Introducing object detection to EKF SLAM on a Nao-robot football field

Angela Madelon Bernardy

Layout: typeset by the author using LaTeX.
Cover illustration: Lex Bolt

Bachelor Informatica

University of Amsterdam

Introducing object detection to EKF SLAM on a Nao-robot football field

Angela Madelon Bernardy

June 24, 2024

Supervisor(s): Dr. A. Visser

Signed: Drs. T.R. Walstra

Abstract

Robot football is a complex and dynamic environment, causing localisation to be a challenging task. This thesis aims to improve the localisation of a robot during robot football. Therefore, the context of this thesis is the standard platform league, a robot competition that uses Nao robots. The goal is to research how SLAM can be implemented on a Nao robot to improve localisation. SLAM is chosen instead of only localisation because maps can have several practical applications in robotics, such as tracking other objects and path planning. A stable map could also improve the robustness of robot localisation over multiple environments. These advantages are why the focus lies both on improving robot localisation and on improving the map of the environment.

A classic EKF filter will be used for the back end of the SLAM algorithm. Different versions of the EKF filter, such as FEJ-EKF SLAM, are created to see if these changes make the filter more accurate. Other additions to the extended Kalman filter were the *minimal view* filter, where a landmark is only added to the map if it is seen n times in a similar location, and filtering out measurements by rejecting the measurement when it is too far from the predicted measurement. These back ends are tested on the Gutmann dataset. Although the average robot and landmark localisation errors are the smallest on a minimal view filter with $n = 3$, the FEJ extended Kalman filter showed great potential in stabilising itself after large peaks in localisation errors.

The front end has two versions: a colour filter and YOLO. The colour filter proposed has a precision of 97.4% and a recall of 65.8% on the validation dataset. A small and a medium-sized YOLOv8 model have been tested on the same set. The small-sized model reached a precision of 92.9% and a recall of 98.3%. The medium-sized model reached a precision of 94.4% and a recall of 99.2%.

As a final test, SLAM is performed on a dataset recorded on a Nao on a robot football field, with the back ends and front ends previously tested. Both YOLO models outperformed the colour filter. YOLOv8s was more accurate regarding robot localisation, while YOLOv8m was more accurate regarding landmark localisation. Based on the results, it is most likely that YOLOv8m would also perform better regarding robot localisation on longer datasets. The FEJ-EKF back-end achieved the best results. The mean robot localisation error was 0.17 metres when this back end was combined with YOLOv8s. The mean landmark localisation error was 0.21 metres when this back end was combined with YOLOv8m. This thesis concludes that certain additions to the EKF filter, such as filtering measurements or decreasing the overconfidence of the filter, significantly impact the performance of SLAM on Nao robots. Next to improvements in the back end, improvements in the front end, such as using state-of-the-art object detection models, also improve the accuracy and robustness of SLAM.

Contents

CHAPTER 1

Introduction

In most human football matches, it is uncommon for a player to be lost or walk off the field. However, the opposite is true in robot football, where not getting lost proves to be the greater challenge. During robot football matches, Nao robots often walk off the field or score their own goal because they incorrectly estimate where they are on the map. This thesis aims to improve the localisation of these Nao robots in the Standard Platform League by using localisation and mapping techniques.

The Standard Platform League is a robot football competition where two teams of five or seven Nao robots compete against each other. Aldebaran United Robotics Group designs these humanoid robots equipped with sensors such as cameras, sonars, and microphones[1]. Figure 1.1 shows two Nao robots during a football match.

Next to improving the robot's localisation, this thesis also focuses on creating an accurate

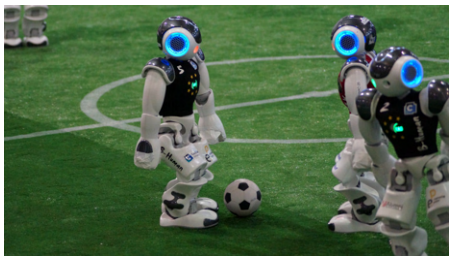


Figure 1.1: Two Naov6 robots playing football in the standard platform league. Source: [1]

map of the environment. A map of the environment would allow a robot to track objects, create paths, and make its location more reliable, even in slightly different environments. SLAM, simultaneous localisation and mapping, is a term that describes algorithms that perform localisation and mapping at the same time. The *extended Kalman filter* is a classical algorithm and forms the basics of many SLAM systems, some of which even have state-of-the-art performance [2]. The scope of this thesis is thus to implement SLAM based on an extended Kalman filter in the context of the Standard Platform League.

