2023

Authors:

Alexandru Gumaniuc

Jadyn Jacques

Ayodeji Olagoke

Rareș Matei Husărescu

Krystian Wiazowski

Ruike Yuan

Advisory Report

MINA – IT2D

VERSION 1.2

DIYALO

Table of Contents

Version Management

Version number	Changes
1.0	Initial version of the document
1.1	Added Advice for detection module
1.2	Added Advice for tracking module

Chapter 1: Introduction

This document summarizes the requirements gathered for the project, stated by Diyalo Foundation. The subject is autonomous robot – Mina, that is supposed to navigate the environment with the use of AI Human tracking. Furthermore, the document will include an advisory report regarding the choice of coding languages and solutions. In addition to the work done by the group, information on the problems that weren't solved or can occur in the future will be provided.

The MoSCoW method is a way of prioritizing the objectives for the project in a manner from least important to most important. In the following order: Must, Should, Could and Won't have.

This document will describe in detail:

- Features according to MoSCoW method,
- Requirements to implement the features listed.

Chapter 2: MoSCoW Prioritization

Must have	• Human recognition • Human tracking
Should have	• Movement system
Could have	• Specific person recognition/Siamese Model
Won't have	• Gesture recognition

Chapter 3: Dependent frameworks and terms

The project will be delivered as an ROS package that can be embedded into the actual robot. To sufficiently test and implement the package into the hardware, or to improve the performance of the robot, detailed information of the system will be described in this chapter, which includes the use of the framework and software for building the package and the system dependencies required for testing and deployment.

Linux Ubuntu (Focal Fossa version 20.04)

There are a lot of reasons for selecting Linux as the operation system for testing and running the simulation for the project. Tons of robotic projects are tested on Linux based on its open-source nature, flexibility and modularity, wide range of hardware support, and real time capabilities (for motion control or sensor feedback).

The project is based on the tested initial ROS package given by the clients, which includes the 3D models of the robot and the simulation environment built for testing. To make the follow-up work compatible with the simulation environment provided, the team decided to use the operating system that clients suggested (Focal Fossa version 20.04).

ROS

ROS is an open-source meta operating system for your robot. It provides the services of an operating system, including hardware abstraction, low-level device control, implementation of commonly used functions, message passing between processes, and package management. It also provides tools and libraries for obtaining, creating, writing, and running code on multiple computers.

As the project is tested in the simulation environment. The chosen robot framework must also be compatible with the environment. ROS integrates well with simulation environments such as Gazebo, allowing developers to test and validate their algorithms in a simulated environment before deploying them on real robots.

The project uses the framework ROS, to connect the input (data from sensors) and the output (command or action for the robot) via ROS topics and messages. The command for the robot is based on the processed data from the Object-Detection framework and the algorithms written in Python to control the movement of the robot.

ROS Noetic

The choice of ROS (Robot Operating System) version and the corresponding Ubuntu distribution depends on various factors, including compatibility, stability, and community support. Choosing ROS Noetic with Ubuntu 20.04 provides a longer period of support and maintenance, as ROS noetic will be supported till 2025, unlike ROS Melodic in Ubuntu 18.04 for which the supports end in 2023.

Gazebo

Gazebo was chosen as the main simulation platform for the project because the original simulation environment provided by the customers was created in Gazebo. Gazebo is a powerful robot simulation environment that allows developers to simulate robots and their

interactions with the environment. It enables testing and validation of algorithms and control strategies before they are deployed on real hardware, reducing costs and development time.

Python 3

Choosing Python 3 as the programming language for various aspects for the project, including robotics and computer vision, offers several advantages:

- Integration capabilities: Python's versatility and integration capabilities make it well-suited for integrating ROS with other tools and frameworks. Python can easily interface with C/C++ code, allowing for performance-critical sections to be implemented in lower-level languages while using Python for high-level control and application logic. Python also integrates well with machine learning frameworks like TensorFlow and PyTorch, enabling seamless integration of AI capabilities in ROS projects.
- ROS support: ROS provides extensive support for Python, making it one of the officially supported programming languages. ROS provides Python bindings (rospy) that allow developers to interact with ROS nodes, topics, services, and other ROS components seamlessly. The ROS ecosystem has a wide range of Python libraries and tools that enable developers to build and integrate components easily.

OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a comprehensive set of tools and functions that enable developers to build applications and systems for image and video processing, object detection and tracking, feature extraction, and more. Since this library is widely used and open-sourced, and it can be easily managed and used in Python 3, OpenCV is chosen as the library for processing the images captured by the real-sense camera.

Object detection and image segmentation model

The team has tested several ways for human tracking and facial recognition. From the basic cascade model to the Single Shot MultiBox (SSD) model and the popular object detection algorithm YOLO (You Only Look Once).

YOLOv8

YOLOv8 is the latest version of YOLO by Ultralytics. As a cutting-edge, state-of-the-art (SOTA) model, YOLOv8 builds on the success of previous versions, introducing new features and improvements for enhanced performance, flexibility, and efficiency. YOLOv8 supports a full range of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. This versatility allows users to leverage YOLOv8's capabilities across diverse applications and domains.

The team selected the latest version of YOLO(YOLOv8) as the model to be deployed into the system, which will serve as the main model for facial recognition and object tracking.

Chapter 4: Advice for detection modules

4.1 Comparatives of different real-time object detection frameworks

Real-time Object Detection is the process of detecting objects in images or video frames. This can be done using various methods, but two of the most popular are SSD and YOLO. Both methods have pros and cons, so let's check out the Difference between SSD & YOLO.

YOLO (You Only Look Once) system, an open-source method of object detection that can recognize objects in images and videos swiftly whereas SSD (Single Shot Detector) runs a convolutional network on input image only one time and computes a feature map. SSD is a better option as it can be run on a video and the exact trade-off is very modest.

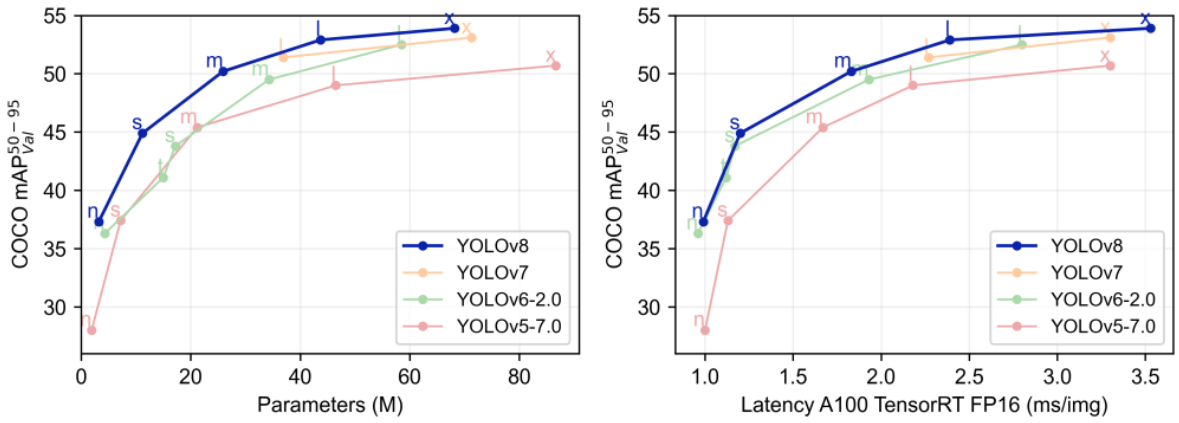
SSD is a healthier recommendation. However, if exactness is not too much of disquiet but you want to go super quick, YOLO will be the best way to move forward. First, a visual thoughtfulness of swiftness vs precision trade-off would differentiate them well.

While dealing with large sizes, SSD seems to perform well, but when we look at the accuracy numbers when the object size is small, the performance dips a bit.

SSD	**YOLO**
The full Form of SSD is Single Shot Detector	The full form of YOLO is You Only Look Once
It takes the input photos and performs a single pass through a convolutional network, generating a feature map.	The open-source method of object recognition can quickly identify subjects in static images and moving footage.
SSD network could be a better alternative because we can run it on a video, and the real trade-off is very small. This makes the SSD a very viable option.	When exactness is not a significant source of disquiet, yet you still want to go very quickly, YOLO is a better choice.
When the object size is very small, there is a slight decrease in performance.	YOLO can be the better option, even if the object in question is relatively small.

4.2 Comparatives of different versions of Yolo

YOLOv8 vs YOLOv7 vs YOLOv6 vs YOLOv5
Right away, YOLOv8 models seem to perform much better compared to the previous YOLO models. Not only YOLOv5 models, YOLOv8 is ahead of the curve against YOLOv7 and YOLOv6 models also.



