DELFT UNIVERSITY OF TECHNOLOGY

BIOMEDICAL ENGINEERING

M.SC. THESIS

Video-based Event Detection in Catheterization Laboratory

Author : Yiheng Chang

TU Delft Supervisor : Prof. John van den Dobbelsteen
Prof. Justin Dauwels

Philips Supervisor : Chavdar Bachvarov

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Biomedical Engineering

June 29, 2023

Acknowledgements

This thesis is a sign of the completion of many things, my graduation project, my internship at Philips and my two years of Biomedical Engineering master studies at Delft University of Technology. These past nine months have been a truly special journey for me.

I am deeply grateful to my supervisors at TU Delft, Prof. John van den Dobbelsteen and Prof. Justin Dauwels, who provided me with valuable assistance, constructive suggestions, and stimulating discussions that greatly enriched the research. Their contributions and collaboration were essential in advancing my understanding and improving the quality of this thesis. I would like to express my heartfelt gratitude to my Philips supervisor, Chavdar Bachvarov, for all his unwavering guidance, invaluable expertise, and continuous support throughout the entire process of my internship at Philips. I would also like to thank all my Philips colleagues and fellow interns for their help and support. Finally, I would like to express my love and gratitude to my family and my girlfriend, Jiayu Zhang, who supported me morally every day during my master studies.

Yiheng Chang
Tilburg, The Netherlands
June 29, 2023

Abstract

Catheterization Laboratory (Cath Lab) is a hospital examination room equipped with diagnostic imaging equipment to display the heart arteries and to detect and treat any abnormality or stenosis. This thesis addresses the urgent need to improve the efficiency of operating rooms in Cath Labs due to the increasing number of patients with cardiovascular disease. Currently, measuring key performance indicators (KPI) for efficiency in Cath Labs requires manual observation and registration of patient entry and exit times, as well as the establishment and closure of access points. This process is time-consuming, costly, and prone to errors. To overcome these challenges, we conducted a study at the Reinier de Graaf Groep Hospital in Delft, where we recorded activities in the Cath Lab and developed a video-based event detection framework to automate the identification of these crucial events. The proposed framework consist of a fine-tuned YOLO v8 object detector, a popular object tracker ByteTrack, a post processing step and different event detection algorithms for different events of interest. This thesis presents the related video-based event detection framework and a detailed introduction for the development of object detector YOLO and object tracking technologies. The materials used in the project and how they work are clearly described. Events of interest are determined as needed for KPI calculations and are defined by describing the workflow in the target Cath lab. The steps taken to prepare the event dataset are also described in detail. The process of training, the connection between the tracker and the detector and the algorithms to detect different events for reproduction are explained. A series of performance tests on the target detector and tracker concluded that the tracker can handle the occlusion to some extent, but the performance of the object detector determined the performance of the tracker. It was also found that the simpler network structure in YOLOv8 outperformed the more complex network structure in detecting smaller objects on our dataset. In addition, the event detection algorithm is tested on multiple videos to evaluate its robustness and result showed that the algorithm can correctly and accurately detect the target events. The practical applicability of the framework and some future work that could enhance its performance are discussed. In conclusion, this thesis presents a video-based event detection framework that can detect events of interest in the Catheterization Lab at Delft Reinier de Graaf Groep Hospital.

Contents

Chapter 1

Introduction

1.1 Background

The Catheterization Labs are hospital examination rooms equipped with diagnostic imaging equipment to visualize the heart arteries and diagnose and treat any abnormalities or stenosis. Currently, the increasing number of cardiac patients requiring care is causing the laboratory to be overloaded. Hospitals struggle to provide patient-centered high-quality care while facing financial constraints. As a result, time and costs need to be minimized to achieve optimal efficiency. A series of metrics have been established by researchers and community to measure the efficiency of Cath Labs and the methods to improve the efficiency. As shown in the Figure 1.1, Reed et al. [1] proposed different key performance indicators (KPI) to assess efficiency along with the corresponding goals. Some of these focus on overall room operations, such as room utilization, case volume, on-time start rate, percentage of days at full capacity, turnaround time and percentage of sheath pulls in the room. Other indicators focus on the efficiency of workforce which includes productivity per full-time employee, hours of overtime and percentage of after-hours cases. Most of these indicators require to know the events that occur in the lab and the exact time of their occurrence. For example, turnaround time requires knowing when the previous patient left and when the next patient will arrive in the lab. In addition, the event-based workflow analysis in healthcare is the study of analyzing the tasks and processes that make up a particular clinical workflow, with the goal of identifying inefficient processes and improving the quality of care [2]. Vankipuram et al. [3] stated that monitoring and evaluation of workflow in the clinical setting can improve the effectiveness of patient management. For example, if we know how long a particular type of procedure will take, we can schedule it better in advance. If we know there is a delay in a procedure, we can prepare another room for the next patient.

Therefore, measurement of efficiency and event-based workflow analysis in the Cath Labs are essential to optimize resource allocation and enhance overall patient care. Currently, observing events in Cath Lab is done manually by nurses stationed outside the lab and documented in procedure files. This approach not only imposes a signif-

icant administrative burden on the hospital, but may also lead to errors due to staff negligence and different subjectivity of staff. To reduce the administrative burden and provide accurate data, Philips proposed a potential solution of installing five cameras at various angles to record activities in the Cath Lab. We hope to use computer vision and machine learning techniques to automatically detect the events of interest and obtain corresponding temporal data. We collaborated with the Reinier de Graaf Groep Hospital in Delft, which has a Philips Cath Lab equipped with a C-arm X-ray machine for rotational angiography. The obtained 3D angiographic images are displayed in real time on a monitor, providing cardiologists with additional information to diagnose cardiovascular disease. After 16 months of data collection, radiographic and video data were collected on more than 300 interventions, most of which were diagnostic coronary angiography. By performing event detection on the video and X-ray images collected in the target Cath Lab, data from the workflow shown in the Figure 3.1 can be obtained and used for analysis. Hence, this thesis focus on automatic event detection with high accuracy from collected video data and proposes an event detection framework consisting of a fine-tuned YOLO v8 object detector connected to an object tracker ByteTrack, a post-processing method and different event detection algorithms to detect different events of interest. The following subsection will present the related research on event detection in videos.

	Definition	Example of Change	Goal
Case volume	Number of cases in a period of time.	731 ± 50 cases/month No significant change	Stable or increased
Room utilization*	Ratio of number of hours each lab room is staffed to the number of hours the lab room is utilized.	91% to 106%	≥100%
% Days at full capacity	Proportion of days in which the entire lab is at full operating capacity.	8% to 77%	≥75%
On-time start	On-time patient and physician arrival for the first case of the day in a given lab room.	62% to 89%	≥85%
Turn-around time	Duration of time between exit of the prior patient and arrival of the next patient to the cath lab.	21.6 to 16.4 min	≤17 min
% Sheath pull in room	Proportion of nonradial cases in which the vascular access sheath is pulled in the lab room.	10% to 3%	≤5%
Productivity per FTE	Ratio of case volume per non-physician FTE.	20.2 to 24.6 cases/month/FTE	Stable or increased
% After-hours cases	Proportion of scheduled cases that occur after normal operating hours (i.e., after 5 PM).	1.7% to 1.8%	Stable or reduced
% Overtime hours	Proportion of hours paid to FTE's classified as overtime.	5.1% to 3.8%	Stable or reduced

Figure 1.1: Metrics of Cath Lab efficiency and proposed goal by Reed et al. [1]

1.2 Related research

The definition of an event in a video varies depending on the complexity of the event. Francois et al. [4] defines an event as a change of state in a multimedia entity, such as a car stopping or moving. Hakeem et al. [5] defines an event as the set of actions performed between agents, such as a person picking up a shopping bag. Jiang et al. [6] focuses on the temporal and semantic connections between the activities carried out throughout the event, for instance an offensive-defensive round in a basketball game. As a result, multiple definitions and detection techniques are

needed depending on the nature of the event. The use of machine learning for video-based event detection is used in a wide range of applications, such as autonomous driving, surveillance, etc.

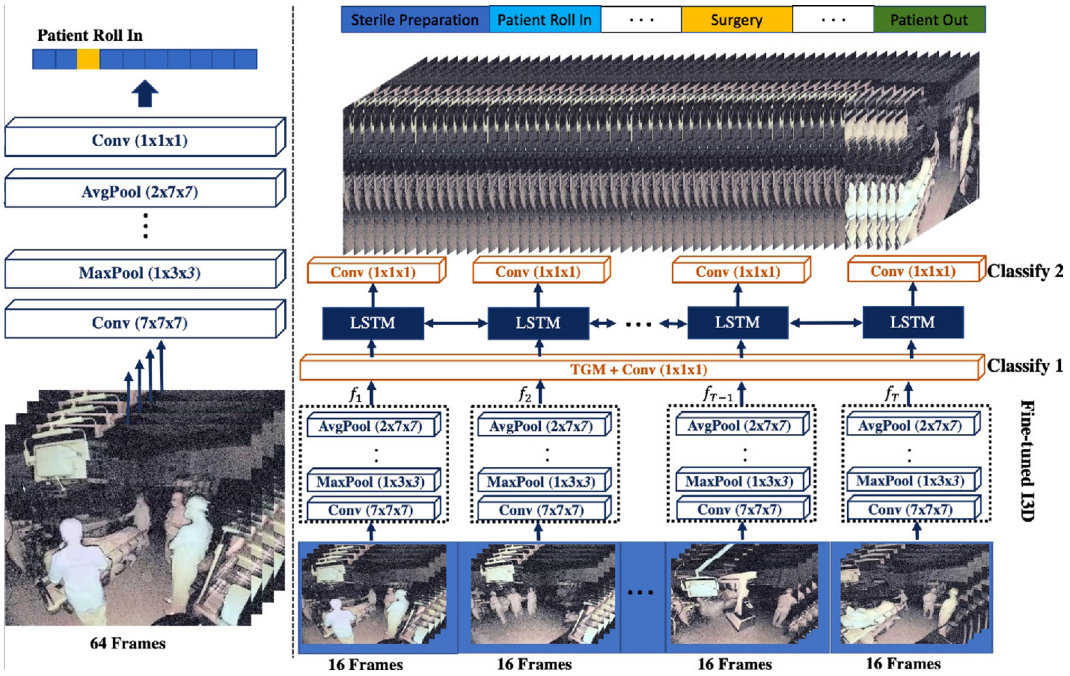

Figure 1.2: A framework for automatic recognition of surgical activities in the robot-assisted operating room proposed by Sharghi et al. [7]

Zhao et al. [8] proposes a unified framework to classify events occurring in surveillance cameras into short-term and long-term events. Short-term events are represented by the static key pose of a person and two convolutional neural networks are used. The pedestrian are first detected and then pose estimation is performed to classify the events. For events that consist of various actions, objects and occur over a long period of time, they use a trajectory-based approach to extract the temporal features of the events [9]. After using Fisher vector to encode the extracted features, the events are classified by a support vector machine. The performance of their framework depends on the ability of the detector to detect each person in a pedestrian, a very challenging task because people obscure each other in the crowd. Therefore, Phan et al. [10] proposed different framework to detect individuals using a combination of detectors and trackers. They utilize a high precision head detector [11] and use a tracker to track the detected head. From the tracking stream, still-images and motion-images of the whole body or upper body are extracted based on the head and they are classified by different pre-trained event classifiers. In addition, there are several studies on automatic recognition of surgical activity in robot-assisted operating rooms. Sharghi et al. [7] presented a large-scale benchmark dataset including 400 full-length multi-angle videos, similar to this project. As shown on the left side of Figure 1.2, they fine-tuned the Inflated 3D ConvNet (I3D) network [12] to analyze the activity occurring in short video clips. The activity extracted from each

clip is mixed by a temporal Gaussian mixing layer, and the order of the workflow is learned by a Long and Short Term Memory network. The network mimics the way humans understand videos. However, the performance of the whole framework can only reach 63.47% which is limited by the performance of the I3D activity extraction module. Therefore, based on these studies and our use case context, a framework consisting of an object detector, a tracker, and an event analyzer is proposed. The following subsections introduce the development of the object detector YOLO and multiple object tracking techniques applied in the proposed framework.

1.3 Object detector YOLO

You look only once (YOLO) is a neural network-based object detection technique that was first introduced by Redmon et al. [13]. It simulates the human visual system and allows you to rapidly identify the objects and their associated positions in the image at a glance. YOLO transforms image pixels directly into bounding box coordinates and class probabilities, redefining object detection as a single regression problem. YOLO is a single-stage object detector, which means the same input image is only processed through the network once. However, common two-stage object detectors, such as Faster R-CNN, Mask R-CNN, and Cascade R-CNN, use input images twice through the network for prediction. Faster R-CNN utilizes a Region Proposal Network (RPN) to generate object proposals, which are then refined by a detection network [14]. It achieves cutting-edge precision but is slower than YOLO. Mask R-CNN is an extension of Faster R-CNN that adds another segmentation branch, enabling the network to construct pixel-level masks of observed objects [15]. It is slower than YOLO but generates more precise segmentation. Cascade R-CNN is a variation of Faster R-CNN that utilizes a cascade of detectors to progressively enhance detection accuracy. It achieves cutting-edge accuracy but is slower than YOLO. YOLO is optimized for speed and precision, making it excellent for real-time applications [16].

YOLO can produce less background error than systems based on region proposal-based and sliding window techniques because YOLO evaluate the full images while formulating predictions. During prediction, YOLO analyzes picture-wide information to determine the bounding boxes of all objects in the image. As shown in the Figure 1.3, YOLO partitions the input image into S x S grids, and if the center of an object falls within one of these grid cells, then that cell is responsible for detecting the object. The cell will generate several bounding boxes and verify the presence of an object. This cell will also provide a number of conditional class probabilities that represent the confidence from the model in the presence of an item and its class. The intersection of the conditional class probability of the object multiplied by the box union between the predicted and true values will give the confidence if the object exists. Thus, the confidence for each box indicates the chance of a certain class within the box and the degree to which the box is predicted to contain the object.