The relevance of this research lies in its context. SLAM based on extended Kalman filters has already been extensively studied [2]. However, the context of robot football provides a set of challenges, making solving SLAM in this environment an interesting topic. First, the Nao robots have some technical challenges. The robots have legs instead of wheels, causing the odometry information to be less reliable [3]. Another challenge of Nao robots is their limited processing power, meaning the algorithms running on the robot must be efficient. Secondly, robot football is a dynamic environment with many changing elements, such as other robots or the ball. These

[1] https://unitedrobotics.group/en/robots/nao

dynamic elements make determining the robot's velocity difficult because the different objects' velocities are unknown. Next, the field is symmetrical, causing no landmark to be unique. For example, the field's top-left and bottom-right corners can look identical. The field also might have structural defects, such as holes or bumps, making the robot's walk unstable.
Furthermore, the Nao robot only has monocular cameras. Since the field of view of these cameras does not overlap, these cannot be combined to retrieve depth information. The lack of depth information causes SLAM in this context to be monocular visual SLAM, which is more challenging to solve due to the lack of depth measurements [4]. Considering the aspects of computer vision in this context further highlights the relevance of this thesis. Beghdadi and Mallem [5] denote the substantial link between visual SLAM developments and computer vision developments. Computer vision in visual SLAM can help detect landmarks and add context to the SLAM algorithm, such as which objects in the image are most likely dynamic. YOLOv8, *You Only Look Once* version 8, is an object detection model which can detect objects in real-time [6]. This research uses this state-of-the-art image detection and segmentation model to recognise and measure landmarks. Thus, this research is relevant since it considers a challenging and unique environment and uses state-of-the-art computer vision, contributing to the knowledge of how these can best be combined. Although this thesis does not consider the dynamic environment and symmetrical aspects of the field, this research does provide a stepping stone to implement these aspects into SLAM as well.

This thesis aims to determine how YOLOv8 and additions to the extended Kalman filter impact SLAM on Nao robots by testing different versions of YOLOv8 and various versions of the Kalman filter on a dataset recorded on a Nao robot. A colour filter will also be tested on this dataset to determine the impact of YOLOv8 on SLAM compared to more classic methods. The performance will be measured in terms of mapping and robot location accuracy, using a dataset recorded on a Nao robot with the same landmarks as used by Gutmann and Fox [7]. The research question of this thesis is *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*. The following two sub-questions will aim to answer this research question: *How do different versions of the Extended Kalman filter impact the performance of SLAM in the context of robot football?*, and: *How do YOLOv8s and YOLOv8m improve the performance of EKF-based SLAM compared to a classic colour filter?*. The first sub-question corresponds with the objective of testing additions to the extended Kalman filter. The second sub-question corresponds with testing different methods to handle the visual data. This question has been split to handle both a small and medium-sized model of YOLOv8. This comparison will give insightful information for implementing YOLO into robotics since robots' processing power is often limited.

This thesis will start with the theoretical background, which explains the algorithms used later, such as extended Kalman Filter SLAM, YOLO and colour filters. The methods section is the next chapter, which details the setup and execution of each experiment. The experiments chapter will then contain the results of the experiments described in the methods chapter. The conclusion will answer the research questions and sub-questions. Finally, the discussion chapter will conclude this thesis by interpreting the previously gathered results, reflecting on the research and proposing possible future research topics.

CHAPTER 2

Theoretical background

This section will go over the algorithms used in this thesis. Because the research question is *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*, both YOLO, EKF-SLAM and colour filters, will be explained. First, this section will review the extended Kalman Filter SLAM, including all the models needed to create this algorithm and discuss one possible alteration. Secondly, the methods for feature extraction will be discussed: YOLO and the colour filter.

2.1 Simultaneous localisation and mapping

In *SLAM, Simultaneous Localisation and Mapping*, the robot tries to create a map and place itself on this map simultaneously. Cadena et al. [2] states that this technique is needed if a globally consistent map is required. A globally consistent map is a map where the spatial relationship between features is consistently maintained. A map can be needed for several reasons, such as path planning and registering observations. Registering observations on the map, when and where objects are observed, allows for intuitively logging of the localisation during a robot football match. Path planning in robot football has a vital role: to go to the goal or the ball while avoiding obstacles. Creating a map might not be needed if the map can be known beforehand and is always consistent. It can be argued that this deems SLAM practical for robot football: the map can be known but is symmetric and changes due to moving objects. Although the floors of football fields are similar, the rest of the environment can differ.

The development of SLAM can broadly be divided into two eras: *the classic age* (1986 - 2004) and *the algorithmic-analysis age* (2004-2015). In the first, the mathematical foundations were discovered for SLAM, including the *Extended Kalman Filter*, which this section will discuss later. The second era focused on studying the properties of SLAM, such as its consistency. The foundations of visual SLAM were also laid in this era [2]. Beghdadi and Mallem [5] added one more era, from 2014 until now, in which learning improves and stabilises SLAM. Currently, SLAM consists of three modules: *initialisation*, *localisation* and *mapping*. Many algorithms implement these steps using an extended Kalman filter [5]. Therefore, to understand SLAM, the extended Kalman filter will be discussed, as well as how this filter can be used to perform SLAM.