YOLOv8 compared with other YOLO models.
When compared with other YOLO models trained at 640 image resolution, all the YOLOv8 models have better throughput with a similar number of parameters.

Now, it is important to see a detailed view of how the latest YOLOv8 models perform in comparison with the YOLOv5 models from Ultralytics. The following tables show a comprehensive comparison between YOLOv8 and YOLOv5.

Performance Comparison of YOLOv8 vs YOLOv5

Model Size	Detection#	Segmentation#	Classification*
Nano	+33.21%	+32.97%	+3.10%
Small	+20.05%	+18.62%	+1.12%
Medium	+10.57%	+10.89%	+0.66%
Large	+7.96%	+6.73%	0.00%
Xtra Large	+6.31%	+5.33%	-0.76%

#Image Size = 640 *Image Size = 224

YOLOv8 models compared with YOLOv5 models.

Object Detection Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	28	37.3	+33.21%
Small	37.4	44.9	+20.05%
Medium	45.4	50.2	+10.57%
Large	49	52.9	+7.96%
Xtra Large	50.7	53.9	+6.31%

*Image Size = 640

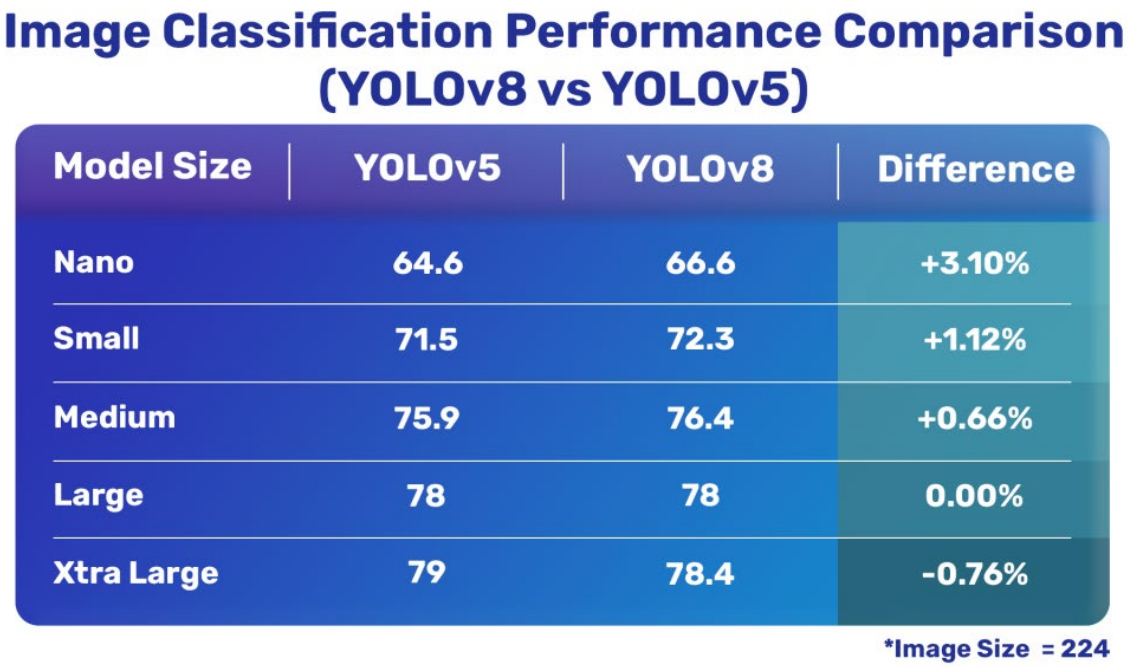
YOLOv8 vs YOLOv5 object detection models.

Instance Segmentation Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	27.6	36.7	+32.97%
Small	37.6	44.6	+18.62%
Medium	45	49.9	+10.89%
Large	49	52.3	+6.73%
Xtra Large	50.7	53.4	+5.33%

*Image Size = 640

YOLOv8 vs YOLOv5 instance segmentation models.



YOLOv8 vs YOLOv5 image classification models.

