toward the thorax. In contrast, during expiration the intrathoracic pressure increases while the intra-abdominal pressure decreases, resulting in the expansion of the IVC and a slight shift toward the abdomen.
Dynamic changes in IVC diameter are influenced by the respiratory cycle and cardiac activity. Pulsatility refers to changes related to heartbeat, while collapsibility concerns changes during respiration.

Typical analysis of IVC performed by physicians is done manually by calculating the caval index (CI), which is defined as the change in vessel diameter during one breathing cycle relative to the maximum diameter and is calculated as:

$$CI = \frac{max(D) - min(D)}{max(D)} \tag{1.1}$$

where max(D) and min(D) represent the maximum and the minimum diameters of the vessel during a breathing cycle. This approach is not standardized, is operator-dependent—affected by the operator's experience—and prone to measurement errors. The caval index provides information about the collapsibility of the investigated blood vessel, tissue compliance and transmural pressure [14].

The transmural pressure P_{tm} is defined as the difference between the internal (P_{int}) and external (P_{out}) blood pressure across the vessel wall, allows variations in vessel size to be assessed :

$$P_{tm} = P_{in} - P_{out} \tag{1.2}$$

Vessel dimensions increase with rising P_{tm}. The change in P_{tm} in the IVC is influenced by the change in P_{out}. The relationship between vessel dimensions, expressed in terms of volume V, and P_{tm} is typically represented by a pressure-volume curve, as shown in a simplified manner in Figure 1.14.

As shown in the Figure 1.14, if the average P_{tm} is low (as in the case of the IVC), size variations will be large, whereas with higher P_{tm} values, the vessel dimensions will be greater and the pulsatility will be lower. The slope of the curve defines vessel compliance,

which is a measure of how easily a blood vessel can dilate when internal pressure increases. An increase in P_{tm} implies a decrease in compliance.

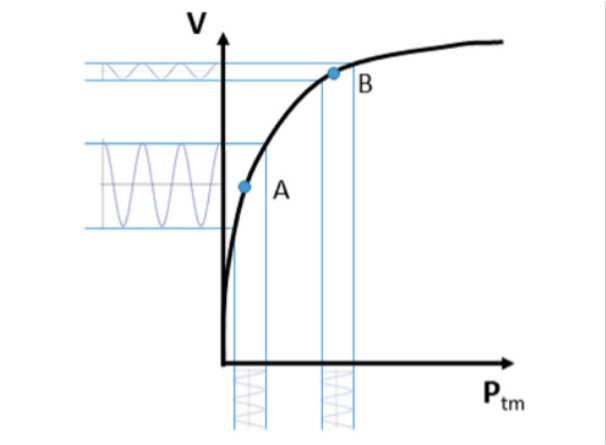

Figure 1.14. Curve defining the relationship between the volume of the venous blood vessel (volume V) and the P_{tm} [16].

An additional factor must also be considered is the extravascular compliance, which is the ability of extravascular tissues to adapt to vessel expansion. Thus, when measuring vessel volume variations in response to specific arterial pressure changes, the total compliance (C_{tot}) is evaluated. This accounts for both vascular compliance (C_v) and extravascular compliance (C_{ev}), according to the formula:

$$C_{tot} = \frac{1}{\frac{1}{C_{\mathrm{v}}} + \frac{1}{C_{\mathrm{ev}}}} \tag{1.3}$$

Low extravascular compliance can lead to an underestimation of the actual vessel compliance.

The analysis of ultrasound images involves not only simple visualization but also tissue tracking, which is crucial in clinical contexts such as lesion assessment, cardiac dynamics monitoring, or observing moving structures during surgical procedures. Tracking in ultrasound imaging allows for monitoring tissue movements to evaluate their overall position, allowing identification of useful parameters for clinical analysis.

Over the years, various object tracking algorithms have been developed. These algorithms differ in complexity and accuracy, and their effectiveness depends on factors such as image quality, the presence of artifacts, and the object's motion dynamics. Comparing these

algorithms is essential to identifying the most promising methods for the chosen purpose. In this work, several algorithms have been analyzed for tissue monitoring in ultrasound videos. The research is based on the implementation of algorithms available in OpenCV, leveraging their ease of use and robustness. These algorithms, originally designed for different purposes, such as tracking people in crowded squares or train stations [17], recognition people walking in different directions [18], have been implemented and tested on ultrasound videos for tracking the IVC. Their comparative analysis allowed for the evaluation of performance in terms of accuracy, robustness, and speed, aiming to identify the most suitable algorithm for automatic tracking of the inferior vena cava in ultrasound videos.

Chapter 2

Materials and methods

2.1 Dataset

The dataset used consists of 13 videos in .mp4 format, containing ultrasonic scans of the inferior vena cava in longitudinal section. To increase the variability of the dataset, the scans come from different sources: specifically, different ultrasound probes were used, and the videos represent inferior vena cava from different subjects. An example of an ultrasound scan is shown in Figure 2.1.

Figure 2.1. Example of an ultrasound scan, frame 0 of video `long 1.mp4`.

The 13 videos are composed of a different number of frames, a total of 3054, and are characterized by different resolutions and durations, as shown in the Table 2.1.

Video	Number of frames	Resolution	Duration (s)
long 1	189	1172x608	9
long 2	300	640x480	10
long 3	173	1172x608	9
long 4	217	1172x608	10
long 5	155	1172x608	10
long 6	125	1172x608	8
long 7	200	1172x608	9
long 8	116	1172x608	6
long 9	95	1172x608	6
long 10	398	1172x608	21
long 11	136	800x800	7
long 12	190	1172x608	9
long 13	758	800x800	15

Table 2.1. Name of videos with their number of frames, resolution and duration.

Each algorithm is tested on all videos, and performance is evaluated by comparison with ground-truth ROIs. For the construction of the ground-truth, the user manually selects the points that delimit the upper and lower edges of the vena cava. Based on these extreme points, the ROI is defined in the first frame, recording the coordinates of its center, width, and height. The point selection is repeated at regular frame intervals, allowing the ROI position to be updated and the new coordinates to be saved. The process is shown in Figure 2.2.

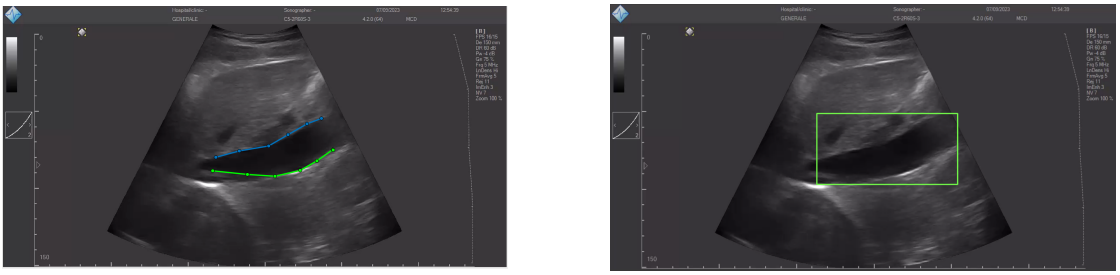

Figure 2.2. Frame 1 of video `long 9.mp4`. In the figure on the left, the manual choice of points delimiting the upper (blue points) and lower (green points) part of the vena cava. In the figure on the right, the ROI obtained from the identified points.

Considering the uniformity of the inferior vena cava movement, which changes little between consecutive frames, it was decided to keep the ROI unchanged for intervals of 10 frames. However, in the case of video `long 13.mp4`, due to the large number of frames compared to the other videos, the ground truth update interval was set to 30 frames.

2.2 Tracking algorithms

Tracking algorithms are used to follow the movement of the IVC in ultrasound videos to obtain a complete view of the vein. They can be useful in assessing the inferior vena cava, allowing the study of vein movements caused by breathing and cardiac activity. Furthermore, it is possible to analyze the mechanical response of the IVC to changes in transmural pressure. These changes provide insights into venous compliance, which is crucial for understanding how veins respond to different filling conditions and how they expand, acting as a reservoir for venous return [16]. Therefore, tracking allows for the automatic analysis and monitoring of the variations in the diameter of the blood vessel, overcoming the limitations of static measurement and operator dependence.

The tracking algorithms used to monitor the IVC in ultrasound videos belong to the OpenCV library, which is widely used in various fields such as image and video processing, and object recognition. These are semi-automatic algorithms that require input. In the case at hand, starting from the ground truth ROI position identified in the first frame of each video, various tracking algorithms are applied to compare their performance using temporal, accuracy, and precision metrics.

2.2.1 Lucas-Kanade algorithm

The Lucas-Kanade algorithm is one of the most widely used techniques for calculating optical flow, which measures the apparent motion of objects or surfaces between two consecutive frames in a video. It is assumed that the pixel intensity of the object of interest does not change between consecutive frames and neighboring pixels exhibit similar motion.

This method, as described in [19], uses a subsection of a video frame, $T(x)$, and aligns it with a target image, $I(x)$. Specifically, it aims to find the corresponding position of $T(x)$ in the next frame. This is done using the parameterized warp matrix $W(x;p)$, which allows the calculation of the new position of the pixel x in the next image after applying the transformation. p is a parameter vector that describes the transformation. The goal of the algorithm is to determine the parameters p that minimize the sum of squared errors between $T(x)$ and the warped image $I(x)$:

$$error = \sum_{x} |T(x) - I(W(x;p))|^2 \tag{2.1}$$

where $T(x)$ is the pixel intensity of the model and $I(W(x;p))$ is the pixel intensity of the deformed image. The transformation is not linear; at each iteration, the parameter vector p is updated incrementally [19].

The OpenCV adaptation of the library simplifies this method by minimizing the sum of

squared errors between a model $T(x)$ and the subsequent image in a video I, without considering deformations. Considering a pixel $I(x, y, t)$ in the first frame, in the next frame it will have moved by an amount equal to $(\Delta x, \Delta y)$ after a time Δt. The idea is that if a point moves slightly from one position to another between two consecutive images, its light intensity remains constant, thus [20]:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.2}$$

The algorithm uses the Taylor expansion to expand the image intensity:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{2.3}$$

By removing the common terms and dividing by Δt, the optical flow equation is obtained:

$$f_x u + f_y v + f_t = 0 \tag{2.4}$$

f_x, f_y and f_t are the image gradients in the x, y and time directions, respectively. To solve this, the Lucas-Kanade algorithm can be used in order to track objects or ROI within videos.