2.1.1 From a robot's state to the Extended Kalman filter

Before describing how robots update their state, some definitions should be explained first. Robots typically have access to two types of data: the measurement data and the control data. The measurement data is denoted as z_t for the measurements at time t. Control data represent the change of the *state* of the robot. Like the measurement data, control data is denoted as u_t for the change of the state during the interval $< t-1; t]$ [3]. The *state* of the robot is defined by Thrun, Burgard, and Fox [3] as the collection of all aspects of the robot and its environment

that might impact the future. The state of the robot at time t will be denoted by x_t. A state is complete and fulfils the *Markov assumption* if the entire next state x_{t+1} can be determined by only using the previous state x_t and the data from that time point z_t and u_t. How control influences state and how state influences the measurements is described in figure 2.1. Note that it is assumed that the state is complete, the control is executed first, and the measurement is taken after. These assumptions align with the Bayes filter, which will be discussed later and describes how to calculate the most likely next state.

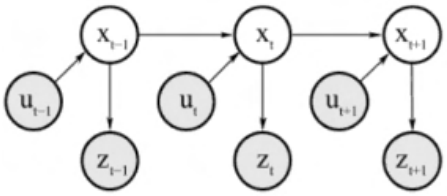


Figure 2.1: The evolution of state, control and measurements. Source: [3]

Belief

The belief of a robot is what the robot believes to be its state and its environment. The belief calculation can be divided into two parts: calculating the state based on the control data, using the *state transition probability*, and updating that prediction with the measurement data, using the *measurement probability*. The state transition probability is defined as $p(x_t|x_{t-1}, u_t)$ if the Markov assumption is fulfilled. It corresponds with updating the state based on the control data. The measurement probability is described as $p(z_t|x_t)$ and thus describes how measurements follow from the current state. The believe is thus described as $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$. When the Markov assumption is fulfilled, it can be rewritten as $bel(x_t) = p(x_t|x_{t-1}, z_t, u_t)$ When calculating the belief without the measurement z_t, this is called the *prediction*. The prediction is denoted as $\bar{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$, and if the Markov assumption holds, as $\bar{bel}(x_t) = p(x_t|x_{t-1}, u_t)$. Going from the prediction to the belief is the *measurement update*.

Bayes Filter Algorithm

The Bayes filter algorithm calculates the robot's belief. The algorithm is described in algorithm 1. Line 3 calculates the prediction for a state x_t by combining all probabilities for x_t if x_{t-1} was the previous state, multiplied by the belief that x_{t-1} was the actual previous state. Line 4 then corrects this belief by incorporating the measurement. This is done by taking the probability that the measurement occurred if x_t was the state and multiplying this by the prediction x_t is the actual state. The η represents the normalisation factor, as the belief any state is the case should always be one, while the multiplication for each state does not always equal one [3].

Algorithm 1 Bayes Filer algorithm by Thrun, Burgard, and Fox [3]

1: **procedure** BAYES FILTER($bel(x_{t-1}), u_t, z_t$)
2: **for all** x_t **do**
3: $\bar{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$
4: $bel(x_t) = \eta p(z_t|x_t) \bar{bel}(x_t)$
5: **return** $bel(x_t)$

The Kalman Filter

The Kalman filter is an implementation of the Bayes filter. It is a Gaussian filter, meaning the belief is represented by a multivariate normal distribution and is represented by its mean μ and covariance Σ. The Kalman filter assumes all arguments are linear Gaussian, implying the following:

- The state transition property $p(x_t|x_{t-1}, u_t)$ is linear in its arguments: $x_t = A_t x_{t-1} + B_t u_t + \eta_t$. The state x_{t-1} and control u_t are vectors. The added η_t is a random Gaussian vector with a mean of zero and a covariance of R_t. This means that the mean of the transition state property becomes $A_t x_{t-1} + B_t u_t$ and the covariance R_t.
- The measurement probability $p(z_t|x_t)$ is linear in its arguments: $z_t = C_t x_t + \delta_t$. The added δ_t is a random Gaussian vector with a mean of zero and a covariance of Q_t. The mean of the measurement probability thus is $C_t x_t$ and the covariance Q_t.

The Kalman filter algorithm is described in algorithm 2. Lines two and three represent the prediction. Line four calculates the *Kalman gain*. The Kalman gain represents how much the measurement should be taken into account when calculating the belief. In line five, the mean is adjusted by the *innovation* of the measurement. Innovation means the difference between the expected measurement $C_t x_t$ and the actual measurement z_t. The mean is adjusted by this difference, depending on how much weight should be given to the measurement, which is determined by the Kalman Gain. The same procedure happens in line six, only this time concerning the covariance of the belief. Finally, the belief for time t is returned [3].