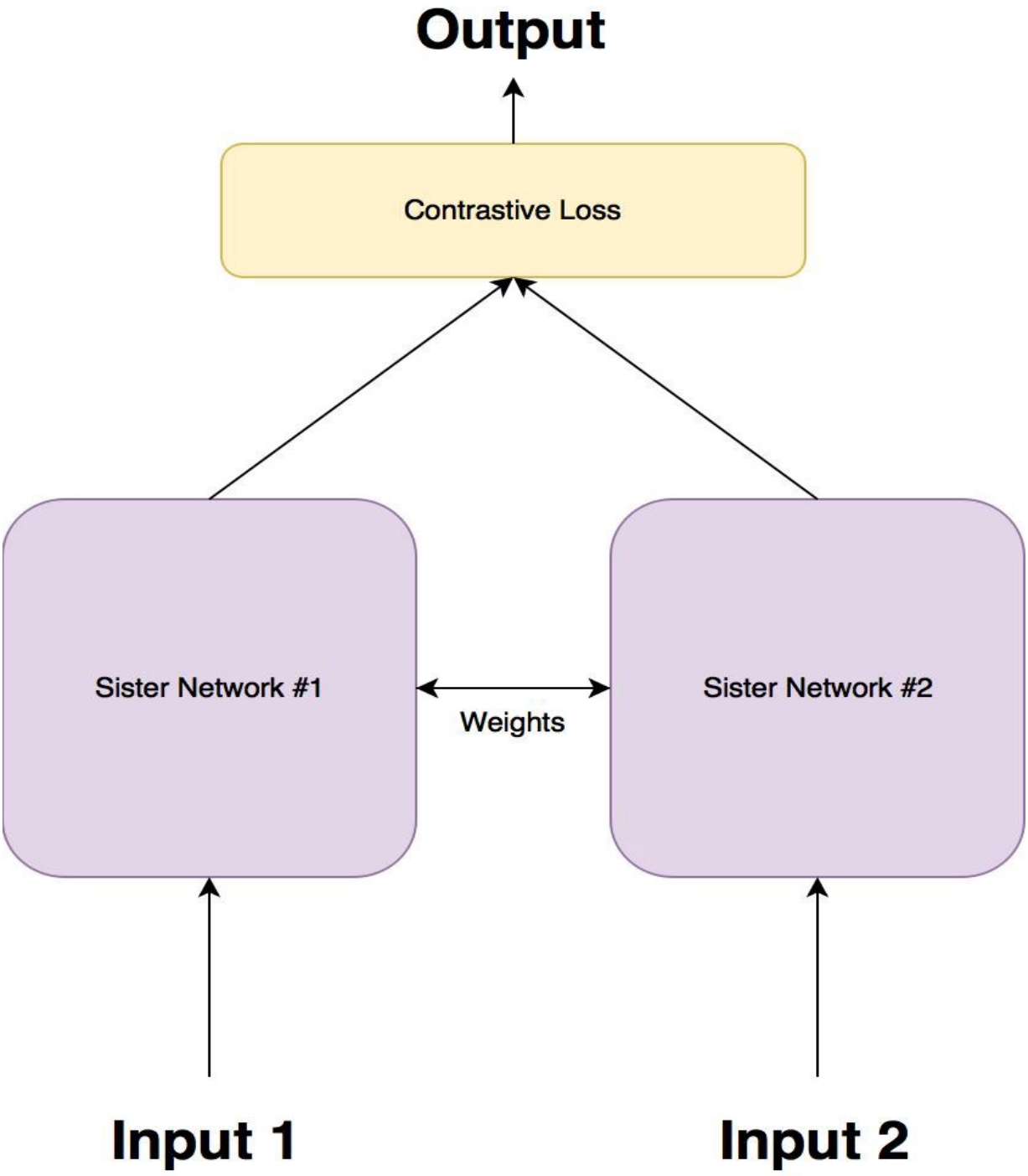
It is clear that the latest YOLOv8 models are much better compared to YOLOv5 except for one of the classification models.

4.3 Siamese model

With Yolo or SSD, it is possible to detect a human object but not a specific person. To achieve the goal of identifying a specific person in a frame to track, some other methods need to be introduced. The team considered to use Siamese model for training data to detect a person with a specific facial feature (e.g., detecting Rob instead of Jack, though Rob and Jack are people). Data will be integrated with the real-time object detection framework like YOLO or SSD (Utilize the YOLO detection head to perform object detection based on the feature representations obtained from the Siamese comparison).

- **Pros of Siamese Network**
  - Siamese network is a one-shot classification model and can perform prediction with just a single training example.
  - More robust to class imbalance as it requires very little information. It can be used on a dataset where very few examples exist for some classes.
  - The one-shot learning feature of the Siamese network does not rely upon domain-specific knowledge but exploits deep learning techniques.