To track the inferior vena cava within ultrasound videos, the function `cv2.calcOpticalFlowPyrLK( prev_Img, next_Img, roi, winSize=(15, 15), maxLevel=2, criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 15, 0.02))` [20] is used, where:

- `prev_Img` is the first grayscale frame used as a reference to calculate the movement of the points.
- `next_Img` is the second grayscale frame where the position of the points, which have moved relative to the previous frame, is to be found.
- `roi` is an array of coordinates representing the initial points to be tracked in the previous frame.
- `winSize=(15, 15)` represents the size of the search window for each level of the pyramid, which is used to calculate the optical flow. The pyramid is a data structure used to represent an image at different resolutions, with the full image at the base and progressively smaller images as as the levels of the pyramid advance.
- `maxLevel=2` represents the maximum number of levels in the pyramid used by the algorithm.

- `criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 15, 0.02))` defines the stopping criteria of the algorithm; in this case, a maximum number of iterations equal to 15 and a minimum change between iterations of 0.02.

2.2.2 Kernelized Correlation Filters algorithm

The Kernelized Correlation Filters (KCF) algorithm is available in OpenCV and is used for object tracking in videos. It employs a correlation filtering approach where a model of the object to track is identified in the first frame and then used to detect its position in subsequent frames [18].

The algorithm is based on the use of a trained classifier capable of distinguishing between positive examples (region of interest) and negative examples (background). The classifier is based on features derived from the calculation of the Histogram of Oriented Gradients (HOG). Specifically, the HOG features are identified through the following steps [21]:

- **Color space normalization** to reduce lighting and shadow effects. The gamma correction technique is used.
- **Calculation of horizontal and vertical gradients**.
- **Construction of local histograms based on gradients**. The frame is divided into cells, and a local histogram describing the distribution of gradient directions within the cell is constructed for each.
- **Normalization and combination of histograms** to obtain a feature vector. Cells are overlaid and histograms combined. Normalization helps to reduce the impact of local lighting variations.

The global HOG vector represents the input for the KCF classifier. To improve accuracy, circulant matrices [21] are used to cyclically shift the tracking area in order to generate synthetic samples. They are computationally expensive to manipulate, so to speed up the correlation calculation, the Discrete Fourier Transform (DFT) is applied.
The algorithm uses HOG features to construct the correlation filter, whose response provides a probability map of where the object is located in the analyzed frame. Each candidate region is compared with the target model, and the position of the target in the next frame corresponds to the region with the highest correlation.

KCF [18] may be subject to undesirable edge effects, which can compromise accuracy. However, for tracking the inferior vena cava in ultrasound images, the vein is almost always positioned at the center, significantly reducing the impact of edge-related problems.

The Python function `cv2.TrackerKCF` [22] is employed to track objects in US videos. After the tracker has been created, the method `init()` is used to set the initial ROI ,

and the `update()` method is used to track the object in subsequent frames. This approach allows for the tracking of the object as it moves through the video, leveraging the KCF tracking algorithm's features for robust tracking even in challenging conditions.

2.2.3 Multiple Instance Learning algorithm

The Multiple Instance Learning (MIL) tracker is part of the OpenCV library and allows tracking of the inferior vena cava in ultrasound videos. The workflow is illustrated in the Figure 2.3.

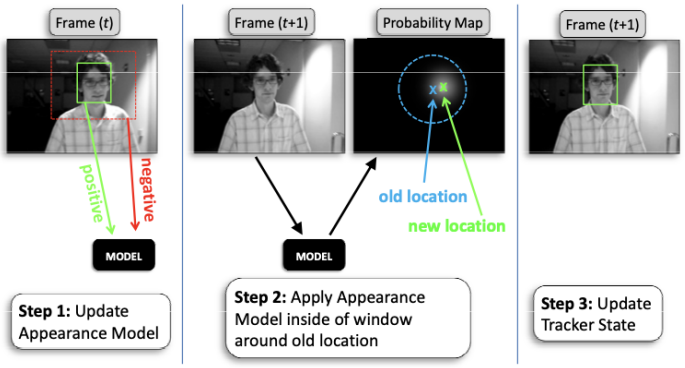

Figure 2.3. Illustration of MIL tracker operation [23].

The image is divided into patches, each represented by a set of Haar-like features [23]. To obtain an overall estimate of the features in a patch, the weight values of all the rectangles into which the patch is divided are summed. Each rectangle has a weight that reflects importance relative to the whole patch.

The MIL tracker is based on a supervised learning approach, where a classifier returns a probability function $p(y|x)$, where x is a patch into which the image is divided and y is a binary value indicating the presence of the object of interest in x [23]. The probability function $p(y|x)$ takes continuous values between 0 and 1. Starting from the manual identification of the ROI, a set X_s of patches is defined to be analyzed in the new frame to search for the object within a radius s. Specifically, X_s is defined as:

$$X_s = \{x \mid s > \|l(x) - l^*_{t-1}\|\} \tag{2.5}$$

where l^*_{t-1} is the estimated position of the object in the previous frame, and $l(x)$ is the position of the patch in the image. For each new frame, $p(y|x)$ is calculated for all x belonging to X_s.

The patches containing the object are labeled as positive. A radius $r < s$ is considered to identify the set of positive patches:

$$X_r = \{x \mid r > \|l(x) - l_t^*\|\} \tag{2.6}$$

The patches that do not contain the object are labeled as negative. A region with a radius between r and β is considered to identify the set of negative patches:

$$X_{r,\beta} = \{x \mid \beta > \|l(x) - l_t^*\| > r\} \tag{2.7}$$

The patches are collected into two bags: the positive bag contains at least one positive patches, while the negative bag contains the negative patches. MIL operates on bags of patches.

During tracking [23], the trained MIL classifier is used to predict the position of the object in subsequent frames. The tracker analyzes a series of positions within the frame, each of which is evaluated by the classifier, which returns the probability of belonging to the object or the background. For each new frame, the tracker identifies the maximum value of the probability function as the position of the object.
It is an iterative process, where the classifier is updated when the patch with the highest score is identified.

To track the IVC, the `cv2.TrackerMIL` function [24] is used similarly to `cv2.TrackerKCF`, as described in Section 2.2.3. This tracker relies on a multiple instance learning approach, where the object is tracked by distinguishing between positive and negative bags.

2.2.4 Tracking- Learning - Detection algorithm

The Tracking - Learning - Detection (TLD) framework is an object tracking algorithm for videos, composed of three independent components [25]: tracking, learning, and detection. These three phases operate independently but interact with each other as shown in the Figure 2.4. These three phases can be performed simultaneously on separate processors or as separate tasks [26].

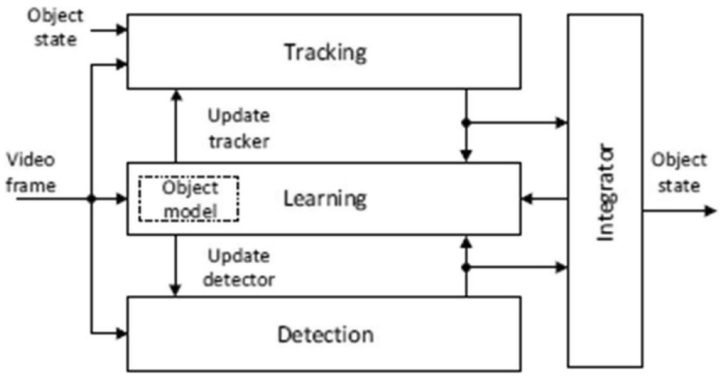

Figure 2.4. Three components of TLD framework [27].

In the tracking phase, it is assumed that the object is always visible and its movement is limited. Based on information from previous frames, the system can detect the motion of the object of interest [25] using the Lucas-Kanade algorithm [27]. However, if the object moves out of the field of view, tracking may fail.

The learning phase [27] improves the system by assuming that the other two phases may fail in certain situations. It identifies and learns from cases where the tracker makes errors, such as when it incorrectly labels positive or negative patches. Positive samples are patches where the object is actually present whereas negative samples are patches where the object is absent. When the tracker identify incorrect negative samples, which the algorithm adds to the training set with a positive label, thereby refining the classification model. Conversely, when the tracker incorrectly identifies a patch as positive, the algorithm labels it as negative and adds it to the training set.

The detection phase is responsible for locating the object when it is either occluded or completely absent from the frame, performing a full scan [25]. Detection relies on three cascading classifiers [27], as shown in the Figure 2.5 and described below:

- **Variance Classifier**: compares the variance of a candidate window with that of the initial target window. If the variance of the candidate window is greater than half of the initial target's variance, the window proceeds to the second classifier. This step helps remove background regions, which are characterized by low variance.
- **Random Forest Classifier**: the object model is extracted and compared with previously stored models in the pool (a collection of target models). If the similarity exceeds a certain threshold, the window is labeled as positive; otherwise, it is labeled as negative. At this stage, if the similarity between the current model and those in the pool exceeds a second threshold, the window is considered as the target by the detection system.

- **Nearest Neighbor Classifier**: a classification algorithm that stores existing cases and classifies new data based on a similarity measure. If the similarity is high, the window is considered as belonging to the same class as the target.

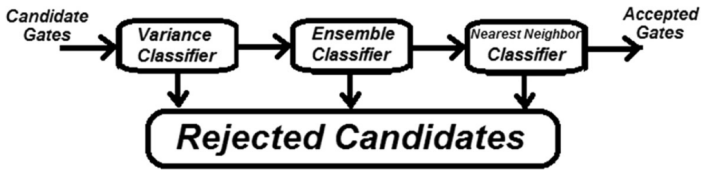

Figure 2.5. Block diagram of the detection phase in the TLD algorithm [27].

From an implementation perspective for tracking the IVC in ultrasound videos, the Python function `cv2.TrackerTLD` is used [28]. The process begins with the creation of the tracker: `cv2.TrackerTLD_create()`. Then is performed its initialization using the `init()` method, applied to the initial frame of the video where the tracking starts. Subsequently, the `update()` method is used to update the tracker in the following frames, allowing continuous monitoring of the ROI's position throughout the video.