The original YOLO has two flaws: a lower recall rate compares with the region proposal network and incorrect location [17]. Hence, the subsequent version of YOLO

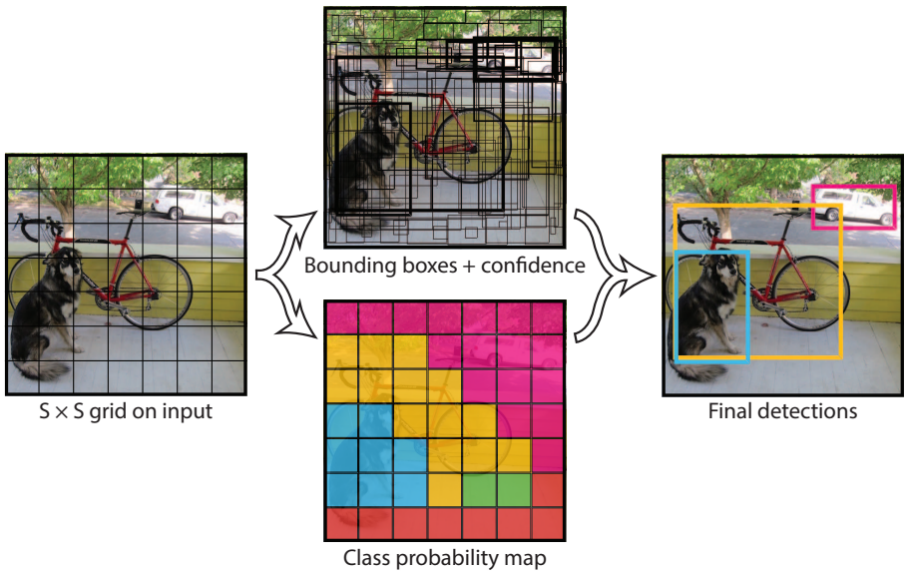

Figure 1.3: The Original YOLO model [13]

attempts to address these two deficiencies. For instance, beginning with YOLO v2, anchor boxes replace the fully connected layers to increase the ability to identify a larger variety of object categories [18]. Anchor boxes are a collection of preconfigured bounding box with various aspect ratios and sizes. When anchor boxes are employed, the class prediction mechanism of spatial position in the original YOLO is replaced with each anchor box predicting both category and object. Hence, based on the displacement of the anticipated box and anchor box, YOLO make optimal prediction. With the update of the YOLO version, the usage and functions of anchor boxes have become more complex and diverse. YOLO v3 incorporates scaled anchor boxes to improve the algorithm's ability to recognize objects of various shapes and sizes [19]. As shown in the Figure 1.4, YOLO v4 uses the k-means clustering approach to construct anchor boxes by grouping the ground truth bounding boxes into clusters and using the cluster centroids as the center of anchor boxes [20].

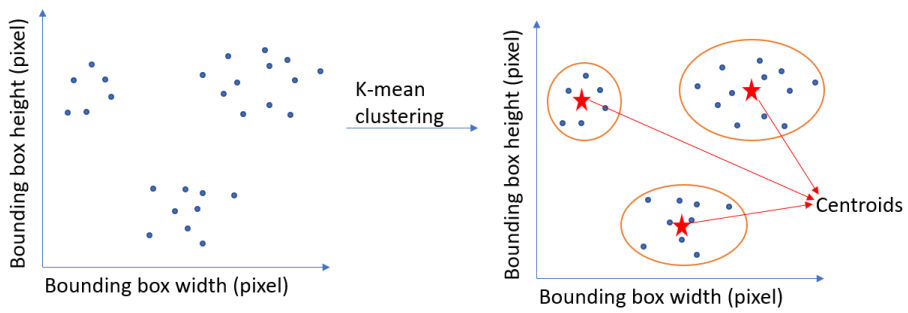

Figure 1.4: K-means clustering approach to construct anchor boxes

YOLO v5 introduces an approach for generating anchor boxes that is similar to YOLO v4 but it can dynamically generate the anchor boxes [21]. Before training, YOLO v5

and YOLO v7 will examine the degree of fit between the predefined anchor boxes and the data. The script will use evolutionary algorithm as shown in the Figure 1.5 to optimize the anchor points. The algorithm will randomly change the original anchor set and recalculate the fitness of the mutated anchor set. If the fit is unsatisfactory, the process is generated again and repeated until the most descriptive anchor set is found for training. By using the best variant anchor set for training, the efficiency of training will increase and the accuracy can be improved [22]. YOLO v8 presents an anchor-free detection head that predicts the bounding box pixel-by-pixel.

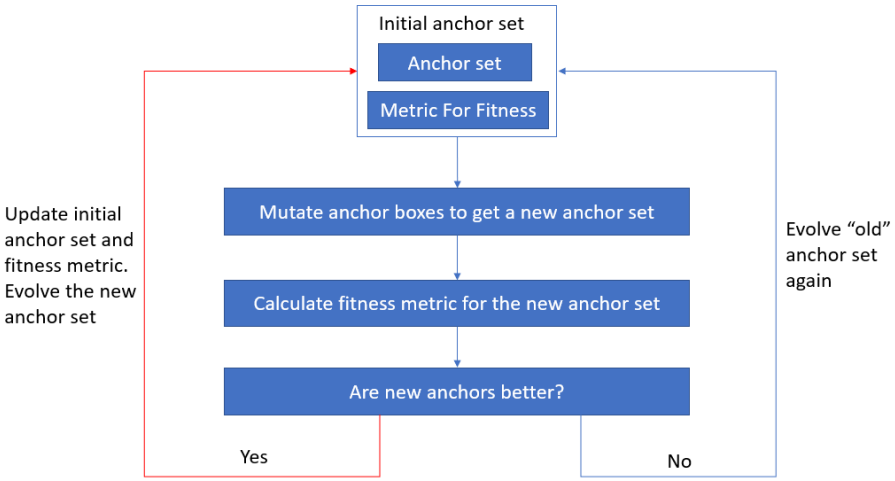

Figure 1.5: Evolution algorithm flow chart for generate dynamic anchor boxes

The model also improved by introducing new loss function into algorithm to address the incorrect location problem. The loss function in YOLO consists of classification part and localization part. The localization loss quantifies the mistake in predicting the bounding boxes of the image's objects, whereas the classification loss quantifies the error in guessing the class probabilities of the objects. The localization loss is computed utilizing the Mean Squared Error (MSE) between the predicted and ground truth bounding box coordinates, whilst the classification loss is computed utilizing the binary cross-entropy loss between the predicted and ground truth class probabilities. The total loss function in YOLO is a weighted sum of the classification and localization losses, with the weights defined by hyperparameters that indicate the relative significance of each loss item. In addition, YOLO v5 incorporates a new loss function known as 'CIoU loss', a version of the IoU loss function aimed to improve the performance of model on unbalanced datasets.

1.4 Multiple object tracking

The primary goal of object tracking in computer vision task is to predict the position of objects throughout a video using spatial and temporal features. Object tracking begins with the collection of detections of the target objects, typically the

coordinates of objects as input. The tracker then allocates unique identifiers to those objects and tracks them through the video frame while maintaining the assigned identifiers. Consequently, object tracking is typically a second-order structure: the first stage typically utilizes an object detector to locate the object that need to be tracked and extract desired features from the object. The second stage is to use a motion predictor to predict the future motion of the object based on its historical data. Object trackers are required for a variety of reasons, the most essential of which is to compensate detection results when object detectors fail. As shown in the Figure 1.6, the object detector may fail to detect objects in video feeds due to occlusions, motion blur, camera viewpoint changes or illumination changes etc. These factors can make it more challenging for object detectors to precisely identify and trace video objects.

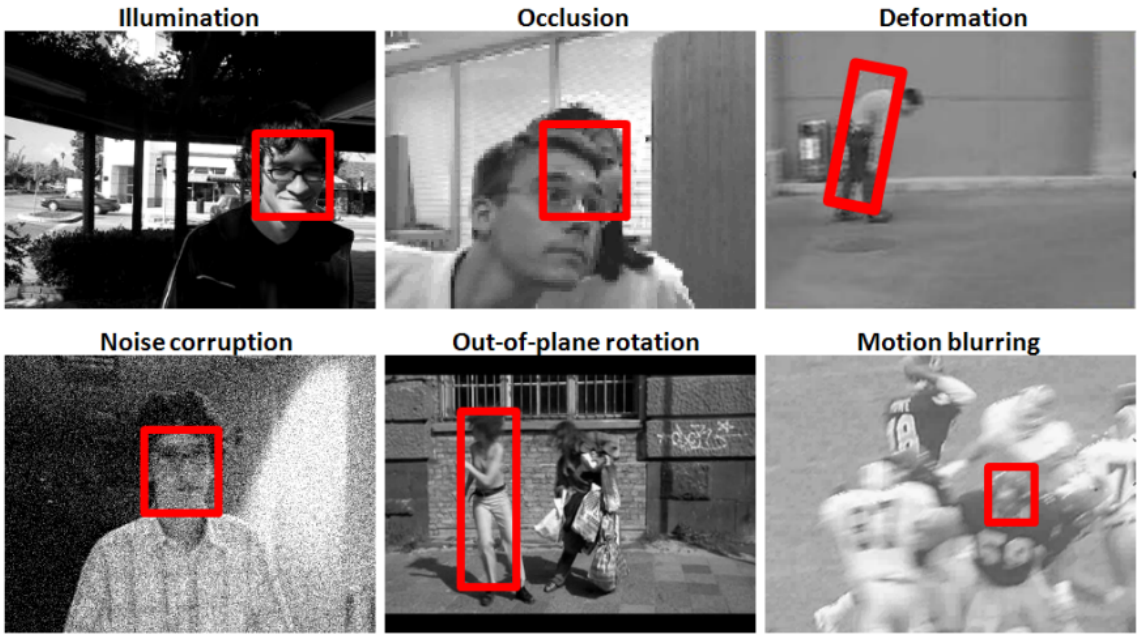

Figure 1.6: Cases where the object detector fails on video input [23]

One of the useful applications of object tracking is re-identification (ReID) which indicates whether the detected objects are the same object as previous object. This is particularly useful in situations where an object reappears after a brief absence which is very challenging for general object detector to recognize that it is the same object. For instance, consider the scenario in which a video depicts an individual wandering around. An object detector may detect presence of a person in the frame, but it may not be able to determine whether it is the same person who reappears into the frame after a few seconds, and this is where object trackers come into play. As mentioned previously, the object tracker allocates a unique ID to the monitored object, and when the object reappears in the frame, the object tracker recognizes that the object has appeared in the video before. Figure 1.7 illustrates the difference between object detectors and object trackers when dealing with identities.

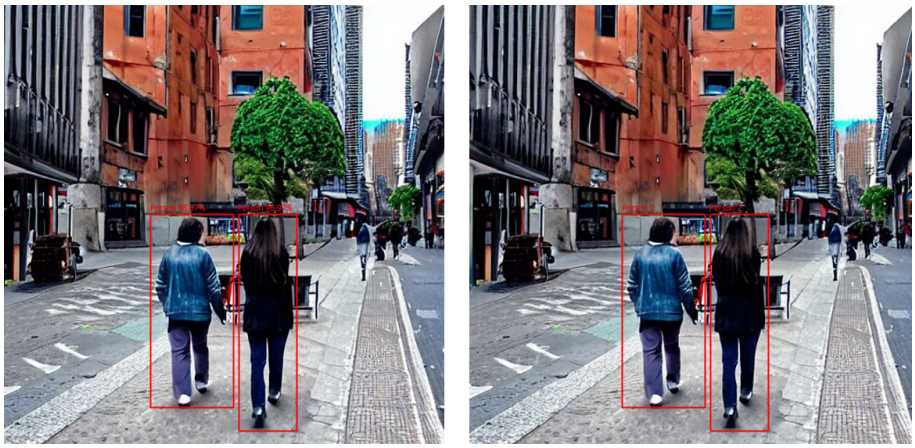

Figure 1.7: The object tracker assigns a unique ID to each object (image generate by stable diffusion) Left: Object detector can detector two people in the image. Right: Object tracker will identify the people by assigning different IDs.

Object trackers outperform general object detectors due to their simple architecture, which enables them to be applied to real-time predictions. Unlike object detectors, which need to analyze the entire video sequence to detect and classify objects, object trackers only need to track the identified object, reducing the computational requirements and enabling faster processing times. Moreover, object trackers can be utilized in numerous applications, such as robotics, autonomous vehicles and surveillance systems. In robotics and autopilot, object trackers can augment the object detection and collision avoidance by tracking the obstacles or the movement of other vehicles in the environment [24]. In surveillance system, object trackers can help identify suspicious behavior by tracing the movements of objects or individuals [25].

The trackers can be classified into different categories based on their input and the number of objects they can track. In conventional computer vision, trackers are utilized without object detection, where the coordinates of the objects are manually initialized and monitored in subsequent frames. In contrast, trackers receive temporal information from detectors and execute data association among the stream in deep learning tasks. This approach has several advantages such as the ability to track multiple objects and adapt to shifting environments by tracking newly introduced objects. These trackers are known as multiple object trackers (MOT) [26].

Simple Online and Realtime Tracking (SORT) is one of the earliest deep learning-based object tracking methods. Appearance modeling is the process of modeling the visual appearance of an object. It attempts to resolve the fact that the appearance of target object may change as it traverses various environments, yielding deceptive information and causing the tracker to lose track of the object. SORT disregards appearance features and only utilizes the position and size of the bounding box from detector for data association and motion estimation. It neglects both short-term and long-term occlusion issues to reduce the overhead of the tracker framework caused

by object re-identification [27].

SORT utilize the Faster Region CNN (FrRCNN) detection framework which is a two-stage end-to-end detector. The first stage extracts features and proposes the region of interest, while the second stage classifies the object within the proposed region [28]. SORT will calculate the inter-frame displacement of each detected object using a linear constant velocity model after receiving the detection results from FrRCNN. The model is independent of camera motion and other objects in the frame. The following equation is used to represent the condition of each detected object. Variable u represents the horizontal pixel position at the target's center, while v represents the vertical pixel position. The scale s represents the area of the object's bounding box. Variable r represents the aspect ratio which is set as a constant.

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{1.1}$$

The velocity component is optimally resolved by the Kalman filter [29] and the detected bounding box is used to update the object condition when a new detection is associated with the target. If no target is detected, its state is merely predicted using the linear velocity model without any corrections. Then SORT calculate the intersection-over-union between detection and predicted objects as the assignment cost matrix which used as the input for Hungarian algorithm [30] to assign the objects to their corresponding tracks. Due to the simple implementation of SORT, it reaches the rate of 260Hz and 20 times faster than the state-of-the-art trackers at that time with the similar accuracy [27]. Although, it has been shown to be effective in tracking objects in crowded scenes, but it has limitations in handling occlusions and re-identifying objects that leave the scene and re-enter later.

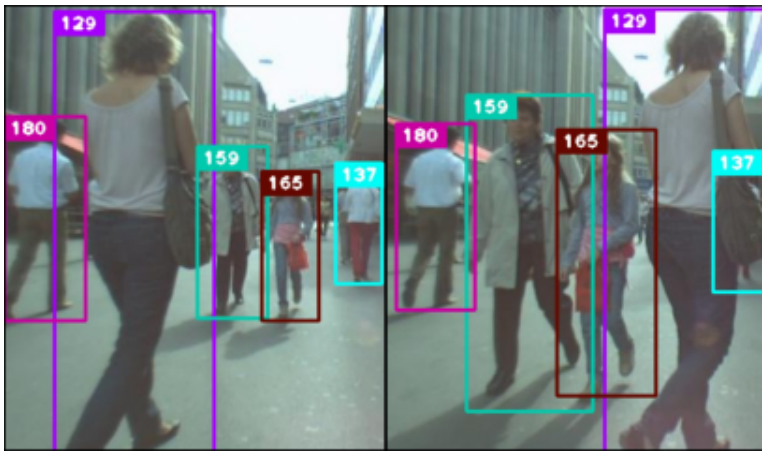

Figure 1.8: The ability of DeepSORT handling occlusions [31]

Deep learning-based SORT (DeepSORT) is an extension of SORT that integrates appearance features derived from deep learning to enhance tracking performance [31]. DeepSORT also employs the Kalman filter to predict where an object will be in the following frame and Hungarian algorithm to designate objects to their corresponding trajectories. However, the input metric of the Kalman filter has changed by replacing the association metric with a more sensible metric that incorporates information

about motion and appearance. Appearance information refers to high-dimensional feature vectors extracted from objects by convolutional neural networks, which are also called deep appearance descriptors. Then, these features are utilized to compute similarity metrics between objects and associate them with their corresponding trajectories. Their CNN network is trained on a large dataset of images to learn discriminative features. This pretrained network enables the use of the algorithm in real-time applications requiring only nearest neighbor queries in visual appearance space. As shown in Figure 1.8, DeepSORT has been shown to be more accurate and robust than SORT, particularly when handling occlusions and re-identifying objects.