Algorithm 2 The Kalman Filter by Thrun, Burgard, and Fox [3]

1: **procedure** KALMAN FILTER(μ_{t-1}, Σ_{t-1}, u_t, z_t)
2: $\bar{\mu_t} = A_t \mu_{t-1} + B_t u_t$
3: $\bar{\Sigma_t} = A_t \Sigma_{t-1} A_t^T + R_t$
4: $K_t = \bar{\Sigma_t} C_t^T (C_t \bar{\Sigma_t} C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu_t} + K_t(z_t - C_t \bar{\mu_t})$
6: $\Sigma_t = (I - K_t C_t)\bar{\Sigma_t}$
7: **return** μ_t, Σ_t

The extended Kalman filter

In reality, not all measurements are normally distributed. Therefore, the Kalman filter rarely works in practice. The extended Kalman filter linearises its arguments so that the Kalman filter can be applied. The extended Kalman filter uses first-order Taylor expansions to linearise both the state transition property and the measurement update property:

- The state transition property $p(x_t|x_{t-1}, u_t)$ is not linear in its arguments and described by $x_t = g(u_t, x_{t-1}) + \eta_t$. How this function is linearised is shown in equation 2.1. The partial

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{2.1}$$

Figure 2.2: The linearisation of g.

derivative g' is taken with regards to the mean of the state, μ, because this is the most likely state. The function $g'(u_t, \mu_{t-1})$ is often abbreviated by G_t, also called a Jacobian. How this Jacobian is calculated is shown in equation 2.2.

$$G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial \mu_{t-1}} = \begin{bmatrix} \frac{\partial x'}{\partial \mu_{t-1,x}} & \frac{\partial x'}{\partial \mu_{t-1,y}} & \frac{\partial x'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial y'}{\partial \mu_{t-1,x}} & \frac{\partial y'}{\partial \mu_{t-1,y}} & \frac{\partial y'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial \theta'}{\partial \mu_{t-1,x}} & \frac{\partial \theta'}{\partial \mu_{t-1,y}} & \frac{\partial \theta'}{\partial \mu_{t-1,\theta}} \end{bmatrix} \tag{2.2}$$

Figure 2.3: The Jacobian G_t. The variable x' represents the function g along the x-axis

$$h(x_t) \approx h(\bar{\mu}_t) + h'(\bar{\mu}_t)(x_t - \bar{\mu}_t) \tag{2.3}$$

Figure 2.4: The linearisation of h.

- The measurement transition property $p(z_t|x_t)$ is not linear in its arguments and described by $z_t = h(x_t) + \delta_t$. How this function is linearised is shown in equation 2.3. The partial derivative is taken with the mean prediction since this is the most likely state. The function $h'(\bar{\mu}_t)$ is often abbreviated by H_t, also called a Jacobian. How this Jacobian is calculated is shown in equation 2.4.

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} = \begin{bmatrix} \frac{\partial x'}{\partial \bar{\mu}_{t,x}} & \frac{\partial x'}{\partial \bar{\mu}_{t,y}} & \frac{\partial x'}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial y'}{\partial \bar{\mu}_{t,x}} & \frac{\partial y'}{\partial \bar{\mu}_{t,y}} & \frac{\partial y'}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \theta'}{\partial \bar{\mu}_{t,x}} & \frac{\partial \theta'}{\partial \bar{\mu}_{t,y}} & \frac{\partial \theta'}{\partial \bar{\mu}_{t,\theta}} \end{bmatrix} \tag{2.4}$$

Figure 2.5: The Jacobian H_t. The variable x' represents the function h along the x-axis

The extended Kalman filter described in algorithm 3 closely represents the Kalman filter described by algorithm 2. The main differences are the calculations of the expected mean and the replacements of the matrices [3].

Algorithm 3 The Extended Kalman Filter by Thrun, Burgard, and Fox [3]

1: **procedure** Extended Kalman Filter(μ_{t-1}, Σ_{t-1}, u_t, z_t)
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
7: **return** μ_t, Σ_t

2.1.2 From the Extended Kalman filter to Extended Kalman Filter SLAM

Now, it is clear how EKF SLAM updates its state. However, in SLAM, the motion, localisation and mapping need to take place as well. In order to explain EKF SLAM, first, the motion model will be discussed, then the localisation model, then EKF-localisation and finally EKF SLAM.

The velocity motion model

In the motion model, the current pose of the robot is described by $(x, y, \theta)^T$, with x, y as its 2D coordinates and θ as its direction, also called its *bearing*. The current hypothesised state x_t is denoted as $(x', y', \theta')^T$. The previous state x_{t-1} is denoted as $(x, y, \theta)^T$. How the believed position can be calculated is described in equation 2.5. Note that v is the forward velocity, and ω is the rotational velocity [3].