- **Cons of Siamese Network**
    - **Outputs only Similarity score and not probabilities**.: Probabilities of mutually exclusive events sum to 1. whereas distances are not constrained to be less than or equal to 1.



* They consist of two sub-models whose outputs are evaluated using contrastive loss. Contrastive loss measures how well the network can distinguish between two different pairs of images.

The main reason for using the Siam model for person recognition is that it requires fewer data resources, since it only needs to recognize the facial feature to identify a particular person. Also, the Siam model is designed to be trained and recognized based on the similarities of the inputs, which is ideal for face recognition.

However, Siamese model does not follow the GDPR data protection principals (Individual right for the use of data and protection of personal data), and there is a tracking framework that can be well integrated with the Yolov8 detection module, follow the GDPR principal and identify the individuals smartly by getting the ID of an object. This tracking framework will be described in the chapter Advice for Tracking module.

Chapter 5: Advice for Tacking module

5.1 Introduction to Tracking

Tracking in Deep Learning is about predicting the positions of objects in a video based on their spatial and temporal features. Technically, tracking consists of obtaining the initial detections, assigning them unique IDs, and tracking them across all frames of the video while maintaining the assigned IDs. Tracking is generally a two-step process:

- A target localization detection module: the module responsible for detecting and localizing the object in the frame, using an object detector such as YOLOv4, CenterNet, etc. (For more information about the choice of detection module please visit the chapter Advice for Detection module)

- Motion prediction module: This module is responsible for predicting the future motion of the object based on its past information.

5.2 Need for Tracker

A few questions come to mind. Why does an object tracker need to be implemented? Why cannot just to use an object detector? Well, there are many reasons why a tracker is needed.

Tracking when object detection fails there are many cases where an object detector can fail. But if an object tracker is used, it can still predict the objects in the image. For example, take a video where a motorcycle is driving through the forest, and a detector can be applied to detect the motorcycle.

If the motorcycle is obscured or overlapped by a tree, the detector fails. But if there is a tracker, the motorcycle we can still be predicted and tracked.

- **ID assignment:** While using a detector, it only showcases the location of the objects, if we just look at the array of outputs, which coordinates belong to which box will not

be known. On the other hand, A tracker assigns an ID to each object it tracks and maintains that ID till the lifetime of that object in that frame.

- **Real-time predictions:** Trackers are very fast and generally faster than detectors. Because of this property, Trackers can be used in real-time scenarios and has many applications in the real world.

5.3 Types of Trackers

Trackers can be classified based on many categories like methods of tracking or the number of objects to be tracked. In this section, different tracker types with some examples will be discussed.

- **Single and Multiple Object Trackers**
  - Single Object Tracker:

These types of trackers track only a single object even if there are many other objects present in the frame. They work by first initializing the location of the object in the first frame, and then tracking it throughout the sequence of frames. These types of tracking methods are very fast. Some of them are CSRT, KCF, and many more which are built using Traditional computer vision. However, deep dearning based trackers are now proved to be far more accurate than traditional trackers. For example, SiamRPN and GOTURN are examples of deep learning based single object trackers.

  - Multiple Object Tracker:

These types of trackers can track multiple objects present in a frame. Multiple object trackers or MOTs are trained on a large amount of data, unlike traditional trackers. Therefore, they are proved to be more accurate as they can track multiple objects and even of different classes at the same time while maintaining high speed. Some of the algorithms include DeepSORT, JDE, and CenterTrack which are very powerful algorithms and handle most of the challenges faced by trackers.

- **Tracking by Detection and without Detection**
  - Tracking by Detection:

The type of tracking algorithm where the object detector detects the objects in the frames and then performs data association across frames to generate trajectories hence tracking the object. These types of algorithms help in tracking multiple objects and tracking new objects introduced in the frame. Most importantly, they help track objects even if the object detection fails.

- Tracking without Detection:

The type of tracking algorithm where the coordinates of the object are manually initialized and then the object is tracked in further frames. This type is mostly used in traditional computer vision algorithms as discussed earlier.

5.4 Introduction to Sort and Deep Sort

DeepSORT is a computer vision tracking algorithm for tracking objects while assigning an ID to each object. DeepSORT is an extension of the SORT (Simple Online Realtime Tracking) algorithm. DeepSORT introduces deep learning into the SORT algorithm by adding an appearance descriptor to reduce identity switches, hence making tracking more efficient. To understand DeepSORT, How the SORT algorithm works should be understood.