Figure 1.9: The ability of StrongSORT handling complex scenes with multiple objects and occlusions [32]

DeepSORT is the earliest method that apply deep learning model into multiple object tracking task and it set a foundation for the other model because of its expansibility, effectiveness and simplicity. For instance, Strengthened Online Real-time Global SORT (StrongSORT) is another deep learning-based tracker that incorporate global information to add further improvement on tracking performance of DeepSORT. StrongSORT use pretrained ResNet50-TP backbone model [33] instead of simple convolutional neural network in DeepSORT as appearance feature extractor to enhance the ability of distinguish objects. These extracted features are kept as inertia terms and update efficiently as the tracklet increase. StrongSORT also modify the nosie of original linear Kalman filter into a varied version which can adaptive to the confidence level of detection result. This enable the StrongSORT can estimate the position of various complex object motions. In re-identification process, StrongSORT propose an appearance-free link model (AFLink) which ignore the appearance model and only use the spatiotemporal metric to determine whether the two trajectories is

the same object. This design continues the simple and fast concept of DeepSORT. A correction algorithm called ECC is used in StrongSORT to compensate for camera viewpoint changes, but this is beyond the scope of this paper's research interest. As shown in the Figure 1.9, StrongSORT has been shown to be more robust than DeepSORT in handling complex scenes with multiple objects and occlusions [32].

Chapter 2

Material

2.1 De-identified intervention video

The videos used for the experiments are records of angiography procedure performed in a Catheterization Laboratory (Cath Lab) at the Reinier de Graaf Groep Hospital in Delft. Angiography is a diagnostic procedure that uses X-ray imaging to display blood vessels in the heart, helping to identify any clogs or restrictions. A catheter is inserted through a small incision made at the arm or groin and guided through blood vessels to the coronary arteries during the procedure. The contrast agent is then injected through the catheter to facilitate X-ray imaging, which aids in the provision of an accurate diagnosis. Patients may undergo additional catheterization procedures, such as stenting or balloon angioplasty, to dilate restricted arteries if blockages or constricted areas are discovered. Due to the presence of medical equipment and staff, the intervention was recorded from five different camera angles to resolve occlusions of important occurrences. The camera installation setup and various camera perspectives are represented in Figure 2.1. The center of the image is the operating table where the patient lies, and the C-arm is the X-ray machine. The patient will enter the room through the sliding door from the southeast camera angle as shown in top right corner of Figure 2.1 and lying on the operating table. The event of the patient entering, lying on the operating table, and exiting is what we are interested in, therefore southeast video is a perfect fit for these events. Video captured from the south wall, which the patient is not occluded by any medical equipment, is great for detect the progress of interventions. All videos are processed by the de-identification tools to blur the face of the personnel and patients in the intervention room to protect the privacy.

2.2 Computer Vision Annotation Tool (CVAT)

CVAT is an open-source image annotation tool for annotating images in computer vision tasks such as object detection, image segmentation, and image classification. CVAT supports exporting annotations in YOLO format, which is exactly what we requested for YOLOV8. In keeping with the confidentiality of the intervening video, CVAT can only be used for local installations. To install CVAT locally, please follow

the official installation guide. After the video is compressed and uploaded, each frame can be viewed and manipulated like a video player. Built-in AI annotation capabilities expedite the CVAT annotation process. It features YOLO, Mask R-CNN and Faster R-CNN detectors to automatically label all common objects in a frame. It also has a tracker that automatically follows the object as it moves from one frame to the next. Additionally, the tracker allows multi-frame stepping, marking objects only at the beginning and end frames, and fine-tuning the bounding boxes of intermediate frames to accelerate the labeling process. In general, the use of these AI tools has greatly improved the efficiency of annotation.

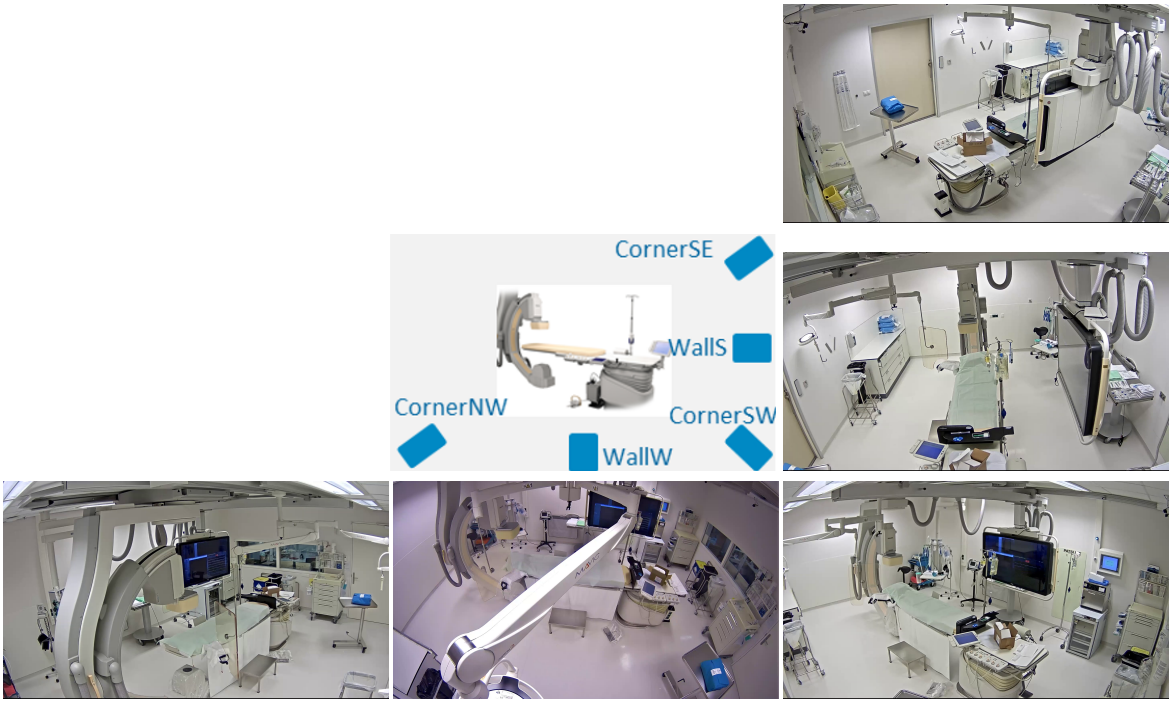

Figure 2.1: Camera installation setup and views from different camera angles, the image at the center illustrates the camera system setup

2.3 Object detector YOLO v8

Ultralytics release the official YOLO v8 in January, 2023. Compared to the performance of other YOLO models on the COCO (Common Objects in Context) dataset, YOLO v8 can achieve higher accuracy with the same number of parameters in the model and faster inference per image. Therefore, this project is developed based on the YOLO v8 as it is the state of the art.

The primary components of YOLO network architecture are the backbone, neck, head and loss. The backbone is the initial part of the network which consists of multiple layers of pre-trained deep convolutional neural network. The backbone can perform hierarchical image processing to reduce the spatial resolution of the im-

age. It can also improve the feature resolution by capturing low-level and high-level features from the images. The neck serves as an intermediate component between the backbone and the detection head. The neck can further process the features extracted by backbone and enhance the ability of models to generalize at various scales, sizes and levels of detail. The head of the network can receive the processed features from the neck and predict the bounding boxes, class probabilities for each object in the image. The loss function can quantify the difference between the predicted output and ground truth to optimize the parameters in the network via backpropagation during training. The model architecture of YOLO v8 is shown in the Figure 2.2.

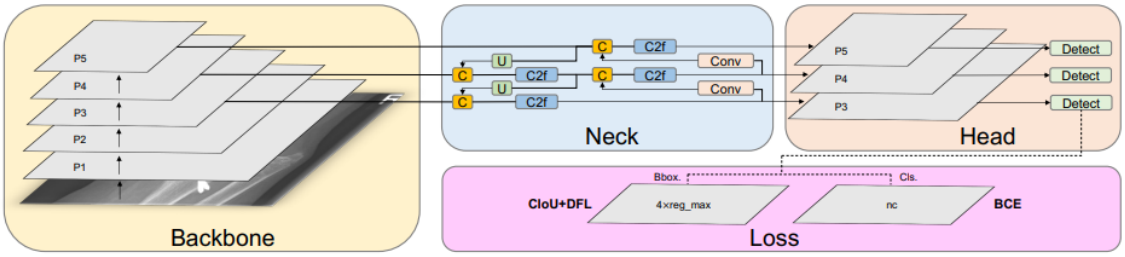

Figure 2.2: The overall structure of YOLO v8 [34]

2.3.1 Backbone

As shown in Figure 2.3, the backbone of YOLO v8 employs the Cross Stage Partial network strategy proposed by Wang et al. [35] and divides the extracted feature map into two sections. The first section is a standard convolution operation, and the second section is a C2f module that receives the output from the previous section as input using a concatenation operation. The C2f module is improved from the C3 module used in YOLOv5 and enhanced by using the ELAN concept introduced in YOLOv7 [22]. It is composed of 2 ConvModule and n BottleNecks connected via Split and Concat between the two ConvModule. The BottleNeck module consists of two Conv modules, with a variable to determine whether to use the shortcut connection and it is set to False by default. The ConvModule module includes a layer of 2D convolution, a layer of batch normalization and a Sigmoid-Weighted Linear unit. The Conv2d can output a transformed feature map that captures spatial relationships and patterns. These features are fed into the bath normalization to ensures that the mean and variance of each feature channel in the output feature map are approximately normalized during training. This process helps stabilize and speed up the training process by reducing internal covariate shift and improving gradient flow. The Sigmoid-Weighted Linear unit serve as an activation function for the output of BathchNorm2d and introduce the non-linearity to the network during activation [36]. By utilizing this structure, the C2f module is able to integrate high-level features with contextual data to enhance detection accuracy.

Another important structure component used in the backbone of YOLO v8 is the Spatial Pyramid Pooling Fusion (SPPF) module. The SPPF module is improved from the

Spatial Pyramid Pooling (SPP) to enhance the inference speed of the model. The SPP module can generate a fixed-length output by aggregating the received feature map of any size [37]. This enables the network to generate feature vectors with consistent size regardless of the dimensions of the input image. Therefore, it is utilized for the fully connected layers or other classifiers that require a fixed input size. The main strength of this module is its ability to extract the most important local and spatial information features without compromising the network speed. The SPPF module achieves its improved inference speed through the incorporation of two Convolutional (Conv) layers and three MaxPool2d layers connected via the Concat. The Conv layers perform spatial transformations and feature extraction, while the MaxPool2d layers perform down sampling operations to reduce the spatial dimensions of the input. The combination of these layers enables the SPPF module to efficiently extract and fuse the contextual information from multiple spatial scales and minimize the computational overhead at the same time. By utilizing the SPPF module into the backbone of YOLO v8, the model can inference faster while preserving its ability to capture relevant contextual features which can enhance the effectiveness and accuracy in object detection tasks.

2.3.2 Neck

Deeper neural networks can encompass more abstract, higher-level features, which is advantageous for tasks such as semantic understanding and image classification. However, in networks for object detection, deeper layers present the challenge of reduced object location information. Deeper layers of the network tend to have a larger field of receptive, which means they extract information from a larger region of the input image. While this facilitates the capture of global context, it can lead to the loss of fine-grained spatial information, such as the precise locations of objects. The reduction in object location information can make it more difficult for the network to precisely locate objects in an image, particularly for objects with intricate details or smaller objects. Object detection architectures typically include skip connections [38], multi-scale features [39], or feature fusion techniques [40] to address this challenge by integrating high-level feature information with fine-grained spatial details. These mechanisms can preserve and incorporate the object location data in shallower layers, mitigating for the potential loss of spatial precision in deeper layers. YOLO v8 use the Feature Pyramid Networks (FPN) and the Path Aggregation Network (PAN) for the neck part of the network to address this challenge. YOLO v8 utilizes the FPN structure to give shallower layer more feature by up sampling from deeper layers [41]. In order to compensate for the loss of spatial information in the deeper layers, YOLO v8 uses the PAN structure to extract more location information from the shallower layers through down sampling[42]. As mentioned before, the neck is a tool that connects the head and the backbone in network structure. Hence, the FPN and PAN structure are only applied to the backbone layers that will be connected to the head.

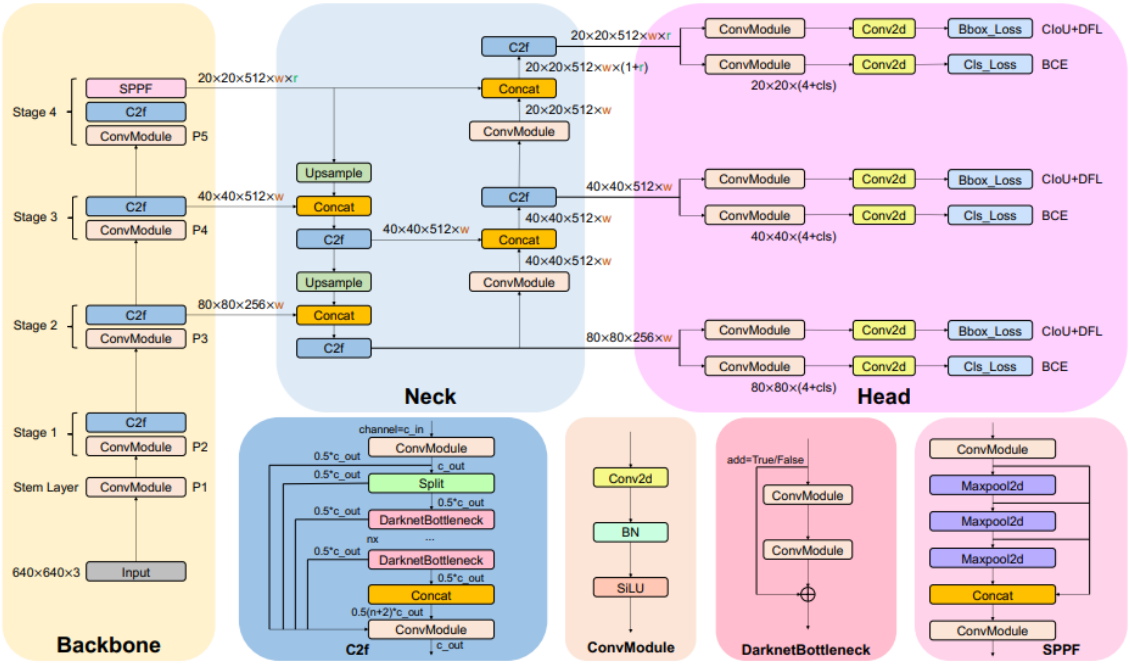

Figure 2.3: The detail structure of YOLO v8 [34]

2.3.3 Head

The head is used to predict the object scores, class probabilities and bounding box information for detected objects. The head of YOLOv8 consists of multiple convolutional layers and fully connected layers. The head module in YOLOv8 is a detect module that takes feature maps from different stages of the neck network as input to perform object detection. Feature maps from various phases of the backbone network extract features at varying spatial resolutions. Feature maps with a lower resolution, such as P3, contain more high-level semantic information and can capture large objects more accurately. In contrast, the P5 which is higher resolution feature maps contain more information and can capture small objects more effectively. By integrating feature maps from varying scales (P3, P4, and P5) at the head, YOLOv8 can utilize higher resolution feature maps for smaller objects and lower resolution feature maps for larger objects. This multi-scale feature fusion enables the model to detect objects of various sizes and scales, thereby enhancing the detection performance.