2.2.5 Minimum Output Sum of Squared Error algorithm

The Minimum Output Sum of Squared Error (MOSSE) tracker is based on the correlation between the object model and the analyzed frame in order to determine its position. It uses adaptive correlation filters [29]; to perform tracking, a filter is created that represents the object's appearance. This filter is trained so that, when convolved with the object's image, the analysis of the filter's response allows the identification of the target's position in the frame. To train the filter, MOSSE generates 8 affine transformations of the template, resulting in a total of 9 input images. Each image corresponds to an output response that takes the shape of a two-dimensional Gaussian, with the peak indicating the object's position. The convolution operation between the filter and the object's image is expressed as:

$$f_i \otimes h = g_i \tag{2.8}$$

The symbol $\otimes$ denotes the convolution operation.
To optimize tracking, the tracker uses the Fast Fourier Transform (FFT). The convolution operation is transformed into point-wise multiplication, as stated by the convolution theorem:

$$F_i \odot H^* = G_i \tag{2.9}$$

The symbol $\odot$ indicates the dot product of the elements and the symbol $*$ represents the complex conjugate of the filter.
The goal in training the filter is to minimize the sum of squared errors between the convolution output and the specific response image:

$$H = \min_H \sum_i |F_i \odot H^* - G_i|^2 \tag{2.10}$$

Once the target is located, the filter is dynamically updated to adapt to changes in the object's appearance. The workflow of the MOSSE algorithm is illustrated in Figure 2.6.

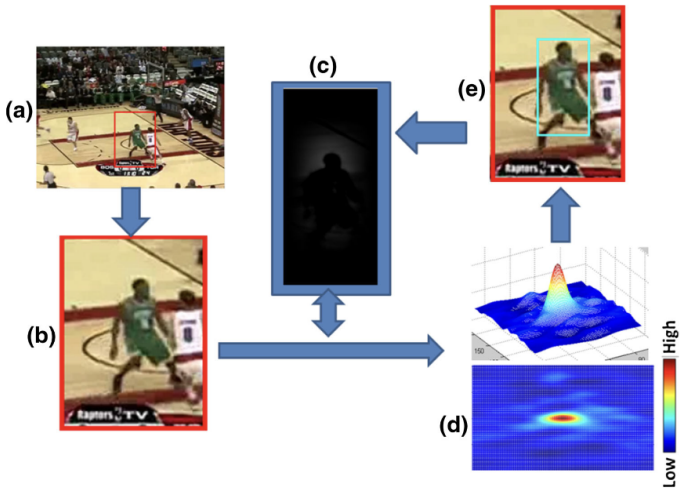

Figure 2.6. The workflow of MOSSE tracking. a = frame, b = search region, c = correlation filter, d = respond output, e = target in green box [29].

This tracking algorithm is robust to variations in lighting, scale, and deformations. It is capable of following objects that change shape over time, making it suitable for dynamic scenarios such as tracking the inferior vena cava. It is also useful for detecting occlusions, by analyzing the ratio between the peak of the correlation and the lateral lobe [18], the algorithm works even when the object is not visible.

The Python function `cv2.TrackerMOSSE` [30] is used for the purpose of this thesis project.

2.2.6 Channel and Spatial Reliability Tracker algorithm

The Channel and Spatial Reliability Tracker (CSRT) is a tracker that combines learning techniques based on spatial features and channels to track an object in a video. It builds a representation of the object to be tracked using a grid of channels, each describing a different feature of the object, such as color, texture, or contours. These features are combined with spatial information to create a representation that includes both the shape and the position of the object in the frame.

The CSRT tracker not only uses the filter but also relies on two types of reliability that allow it to adapt to changes in the appearance of the object [18]:

- **Spatial reliability**: the spatial reliability map dynamically adapts the filter to the shape and position of the object, and is used to determine the filter's size. Additionally, the map adjusts the filter so that it is not bound by the object's periodic movement, allowing it to search within an unconstrained area in the frame [31].
- **Channel reliability**: each channel is weighted according to its importance for tracking. The algorithm measures the reliability of each channel with a score. These scores allow the proper weighting of the various channels when calculating the object's position.

The filter's response is a correlation map that represents the probability that a particular area in the frame contains the object. The peak of this map indicates the most probable position of the object within the frame.
After each frame, the filter is updated by combining the spatial and channel features. This continuous update process allows the tracker to maintain high performance even when the object undergoes deformations or scale variations.

To track the IVC, the tracker is created using the Python function `cv2.TrackerCSRT_create()` [32]. It is then initialized with the bounding box in the first frame of the video and updated as the subsequent frames are analyzed.

2.2.7 Block Matching algorithm

The Block Matching Algorithm (BMA) is based on dividing the frames into non-overlapping blocks and searching for correspondences between the pixel blocks of consecutive frames. Each block in the current frame is compared with blocks in the subsequent frame by shifting it within a predefined search area. For each possible shift of the block, a measure of the distance between the intensity values of the pixels in the two blocks is calculated. The position that results in the smallest distance is identified as the best match [33].

In the context of IVC tracking, normalized cross-correlation (NCC) was used to identify the best match between the blocks, as indicated in the Formula 2.11.

$$NCC = \frac{\sum_{i,j}(I_1(i,j) - \mu_1)(I_2(i,j) - \mu_2)}{\sqrt{\sum_{i,j}(I_1(i,j) - \mu_1)^2 \sum_{i,j}(I_2(i,j) - \mu_2)^2}} \tag{2.11}$$

Where:

- I_1 and I_2 are the two blocks being compared,
- μ_1 and μ_2 are the mean values of the blocks I_1 and I_2 respectively.

The NCC value returns a number between -1 and 1. 1 indicates perfect correlation, that is, the two blocks have exactly matching pixels, and -1 indicates perfect negative correlation.

The BMA is very fast and suitable for real-time applications, but its accuracy can decrease in the presence of object deformations, lighting changes, noise, occlusions, or the appearance of new objects.

In the specific case of IVC tracking, starting with the ROI in the first frame of the video Block Matching is update to track the movement of the ROI. The NCC coefficient is used to measure the similarity between the examined block and the ROI in previous frames. A higher value indicates a better match, helping to identify the most probable position of the object.

2.2.8 Mean Shift algorithm

Mean Shift is a probabilistic method based on a probability distribution, used to find the optimal position of an object through an iterative process of shifting the search window towards the region of highest density.

First, the histogram of the intensity values of the ROI identified in the first frame of the video is built, which represents the target model to be tracked. For the subsequent frame, the histogram of the region where the object is presumed to be located is calculated, and the similarity between the target model and the candidate model is measured using the Bhattacharyya coefficient [34], defined by the Formula 2.12:

$$\rho(p,q) = \sum_{i=1}^{m} \sqrt{p_i \cdot q_i} \tag{2.12}$$

where p is the histogram of the target model, q is the histogram of the candidate model and m is the number of the bins in the histogram.

To reduce the influence of pixels far from the center of the ROI, the Epanechnikov function [34] is employed, assigning greater weight to pixels closer to the center of the search window compared to those farther away.
The maximum value of the Bhattacharyya coefficient guides the Mean Shift vector, which determines the direction and magnitude of the shift needed to center the ROI on the object. The algorithm is iterative and updates the target's position by calculating the Mean Shift vector at each iteration. The process ends when the vector becomes very small or zero, indicating that the object has been correctly located. The new position of the ROI represents the object's position in the current frame. For subsequent frames, the process is repeated.

The Mean Shift algorithm has been implemented to perform tracking of the IVC within the ultrasound videos. The ROI in the first frame is used to extract color features by calculating the histogram in the HSV (Hue Saturation Brightness) color space. After applying a mask to exclude overly dark or bright values, the tracking begins. For each subsequent frame in the video, a back-projection of the histogram is performed on the current frame, producing a probability map that allows us to understand which areas are most likely to be where the object is located.

To apply the algorithm, the Python function `cv2.meanShift(back_proj, track_window, term_crit)` [35] is used, where:

- `back_proj` is the probability map calculated using the histogram of the ROI defined in the first frame.
- `track_window` is a rectangle that defines the search window in the current image. It is the starting point of the tracking and is updated at each iteration of the algorithm.
- `term_crit` is the termination criteria that determines when the algorithm should stop. In this case, the algorithm stops when the position of the window changes by less than 1 and after 10 iterations.

2.2.9 Boosting algorithm

The Boosting algorithm in OpenCV is based on AdaBoost with a HAAR cascade classifier [31]. AdaBoost is a machine learning algorithm that combines multiple weak classifiers into a strong classifier, thus increasing the overall accuracy of the model. Tracking is performed as illustrated in the Figure 2.7.

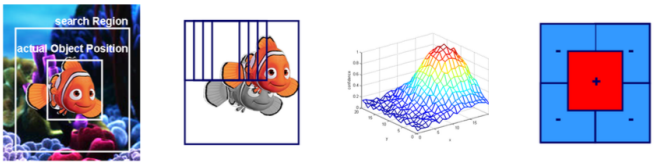

Figure 2.7. Steps of Boosting algorithm. Given the initial position of the object (a) in frame t, the classifier is evaluated at various candidate positions within a surrounding search area in frame t + 1. The confidence map (c) obtained is analyzed to identify the most probable location, and the tracker (classifier) is then updated based on this estimation (d) [36].

During initialization phase, it is assumed that the target object has already been detected. The region containing the IVC is considered a positive sample for the tracker. To enable the model distinguish the object from the background, negative samples are extracted from non-overlapping regions around the target area [18].

Each weak classifier [36] is trained to evaluate the visual features of the target area and determine whether each pixel belongs to the tracked object or not. When analyzing a new frame, for each position of the ROI, the response of each weak classifier is evaluated, which returns a confidence map. This map assigns a score to each position in the image based on its similarity to the target object.
Each weak classifier is assigned a weight based on its accuracy. A classifier with a lower error rate receives a higher weight. The final combination of weak classifiers to form the strong classifier is based on the weights assigned. The confidence values from all the weak classifiers are combined using a weighted sum, in order to select the area with the highest confidence value. Then, the tracking window is moved to the position corresponding to the maximum confidence value.

In every frame, the tracker updates the position estimate of the object by analyzing the ROI based on the output of classifiers trained in the previous frame.

The Python function `cv2.TrackerBoosting_create()` [37] is used to create the tracker. Starting from the ROI identified in frame 0, tracking begins and the tracker is updated in subsequent frames.

2.2.10 Median Flow algorithm

Median Flow is a visual tracking algorithm implemented in OpenCV, designed to follow the movement of an object between frames of a video. It is based on the principle

of bidirectional optical flow consistency, using the points tracked in previous frames as references to estimate the movement of the object [38].