The odometry motion model

The velocity motion model uses translational and rotational velocities to determine the most likely next robot position. Some robot interfaces provide calculated robot positions based on the robot's movements. These are odometry measurements. The odometry motion model uses the current believed pose and the previous believed pose from this data to generate two rotational

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix} \tag{2.5}$$

Figure 2.6: The velocity motion model. Source: [3]

and one translational velocities. These are shown in figure 2.7. The first rotational velocity moves in the heading direction, and the second rotational velocity moves from the heading direction to the orientation of the final pose. The translational velocity describes how far the robot moves.

These velocities are calculated based on the odometry information for the current timestamp

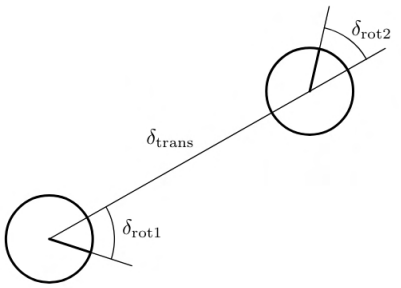


Figure 2.7: The odometry motion model. Source: [3].

and the previous timestamp. When the previous odometry pose is given by $\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})$ and the current odometry pose is given by $\bar{x}_t = (\bar{x}', \bar{y}', \bar{\theta}')$, then the velocities are calculated as shown in equations 2.6, 2.7, and 2.8 [3].

$$\delta_{rot1} = atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \tag{2.6}$$

$$\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2} \tag{2.7}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1} \tag{2.8}$$

These velocities are combined to update the robot state as shown in equation 2.9, where the estimated robot position at time x_t is represented by $(x', y', \theta')^T$ and x_{t-1} is represented by $(x, y, \theta)^T$ [3].

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \delta_{trans}\cos(\theta + \delta_{rot1}) \\ \delta_{trans}\sin(\theta + \delta_{rot1}) \\ \delta_{rot1} + \delta_{rot2} \end{bmatrix} \tag{2.9}$$

Feature-based measurement model

When measuring data, either the entire raw data is processed, or certain *features* from the measurement data are selected, such as a wall or a corner. This reduces the computational complexity. Features might be based upon *landmarks*, which are physical objects. When the feature extractor is given as a function f, the feature vector for measurement z_t becomes as in equation 2.10, with r being the range of a feature, ϕ being its bearing and s its signature.

$$f(z_t) = \{f_t^1, f_t^2, ...\} = \{ \begin{bmatrix} r_t^1 \\ \phi_2^1 \\ s_t^1 \end{bmatrix}, \begin{bmatrix} r_t^2 \\ \phi_2^2 \\ s_t^2 \end{bmatrix}, ... \} \tag{2.10}$$

Because conditional independence is assumed, the probability of getting a feature vector can be calculated by calculating the probability for each feature separately.

Feature-based maps consist of a list of features, with each feature having a signature and a

location coordinate. The range and bearing from the robot's pose to the feature of the map m_j is given by equation 2.11.

$$\begin{bmatrix} r_t^i \\ \phi_2^i \\ s_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ atan2(m_{j,y} - y, m_{j,x} - x) - \theta \\ s_j \end{bmatrix} + \begin{bmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\phi^2} \\ \epsilon_{\sigma_s^2} \end{bmatrix} \tag{2.11}$$

Furthermore, when discussing feature-based models, the correspondence variable c_t^i denotes the corresponding map feature of the measured feature i at time t [3].

Extended Kalman filter localisation

When incorporating the feature selection and the motion model into the EKF algorithm, a localisation algorithm can be formed. This algorithm is closely related to extended Kalman filter SLAM since the only aspects that differ are related to the map.

The Extended Kalman Filter localisation can be seen in algorithm 4. The prediction step is executed in lines three through four based on the control. Line three calculates the Jacobian of the movement. When the motion model in equation 2.5 is taken as the function for the Jacobian calculation in equation 2.2, the matrix in line three follows. Lines four and five describe the covariance of the noise: M_t denotes the noise in the controls, and V_t maps this to noise in the state by creating a matrix of the derivative of motion regarding the control. Lines six and seven represent the standard EKF prediction step as described in algorithm 3.
Lines eight through twenty represent the correction step. The linearisation of the measurement is done as follows: from equation 2.11, the vector without the error vector can be seen as the h function as described in section 2.1.1. Then, it follows that the H_t^i matrix will be the derivative of this vector taken over the predicted mean $\bar{\mu}_t$, much like in equation 2.4. Lines fourteen through seventeen represent the correction step as shown in algorithm 3. The variable $\hat{z}$ used in these calculations is the expected feature, q is the quadratic distance to the feature, and j is the true identity of the feature. Finally, in line twenty, the probability of measuring z_t is calculated and returned alongside the new robot state [3].