Another enhancement introduced in the head of YOLO v8 architecture is adopting the anchor-free method instead of the anchor-based method. As mentioned in introduction of object detector YOLO, the anchor-based approach relies on pre-defined anchor boxes to compute the offset between the predicted box and the anchor box, allowing for more accurate object localization. However, the model can directly predict the bounding boxes without anchor boxes in the anchor-free method. This

eliminates the step of computing the offset, resulting in a reduction in the number of predicted boxes. Hence, the input to the Non-Maximum Suppression (NMS) algorithm which is used to filter out the redundant detections is reduced, leading to improved efficiency in the post-processing stage. By employing the anchor-free method, YOLO v8 achieves accurate object detection while simplifying the detection pipeline.

2.3.4 Loss

The Binary Cross-Entropy (BCE) loss function is used as the classification loss function in YOLO v8. The function is shown below, where *N* is the number of training samples, *y* is the ground truth label and *p* is the prediction. The BCE loss function measure the difference between the predicted class probabilities and ground truth labels for each class *i* and each grid cell *j*. The result of BCE loss function can penalize the model for predicting low probabilities for correct classes or high probabilities for incorrect classes.

$$\text{BCELoss}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{S^2}[y_{ij}\log(p_{ij}) + (1-y_{ij})\log(1-p_{ij})] \tag{2.1}$$

The Distribute Focal Loss (DFL) and Complete IoU (CIoU) function is used for the bounding box regression in YOLO v8. The equation for CIoU loss function is shown as follow, the *IoU* represent the intersection over union and the *v* is the aspect ratio. The first item of the equation is the overlapping area, the second item is the normalized distance between the center points of the ground truth box and predicted box and the third item is the term for aspect ratio consistency. For the non-overlapping case, the CIoU loss can quickly converge the predicted bounding boxes to the ground truth bounding boxes. The CIoU loss requires fewer iterations in converging process which makes regression very fast for various aspect ratios [43].

$$\text{CIoULoss} = (1 - IoU) + \frac{Distance^2}{Distance_c^2} + \frac{v^2}{(1-IoU)+v} \tag{2.2}$$

The equation for DFL loss function used in YOLO v8 is modeled by Ju and Cai [34] as follows and *y* represent the object. They claim that the effect of DFL is to expand the probability values around objects, but this is not confirmed as there is no official YOLO v8 paper yet and no other paper prove they are correct.

$$\text{DFL}(S_i, S_j) = -((y_j - y)\log(S_i) + (y - y_i)log(S_j)) \tag{2.3}$$

2.4 Multi-object tracker ByteTrack

This project use the state-of-the-art multi-object tracker ByteTrack which proposes a new data association framework, BYTE. BYTE associates every detection box from the object detector, unlike other tracking methods that only perform tracking of high scoring detections [44]. BYTE can remove background detection and recover the

occluded objects by using similarities to the tracks of low scoring detections.

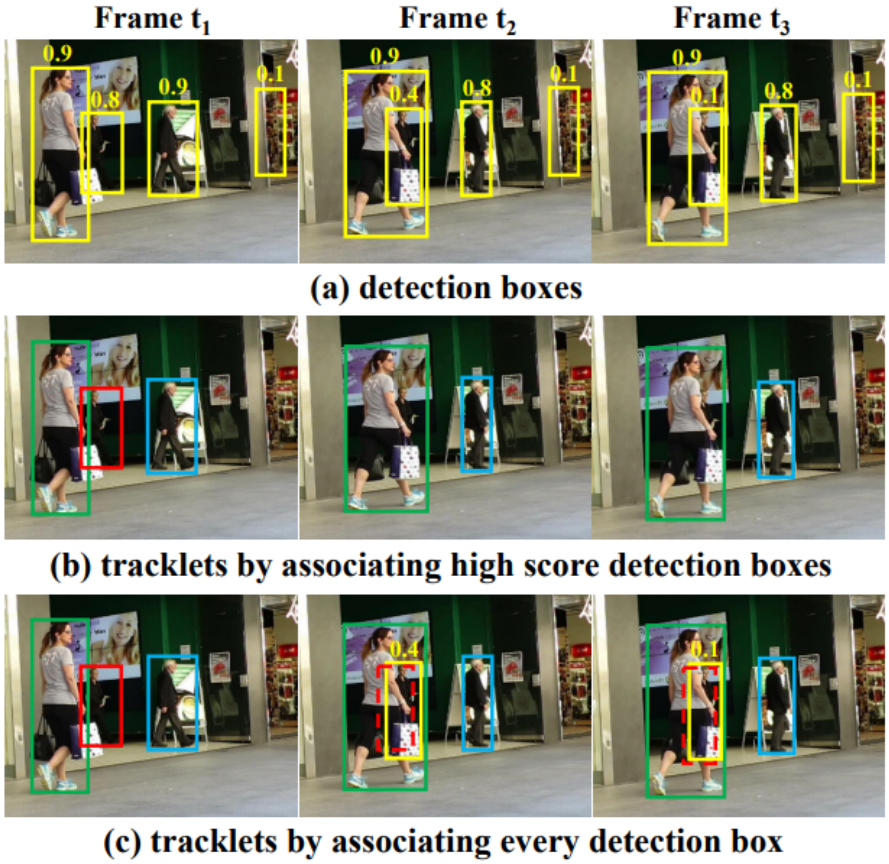

Figure 2.4: The different matching strategy used by ByteTrack and other tracking method [44]

The Figure 2.4 is a good example to show the problem that ByteTrack solves. The first row of images shows the detection results of the object detector on the video stream. It is obvious that the object detector is robust enough to detect the man in the middle even though it is occluded by the woman with shopping bag and there is an error detection that detects the background noise as an object on right side of the image. The second row of images shows the tracking strategy of other trackers which only perform tracking on the detected box over a certain confidence threshold and the same identity is represented by the same bounding box color. The third row of images shows the result of ByteTrack, which takes all the bounding box into account and can track the man in the middle even with severe occlusion. The dotted bounding box in the middle is the predicted box of Kalman filter, which is reserved as a tracklet because of the large Intersection over Union (IoU) value with the detected result. In addition, ByteTrack also utilize the motion model to distinguish the background and the object of interests in the detection boxes with low scores. This approach solves the problem of avoiding false positives, which is why most trackers set a threshold for the object to be tracked.

```
1 # Input
```

```
Input Video = V
Object detector = Det
Detection score threshold = t
# Output
Track result = T_result
T_result = 0 # Initilize
for frame in V:
    # Perform detection, get confidence and box
    D_result = Det(frame)
    # Initialize empty subset for high/low confidence
    D_high, D_low = 0
    # Split detection result
    for d in D_result:
        if d.confidence > t:
            D_high.append(d)
        else
            D_low.append(d)

    # Perform prediction by Kalman filter
    for track in T_result:
        KalmanFilter(track)

    # First association using Hungarian algorithm
    Similarity_First = SimilarityMetric(T_result, D_high)
    T_result = Hungarain(Similarity)
    D_remain = unmatched detection from D_high
    T_remain = unmatched track from T_result

    # Second association for low confidence
    Similarity_Second = IoU(D_low, T_remain)
    T_result = Hungarain(Similarity)
    D_left = unmatched detection from D_low
    T_left = unmatched track from T_remain

    # Clean up
    Delete D_left
    Delete T_left after n frames

    # Intial new tracks for high confidence
    For d in D_high:
        initial new track in T_result

return T_result
```

The pseudo code of BYTE shown above, takes a video sequence, an object detector and a confidence threshold as input. The output of the algorithm will be the tracklet of all the interested object in the video with the temporal and spatial information and corresponding identity. For each frame in the video sequence, the algorithm will first apply the object detector and separate the detections into the high confidence detection subset *D_high* and low confidence detection subset *D_low* based on the set confidence threshold. After the separation, BYTE use the Kalman filter to predict the new location for each track in the current frame. Then the first association is performed among the detected boxes by first computing the similarity to identify

which detection belong to the same objects for the high confidence detection subsets. The similarity can be calculated by the Re-ID feature distance or the IoU values between detection and prediction from Kalman filter. The computed similarity is used as a cost matrix for the Hungarian algorithm to optimally assign the detection to tracks. The unmatched high confidence detections *D_remain* and unmatched tracks *T_remain* are stored for further processing. For the low confidence detection subset, the second association is conducted with the unmatched tracks from the first association. The unmatched low confidence detections *D_left* are considered as background or error detection and deleted from the result. For the remaining tracks *T_left* after the second association are kept for a certain number of frames and deleted if no association is performed. After the two association, new tracklet for the unmatched high confidence detections will be initialized and add to the tracking result. In this way, BYTE has performed data association for all detection results. BYTE is highly scalable and have flexibility to configure different similarity metrics in two associations based on the using scenarios. BYTE can be connected to any object detector as long as the input is correctly feed and the detection performance of the detector will affect the upper limit of the tracking results. ByteTrack uses fine-tuned YOLOX [45] as object detector, while YOLO v8 is used in this project for better performance.

Chapter 3

Method

3.1 Target event and definition

Most of the recorded videos are of diagnostic Coronary Angiography (CAG) procedures and the workflow in the target Cath Lab is shown in Figure 3.1. During the patient preparation phase, the patient will enter the intervention room via a sliding door, walk to the operating table, and lie down. The nurse will next prepare the necessary medical instruments and drape the patient. At this time, the cardiologist will enter the room and attempt to identify the best access point on target arm or groin vein by manually loosening the vein. The local anesthetic will be administered by an intravenous catheter, and an incision is made at the entrance to place the sheath, which is a small plastic tube used to insert. Moreover, doctors inject vasodilators and anticoagulants to dilate veins and inhibit the blood clotting response that closes the wound. The catheter is thereafter inserted and directed to the aortic root. During the insertion, X-ray pictures of coronary arteries are taken using a technique called Coronary Angiography. Throughout this procedure, cardiologists may switch access points to gather more information. After sufficient information has been collected, the catheter will be withdrawn and the sterilize nurse will commence suturing, which is the patient aftercare module in workflow. If the access point is the arm, an inflatable wristband will be applied over the incision to raise pressure, so preventing bleeding and clot formation. After a period of recovery, the patient will leave the room with the support of the nurse. If the entry area is the groin, the incision is closed by manual pressure and the patient is lifted out of the room on a transport bed to avoid bleeding.

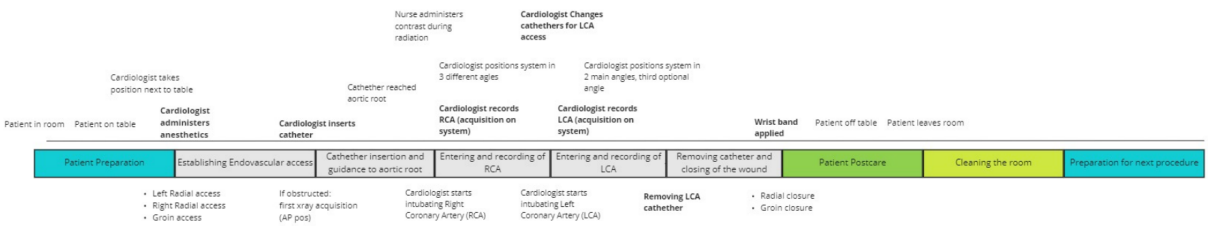

Figure 3.1: Workflow in Cath Lab

As mentioned in section 1.1, most of the KPI measurements in Cath Labs require knowing when patients enter and leave the room and when access is established and closed. Therefore, this thesis regards these four events as the target events and the composition of the video are described below. As seen in the Figures 3.2, patients are walking through sliding doors or transport by shuttle beds for patient entry and exit events.

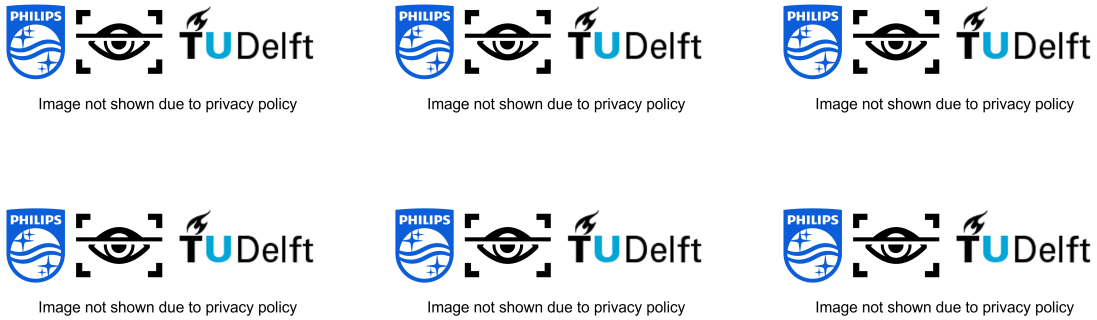

Figure 3.2: Patient enters and exits.

The Figure 3.3 show the events before the access point is created. After entering the room, the patient will lie down on the operation table undraped. While the patient is lying on the bed, the sterile nurse usually prepares the anesthetic, vasodilator and anticoagulant to be used at the start of the procedure. After preparation, the nurse will drape the patient and expose the point of access. The cardiologist will then come in and put on the surgical gown and begin to feel the position of the veins.

Figure 3.3: Events before the access point is made.

As shown in Figure 3.4, these are the events that occur after the access point is created. The first image shows the cardiologist injecting the prepared anesthetic. The cardiologist then uses a scalpel to create the access point, while the bleeding in the second picture is a sign of access, followed by the catheter insertion process shown in the third picture. The zoomed-in images in the second row visually show what happened at the access point.