In the current frame, the ROI containing the object to be tracked is identified, and a grid of points is initialized within it. These points serve as references for calculating the object's movement, with each point being tracked through the Lucas-Kanade algorithm [38]. The quality of the optical flow predictions for each point is evaluated through two filtering processes [39]:

- **filtering via normalized cross-correlation (NCC)** : points are excluded if the correlation between pixel blocks in the two frames is below a certain threshold.
- **filtering via Forward-Backward (FB) error** : points with excessive propagation error are discarded. The FB error is the Euclidean distance between the tracked points in the forward direction (from the current frame to the next) and in the reverse direction.

The worst 50% of the predictions are discarded [38], while the remaining ones are used to estimate the displacement of the entire ROI. This estimate is calculated by computing the median of the displacements of each point.

To describe how the tracked object changes in size or deforms over time, the scale variation [38] is used, defined as the ratio between the medians of the distances between pairs of points across frames. If the value is greater than 1, it indicates magnification, while if it is less than 1, it indicates reduction.
The algorithm assumes that the object to be tracked consists of small rigid patches [38], in fact flexible parts or edges of the object are excluded from the process as they could compromise the accuracy of tracking.

To track the IVC using the Median Flow algorithm, the Python function `cv2.TrackerMedianFlow` [40] is used. After creating and initializing the tracker, tracking continues with monitoring the position of the object in the subsequent video frames.

2.2.11 Oriented FAST and Rotated BRIEF

The Oriented FAST and Rotated BRIEF (ORB) algorithm belongs to the OpenCV library, used to detect, describe and compare image features, which are useful for tracking objects in videos.

ORB combines two techniques [41], as shown in Figure 2.8:

- FAST (Features from Accelerated Segment Test): allows to detect points of interest, known as corners, within an image.

- BRIEF (Binary Robust Independent Elementary Features): generates binary descriptors that enable the comparison of the corners detected.

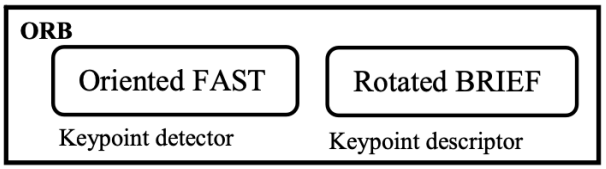

Figure 2.8. The component diagram of ORB, which include FAST and BRIEF [42].

The first step involves applying the FAST algorithm to identify corners. This algorithm analyzes an imaginary circle around each candidate point and compares the intensity of the pixels along the circle with that of the central pixel. A point is classified as a corner if at least a certain number of pixels in the circle have intensities significantly higher or lower than that of the central pixel. A corner is determined using Formula 2.13 [41]:

$$N = \sum_{A \in \text{circle}(p)} \text{f}_{\text{CR}}(I(p), I(x)) \tag{2.13}$$

where:

$$f_{\text{CR}} = \begin{cases} 1 & |I(x) - I(p)| > \epsilon_{\text{d}} \\ 0 & \text{others} \end{cases} \tag{2.14}$$

$I(x)$ is the gray value of any point in the circle, $I(p)$ is the gray value of the center of the image, p is the center point, and ϵ_{d} is the minimum threshold given. With the Formula 2.13 the number N of pixels is calculated, if $N > \epsilon_{\text{d}}$ then the point will be a corner.

To define the dominant angle around the corner ORB uses the intensity centroid [42]:

$$\theta = \text{atan2}(m_{01}, m_{10}) \tag{2.15}$$

where moments are defined as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{2.16}$$

I(x, y) is the pixel intensity in the patch surrounding the keypoint.

The BRIEF algorithm randomly selects a number of pairs of points around the identified corners [41]. The goal is to obtain a binary descriptor for each identified corner, it is constructed by comparing the pixel intensities for each pair. If one pixel is brighter than the other, 1 is recorded otherwise 0.

To associate points between successive frames, the obtained descriptors are compared using the Hamming distance [41], defined as the sum of the results of the XOR operation between the corresponding bits of the two descriptors:

$$D(K_1, K_2) = \sum_{i=1}^{n} K_1[i] \oplus K_2[i] \tag{2.17}$$

K_1 and K_2 are the descriptors and n is the descriptor length.

Additionally, ORB ranks the keypoints based on the Harris score [41], to reduce the number of keypoints to compare in different frames. The Harris score identifies how much a point is a corner: a point is defined as such if a shift in any direction around it causes a significant change in pixel intensity.

From the implementation point of view [43], starting from the ROI in frame 0 of the video, the keypoints in it are identified. For each subsequent frame, the ROI corresponding to the updated position of the precedent frame is extracted, taking into account the observed movements. ORB is used to detect the keypoints in the new ROI, and the descriptors computed in the first frame are compared with those obtained in subsequent frames through Hamming distance.

2.3 Metrics

To determine which of the algorithms described in Section 2.2 is best suited for tracking the inferior vena cava in ultrasound videos, several evaluation metrics are considered.

For each video, the execution time of the algorithms is calculated, defined as the time taken by the tracker to track the object throughout the video. Consequently, frames per second (FPS) is used to determine how many frames the algorithm is capable of processing per second. Another metric used is the percentage of failed frames, which is used to understand whether the tracker is able to track the object of interest within all frames of the video.

The area of the ROI resulting from the tracking of each algorithm is then compared with the one containing the IVC, defined as the ground truth. To this end, the Intersection over Union (IoU) is calculated, defined as the ratio between the intersection area and the union area of the two ROIs. The IoU takes values between 0 and 1, where 1 indicates perfect overlap. By setting a threshold of 0.6 for the IoU, the percentage of false positives

(FPP) is calculated for each algorithm. The FPP indicates how many times the tracking algorithm incorrectly detects an ROI that does not match the ground truth, so it does not recognize the area containing the IVC. Additionally, considering the centers of both the predicted and ground truth ROIs, their Euclidean distance was calculated. Finally, the Euclidean distance between the ROI centers in successive frames is calculated, both for the tracker-detected and ground truth ROIs. The corresponding distances are compared and, by applying a threshold of 0.5 cm, the number of cases where the distance exceeds this value is determined.

Mean values and standard deviations (std) of IoU, FPP and Euclidean distance are calculated for each algorithm. An example of the calculation of some of these parameters is shown in the Figure 2.9.

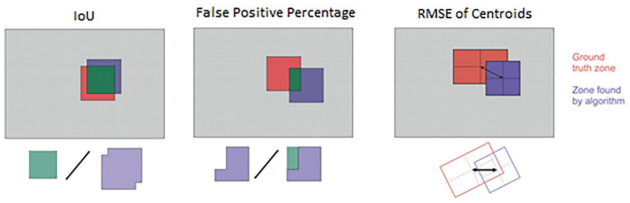

Figure 2.9. Metric visual representation [31].

2.4 Python implementation

The objective of this thesis project is to evaluate the tracking algorithms described in Section 2.2, in order to determine which is the most effective for tracking the inferior vena cava in ultrasound videos. The evaluation is carried out by calculating the metrics described in Section 2.3.
To achieve this goal, a step-by-step approach was adopted, dividing the flowchart into three main phases.

First phase Starting from the initial frame of the video (frame 0), the first phase involves (Figure 2.10) the identification of points to define the conversion from pixel to cm: the physician selects two points on the scale present in frame 0 to define a distance equal to 1 cm, allowing calculation of the conversion factor cm/pixels. Using the distance between these points, the conversion factor is calculated as the ratio between 1 cm and the corresponding length measured in pixels in frame 0. The conversion factor is used to limit the area of tracker movement in the video to calculate some metrics.

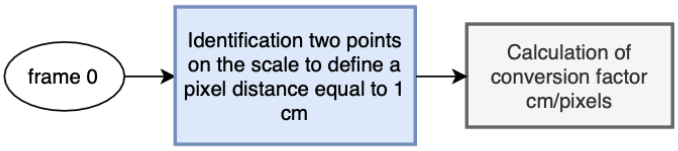

Figure 2.10. Flowchart for the calculation of the cm/pixel conversion factor. In blue, steps performed by the clinician.

An example of the conversion factor calculation is illustrated in Figure 2.11.

Figure 2.11. Frame 0 of video `long 1.mp4`. The distance (in yellow) between the red points, identified by the clinician, is equal to 10 mm, in this case 45.045 pixels. Calculation of conversion factor: 1 cm/ 45.045 pixels = 0.0222 cm / pixels

Second phase As illustrated in the flowchart in Figure 2.12, starting from the position of the ROI ground truth determined in the first frame of the video, the tracking algorithms described in Section 2.2 are applied. For each algorithm, the following metrics are calculated: execution time, frame rate (FPS), and the number of frames where the tracker fails. These parameters are stored in a file named `temporal_metrics.txt` (green in the Figure 2.12). Additionally, for each frame, the center coordinates of the ROI, along with its width and height in pixels, are saved in a .txt file.

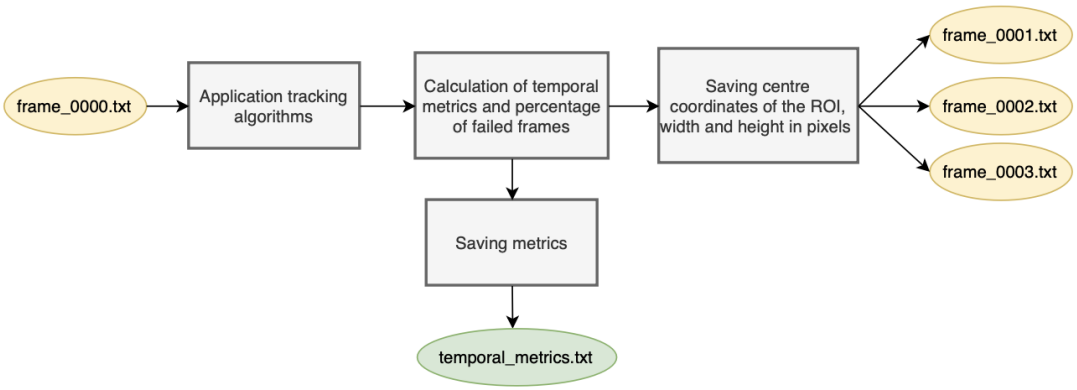

Figure 2.12. Flowchart for calculating center, width and height coordinates of the ROI for each frame of the video and for calculating metrics.

Third phase To evaluate the performance of each algorithm, the data of the ROI considered as ground truth (shown in orange in Figure 2.13) is compared with the data of the ROI obtained from the tracking algorithm (shown in yellow in Figure 2.13). The accuracy metrics calculated include IoU, FPP, Euclidean distance of the centroids and the percentage of unreliable distances, as specified in Section 2.3. These metrics are saved in the `metrics_results.txt` file, reporting the average values and std for each video.