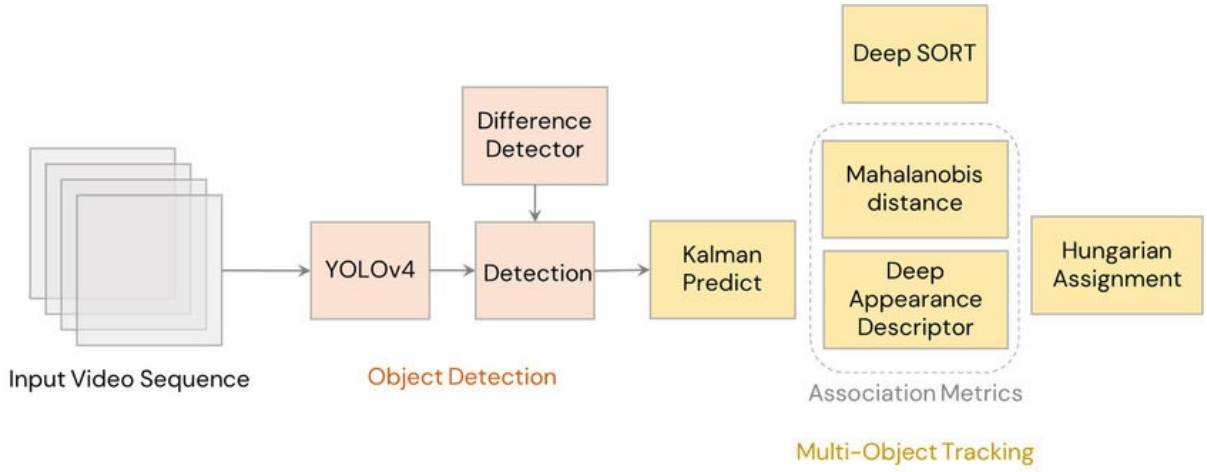
Simple Online Realtime Tracking (SORT)

SORT is an approach to Object tracking where rudimentary approaches like Kalman filters and Hungarian algorithms are used to track objects and claim to be better than many online trackers. SORT is made of 4 key components which are as follows:

- **Detection:** This is the first step in the tracking module. In this step, an object detector detects the objects in the frame that are to be tracked. These detections are then passed on to the next step. Detectors like FrRCNN, YOLO, and more are most frequently used (see chapter Advice for Detection module).
- **Estimation:** In this step, we propagate the detections from the current frame to the next which is estimating the position of the target in the next frame using a constant velocity model. When a detection is associated with a target, the detected bounding box is used to update the target state where the velocity components are optimally solved via the Kalman filter framework.
- **Data association:** Target bounding box and the detected bounding box are now introduced. So, a cost matrix is computed as the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing targets. The assignment is solved optimally using the Hungarian algorithm. If the IOU of detection and target is less than a certain threshold value called IOUmin then that assignment is rejected. This technique solves the occlusion problem and helps maintain the IDs.
- **Creation and Deletion of Track Identities:** This module is responsible for the creation and deletion of IDs. Unique identities are created and destroyed according to the IOUmin. If the overlap of detection and target is less than IOUmin then it signifies the untracked object. Tracks are terminated if they are not detected for TLost frames, you can specify what the amount of frame should be for TLost. Should an object reappear, tracking will implicitly resume under a new identity.

The objects can be successfully tracked using SORT algorithms beating many State-of-the-art algorithms. The detector gives us detections, Kalman filters give us tracks and the Hungarian algorithm performs data association. So, why the DeepSORT is even needed?

5.5 Reason For using DeepSort



SORT performs very well in terms of precision and accuracy of tracking. However, SORT provides tracks with a high number of ID and fails in occlusions. This is due to the association matrix used. DeepSORT uses a better association metric that combines both motion and appearance descriptors. DeepSORT can be defined as a tracking algorithm that tracks objects based not only on the speed and motion of the object, but also on the appearance of the object.

For the above purposes, a well-discriminating feature embedding network is trained offline before tracking is implemented. The network is trained using a large re-identification dataset so that it is suitable for the tracking context. To train the deep metric association model in DeepSORT, the cosine metric approach is used. According to the DeepSORT paper, "Cosine distance accounts for appearance information that is particularly useful for recovering identities after long-term occlusions when movements are less distinctive."