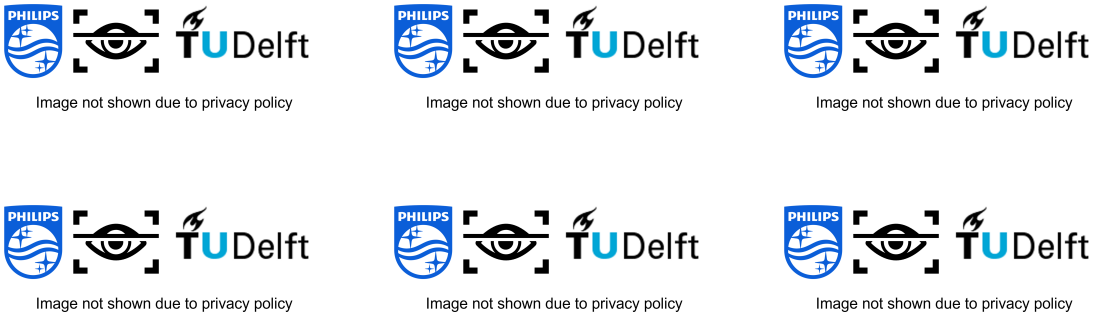

Figure 3.4: Events after the access point is made.

The Figure 3.5 depicts what occurs after the cardiologist retrieves the catheter and before the patient leave the bed. Suturing is typically performed by the sterile nurse, but occasionally by a cardiologist. When the nurse let the patient grasps the syringe is the sign that the suturing process is complete and the patient is ready to leave.

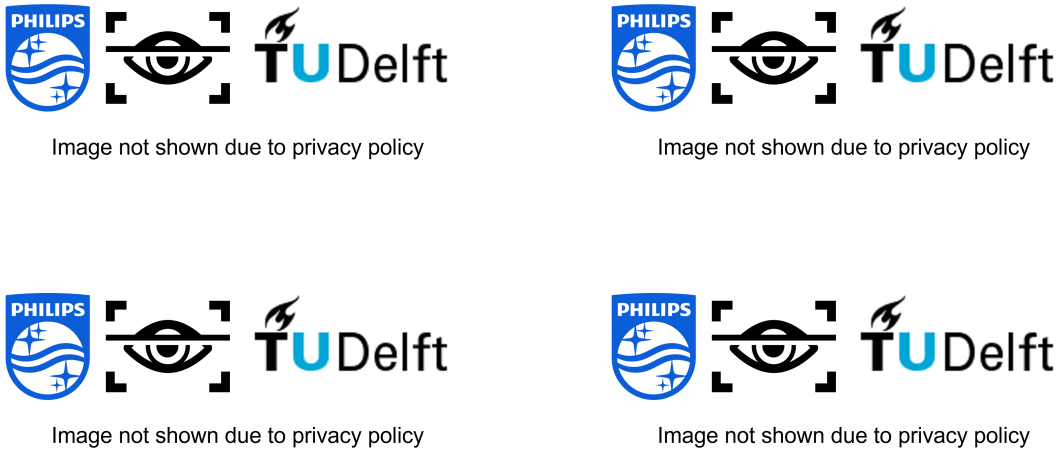

Figure 3.5: Events for closing the access point.

3.2 Dataset preparation

3.2.1 Dataset for patient enters and exits

As mentioned previously, the best camera angle to use for detecting patient entry and exit events is the southeast angle as shown in top right corner of Figure 2.1. The goal of this project is to detect events by using information from object detection results, so defining the appropriate object classes is the first step. The project for patient entry and exit were created in CVAT along with 5 object class: 'Patient', 'Door', 'Cardiologist', 'Nurse' and 'Sterilize Nurse'. For the events of patients entering and exiting the room, we were most interested in the patient and the door, but other personnel in Cath Lab were also annotated to test the generalization capability of YOLO. In order to obtain improved outcome, the following basic rules was performed when annotating the images:

1. Be accurate: Accuracy in annotation process is crucial for object detection. Include as few background elements as possible while annotating borders to precisely include objects. Considering target object are large objects with numerous special cases, this project uses **Rectangle** rather than **Polyline** to label all objects. When objects of interest overlap each other, if the overlap is not severe meaning that most of the features of the objects are preserved, then the bounding box is marked as overlapping and the labeling is artificially predicted to contain the entire object. If the overlap is severe, only the unobscured objects are labeled and the rest of the obscured objects are considered as noise for the input algorithm, which can enhance the robustness of the algorithm.

2. Be consistent: Maintain consistency throughout the entire annotation process. This includes applying the same labeling conventions for similar objects and using the same shapes, colors and labeling procedure for the bounding boxes.

3. Use multiple annotations: Label as many different states of each object as possible such as different orientations, dimensions and lighting conditions. Thus, the videos are randomly selected from different recorded dates, which ensures that the videos have different clarity, lighting conditions and camera views. This will improve the accuracy of the model predictions.

4. Validated annotations: CVAT can output coordinates in YOLO format for each annotated frame and save them in the txt file name with corresponding frame number. These corresponding frames are extracted by a script using OpenCV and together with the txt files are the data input for YOLO v8. In addition, to verify the veracity of the dataset, a script is written to draw the annotated ground truth bounding box in the corresponding extracted frames to verify the annotations. As shown in Figure 3.6, this step will ensure that the annotations are accurate and consistent.

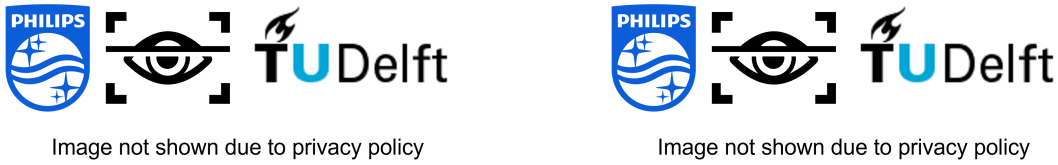

Figure 3.6: Validate process for dataset: Left (Annotation in CVAT), Right (Annotation drawn by validation script)

Based on these guidelines, 12 videos from the southeast corner were randomly selected, two of which labeled all personnel from start to finish, while the remaining ten videos labeled only the frames in which patients entered and exited. A total of 16 patients were labeled, 14 of whom walked in and 2 of whom used transport bed. The specific differences between the two types of entry and exit are shown in Figure 3.7.

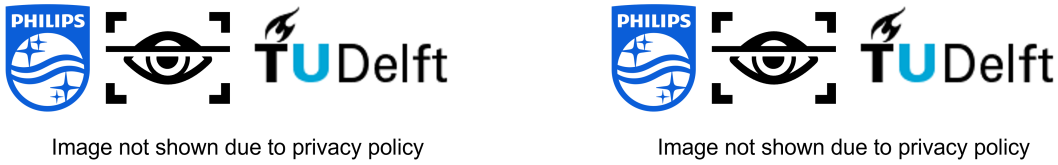

Figure 3.7: Two ways for patient entry and exit the Cath Lab: Left (walk), Right(transport bed)

Initially, only one video containing the entire Cath Lab workflow as previously described was annotated. All personnel were annotated to evaluate the capacity of YOLO to generalize personnel in Cath Lab. Surprisingly, the generalization of YOLO to all individuals exceeded expectations, with the exception of the most important object 'Patient'. Inspection of the dataset revealed that the process of the patient entering and exiting the room took only 20 seconds, resulting in a significantly smaller number of effective frames in comparison to other objects. Therefore, the network's ability to detect the patient is poor, whereas the performance of the other categories is decent. In order to improve the ability to detect patients during enter and exit action, the remaining 11 recordings were labeled to increase the number of frames for patient entry and exit. The Table 3.1 depicting the number of instance labeled for each object class, the number of 'Patient' instances in the final dataset is significantly increased.

A critical step in preparing dataset for object detection is choosing an appropriate background. Background is the area of an image in which there are no objects of

Table 3.1: The statistic table for patient enter and exit dataset. The first dataset is a dataset created to test the capabilities of YOLO and contains 1 patient from 1 video. The final dataset is the one used for training and contains 16 patients from 12 videos.

Object Class	First Dataset (Frame)	Final Dataset (Frame)
Patient	19394	63257
Door	36180	75027
Cardiologist	16695	28078
Nurse	16020	31221
Sterilize Nurse	18751	39675

interest. A well-defined background is essential because it enables the model to learn to differentiate between the foreground (the object of interest) and the background. Following a summary of prominent object detection algorithms [13][46][47], here are some key factors to consider when selecting the background for object detection:

1. Choose a neutral background: The background should be as uniform and neutral as feasible, with minimal variation in color, illumination, and texture. A neutral background allows the model to concentrate on the object in the foreground.

2. Include images with complex backgrounds: While neutral backgrounds are ideal, it is also important to include images with complex backgrounds, as this will help the model learn to distinguish between foreground and background more effectively.

3. Use large number of background images: The more diverse of background images, the better the model can generalize to new images.

4. Avoid using foreground objects as backgrounds: It is crucial that foreground objects are not included in background images, as this can confuse the model and reduce its accuracy.

For our dataset, people walk around the room and the variety of the background is very limited which makes it difficult to create an ideal background images. Therefore, the selected dataset contains some frames where the room is empty and only the 'Door' is labeled. While it is not a perfect background, the results are great when containing these frame as the background and feed into the network.

In object detection, data augmentation is a technique that generates new training images by performing various transformations on the original image. The aim is to improve the robustness and generality of the trained model by introducing variations in the orientation, size and appearance of the objects in images. There are many popular data augmentation techniques that have been shown to improve the performance of detectors, such as **Hide and Seek** [48], **Cutout** [49], **Random Erase** [50], **Grid Mask** [51], **MixUp** [52], etc. However, none of these techniques are utilized in this project. During training, YOLO v8 will instead use the built-in augmentation

method **Mosaic** [20]. The Figure 3.8 explained how the Mosaic work. Mosaic will select four different images randomly from the train dataset and resize these images into a 2*2 grid. The corresponding bounding boxes for the objects in each of the four images are adjusted to account for their new positions. The grid will be process as a single large image and do some basic augmentation such as flip, adjust contrast. Then an area from the grid will be selected randomly as the training data for YOLO.

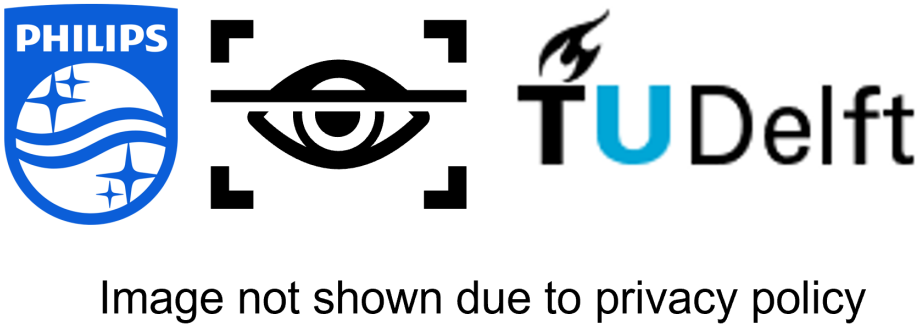

Figure 3.8: Working principle of Mosaic augmentation in YOLO v8

3.2.2 Dataset for procedure events

As mentioned earlier, intervention begins when the cardiologist makes an incision at the access point with a scalpel. Therefore, the object is actually different states of the same area in the dataset for procedure events. The state of the access point is labeled as 'ArmNoAccess' which represent there is no blood at the access point for arm or as 'ArmAccess' represents there is blood at the access point. The presence of blood labeling logic also applies to the access point for groin with the object names 'GroinNoAccess' and 'GroinAccess'. The Figure 3.9 illustrate a few instances of each class.

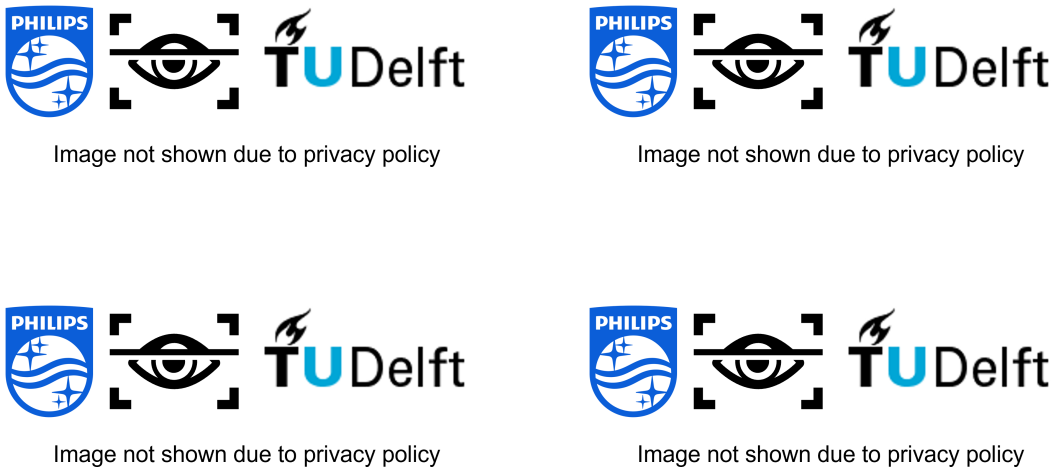

Figure 3.9: Instance for procedure events. First row: No blood at the access point. Second row: Blood appears at different access point

Figure 3.10: Instances for 'Suture' object

It is difficult to observe suturing process in groin type surgery from the video due to its complexity. Therefore, the dataset only contains the 'Suture' object for arm type surgery. The label logic for the 'Suture' object is the area of the patient's arm after the sterile nurse removing the patient's drapes. To distinguish the patient's arm during the suturing process from the patient lying on the bed before the intervention, as shown in the Figure 3.10, the hand of sterile nurse, the syringe, and the arm of patient are labeled as 'Suture' together to add some features to distinguish for the object detector. After following this annotation logic and the annotation ground rules

mentioned before, 14 videos from the south camera angle are randomly selected and 12 interventions are labeled. The detail for the dataset is shown in Table 3.2.