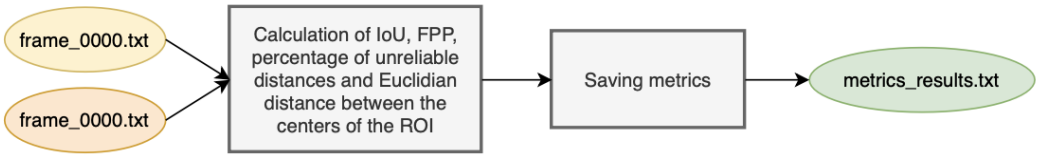

Figure 2.13. Flowchart for comparing the ground truth ROI (in orange) with the ROI identified by the algorithm (in yellow) and subsequent calculation of metrics.

Chapter 3

Results and discussion

The 11 algorithms described in Section 2.2 were tested on the videos from the dataset.

For each video, the time interval between the start of the tracking (corresponding to frame 1 of the video) and its conclusion is measured. The results show that the fastest algorithms in completing the tracking are Mean Shift, Lucas-Kanade, Median Flow, and MOSSE, while the slowest algorithm is MIL. Considering the total number of frames that make up the video, it can be observed that the greater the number of frames, the longer it will take for the algorithm to perform the tracking. The execution times are consistent across the videos. The Figures 3.1 shows the execution time graphs of the various algorithms for all the videos.

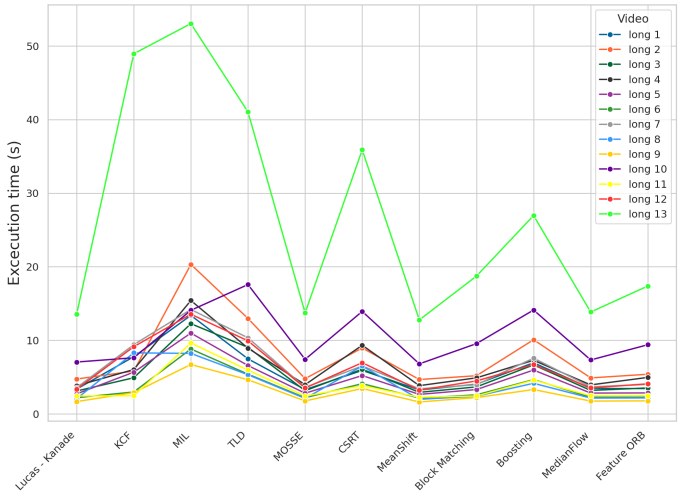

Figure 3.1. Scatter plot representing for each video the execution time of the algorithms.

To assess how quickly the algorithms can process each frame of the video, the FPS value is calculated, defined as the ratio between the total number of video frames and the execution time of the tracking. The average results are shown in Figure 3.2.

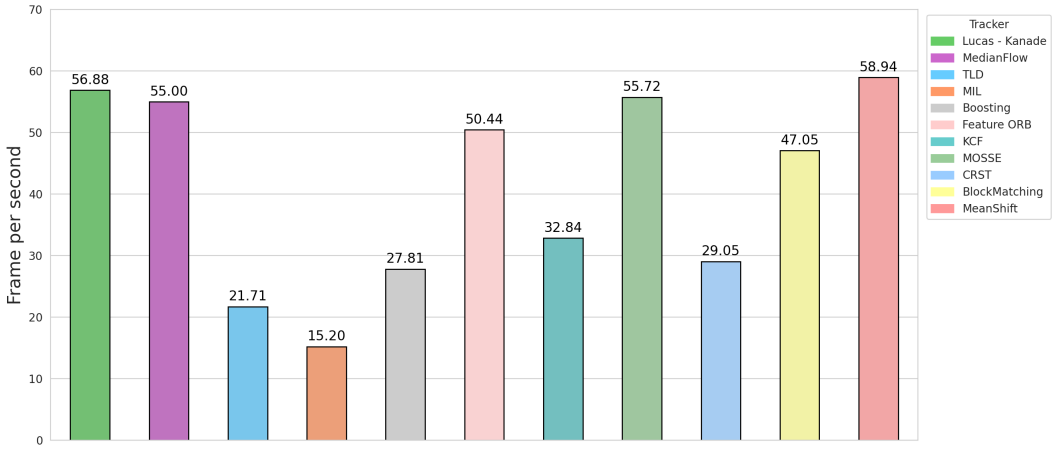

Figure 3.2. Bar plot of average FPS values for each algorithm.

In IVC tracking, the FPS value of an algorithm is a crucial indicator for assessing its usefulness in different applications. A high FPS ensures smooth and real-time monitoring, allowing physicians to observe the overall position of the IVC in the videos. The average FPS values were calculated for each algorithm.
The Figure 3.2 shows the results: the Mean Shift tracker achieves the highest FPS (approximately 59 frames per second), followed by Lucas-Kanade, MOSSE, Median Flow, Feature ORB, Block Matching and KCF. The MIL tracker stands out negatively, processing only 15 frames per second.

Based on the dataset videos, the percentage of failed frames is calculated for each algorithm, indicating the inability of the tracker to detect the vein in all the frames of the video. The algorithms affected by this issue are Median Flow, MOSSE and Lucas-Kanade, while the remaining ones successfully detect the IVC in all frames.
For the algorithms with this metric not equal to zero, a scatter plot is created to show the value of this percentage for each video. The reference Figures are 3.3, 3.4 and 3.5.

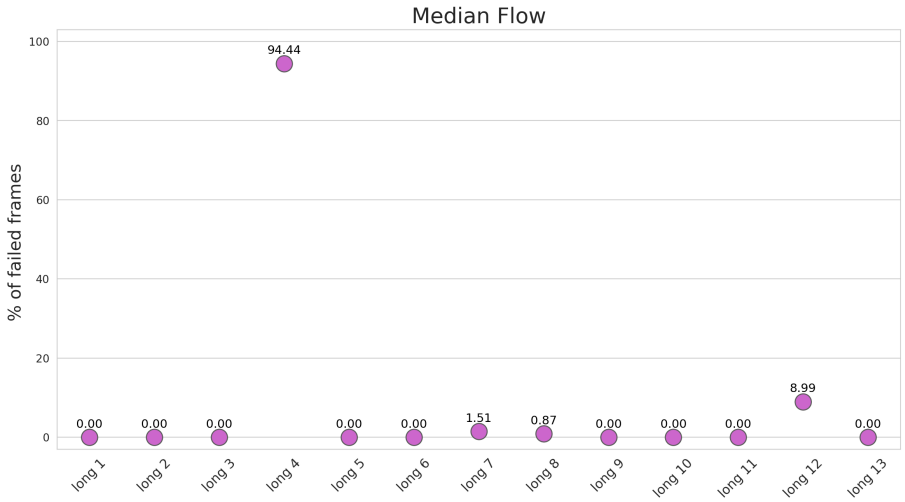

Figure 3.3. Scatter plot of the Median Flow algorithm showing the percentage of failed frames for each video.

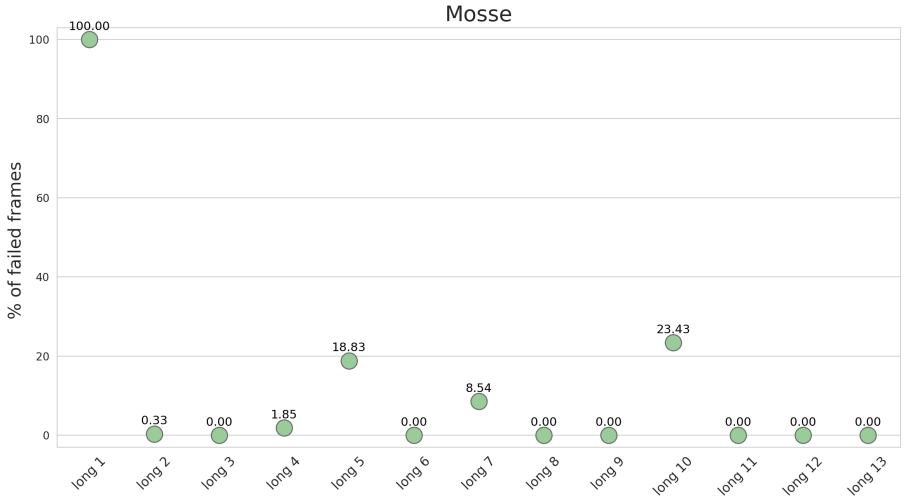

Figure 3.4. Scatter plot of the Mosse algorithm showing the percentage of failed frames for each video.

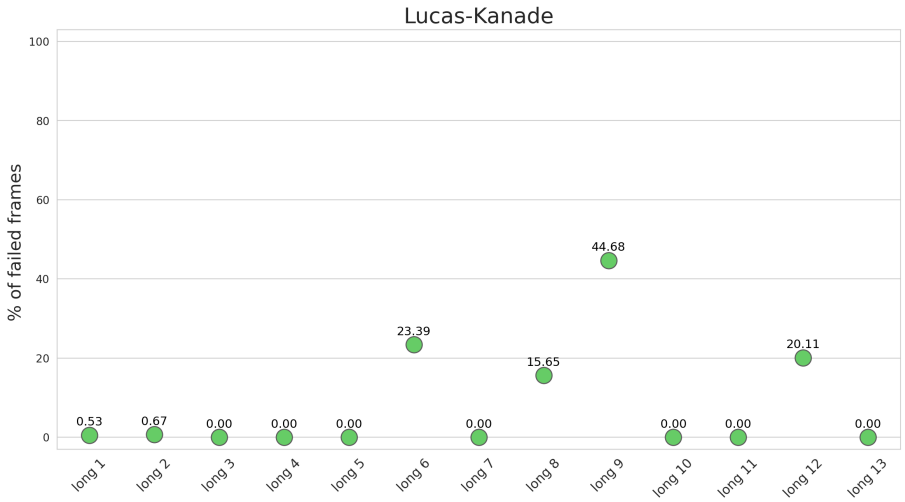

Figure 3.5. Scatter plot of the Lucas-Kanade algorithm showing the percentage of failed frames for each video.

From the reference figures it is clear that:

- The Median Flow algorithm (Figure 3.3) shows a high percentage (greater than 90%) in video `long4.mp4` and a lower percentage for the video `long12.mp4`, it is about 9%.