Extended Kalman Filter SLAM

The extended Kalman Filter SLAM has a slightly different goal than extended Kalman filter localisation. In SLAM, the robot has to construct the map and, thus, keep track of the features and their likely location. In algorithm 5, this can be seen in that the map is not part of the arguments anymore. Note how this algorithm is very similar to algorithm 4. In this algorithm, the mean and covariance matrix not only contain the pose but all the landmarks as well. The F matrices select either only the robot pose, x, and a feature with index j. Then, lines two through five contain the prediction step using the motion model.
In the update step, the matrix H_t^i now depends on only two parts of the vector, thus it is split up in a selection matrix $F_{x,j}$ and a more compact matrix. Another important note is that when a feature is seen for the first time, it is initialised based on the robot's believed position and orientation [3].

2.1.3 Confidence in EKF SLAM

Although EKF SLAM is a widely used method for simultaneous localisation and mapping, the filter is known to be inconsistent. That is, the extended Kalman filter can be overconfident in its estimates, leading to inconsistency. Huang, Mourikis, and Roumeliotis [8] perform an observability analysis on both the real non-linear SLAM model and the EKF SLAM model. Observability in this context means the ability to infer the entire state of the system from the measurements. They conclude that the real SLAM model has an unobservable subspace of dimension 3, representing the x and y position and rotation in the real-world coordinate frame. Contradictory to this finding, the EKF SLAM model has an unobservable subspace of dimension 2, representing the x and y position in the real-world coordinate frame. This conclusion entails

Algorithm 4 EKF localisation with known correspondences by Thrun, Burgard, and Fox [3]

procedure EKF LOCALISATION(μ_{t-1}, Σ_{t-1}, u_t, z_t, c_t, m)

$\theta = \mu_{t-1,\theta}$

$$G_t = \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos(\theta) + \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 1 \end{bmatrix}$$

$$V_t = \begin{bmatrix} -\frac{\sin(\theta) + \sin(\theta + \omega_t \Delta t)}{\omega_t} & \frac{v_t(\sin(\theta) - \sin(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t \cos(\theta + \omega_t \Delta t)\Delta t}{\omega_t} \\ \frac{\cos(\theta) - \cos(\theta + \omega_t \Delta t)}{\omega_t} & -\frac{v_t(\cos\theta - \cos(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t \sin(\theta + \omega_t \Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{bmatrix}$$

$$M_t = \begin{bmatrix} \alpha_1 v_t^2 + \alpha_2 \omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4 \omega_t^2 \end{bmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix}$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T$$

$$Q_t = \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{bmatrix}$$

for all $z_t^i = (r_t^i, \phi_t^i, s_t^i)^T$ **do**

$j = c_t^i$

$q = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$

$$\hat{z}_t^i = \begin{bmatrix} \sqrt{q} \\ atan2(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \end{bmatrix}$$

$$H_t^i = \begin{bmatrix} -\frac{m_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \bar{\mu}_{t,y}}{q} & -\frac{m_{j,x} - \bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$S_t^i = H_t^i \bar{\Sigma}_t [H_t^i]^T + Q_t$

$K_t^i = \bar{\Sigma}_t [H_t^i]^T [S_t^i]^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$

$\bar{\Sigma}_t = (I - K_t^i H_t^i) \bar{\Sigma}_t$

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

$p_{z_t} = \Pi_i det(2\pi S_t^i)^{-\frac{1}{2}} exp\{-\frac{1}{2}(z_t^i - \hat{z}_t^i)^T [S_t^i]^{-1} (z_t^i - \hat{z}_t^i)\}$

return μ_t, Σ_t, p_{z_t}

Algorithm 5 EKF SLAM with known correspondences by Thrun, Burgard, and Fox [3]

procedure EKF SLAM(μ_{t-1}, Σ_{t-1}, u_t, z_t, c_t)

$\theta = \mu_{t-1,\theta}$

$$F_x = \begin{bmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 \end{bmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix}$$

$$G_t = I + F_x^T \begin{bmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos(\theta) + \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 0 \end{bmatrix} F_x$$

$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_x^T R_t F_x$

$$Q_t = \begin{bmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{bmatrix}$$

for all $z_t^i = (r_t^i, \phi_t^i, s_t^i)^T$ **do**

  $j = c_t^i$

  if landmark j never seen before **then**

$$\begin{bmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{bmatrix} + \begin{bmatrix} r_t^i \cos(\phi_i^t + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_i^t + \bar{\mu}_{t,\theta}) \\ 0 \end{bmatrix}$$

$$\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{j_x} - \bar{\mu}_{t_x} \\ \bar{\mu}_{j_y} - \bar{\mu}_{t_y} \end{bmatrix}$$

$q = \delta^T \delta$

$$\hat{z}_t^i = \begin{bmatrix} \sqrt{q} \\ atan2(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{bmatrix}$$

$$F_{x,j} = \begin{bmatrix} 1 & 0 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 1 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 1 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 0 & 1 & 0...0 \end{bmatrix}$$

$$H_t^i = \frac{1}{q} \begin{bmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & -\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & \delta_x & -1 & -\delta_y & -\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} F_{x,j}$$