That means cosine distance is a metric that helps the model recover identities in case of long-term occlusion and motion estimation also fails. Using these simple things can make the tracker even more powerful and accurate.

Chapter 6: Future advice

6.1 Improve adaptability to different environments.

Adaptability to different environments: Consider the various environments in which the robot may operate, such as crowded areas, narrow corridors, or uneven surfaces. Ensure the robot's navigation system is versatile enough to handle these different conditions effectively. Test and calibrate the robot's sensors and algorithms in various scenarios to ensure reliable performance.

6.2 Design an Intuitive User Interface and Control panel

Develop a user-friendly interface for human operators to interact with robots effectively. Prioritize simplicity, clarity, and ease of use to facilitate intuitive control and monitoring.

6.3 Improve Data Processing and Analysis

Real-Time Processing: Implement efficient real-time data processing algorithms to analyze and interpret the data collected by the robot. Prioritize low-latency processing to enable timely responses to detected events or anomalies.

6.4 Data Fusion

Integrate data from multiple sensors to create a comprehensive situational awareness for the robot. Develop algorithms that can fuse data from different sources, such as cameras and LiDAR, to enhance tracking accuracy and reduce false positives.

Chapter 7: Instruction Advice for Running the Project

Configuration and instruction for implementing the project.

Set up the simulation environment:

- Install Ubuntu
  Ubuntu should be installed prior to simulation environment for testing.
  There are two ways for the ubuntu installation, either to install it as a separate operating system or install it as a virtual machine.

Ubuntu for installation:
https://releases.ubuntu.com/focal/

- Install Ros
  Ros noetic should be configured and installed after the installation of Linux Ubuntu 20.04.
  Instructions for installing the ROS noetic:
  https://wiki.ros.org/noetic/Installation/Ubuntu

- Configure Gazebo
  The necessary dependencies and packages like rosdep and gazebo_ros_pkgs should be installed as well. Also, Gazebo needs to be tested after configuration.
  Follow each step in the following link to install packages and to configure and test Gazebo:
  https://classic.gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros

- To Create a catkin workspace for testing.
  After installing ROS and configuring Gazebo and related packages. A catkin workspace has to be set up. A catkin workspace is a folder to modify, build, and. install Ros packages, the python codes for the project will be included in the workspace as well.

To catkin workspace setup, try the following commands:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
cd ~/catkin_ws
catkin_make
```

- Clone the repositories for the robot as the catkin workspace to run.
  Repositories of the project:
  https://gitlab.com/nhlstendenIT/project-6.2/diyalo/mina-autonomous-service-data-driven-robot
  Run catkin_init_workspace command in the catkin_ws/src folder to initial the workspace.

Run the catkin_make or catkin build to set up the workspace.

- Advice for debugging:
  If there are errors occurring when running catkin_make and catkin build due to the ram space and speed of execution, try to run the following command:

  catkin build -j1

- Run the project by typing a few commands.
  Make sure the commands are executed in the root folder of the workspace(mina_ws)

Chapter 8: References

Mina – autonomous data driven service robot.

Osrf. (n.d.). *Gazebo : Tutorial : Installing gazebo_ros_pkgs (ROS 1)*. https://classic.gazebosim.org/tutorials?tut=ros_installing&cat=connect_ros

noetic/Installation/Ubuntu - ROS Wiki. (n.d.). https://wiki.ros.org/noetic/Installation/Ubuntu

Canonical. (n.d.). *Ubuntu 20.04.6 LTS (Focal Fossa)*. https://releases.ubuntu.com/focal/ https://releases.ubuntu.com/focal/

Rath, S., & Rath, S. (2023). YOLOv8 Ultralytics: State-of-the-Art YOLO Models. *LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Examples and Tutorials*. https://learnopencv.com/ultralytics-yolov8/

Technostacks. (2023). YOLO Vs. SSD: Choice of a Precise Object Detection Method. *Technostacks Infotech*. https://technostacks.com/blog/yolo-vs-ssd/

Residentmario. (2019). Siamese networks. *Kaggle*. https://www.kaggle.com/code/residentmario/siamese-networks

Khandelwal, R. (2021, December 28). One-Shot Learning With Siamese Network - The Startup - Medium. *Medium*. https://medium.com/swlh/one-shot-learning-with-siamese-network-1c7404c35fda

Sanyam, & Sanyam. (2022). Understanding Multiple Object Tracking using DeepSORT. *LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Examples and Tutorials*. https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/