- The MOSSE algorithm (Figure 3.4) is unable to detect the IVC in any frame of the `long1.mp4` video. The most problematic video is that mentioned above, along with `long5.mp4`, `long7.mp4` and `long10.mp4`, where the detection rate is 18,83%, 8.54% and 23.43% respectively.

- The Lucas-Kanade tracker (Figure 3.5) has zero detection rates only for a limited number of videos, specifically `long3.mp4`, `long4.mp4`, `long5.mp4`, `long7.mp4`, `long10.mp4`, `long11.mp4` and `long13.mp4`. The `long9.mp4` video exhibits the worst tracking performance with this algorithm.

The overlap between the ROI identified by the tracking algorithm in each frame of the video and the corresponding ROI defined as ground truth is calculated using IoU. The distribution of values of all the frames is shown through box plot in Figure 3.6.

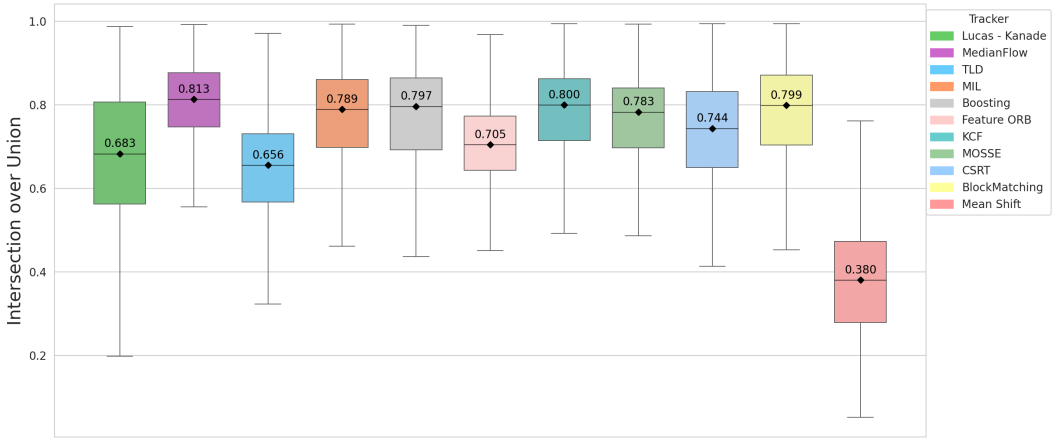

Figure 3.6. Box plot showing the distributions of the IoU values for the different algorithms used for tracking the IVC.

The IoU has values between 0 and 1; values close to the maximum value are preferred to identify which algorithm most accurately selects the area containing the IVC. The standard deviation, on the other hand, measures the variability of the overlap in the frames of the video; a low value is preferred as it indicates greater stability, thus avoiding sudden errors.

For each tracker, the IoU is calculated for all frames of the video. Through the box plot, the median and inter-quartile range (IQR) are calculated for each algorithm. The IQR represents the dispersion of data around the median. The goal is to identify the algorithm that exhibits a high IoU value with narrow IQR and a low standard deviation, ensuring both accuracy and stability in IVC tracking.

The IQR values are shown in Table 3.1.

Algorithm	**IQR**
Lucas - Kanade	0.244
KCF	0.149
MIL	0.162
TLD	0.164
MOSSE	0.143
CSRT	0.182
Mean Shift	0.195
Block Matching	0.167
Boosting	0.173
Median Flow	0.13
ORB	0.129

Table 3.1. IQR of IoU of each algorithm

The bar graph with errors (Figures 3.7) shows the average IoU value for each algorithm calculated over the entire dataset and the std value that measures the dispersion of IoU values around the mean.

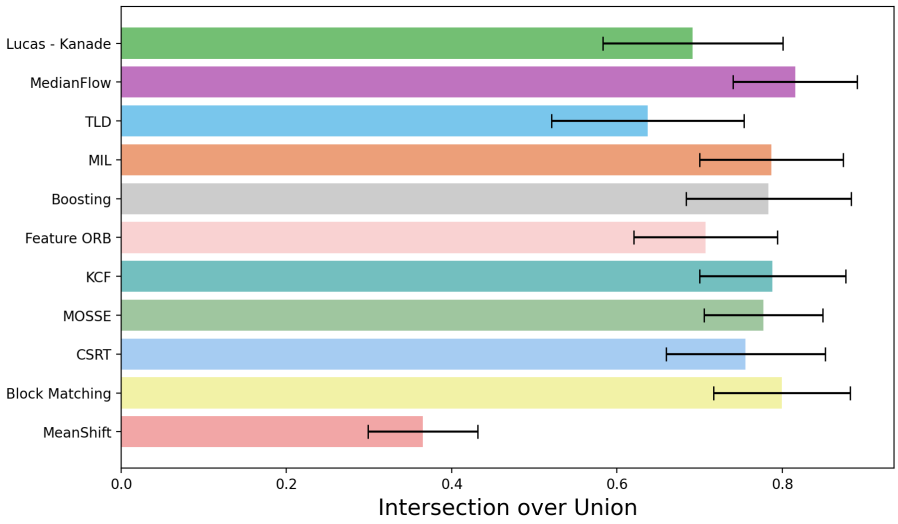

Figure 3.7. Bar plot with errors showing the dispersion of IoU values around the mean for the different algorithms.

It can be observed that Median Flow, KCF and Block Matching are the algorithms with the highest median IoU values: 0.813, 0.8 and 0.799 respectively. MIL, Boosting, and MOSSE have a slightly lower median IoU of about 0.79. As shown in Table 3.1, KCF, MOSSE, and Median Flow have narrow IQRs (between 0.13 and 0.149), indicating that their performance is fairly consistent across video frames. These algorithms, along with Block Matching, also show a low standard deviation, with values ranging from 0.075 to 0.088.

In conclusion, the best trackers for this selection method are:

- Block Matching: median IoU = 0.799, IQR = 0.167, std = 0.082
- Median Flow: median IoU = 0.813, IQR = 0.13, std = 0.075
- KCF: median IoU = 0.8, IQR = 0.149, std = 0.088

The least performing tracker is Mean Shift, whose median IoU is 0.38. The standard deviation and IQR, respectively 0.066 and 0.195, show that the tracker is stable in making errors, indicating its incompatibility with the thesis objectives.

The percentage of false positives is calculated based on the IoU value. For each frame of the video, the IoU was calculated, using a threshold of 0.6 to define the number of

frames in which the tracker cannot correctly identify the location of the ROI. For each algorithm, considering all frames of the videos, mean valor and mean standard deviation are calculated. The Figure 3.8 illustrates the results obtained, it shows only the mean values, as the standard deviations are very low (on the order of 0.1%).
The best tracker will be the one with a combination of low average value and low standard deviation. In fact, a low percentage of false positives indicates that the tracker can identify the position of IVC in most frames, and a low standard deviation value reflects high tracking stability over time, suggesting the absence of random errors.

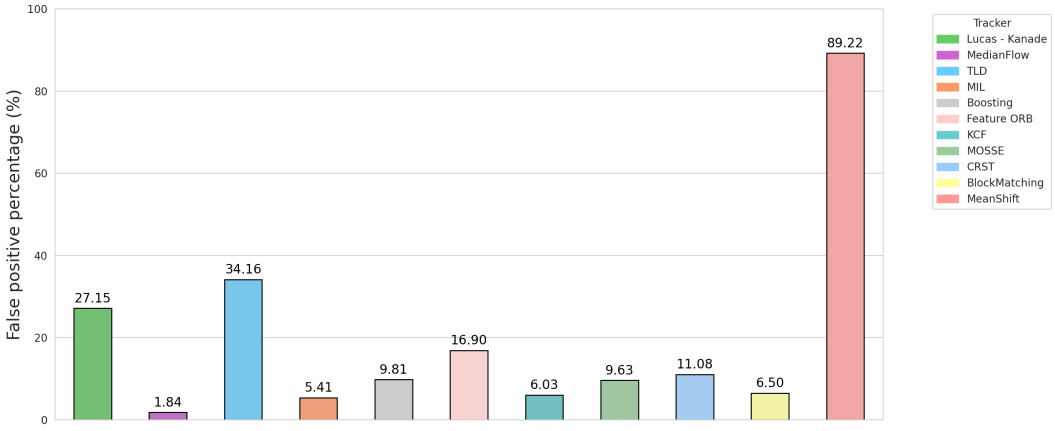

Figure 3.8. Bar plot of mean values of FPP values.

The results show that the Mean Shift tracker performs the worst, with an FPP of 89.22%, indicating poor IVC detection. The best tracker, on the other hand, is Median Flow which records a false positive rate of 1.84%.
Other algorithms with good performances in detecting the IVC across video frames include KCF, MIL and Block Matching (FPP between 5.41% and 6.5%).

The Euclidean distance between the centers of the ROIs drawn by each algorithm and the center of the ROI ground truth is calculated to evaluate the accuracy with which the algorithm follows the center of the ROI between frames. The distance is measured in cm using the conversion factor cm/pixels. This calculation is done for each frame of the video; the distribution of values for each algorithm is shown in Figure 3.9.

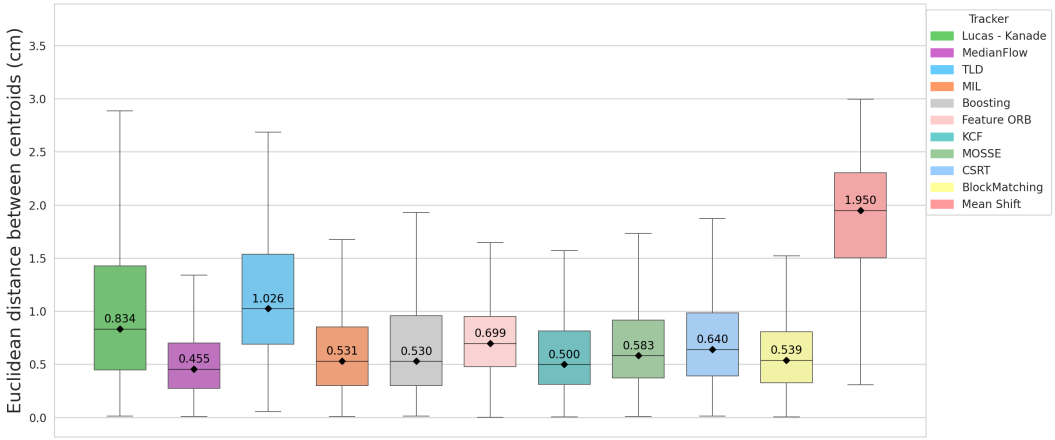

Figure 3.9. Box plot showing the distributions of Euclidean distance values for the different algorithms used for tracking the IVC.