$K_t^i = \bar{\Sigma}_t H_t^{iT} (H_t^i \bar{\Sigma}_t H_t^{iT} + Q_t)^{-1}$

$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$

$\bar{\Sigma}_t = (I - K_t^i H_t^i) \bar{\Sigma}_t$

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t

that the EKF SLAM system receives nonexistent information about the rotation of the robot. This is proposed as the core cause of the inconsistency in SLAM and might lead to overconfidence. To prevent this overconfidence, Huang, Mourikis, and Roumeliotis [8] propose a SLAM filter that uses not the latest state estimates in the Jacobians, but the first, leading to the name *first estimates Jacobian EKF SLAM*. They state that this should lead to the unobservable subspace of EKF SLAM having dimension 3. Two changes are needed to transform EKF SLAM to FEJ-EKF SLAM;

- In the G Jacobian for the state transition probability, instead of using the previous believed state, the previous predicted state has to be used.
- In the H Jacobian for the measurement transition property, instead of using the latest predictions for the landmark positions, the first predictions for the landmark positions have to be used in the parts of the Jacobian which correspond to the robot's position.

2.2 Colour filters

A colour filter will be used to compare YOLO to a more traditional method of extracting features from an image. Therefore, this section will discuss colour spaces and colour filters.

2.2.1 Colour spaces

Colours that are perceived, are combinations of multiple lightwaves. The spectral decomposition of amplitude by wavelength determines the hue, colourfulness and intensity. In human eyes, cones and rods determine how light is percieved. The cones determine the colour we perceive, of which there are three types; one mostly sensitive to red hues, one mostly sensitive to green hues and one mostly sensitive to blue hues [9].

These three types of cones then lead to the most well-known colour space, namely the RGB colour space. In the RGB colour space, a composition of red, green and blue light represents each colour in this colour space. This colour space can be represented by a three-dimensional cube, where the x, y and z axes represent the R, G, and B colours, respectively [9]. The RGB colour space is shown in figure 2.8.

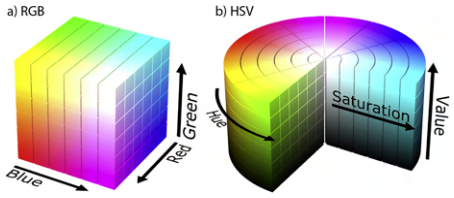


Figure 2.8: The RGB colour space (a) and HSV colour space (b). Source: [10]

However, people don't typically view colours as a mixture of red, blue, and green. Instead, most people first describe a colour by its hue and then by other characteristics such as intensity and brightness. Therefore, a colour space which describes a colour by these characteristics would be more intuitive [9]. The HSV (Hue Saturation Value) colour space is such a colour space where the hue is described by an angle, making it a cylindrical colour space. The HSV colour space can be viewed in figure 2.8.

2.2.2 Colour filters

Detecting objects based on their colour has many applications. For example, colour filters can help to determine whether food has gone bad [11]. Another application of colour filters is in fire detection. Celik and Ma [12] describe how colour masks can detect fire. They note as well that for fire detection, the RGB colour space can be less intuitive and less effective for creating

colour masks, in comparison with more intuitive colour spaces, based on values such as hue and illumination.

2.3 YOLO: You Only Look Once

YOLO, You Only Look Once, is a powerful object detection model that runs in real time. Redmon et al. [13] state that the first YOLO model could run at 45 frames per second, making it fast enough for real-time applications. Since then, YOLO has only gotten faster and is now widely used in many applications [6].

2.3.1 The inner workings of YOLO

YOLO consists of a single convolutional neural network, hence the name. The advantages include object detection becoming much faster and YOLO being able to include contextual information when analysing an image. An overview of how YOLO works can be seen in figure 2.9. The image is split up into an S by S grid. Each cell predicts B bounding boxes. The bounding boxes are defined by x, y, w, h and confidence. The coordinates of the centre of the bounding box are x, y, relative to the grid cell. The width and height w, h of the bounding box are relative to the image. The prediction represents the *IoU: intersection over union* between the predicted bounding box and the ground truth bounding box. The intersection over union represents the overlap between the predicted box and the ground truth box: zero means no overlap, and one means they are the same box.
Other than predicting B bounding boxes, each cell also predicts the chance a class is in the grid for each class that can be classified, thus for C classes. These probabilities are called conditional class probabilities and are described by $p(Class_i|Object)$ [13].
The network of the first YOLO model consists of 24 convolutional layers followed by two fully

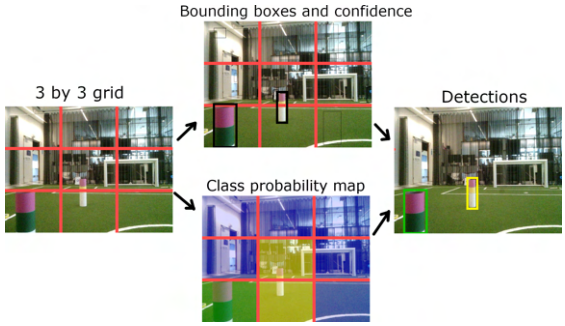


Figure 2.9: A simplified example of the YOLO model, with a three-by-three grid, where each grid only predicts one bounding box. The thickness of the bounding boxes represents their probability. The blue colour in the class probability map corresponds with the background.

connected layers. The convolutional layers are responsible for extracting features from the image, and the two fully connected layers are responsible for the object detection [13]. The fully connected layers were removed in the next version of YOLO, making YOLO a fully convolutional network [6].