Table 3.2: The statistic table for procedure dataset

Object Class	Number of annotated frames
ArmNoAccess	41551
ArmAccess	37252
GroinNoAccess	16487
GroinAccess	2341
Suture	10559

For both dataset, the extracted pictures and corresponding label files are randomly assigned to the following directory tree through a split script according to the ratio of 80% training set, 10% verification set, and 10% test set. The *Dataset.yaml* file will define the path to training, validation, and testing set, declare the number of classes and specify the corresponding names in order.

```
Dataset
├── Dataset.yaml
├── train
│   ├── images:  Folder to store the training images
│   └── labels:  Folder to store the annotations for the training images
├── val
│   ├── images:  Folder to store the validation images
│   └── labels:  Folder to store the annotations for the validation images
└── test
    ├── images:  Folder to store the testing images
    └── labels:  Folder to store the annotations for the testing images
```

3.3 Training of object detection

Another significant improvement in YOLO v8 is the command line interface, which enables all tasks to be executed from the terminal using simple single-line commands. The command line makes YOLO v8 extremely convenient and effective to use and detailed introduction to using command line interface can be found here. The following example shows the simplicity and versatility of using the command line interface. Training can be initialized by specifying the tasks to be performed, setting the mode of the model, and modifying the values of some default hyperparameters.

```
yolo TASK MODE ARGS

Where TASK (optional) is one of [detect, segment, classify]
      MODE (required) is one of [train, val, predict, export, track]
      ARGS (optional) are 'arg=value' pairs that override defaults.
```

The training of this project is carried out on Philips deep learning server, which is equipped with four NVIDIA GeForce RTX 2080Ti with 12GB memory. The training command line is as follows:

```
yolo task=detect mode=train model=yolov8n.pt data=dataset.yaml batch
   =32 epochs=20 imgsz=640 device=0
```

The task of this project is to train a pretrained model for object detection task. Thus, the *task* is detection and the *mode* is train. Ultralytics officially provides 5 different pre-trained models ranging from simple to complex, and their performance is shown in the Table 3.3. The accuracy of the model increases and the speed decreases with the increase of the model parameters. Therefore, the training process mainly uses the YOLOv8n model to detect small objects, and YOLOv8m is used because of the balance of accuracy and speed.

Table 3.3: Five different models provided by Ultralytics

Model	mAP@50-95	Speed CPU (ms)	Parameters (M)
YOLOv8n	37.3	80.4	3.2
YOLOv8s	44.9	128.4	11.2
YOLOv8m	50.2	234.7	25.9
YOLOv8l	52.9	375.2	43.7
YOLOv8x	53.9	479.1	68.2

The *data* argument is a yaml file configured for the created dataset which specifies the path to training, validation and test dataset, the number of classes, and the name of classes. The batch size determines the number of images that are simultaneously fed into the network for computation. Batch size is set to 32 when training on yolov8n model and 16 when training on yolov8m model to prevent running out of memory problem of using CUDA. YOLO v8 provides mosaic, mixup, copy paste image enhancement methods, and only mosaic augmentation is used in the training process. The training epoch is usually greater than 10 because the mosaic dataloader will be turned off when the remaining training epoch is 10. The hyperparameter warmup epoch is set to 3 to avoid initial instability or possible suboptimal convergence due to random weight initialization. The learning rate used the official default setting of 0.01 to avoid affecting the convergence and performance of the model.

3.4 Link object detection and object tracking

As mentioned earlier, the data association method BYTE can be chained to any custom object detector with the correct input. Therefore, the linker script is used to manage the configuration of tracker, connect the tracker with object detector and organize the output. The pseudo code is shown as follow:

```
# Input
Input video - Vin
Object detector - Det
```

```
Detection confidence threshold - t_c
NMS threshold - t_nms
# Output
Saved video with tracklet - Vout
Saved txt with tracklet - TXTout

# Manage arguments using parser
parser = argparse.ArgumentParser()
# Process the input video and create ouput directory
exp_name = increment_path(Path(project) / str(Vin))
vid_path, txt_path = exp_name.mp4, exp_name.txt
# Load YOLO model
model = AutoBackend(Det, device)
# Create dataloader for input video
dataset = LoadImages(Vin, imgsz, model.stride)

# Run tracking
for frame_idx, image in dataset:
    # Convert the image into pytorch tensor
    im = torch.from_numpy(im).to(device)
    # Using YOLO v8 to make prediction
    preds = model(im)
    # NMS to remove error and duplicate
    preds_nms = non_max_suppression(preds, t_c, t_nms)
    # Process detections
    for det in pred_nme:
        scale_box(det) # rescale the boxes to image size
        output = tracker(det) # Feed the detections to BYTE
        # Draw bounding box
        vid_path = annotator(det)
        # Save txt
        txt_path = txt.write(det)

return vid_path, txt_path
```

The algorithm takes an input video, a configured object detector, a detection confidence threshold, and a non-maximum suppression threshold as input. For each frame in the video, the script converts the image into a pytorch tensor and passes it to the object detector for prediction. The generated predictions are filtered out by NMS operations to remove redundant bounding boxes. The filtered bounding boxes are fed to BYTE to perform data association and tracking. The results of the tracker are also written to each frame of the video and to the corresponding text file.

3.5 Event detection

The results of the object tracking are used as input data for event detection. The event detection framework in this project is different from the regular event detection framework due to the different usage scenarios. A conventional event detection framework first detects objects, analyzes the motion of each object, and compares the spatio-temporal information of each object to identify whether an event is occurring. For patient entry and exit event, the motion state patient is relatively simple as

walking or lying down. While it is not difficult to analyze what movement the patient is making, the patient will also be lying on a transport bed outside of the room before and after the procedure begins. In addition, the direction in which the walking movement is performed needs to be determined by a reference object. Therefore, the relative distance between the bounding boxes of patient and the door are used to detect patient entry and exit events without the need for movement analysis of the patient. When the relative distance between the bounding box center of patient and door increases, the patient is considered to enter. The patient is considered to leave when the distance decreases, and the patient is considered to stay when the distance remains stable at a predetermined interval. The pseudo code of the data processing steps and event detection are shown as follow.

```
# Input
Input video from tracking - V_track
Input txt from tracking - Track_txt
# Output
Saved video with event detection result - V_event
Saved txt with event detection result - Event_txt

Window_size = 150 # Window size for moving average filter
# External function
def read_data_from_txt_file(Track):
    # load the data from txt contained the object tracking
    data[frame][class] = bbox information

def calculate_IoU_distance(data):
    # calculate the distance and IoU of input data
    data[frame][distance] = Euclidean_distance(bbox)
    data[frame][IoU] = IoU(bbox)

def event_detection(data):
    # only enter and exit can happen when have IoU
    if intersection in data:
        for time_interval in len(data):
            if next_distance - current_distance > threshold:
                data[frame][event] = 'Patient enters'
            else:
                data[frame][event] = 'Patient exits'
    # stay event only happen if IoU not exist
    else:
        for time_interval in len(data):
            if next_distance - current_distance > threshold:
                data[frame][event] = 'Patient enters'
            elif next_distance - current_distance < threshold:
                data[frame][event] = 'Patient exits'
            else:
                data[frame][event] = 'Patient stays'

def main():
    data = read_data_from_txt_file(Track_txt)
    for frame in data:
        # only process the frame contain both Patient and Door
        if 'Patient' and 'Door' in frame:
```

```
            frame = calculate_IoU_distance(frame)
    # apply the moving average filter to distance data
    data = moving_average(data[frame][distance], Window_size)
    # perform event detection
    data = event_detection(data)
    # edit the video and save the result to txt
    V_event = video.write(data)
    Event_txt = txt.write(data)
```

For each frame containing the object 'Patient' and 'Door', the intersection of the two bounding boxes and the Euclidean distance are calculated. These are variables that are used as conditions for event detection. If there is an intersection, it means that the patient is walking around the door and only entry and exit can occur. If no intersection exists, the patient may be lying on the table or walking. The data is passed through a *convolve* function from Numpy library to perform convolution operation which moves the average filter with a predefined window size to remove noise. For each frame, the intersection is checked and the remaining streams with a preset time interval are processed in a nested loop. If the difference between the distance of the next frame and the previous frame exceeds the threshold, the event key in the frame list is set to 'Patient enters'. Otherwise, the event variable is set to 'Patient exits'. The function then updates the event key of the frame in the data dictionary for a short period of time after this frame to make the algorithm robust to imperfect tracking results due to occlusion. The event variable is set to 'Patient stays' for frames with no intersection and no distance change within the preset time interval.

Image not shown due to privacy policy

Figure 3.11: The result for split red channel. (a) The area of 'Arm No Access' (b) Splitted red channel of 'Arm No Access' (c) The area of 'Arm Access' (d) Splitted red channel of 'Arm Access'

Various techniques were tried to accurately detect access and closure event. The first method experimented with is performing the image processing, as the presence of blood is a sign that access has been established and blood is dark red while most other important objects are blue. Therefore, a script was used to convert each frame in the video from RGB color scale to HSV color scale. Another script lets the user click on the cursor to select an area of blood and get the HSV value for that area. After randomly selecting five videos from different dates with different lighting conditions and obtaining the HSV range of the blood region, the red channel around the selected range values were separated from the image. However, the results were not

as expected, and such a method could not distinguish between no access and access status of the regions as shown in the Figure 3.11. In addition, the gloved hands of cardiologist were difficult to separate from the blood as noise.

If relative distance is used as in the case of patient entry and exit events, the cardiologist and patient need to be detected precisely and the event is judged by the duration of the cardiologist standing next to the patient. However, this approach is unreliable due to the fact that the cardiologist will stand next to the patient for a period of time before access is established and use hands to feel where access should be created, and may suddenly leave during the procedure. In addition, there may be more than one cardiologist present during the procedure and the detection of the cardiologist from object detection is not perfect due to lack of distinctive features as well as severe occlusion. Detecting movements of cardiologist is not feasible because during the procedure, some cardiologists may habitually bend over, while others adjust the height of the operation bed to facilitate standing up straight to perform the procedure. Analyzing the hand gesture of cardiologist is also impractical because the hands are gloved and heavy occlusion makes it difficult to analyze which gesture belongs to which event. Therefore, in order to accurately detect when the access is established, it is most feasible to detect when and at which access point bleeding appears. As previously mentioned, the presence or absence of blood labeling logic at the access point was used to create dataset and train network.

Image not shown due to privacy policy

Figure 3.12: Detection of Suture object has many False Negatives. Left: Arm of patient before procedure (error detection) Right: Arm of patient during suturing (correct detection)

The 'Access established' event can be detected by changing from 'Arm/Groin No Access' object to 'Arm/Groin Access' object and the 'Suture' event is detected by changing from 'Arm/Groin Access' object to 'Suture' object. The object detection results for this event were not as desirable as the patient entry and exit event because the objects were small and represents different states of the same region, which means that they did not have distinct characteristics. For instance, the arm area where the suture event occurred and the patient's arm before procedure were both detected as 'Suture' objects as shown in Figure 3.12. To solve this problem, an off-the-shelf single shot detection network was used to obtain the operation bed information in Figure 3.13. The table states are distinguished into three states based on whether the patient is on the table and whether the patient is draped which are: *TableEmpty*, *TableUndrapedPatient* and *TableDrapedPatient*.

Figure 3.13: The single shot detector classify the table status into three states: TableEmpty, TableUndrapedPatient and TableDrapedPatient

By adding additional information about the table status, the workflow for this event is simplified as the table is first empty, the patient enters the room and lies on the bed undraped. The patient is then draped by nurse and access is established, sutures are made and the empty state of table is restored for the next patient. The recorded video may start and end from any part of this simplified workflow. Therefore, the algorithm for detecting this event must be sufficiently robust to different situations and the pseudo code is shown as follow.

```
# Input
Input video from single shot detector - V_table
Input csv from single shot detector - Table_status
Input txt from tracking - Track_txt
# Output
Saved video with event detection result - V_event
Saved txt with event detection result - Event_txt

def event_detection(data):
    # Detects 'Arm Access established' events based on the changes of object ID from 0 to 1.
    # Detects 'Groin Access Established' events based on the changes of object ID from 2 to 3.
    for frame in data:
        if {0} is in last_object and {1} is in current_object:
        data[frame]['event'] = 'Arm Access Estabiashed'
        elif {2} is in last_object and {3} is in current_object:
        data[frame]['event'] = 'Groin Access Estabiashed'
    return data

def process_data(data):
    # This function will remove the error suture (subkey '4') and detect the correct Suture event.
    change_patient = False
    for frame in data:
        if table_status is 'TableDrapedPatient' and {4} is in frame and change_patient is False:
            data[frame]['event'] = 'Suture'
        else:
            remove suture object from frame
    return data

def main():
    # combine data from txt and csv
    data = read_data(Track_txt, Table_status)
    data = event_detection(data)
```

```
    data = process_data(data)
    # edit the video on the result from table status
    V_event = video.write(data, V_table)
    Event_txt = txt.write(data)
```

3.6 Gaussian smooth interpolation

The values of tracking threshold were found to affect the accuracy of the results. Higher threshold values result in less ability to handle occlusion but more reliable results. While lower threshold values for non maximum suppression of the object detector can handle occlusion well but contained a lot of false positives. Therefore, to balance the ability to handle occlusion and accuracy, the tracking threshold is set high to avoid false positives being sent to ByteTrack. To enhance the low ability to handle occlusion due to the high threshold, Gaussian smoothing interpolation is used to compensate for the heavily occluded frames. This post-processing method consists of two parts: linear interpolation and Gaussian smoothing of the interpolated results. Linear interpolation uses the tracking results as input and calculates the interpolated bounding box coordinates for undetected frames in consecutive frames.

The Gaussian smoothing function uses interpolated data to perform Gaussian smoothing on each individual object trajectory. The function first identifies the unique object classes present in the data. For each class, it retrieves the corresponding interpolated trajectory and applies Gaussian Process Regression (GPR) using the Radial Basis Function (RBF) kernel. The length scale parameter of the kernel is dynamically adjusted according to the track length. The regression operation is applied to the x, y, width, and height coordinates of the object track separately with the function of time which is frame number. The regression can produce a smoothed estimate for each coordinate and then combined with other track information. The method provides hyperparameters for how many frames the algorithm can interpolate and length scale parameters for the desired smoothing effect.

Chapter 4

Result and Discussion

An evaluation was performed to see the performance of the fine-tuned YOLO v8 model and how much the object tracker could improve the results. The reliable tracking results are the cornerstone of an accurate event detection framework. A short video of 26572 frames in length was used as the test video, which included the complete standard workflow. In addition, the test video was chosen to show patients entering and exiting while being occluded by a nurse to test the performance of YOLO v8 and the ability of tracker to handle occlusion. The labeling process for the test video followed the labeling logic used in preparing the dataset and were used as ground truth. The following subsections show the results of each component in the proposed event detection framework and the obtained results are interpreted and analyzed.

4.1 Performance of object detector

The test videos are evaluated using the fine-tuned object detection model, where the IoU parameter is set to 0.45 and the non-maximum suppression threshold is set to 0.4. The evaluation script is used to compute the intersection of ground truth and the bounding boxes generated by object detector with an IoU threshold of 0.5 for counting the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). These statistics are used in the following equation to compute the machine learning metrics.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{F1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

The evaluation is performed on the NVIDIA GeForce RTX 2080Ti resulted in a processing time of approximately 28 milliseconds per frame. Table 4.1 shows the performance of the fine-tuned object detector for each class of interest.

Table 4.1: Performance of fine-tuned YOLO v8 model

Class	Precision	Recall	F1-score	Accuracy
Patient	0.9869	0.3752	0.5437	0.3747
Door	1.0000	0.3908	0.5620	0.3908
Cardiologist	0.2186	0.0172	0.0318	0.0627
Nurse	0.7543	0.0903	0.1613	0.1030
Sterilize Nurse	0.3154	0.0522	0.0896	0.1266

It can be observed intuitively from the table that the object detector has the highest accuracy in detecting 'Patient' and 'Door' classes, which means that the instances detected by the model are mostly correct and have fewer False Positive detection. However, the lower recall and F1-score for "Patient" compared to "Door" means that the model sometimes fails to detect patient which means have more False Negative detection. This was to be expected, as the test video had frames that the patients obscured by nurses and the Figure 4.1 shows this situation. Patients are more perfectly detected not only because the diversity of patients included in the dataset enhances the generalization of the detector to patients, but more importantly because patients are usually bare-legged and wearing socks. These unique features distinguish them from other personnel in Cath Lab. Similarly, door as an static object not only have macroscopic unique features such as door frames but also contain many instances of opening and closing under different levels of occlusion.