IQRs are calculated and described in Table 3.2. The standard deviation shows the distribution of Euclidean distance values with respect to the mean value, the reference Figure is 3.10.

Algorithm	**IQR**
Lucas - Kanade	0.98
KCF	0.505
MIL	0.55
TLD	0.848
MOSSE	0.544
CSRT	0.595
Mean Shift	0.799
Block Matching	0.479
Boosting	0.658
Median Flow	0.43
ORB	0.466

Table 3.2. IQR of Euclidean distance between centers of ROI for each algorithm

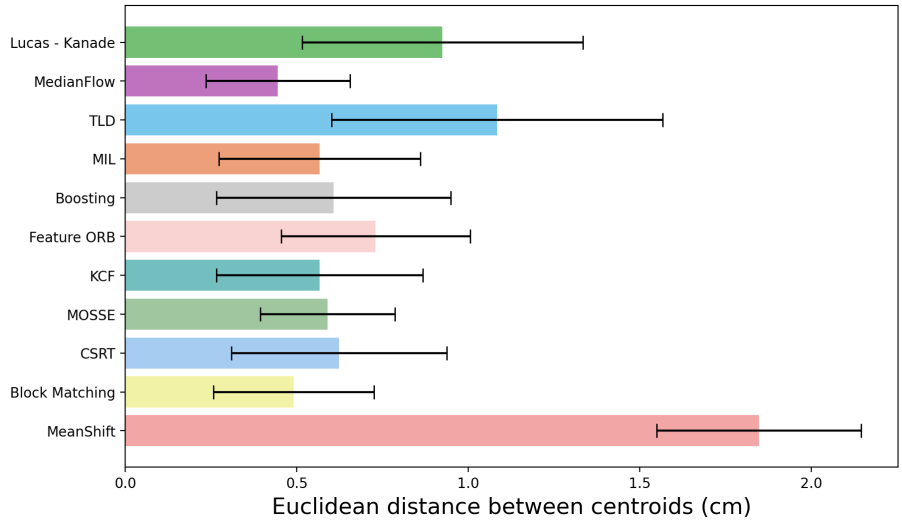

Figure 3.10. Bar plot with errors showing the dispersion of Euclidean distance between centroids around the mean for the different algorithms.

The Euclidean distance is used to quantify the spatial error between the IVC position obtained by the tracker and its actual reference position. The goal is to prioritize lower values of Euclidean distance, IQR, and standard deviation to achieve higher tracking accuracy and avoid fluctuations or unexpected errors.

The references figures show that Median Flow, Block Matching, MIL, and KCF present lower values for median Euclidean distance. In particular, Median Flow records a median of 0.455 cm with an IQR of 0.43 cm. Block Matching has a median of 0.539 cm and an IQR equal to 0.479 cm. KCF presents a median of 0.5 cm and an IQR of 0.505 cm and finally, MIL has a median distance equal to 0.531 cm with a IQR of 0.55 cm.
Considering the distribution of the Euclidean distance values with respect to the mean value, the algorithm that is characterized by the lowest std is MOSSE (std = 0.196 cm). It is followed by Median Flow (std = 0.21 cm), Block Matching (std = 0.233), MIL and KCF (std about 0.30 cm).

In conclusion, considering the values above, the best algorithms are KCF, Block Matching, Median Flow, and MIL, as they demonstrate more stable and precise performance compared to the other algorithms:

- KCF: median distance = 0.5 cm, IQR = 0.505 cm, std = 0.30 cm.
- Block Matching: median distance = 0.539 cm, IQR = 0.479 cm, std = 0.233 cm.
- Median Flow: median distance = 0.455 cm, IQR = 0.43 cm, std = 0.21 cm.
- MIL: median distance = 0.531 cm, IQR = 0.55 cm, std = 0.294 cm.

Mean Shift is the tracker that has the greatest difficulty following the center of the ROI, as evidenced by a median of 1.95 cm and IQR of 0.799 cm, indicating constant errors.

Euclidean distances between ROI centers were calculated between consecutive frame pairs, both for tracker-detected and ground-truth ROIs. Comparison of corresponding distances within the same video allows evaluating the tracker's ability to correctly follow the object over time. Differences greater than a threshold of 0.5 cm indicate tracker instability. These distances have been defined as unreliable. The bar plot describing the mean value of the percentage values of the unreliable distances are shown in Figure 3.11.

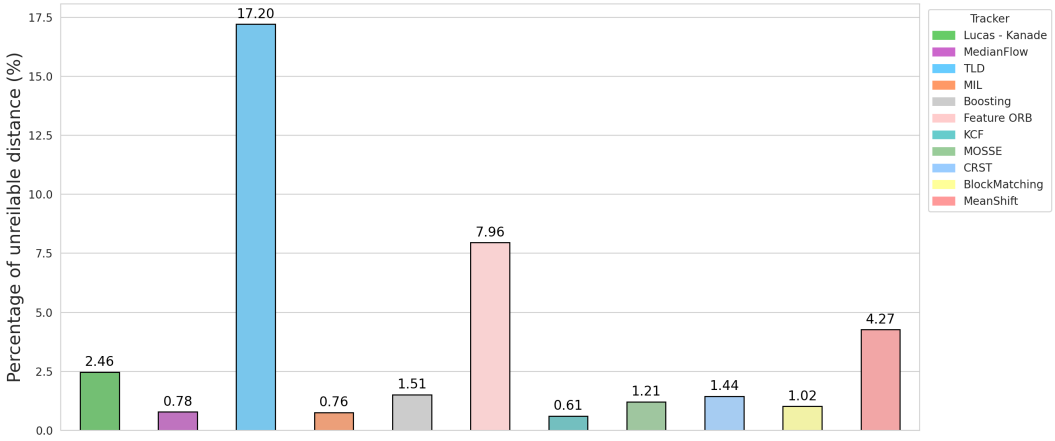

Figure 3.11. Bar plot showing the mean value of unreliable distance for the different algorithms used for tracking the IVC.

The percentage of unreliable distances shows that all algorithms are able to track the object of interest across different frames of the video, as indicated by the values in the Figure above. The best algorithm is KCF, while TLD proves to be the worst in this regard.

The objective of this thesis project is to identify the best algorithm for tracking the inferior vena cava in ultrasound video. Tracking the vein is essential to follow the overall movement of the vessel and therefore to monitor the parameters that characterize it.
The results show that algorithms such as Mean Shift, Lucas-Kanade, Median Flow, and MOSSE have proven to be the fastest, while MIL is the slowest. It is crucial for the selected tracker to process a high number of frames per second, ensuring smooth tracking, especially in dynamic environments. The number of frames in the video affects the tracker's speed, as a higher frame count results in longer processing times. Considering the different videos analyzed for each tracker, it can be observed that the speed performance of the algorithms remains consistent.

In terms of precision the best-performing trackers are Median Flow, KCF and Block

Matching, which stood out for their high IoU values and low Euclidean distance values between the ROI center obtained by the algorithm and that of the ground truth. These algorithms are capable of accurately tracking the ROI associated with the IVC and following the center in the video frames. Although MIL is accurate in detecting the presence of the IVC, it is slow and therefore not suitable for this purpose.
The algorithms that are less suitable for tracking the IVC in ultrasound videos are Mean Shift and Feature ORB. Both are characterized by low accuracy, as evidenced by low IoU values and a high Euclidean distance between the centers of the tracked ROI and the ground truth ROI. Additionally, the low standard deviation indicates systematic errors.

To consistently monitor the vena cava in videos, the tracker must be able to identify the IVC in all frames. Based on the calculation of the percentage of failed frames, Median Flow, MOSSE, and Lucas-Kanade encounter difficulties in tracking the IVC in some videos. The error occurs in some or all frames. Tracking blood vessels is challenging for certain algorithms due to the low contrast between surrounding tissues and, in this case, the IVC. The trackers described above use optical flow models, which may not perform well when the movement of the IVC is barely visible. Additionally, ultrasound images are subject to artifacts that can further complicate tracking. Some algorithms, such as MOSSE, are more effective at tracking objects with a stable shape and may struggle to follow the dynamic movement of the IVC.

The false positive rate indicates how much the algorithm fails to correctly detect the IVC position in the frames. The rate is calculated by imposing a threshold of 0.6 on the IoU, frames that detect a lower value contribute to the calculation of this metric. The Mean Shift tracker showed the worst FPP, failing to maintain stable tracking of the IVC in the frames. On the contrary, Block Matching, Median Flow and KCF stood out as the best, with low false positive rates.

For each pair of successive frames, the Euclidean distance between the centers of the tracked ROIs was compared with the corresponding distance in the ground-truth ROIs. If the difference exceeds 0.5 cm, it indicates a poor ability of the tracker to follow the object movement in the video. Among all the analyzed algorithms, TLD shows the worst performance. The calculation of the distance between the centers of the ground-truth ROIs and those obtained by the tracker allows to evaluate the overall accuracy of the tracking in each frame, measuring the precision with which the algorithm identifies the position of the IVC. Instead, the comparison of the distances between the centers of the ROIs in successive frames provides an indication of the ability of the tracker to follow the object over time, evaluating the consistency of the tracking along the video.

Chapter 4

Conclusion

The aim of this thesis project is to perform tracking of the inferior vena cava in ultrasound videos to analyze vessel movements induced by the respiratory cycle and cardiac activity. The analysis is conducted on a dataset consisting of 13 videos, characterized by different lengths and resolutions. The OpenCV library is the starting point for the selection of tracking algorithms: specifically, 11 trackers are employed to track the blood vessel. These algorithms, commonly used in different fields, belong to different categories, including optical flow, adaptive classifiers, correlation filters, block matching, feature-based tracking, and brightness histogram analysis. The location of the ROI ground truth in the first frame of the video is the start point for tracking. Precision, accuracy, and speed are evaluated for each algorithm using different metrics.
The results show that Mean Shift is not suitable for the use case because the identified ROI is often out of position with the ground truth and generates a very high number of false positives. The tracker, despite being one of the fastest in terms of FPS, tends to frequently lose the object during the video. Mean Shift, therefore, cannot correctly detect the position of the IVC and track its movements.
Analyzing the different metrics, the algorithms with the best performance are Median Flow, KCF, and Block Matching. However, Median Flow proves to be unreliable, as it fails to track the vessel for most frames in some videos. Block Matching offers slight superiority in precision and accuracy over KCF, but the latter makes fewer errors in ROI identification and is more stable.
The choice of the most suitable tracking algorithm for tracking the IVC is made through a statistical test on the calculated metrics. Significant differences emerge in performance, such as the FPS and the percentage of unreliable distances. The FPS of the methods considered allows for real-time tracking, but the KCF algorithm shows a lower percentage of unreliable distances, indicating greater stability in detecting the IVC in video frames. The goal is to identify a robust algorithm that can perform correctly on all videos without failure, consequently KCF is the best choice for tracking the inferior vena cava due to its balance of accuracy, stability, and reliability.

Bibliography

[1] J. E. Aldrich. «Basic physics of ultrasound imaging». In: *Critical Care Medicine* 35.5 Suppl (May 2007), S131–S137. DOI: 10.1097/01.CCM.0000260624.99430.22.

[2] F. M. Abu-Zidan, A. F. Hefny, and P. Corr. «Clinical ultrasound physics». In: *Journal of Emerg Trauma Shock* 4.4 (Oct. 2011), pp. 501–503. DOI: 10.4103/0974-2700.86646.

[3] Kevin Martin. «Introduction to B-mode imaging». In: *Diagnostic Ultrasound: Physics and Equipment*. Ed. by Peter R. Hoskins, Kevin Martin, and Abigail Thrush. Cambridge University Press, 2010, pp. 1–3.

[4] J. A. Jensen. «Medical ultrasound imaging». In: *Progress in Biophysics and Molecular Biology* 93.1-3 (Apr. 2007). Epub 2006 Aug 15, pp. 153–165. DOI: 10.1016/j.pbiomolbio.2006.07.025.

[5] John S. Mattoon and Thomas G. Nyland. *Principi fondamentali di ecografia diagnostica*. CAPITOLO 1: Principi fondamentali di ecografia diagnostica. Milano: Elsevier, 2011. Chap. 1.

[6] H.F. Routh. «Doppler ultrasound». In: *IEEE Engineering in Medicine and Biology Magazine* 15.6 (1996), pp. 31–40. DOI: 10.1109/51.544510.

[7] W. D. Foley and S. J. Erickson. «Color Doppler flow imaging». In: *AJR Am J Roentgenol* 156.1 (Jan. 1991), pp. 3–13. DOI: 10.2214/ajr.156.1.1898567.

[8] R. S. Moorthy. «Doppler ultrasound». In: *Med J Armed Forces India* 58.1 (Jan. 2002), pp. 1–2. DOI: 10.1016/S0377-1237(02)80001-6.

[9] S. Jarvis and S. Saman. «Cardiac system 1: anatomy and physiology». In: *Nursing Times [online]* 114.2 (2018). [Online], pp. 34–37.

[10] National Cancer Institute Surveillance Epidemiology and End Results (SEER) Program. *Cardiovascular System Anatomy*. https://training.seer.cancer.gov/anatomy/cardiovascular/. Accessed January 22, 2025.

[11] SimpleMed. *Anatomy of the Cardiovascular System*. Accessed: 2025-01-22. 2025. URL: https://simplemed.co.uk/subjects/cardiovascular/anatomy-of-the-cardiovascular-system.

[12] Lumen Learning. *Structure and Function of Blood Vessels*. https://courses.lumenlearning.com/suny-ap2/chapter/structure-and-function-of-blood-vessels/. Accessed January 22, 2025.

[13] R. Chaudhry, J.H. Miao, and A. Rehman. *Physiology, Cardiovascular*. Updated 2022 Oct 16. StatPearls [Internet]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK493197/. Treasure Island (FL): StatPearls Publishing, 2025.

[14] P. Policastro and L. Mesin. «Processing Ultrasound Scans of the Inferior Vena Cava: Techniques and Applications». In: *Bioengineering* 10.9 (2023), p. 1076. DOI: 10.3390/bioengineering10091076.

[15] Humanitas. *Vena Cava Inferiore*. Accessed: 2024-11-22. 2024.

[16] Luca Mesin et al. «Assessment of Phasic Changes of Vascular Size by Automated Edge Tracking-State of the Art and Clinical Perspectives». In: *Frontiers in Cardiovascular Medicine* 8 (Jan. 2022), p. 775635. DOI: 10.3389/fcvm.2021.775635.

[17] D. Džigal N. Dardagan A. Brđanin and A. Akagic. «Multiple Object Trackers in OpenCV: A Benchmark». In: *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. 2021, pp. 1–6. DOI: 10.1109/ISIE45552.2021.9576367.

[18] Olfa Haggu, Agninoube Tchalim, and Baptiste Magnier. «A Comparison of OpenCV Algorithms for Human Tracking with a Moving Perspective Camera». In: *2021 European Conference on Visual Information Processing (EUVIP)* (2021). DOI: 10.1109/EUVIP50544.2021.9483957.

[19] Simon Baker and Iain Matthews. «Lucas-Kanade 20 Years On: A Unifying Framework». In: *International Journal of Computer Vision* 56.3 (2004), pp. 221–255. DOI: 10.1023/B:VISI.0000011205.11775.fd.

[20] OpenCV. *Lucas-Kanade Optical Flow - OpenCV Documentation*. https://docs.opencv.org/4.x/db/d7f/tutorial_js_lucas_kanade.html. Accessed: November 30, 2024. 2024.

[21] Zhang Nana and Zhang Jin. «Optimization of Face Tracking Based on KCF and Camshift». In: *Procedia Computer Science* 131 (2018), pp. 158–166. URL: https://api.semanticscholar.org/CorpusID:67214540.

[22] OpenCV. *cv::TrackerKCF Class Reference*. https://docs.opencv.org/3.4/d2/dff/classcv_1_1TrackerKCF.html. Accessed: 2024-11-30. 2015.

[23] B. Babenko, M.-H. Yang, and S. Belongie. «Visual Tracking with Online Multiple Instance Learning». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL, USA, 2009, pp. 983–990. DOI: 10.1109/CVPR.2009.5206737.

[24] OpenCV. *cv::TrackerMIL Class Reference*. https://docs.opencv.org/4.x/d0/d26/classcv_1_1TrackerMIL.html. Accessed: 2024-12-07. 2024.

[25] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. «Tracking-Learning-Detection». In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* (2012).

[26] Peter Janků et al. «Comparison of tracking algorithms implemented in OpenCV». In: *20th International Conference on Circuits, Systems, Communications and Computers*. 2016. URL: https://api.semanticscholar.org/CorpusID:62907943.

[27] H. Moridvaisi, F. Razzazi, M. Pourmina, et al. «An extended TLD tracking algorithm using co-training learning for low frame rate videos». In: *Multimedia Tools and Applications* 82 (2023), pp. 24743–24769. DOI: 10.1007/s11042-022-14106-1. URL: https://doi.org/10.1007/s11042-022-14106-1.

[28] OpenCV. *cv::TrackerTLD Class Reference*. https://docs.opencv.org/3.4/dc/d1c/classcv_1_1TrackerTLD.html. Accessed: 2024-12-04. 2023.

[29] K. Han. «Image object tracking based on temporal context and MOSSE». In: *Cluster Computing* 20 (2017), pp. 1259–1269. DOI: 10.1007/s10586-017-0800-0.

[30] OpenCV. *cv::TrackerMOSSE Class Reference*. https://docs.opencv.org/3.4/d0/d02/classcv_1_1TrackerMOSSE.html. Accessed: 2024-12-04. 2023.

[31] A. A. Levin, D. D. Klimov, A. A. Nechunaev, et al. «Assessment of experimental OpenCV tracking algorithms for ultrasound videos». In: *Scientific Reports* 13 (2023), p. 6765. DOI: 10.1038/s41598-023-30930-3. URL: https://doi.org/10.1038/s41598-023-30930-3.

[32] OpenCV. *cv::TrackerCSRT Class Reference*. https://docs.opencv.org/3.4/d0/d02/classcv_1_1TrackerCSRT.html. Accessed: 2024-12-04. 2023.

[33] A Gyaourova, C Kamath, and S Cheung. «Block Matching for Object Tracking». In: (Oct. 2003). DOI: 10.2172/15009731. URL: https://www.osti.gov/biblio/15009731.

[34] Xiang Xiang, Wenhui Chen, and Du Zeng. «Intelligent Target Tracking and Shooting System with Mean Shift». In: *2008 IEEE International Symposium on Parallel and Distributed Processing with Applications*. 2008, pp. 417–421. DOI: 10.1109/ISPA.2008.167.

[35] OpenCV. *MeanShift Tutorial*. https://docs.opencv.org/3.4/d7/d00/tutorial_meanshift.html. Accessed: 2024-12-05. 2017.

[36] Helmut Grabner, Michael Grabner, and Horst Bischof. «Real-Time Tracking via On-line Boosting». In: vol. 1. Jan. 2006, pp. 47–56. DOI: 10.5244/C.20.6.

[37] OpenCV. *cv::TrackerBOOSTING Class Reference*. https://docs.opencv.org/3.4/d1/d1a/classcv_1_1TrackerBoosting.html. Accessed: 2024-12-12. 2023.

[38] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. «Forward-Backward Error: Automatic Detection of Tracking Failures». In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 2756–2759. DOI: 10.1109/ICPR.2010.675.

[39] A. Varfolomieiev and O. Lysenko. «An improved algorithm of median flow for visual object tracking and its implementation on ARM platform». In: *J Real-Time Image Proc* 11 (2016), pp. 527–534. DOI: 10.1007/s11554-013-0354-1.

[40] OpenCV. *cv::TrackerMedianFlow Class Reference*. https://docs.opencv.org/3.4/d7/d86/classcv_1_1TrackerMedianFlow.html. Accessed: 2024-12-12. 2023.

[41] Meng Fanqing and You Fucheng. «A tracking algorithm based on ORB». In: *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*. 2013, pp. 1187–1190. DOI: 10.1109/MEC.2013.6885245.

[42] Shuang Wu et al. «Object tracking based on ORB and temporal-spacial constraint». In: *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*. 2012, pp. 597–600. DOI: 10.1109/ICACI.2012.6463235.

[43] OpenCV. *cv::TrackerORB Class Reference*. https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html. Accessed: 2025-02-09. 2023.