Figure 4.1: The ability of fine-tuned object detector to handle occlusion. Left: Patient can be detected when minor occlusion is present. Right: Patient cannot be detected when severe occlusion is present (FN)

However, the accuracy, recall and F1-scores for 'Cardiologist', 'Nurse' and 'Sterilize nurse' were low which indicates that the model has difficulty in accurately detecting and classifying instances belonging to these categories. For cardiologists and sterilize nurses, the main reason for the detection failure is that they lack unique features to distinguish them and they are heavily occluded most of the time. As shown in the Figure 4.2, both are occluded by the operating table and their faces are blurred, which makes the two objects less distinguishable in the eyes of the detector. However, in fewer cases the detector can distinguish probably because the

higher level features in the network are different. Nevertheless, it is observed that the algorithm can actually distinguish them when the occlusion is not so heavy. Another interesting point is that the results are very good when running the fine-tuned detector on a video of the same day as the video in the dataset which is shown in Figure 4.2. However, the lighting conditions of this video are different from the test video, which means that the detector can have the ability to generalize people into different identities but requires a lot of annotation. 'Nurse' had better detection results than others because the nurses wore green clothing, making it easier for the detector to detect them. Most of the False Negatives were due to nurses walking around in radiation-protective clothing which can be detected as the cardiologists or sterilize nurses. The mean Average Precision (mAP) value was calculated from the average AP value and the result was **75.86%**, which indicates the accuracy of the model in detecting objects of all categories. The result confirms the ability of YOLO v8 to generalize on our dataset, but requires a large dataset to support it. Due to the satisfy result of patient and door which are most important for event detection, no additional time was spent for labeling more instances of the remaining classes.

Figure 4.2: Different performance of the detector when detecting cardiologists and nurses. Left: Reasons for the poor detection results. Right: Good results of the video recorded on the same day as in the train dataset.

Table 4.2 compares the performance of the fine-tuned YOLO v8 model with the object detector proposed by Dai [53]. His model improves the Scaled-YOLOv4 model to detect medical devices in the same Cath lab using a SIoU loss functions and a visual transformer layer. The Scaled-YOLO v4 column is the result of fine-tuned official model and the YOLO-T column is the result of the model with new loss function and transformer layer. His model is designed to detect medical equipment such as monitors, x-ray detectors and receivers, and classifies sterilize nurses and nurses collectively as lab assistant, while doors are not his object of detection. The table clearly shows that except for the cardiologist, the performance of the fine-tuned YOLO v8 model outperformed his detector in the remaining categories. His detector can find the area where the object is located from different camera angles to reduce the influence of occlusion, and his model is trained on video without face blurring. The large number of patient instances and operation bed instances in his model are overlapped, resulting in poor average precision of patient. This is the main reason that why the bed was not labeled as a class of interest in this thesis. In addition, when testing its network on videos with blurred faces, we found that many nurses and cardiologists were confused with each other, which is consistent with the result of

our fine-tuned YOLOv8 model.

Table 4.2: Object detection results (Average Precision) on Cath Lab dataset using fine-tuned YOLO v8, Scaled-YOLO v4, YOLO-T

Class	**Fine-tuned YOLO v8**	**Scaled-YOLO v4**	**YOLO-T**
Patient	0.987	0.564	0.713
Door	1.0000	No Measurement	No Measurement
Cardiologist	0.219	0.545	0.792
Nurse	0.754	0.682	0.688

As for the events related to procedure, the objects are actually different states of the same area. In addition, they are small objects compared to the people and doors in patient enter and exit event. Therefore, the performance of detector on the procedure related event is not based on IoU analysis as it is for enter and exit events. A video of 106990 frames in length containing the entire procedure was selected as test video.

Table 4.3: Performance of detector on Access/Closure event

Class	**Total frame**	**Successful detection**	**Error detection**
ArmNoAccess	23423	21471	832
ArmAccess	22028	18203	15
GroinNoAccess	93420	82608	0
Suture	3611	2710	10935

The Table 4.3 shows the results on this test video with the fine tuned YOLOv8n model. The detector detect 832 of the frames where the access had not created as having created, while only 15 frames detected the access created as no access. Despite the fact that these values vary across different test recordings, the detector is more prone to identify no access state as access state. This may be because the detector learns the features of the hand of cardiologist as well as the features of blood presence or absence. For objects with no access at the groin, the detector can detect them well, but infusion bottles tend to occlude one access point and cause detection failure. It is reassuring that the detector does a good job of distinguishing between arm access points and groin access points and there is no confusion at all between the two detections. However, the suture object has 10935 frames detected incorrectly which is the False Positives in Figure 3.12 of the Method. The same dataset and training settings are used to train Yolov8m, which is a more complex network to achieve higher accuracy. However, due to the small size of the objects, the detection results are not as good as the simplified network Yolov8n. This is a typical obstacle in object detection. Although YOLO v8 use Feature Pyramid Networks (FPN) and the Path Aggregation Network (PAN) in the neck of the network to address this challenge which is discussed in Material part, the performance on our dataset is not

good. In addition, each frame takes 10 milliseconds to process on Yolov8n model and 30 milliseconds on Yolov8m model which indicate that the simpler network can detect small objects faster and more accurately in our dataset.

4.2 Performance of object tracker

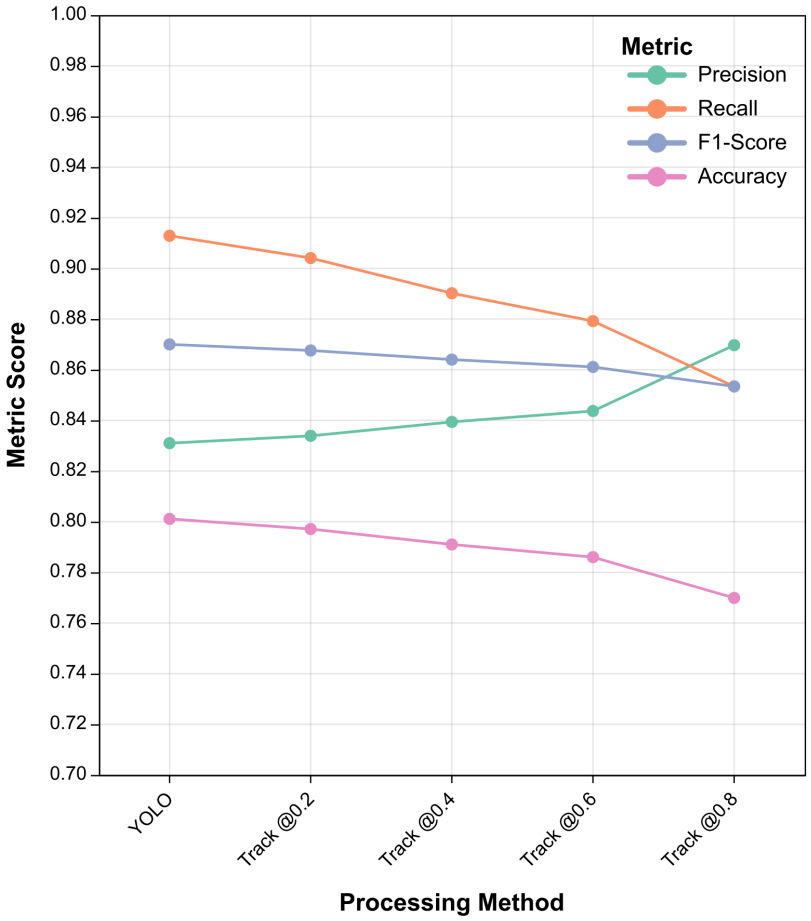

Figure 4.3: Performance comparison of YOLO and tracking algorithm at different confidence thresholds

As a popular field in computer vision tasks, the performance of tracker has many official benchmarks such as the MOT 20 for MOTChallenge [54]. However, these benchmarks are used to evaluate the ability to track and re-identify in crowded scene. In contrast, the number of people in Cath Lab is less and the re-identification of people is largely enabled by the generalization ability of the object detector. In addition, the ID assigned to patient by BYTE often changes during tracking due to severe occlusion. Such changes are more common for other personnel in the Cath Lab because the detector is less accurate to detect them. As described in the Material section 2.4, BYTE performs data association twice and can configured by different association methods which are IoU values or Re-ID feature distance. However, after trying

different combinations of data association, the situation that IDs change frequently did not improve. Other trackers that performed well on re-identification were also experimented and did not improve. However, in our scenario there is only one door and one patient, which makes the ID less important because we know whether the object of interest is still the same object by class. Therefore, when evaluating the performance of the tracker in our project, we are more concerned about the ability of the detector to handle the missing detections as much as possible according to our needs. The object tracking pipeline runs with the same IoU, but uses different thresholds (0.2, 0.4, 0.6, 0.8) when connecting the object detection models. The results of the object detector in the previous subsection are used as a baseline. By changing the processing threshold of the bounding box information, the input to the tracker is changed to see how the performance of the detector affects the tracking results. A comparison of the performance of YOLO and the tracking algorithm at different confidence thresholds is shown in Figure 4.3.

The graph shows that the precision increases as the confidence threshold of the tracker is increased. Increasing the confidence threshold means that the tracker will only consider detections with a higher confidence score. This leads to stricter detection filtering and reduces the number of False Positives. As a result, higher confidence thresholds ensure that only more confident and accurate detections are tracked, resulting in higher precision values. However, since higher confidence thresholds filter out detections with lower confidence scores, it is more likely that some True Positive detections will be missed. Thus it can be observed in the figure that the recall rate decreases as the confidence threshold of the tracker increases. Recall measures the proportion of true positive detections identified by the tracker among all actual positive instances in the ground truth. By increasing the confidence threshold, the tracker becomes more conservative in its detection and may miss some instances, resulting in a lower recall value. The accuracy also drops as the threshold increase for the same reason. Such a result can be explained by the working principle of ByteTrack, where BYTE shines by handling every detected box even if they have a low confidence value. However, the performance of the tracker will be affected by the False Negatives which is reflected in recall. The Table 4.4 shows the tracking results for each class with the same threshold and IoU for the tracker and detector.

Table 4.4: Performance of tracker with same setting as YOLO

Class	**Precision**	**Recall**	**F1-score**
Patient	0.9915	0.3731	0.5421
Door	1.0000	0.3908	0.5620
Cardiologist	0.2828	0.0342	0.0610
Nurse	0.7528	0.0601	0.1113
Sterilize Nurse	0.3543	0.0333	0.0609

Compared to Table 4.1, there is a slight increase in the precision of the 'Patient' class.

The reason for the small magnitude increase in values is because of the large number of patient instances in the video. In order to evaluate the utility of the tracker better, a more detailed investigation was done for the patient entry and exit process. This process occupies a total of 1410 frames in the test video and there are 376 frames that the patient is occluded by the nurse. YOLO can successfully detect 1303 frames, and the tracker can detect 1337 frames under the same setting conditions. This corresponds to an increase of 2.4% on the performance. If the threshold is set to 0.2 it can track up to 1392 frames which corresponds to a 6% increase in performance. It is worth noting that these performance gains come from compensating for occlusion which is our goal with the tracker. The detection result of the door is perfect, so the tracker will not increase the result. The remaining classes have less accurate detection ability, and the trackers can increase the detection performance to a certain extent. Overall, the performance of object detectors is the key to the success of trackers. The good performance of the object detector determines the lower limit of the tracker. On the contrary, the poor performance of the object detector determines the upper limit of the tracker effect.

Figure 4.4: The result of post-processing step GSI augmented tracking. Left: The tracking failed to track patient with severe occlusion. Right: The GSI interpolated bounding box for patient.

A high threshold is applied to make the tracking results of input to the event detection more precise. While it will reduce ability of ByteTrack to handle occlusion, it will contain fewer False Positives and make the bounding box changes less abrupt. The tracker takes about 30 milliseconds to process each frame, adding only 2 milliseconds per frame. The event detection algorithm is built robust enough to use high threshold tracking results as input. The post-processing step to enhance the tracking results is still developed to provide perfect results. The Gaussian smoothing interpolation is performed on the patient and door trajectories as described in the Method. The algorithm will first interpolate the coordinates of the objects in missing frames and then use a regression function to smooth the interpolated results. This method was first proposed by the StrongSORT introduced in the Introduction section and they use it based on the ID assigned to each object. However, IDs change frequently in our use case and cannot be eliminated. Therefore, we took advantage of the fact that Cath Lab has only one door and one patient, and modified the algorithm to process according to the class. The post-processing results are shown in the Figure 4.4. The disadvantage of this post-processing method is that it takes longer

time because it constructs equations relative to the video length for coordinates of the bounding box and perform regression on them. Hence, the longer the video, the longer the post-processing will takes.

The performance of the tracker for access/suture events is essentially the same as for patient entry and exit events. The tracker can influence the result by tracking of True Positive and False Negative instances described in the previous section meaning that by tracking it can make the correct ones more correct and the wrong ones more wrong. Therefore, the threshold is set higher for this event in order to provide less noisy data for the event detection algorithm to process.

4.3 Event detection

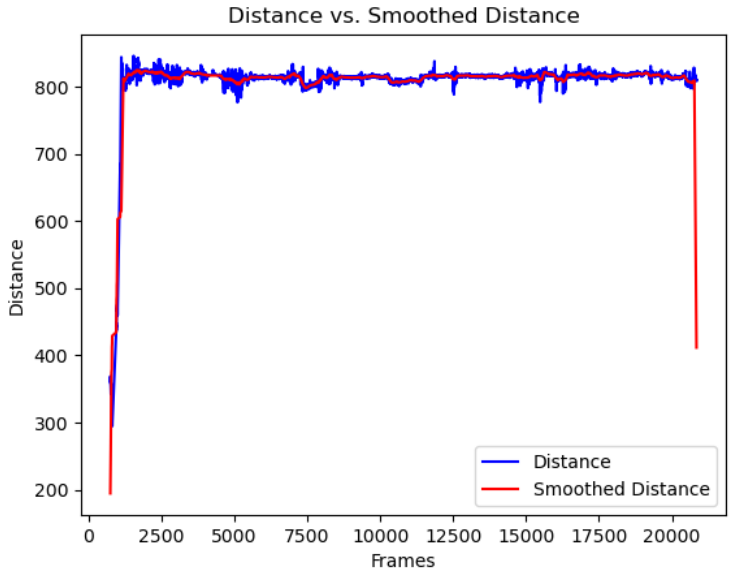

Figure 4.5: Signal processing results for tracking data in event detection algorithm

For patient entry and exit events, the data generated by the tracker is the basis for determining the event. The Figure 4.5 visualizes the design logic of the algorithm. The blue curve is the result from the tracker, and the red curve is the result of filtering the blue curve by the signal processing method described in the method. The ”noise” is actually caused by the constant vibration of the generated bounding box while the patient is lying in bed. In addition, this filtering has the advantage of making the slope of the patient enter and exit more linear, where the non-linearity comes from the sudden loss of detection caused by occlusion. Therefore, the filtering process increasing the robustness of the algorithm. When the distance between the patient and the door increases it means that the patient enters, when the distance decreases it means that the patient exits. When the change in distance stabilizes in a predefined interval over time means the patient is lying down. Another advantage

of the algorithm is that it detects events based on the distance between patient and door, regardless of how many patients may appear in the video. As described in the pseudo code in section 3.5, the distance and intersection are calculated for each frame where both the patient and the door are detected, and the event detected for the corresponding frame is stored in a text file so that the timestamp can be accurately obtained. The Figure 4.6 shows the result of generate text file and the corresponding time stamp.

```
frame: 714, distance: 423.01252032923327, smoothed distance: 192.99116891621912, intersection: 17432.774752, event: None
frame: 715, distance: 421.6395714882677, smoothed distance: 194.84884242828275, intersection: 17965.293176000003, event: None
frame: 716, distance: 413.18231872311526, smoothed distance: 196.65847792197502, intersection: 19597.207318, event: None
frame: 717, distance: 424.41607198744254, smoothed distance: 198.4435951226265, intersection: 17739.073224, event: patient enters
frame: 718, distance: 428.7245361782435, smoothed distance: 200.21219086242715, intersection: 17133.899838000012, event: patient enters
frame: 719, distance: 421.25700519384833, smoothed distance: 201.9748809127992, intersection: 18632.95843499999, event: patient enters
frame: 720, distance: 417.54129484831793, smoothed distance: 203.73163420043488, intersection: 19443.03720000001, event: patient enters
frame: 721, distance: 412.9362790361365, smoothed distance: 205.47714643857233, intersection: 20281.113900000008, event: patient enters
frame: 722, distance: 418.6464872061988, smoothed distance: 207.20179916749433, intersection: 19268.003115, event: patient enters
frame: 723, distance: 417.61274514508057, smoothed distance: 208.9194686004801, intersection: 19318.474127999994, event: patient enters
frame: 724, distance: 418.26047343760564, smoothed distance: 210.62365356137312, intersection: 19274.578511999996, event: patient enters
```

Figure 4.6: Generated text file from event detection algorithm

The result of the event detection algorithm is written in the top left corner of the corresponding frame which is shown in Figure 4.7. The algorithm was performed on multiple videos and the results detected by the algorithm were subjectively correct.

Figure 4.7: The result of event detection. Left: Patient enters. Middle: Patient stay on the table. Right: Patient exits.

As mentioned earlier, the data from the object tracking and the data from the single shot detector are combined and used as the basis for determining access and closure events. The purpose of the event detection algorithm for this event is to determine when an access is established at the same access point based on the change from no access to an access and the correct suture objects are retained. As with the enter and exit events, detection is written to video and saved to text files to obtain the accurate times tamp. The Figure 4.8 shows the different states of the access.

Figure 4.8: Result of detecting whether access is established at which access point. Left: Access is not established and the event is 'None'. Right:Access established and the event is 'Arm Access Established'.

The Figure 4.9 shows that the False Negatives for the suture objects are removed, which can be compared with the Figure 3.12. The algorithm is robust enough to detect patient changes based on whether the procedure has been performed or not, and eliminates False Negatives of sutured objects due to a second patient lying in bed before access established.

Figure 4.9: Result of detecting the suture event. Left: Preserve correct suture objects and correct event detection. Right:False Positive of suture are removed and no event detection.

In general, the detection of access/suture events is not as accurate as patient entry and exit due to the lower performance of the detector. In addition, as described in the Method the algorithm detects 150 consecutive frames of access establishment before determining access established onset. Although this causes a delay of 5 seconds, it enhances the robustness of the algorithm to error detection and it can be negligible compared to a 30-minute procedure. As described in Section 1.2, video-based event detection in the robot-assisted operating room conducted by Sharghi et al. [7] uses the mean Average Precision (mAP) of activity extraction as the performance metric for its framework. However, instead of using the activities that occur in short video clips to represent events, our framework uses a different strategy to detect events. Therefore, it can only be said that the results of object tracking will have a critical impact on the performance of the proposed framework, and we cannot use the performance of the tracker as a substitute for the performance of the whole framework.

Chapter 5

Conclusion

This chapter contains a summary of this project and indicates future research directions for this project. A summary of the project is presented in Section 5.1, and future work is provided in Section 5.2.

5.1 Summary

In summary, this thesis first presents the motivation for this project. The increasing number of patients with cardiovascular disease is making the Cath Lab run overloaded and improving the efficiency of the operating room is imminent. In order to measure KPI for efficiency measurements of Cath Lab , it is necessary to know when patients enter and exit the room as well as when the access established and closed. These data are currently observed and manual registered by the staff in hospital, which is time-consuming, expensive and error-prone. Therefore, we have recorded the activities happened in the Cath Lab of the Reinier de Graaf Groep Hospital in Delft and proposed an event detection framework to automate the detection of these important events. In the introduction, we present research related to event detection in video and describe in detail the development of object detection and object tracking techniques in this framework.

The Material section provides a comprehensive description of the videos used in the project and the video annotation tools used to create dataset. The project uses the YOLO v8 object detector and the components of network are fully explained based on available resources such as papers, official issue communities (Github), and source code reviews since there is no published official article. It also introduces the working principle of the ByteTrack tracker, including the data association method BYTE. The project replaces the YOLOX detector in ByteTrack by the state-of-the-art YOLO v8 to improve performance. In the Method section, the workflow in the target Cath Lab is described. The events of interest and the target objects of the object detector are defined. Detailed steps for the preparation of two event datasets are also provided, emphasizing the importance of labeling logic and dataset balancing. No additional data augmentation is performed on the dataset since built-in augmentation method in YOLO v8 Mosaic augmentation is utilized. By using the new command

line features provided by YOLO v8, the training process is greatly simplified. The connection between trackers and detectors is discussed and algorithms for detecting different events are presented. In addition, problems encountered during the implementation and considerations in the design of the event detection algorithm are discussed. A post-processing method called Gaussian smoothing interpolation is introduced to refine the tracking results which used as input for event detection.

In the Result and Discussion section, performance of the object detector is evaluated by the evaluation metric. The detector shows good performance on detecting patients enter and exit dataset, especially for the target objects patient and door. However, the detection performance is less satisfactory for objects with low distinguishing features. Furthermore, the results of fine-tuned YOLO v8 and results of the object detectors proposed by Dai [53] are compared. It is shown that the fine-tuned model outperforms the scale-YOLOv4 based model except for cardiologist detection. The results of the post-processing step Gaussian smooth interpolation are generally good, but require a longer processing time. Regarding the access and suture detection, the detection is not good enough because the features are less distinct and the objects are smaller in the frame. It was expected to obtain better detection results with a more complex network, but surprisingly the simpler network structure outperformed the complex one. The performance of the tracker was also evaluated, and it was shown to have the ability to handle occlusions on both datasets. Changing the detector input to the tracker by changing the threshold value was found to affect the performance of the tracker. Overall, the performance of object detectors is the determining factor in the success of trackers. The lower limit of the tracker is determined by how well the object detector can perform. On the contrary, the efficacy of the tracker is limited by the poor performance of object detector. Moreover, YOLOv8 is very fast with every frame in the video and the tracker added to the framework is also very fast. Both event detection algorithms were experimented on multiple videos to test robustness and the output of the algorithms is visualized by the figures. Overall, the algorithms can correctly and accurately detect the target events.

5.2 Future work

This is a proof-of-concept project demonstrating the feasibility of using artificial intelligence and computer vision to automatically detect events in the target Cath Lab. The proposed framework can be improved from different modules to achieve better performance, as well as improving the whole framework and exploring ways to apply it to other scenarios. Possible research directions for follow-up are as follows:

- In the proposed framework, the model for object detection is trained only on the Cath Lab dataset in Delft. Therefore, extra annotation and training are required when applying the proposed framework in other labs. In addition, the people in our dataset are processed by face blurring to protect privacy. When applying the proposed framework within a hospital, the performance may be

improved if the detector can be trained on video of non-blurred faces from a specific hospital and recognize each person in the room. For the event detector in the framework, although the patient enter and exit detection may be affected by the room configuration, the detection logic based on distance change can be retained. As for access establish and closure events, the framework can be successfully applied as long as the workflow is the same and the process can be clearly observed in the video.

- The proposed event detection framework only utilizes videos from the south-east and south corners, but we collected videos from five different angles simultaneously. More useful information can be collected from other angles, for example, the enter and exit of cardiologist and sterilize nurse can be detected from the northwest angle. The videos from other camera angles can also be used to enhance the ability of detector handling occlusion or 3D modeling the lab to reconstruct the scene.

- Currently, the event detector analyzes events based on the result from object tracking. The event detector can be implemented in real time when using as hospital clinical applications. The speed of current detection and tracking algorithms makes real-time event detection possible. However, speed and efficiency can be further improved by using the joint detection and tracking model.

- The events detected in this project are based on video with no audio and the audio can be further collected. Moreover, the X-ray images of the interventions are collected with accurate time stamps. From these multi-modal data, we could detect more events that occur in the Cath Lab workflow as shown in the Figure 3.1.

Bibliography

[1] Grant W Reed, Michael L Tushman, Samir R Kapadia, and Congruence Model. Effective operational management in the cardiac catheterization laboratory. *Journal of the American College of Cardiology*, 72(20):2507–2517, 2018. pages 1, 2

[2] Eva Gattnar, Okan Ekinci, Vesselin Detschew, and M Capel-Tunon. Event-based workflow analysis in healthcare. In *IVM/FTMDD/RTSOABIS/MSVVEIS*, pages 61–70, 2011. pages 1

[3] Mithra Vankipuram, Kanav Kahol, Trevor Cohen, and Vimla L Patel. Toward automated workflow analysis and visualization in clinical environments. *Journal of biomedical informatics*, 44(3):432–440, 2011. pages 1

[4] Alexandre RJ Francois, Ram Nevatia, Jerry Hobbs, Robert C Bolles, and John R Smith. Verl: an ontology framework for representing and annotating video events. *IEEE multimedia*, 12(4):76–86, 2005. pages 2

[5] Asaad Hakeem, Yaser Sheikh, and Mubarak Shah. Caseˆ e: a hierarchical event representation for the analysis of videos. In *AAAI*, pages 263–268, 2004. pages 2

[6] Lu Jiang, Alexander G Hauptmann, and Guang Xiang. Leveraging high-level and low-level features for multimedia event detection. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 449–458, 2012. pages 2

[7] Aidean Sharghi, Helene Haugerud, Daniel Oh, and Omid Mohareri. Automatic operating room surgical activity recognition for robot-assisted surgery. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part III 23*, pages 385–395. Springer, 2020. pages 3, 46

[8] Zhicheng Zhao, Xuanchong Li, Xingzhong Du, Qi Chen, Yanyun Zhao, Fei Su, Xiaojun Chang, and Alexander G Hauptmann. A unified framework with a benchmark dataset for surveillance event detection. *Neurocomputing*, 278:62–74, 2018. pages 3

[9] Katsunori Ohnishi, Masatoshi Hidaka, and Tatsuya Harada. Improved dense trajectory with cross streams. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 257–261, 2016. pages 3

[10] Sang Phan, Martin Klinkigt, Vinh-Tiep Nguyen, Tien-Dung Mai, Andreu Girbau Xalabarder, Ryota Hinami, Benjamin Renoust, Thanh Duc Ngo, Minh-Triet Tran, Yuki Watanabe, et al. Nii-hitachi-uit at trecvid 2017. In *TRECVID*, volume 18, 2017. pages 3

[11] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016. pages 3

[12] Qianyu Wu, Aichun Zhu, Ran Cui, Tian Wang, Fangqiang Hu, Yaping Bao, and Hichem Snoussi. Pose-guided inflated 3d convnet for action recognition in videos. *Signal Processing: Image Communication*, 91:116098, 2021. pages 3

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. pages 4, 5, 26

[14] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. pages 4

[15] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. pages 4

[16] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. pages 4

[17] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. pages 4

[18] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. pages 5

[19] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. pages 5

[20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. pages 5, 27

[21] Xingkui Zhu, Shuchang Lyu, Xu Wang, and Qi Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2778–2788, 2021. pages 5

[22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022. pages 6, 14

[23] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *ACM transactions on Intelligent Systems and Technology (TIST)*, 4(4):1–48, 2013. pages 7

[24] Hsu-kuang Chiu, Jie Li, Rareş Ambruş, and Jeannette Bohg. Probabilistic 3d multi-modal, multi-object tracking for autonomous driving. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14227–14233. IEEE, 2021. pages 8

[25] Akshay Mangawati, Mohammed Leesan, HV Ravish Aradhya, et al. Object tracking algorithms for video surveillance applications. In *2018 International Conference on Communication and Signal Processing (ICCSP)*, pages 0667–0671. IEEE, 2018. pages 8

[26] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015. pages 8

[27] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016. pages 9

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. pages 9

[29] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. pages 9

[30] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. pages 9

[31] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017. pages 9

[32] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 2023. pages 10, 11

[33] Jiyang Gao and Ram Nevatia. Revisiting temporal modeling for video-based person reid. *arXiv preprint arXiv:1805.02104*, 2018. pages 10

[34] Rui-Yang Ju and Weiming Cai. Fracture detection in pediatric wrist trauma x-ray images using yolov8 algorithm. *arXiv preprint arXiv:2304.05071*, 2023. pages 14, 16, 17

[35] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020. pages 14

[36] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. pages 14

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. pages 15

[38] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *International Workshop on Deep Learning in Medical Image Analysis, International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 179–187. Springer, 2016. pages 15

[39] Wenqi Ren, Si Liu, Hua Zhang, Jinshan Pan, Xiaochun Cao, and Ming-Hsuan Yang. Single image dehazing via multi-scale convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 154–169. Springer, 2016. pages 15

[40] Diogo Duarte, Francesco Nex, Norman Kerle, and George Vosselman. Multi-resolution feature fusion for image classification of building damages with convolutional neural networks. *Remote sensing*, 10(10):1636, 2018. pages 15

[41] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. pages 15

[42] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. pages 15

[43] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586, 2021. pages 17

[44] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer, 2022. pages 17, 18

[45] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. pages 20

[46] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. pages 26

[47] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. pages 26

[48] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *2017 IEEE international conference on computer vision (ICCV)*, pages 3544–3553. IEEE, 2017. pages 26

[49] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. pages 26

[50] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. pages 26

[51] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020. pages 26

[52] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. pages 26

[53] Renjie Dai. Detecting medical equipment in the catheterization laboratory using computer vision. 2022. pages 39, 48

[54] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020. pages 41