Over the next few years, many new YOLO versions have been released. The concept of scaling up and scaling down was introduced. Scaling up meant a larger model with a slower performance and more accuracy. Scaling down meant a smaller model with lower accuracy and more speed. In these new versions, the separation between the *backbone*, *neck*, and *head* was made. In the backbone, features from the image are found, which are refined in the neck. The head then uses those features to predict the objects. YOLO version 8, released in 2023, includes a detached head to allow for more features than just object detection. Other functionalities such as object tracking, pose estimation, and segmentation are possible as well. YOLOv8 comes in different sizes as well [6].

2.3.2 YOLOv8

YOLO version 8, or YOLOv8, was created by the company Ultralytics and is based on their previous model, YOLOv5. YOLOv8 is more accurate and faster than YOLOv5 [14]. These increases in accuracy and speed are due to some changes in the architecture. As previously mentioned, this architecture uses a back-end to distil the features from the image. In the neck, these features are combined and refined. In the head, the classification then takes place. One difference between YOLOv5 and YOLOv8 is that the backbone's functionality to incorporate features and add contextual information has been improved. Another difference between version 8 and version 5 is that the kernel size in the convolutions decreased from 6 to 3 [6]. For more detail on the model, see section A.2. The segmentation model and object detection model for version 8 largely overlap. The neck is slightly different, and the segmentation model uses a different head [6].

2.4 Summary

This section discussed the theory behind the different elements used in this thesis to create EKF SLAM. First, SLAM was explained by starting from a robot's state, then going over the Bayes filter and then expanding this knowledge to EKF localisation and EKF SLAM. A different version of the EKF filter, the first estimates Jacobian EKF, was explained as well. Next to SLAM, colour filters and YOLO were discussed.

In the next section, these elements will be combined to set up experiments to answer the research question *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*. The versions of EKF SLAM discussed will be used as the back end, whilst the colour filter and YOLOv8 will be used for the front end.

CHAPTER 3

Method

As previously stated, the research question of this thesis is: *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*. This research question is split into two sub-questions: *How do different versions of the Extended Kalman filter impact the performance of SLAM in the context of robot football?*, and: *How do YOLOv8s and YOLOv8m improve the performance of EKF-based SLAM compared to a classic colour filter?*

Experiments are required to answer the research question. Performing EKF SLAM on a dataset recorded on a Nao robot can answer all sub-questions; the front ends and back ends are tested on data generated from a Nao robot. The front end of this algorithm will be both with and without YOLO. The back end of this algorithm will be different versions of EKF SLAM.

A dataset, multiple back ends, and multiple front ends are required to perform this experiment. First, a dataset must be created on the Nao robot. Next, several versions of EKF SLAM are needed for the back end. Finally, the front end of EKF SLAM requires both trained YOLO models and a colour filter. The colour filter will form the baseline against which YOLO can be compared.

This chapter will discuss how the elements mentioned above are created. To ensure that the individual aspects of the experiment function, they also require testing, which will be introduced in this section.

3.1 Creation of the dataset

As the algorithm in section 2.1.2 describes, the back-end needs several arguments: the control data and the measurement data. This section will detail how this data was collected using the Nao robot. The final result will be one dataset recorded on a Nao, consisting of a video and a log file. The log file will contain odometry information and the video's start and end times.

3.1.1 Technical considerations

The measurement data can be extracted from the video stream the robot produces. The type of control data depends on the kind of motion model used. The velocity model needs the rotational and translational velocities. The odometry motion model requires the current and previous poses to be known [3]. The NAOqi[1] API provides an estimated robot position from the motion, making the odometry motion model the logical choice.

[1] http://doc.aldebaran.com/2-8/index_dev_guide.html

3.1.2 The recording setup

The SLAM algorithm will work with unique landmarks. Gutmann and Fox [7] use these landmarks as well to compare different localisation methods. The landmarks used are the same, although their location differs. Figure 3.2 shows the setup used by Gutmann and Fox [7]. Figure 3.3 shows the setup used in this thesis. The field size between the database by Gutmann and Fox and the databases in this paper differ. The field used in this thesis is larger because the field size of the standard platform league has increased over time. The poles have been moved in the field to keep the distance between the robot and the landmarks similar. Figure 3.1 shows the setup with the landmarks in place. One main difficulty in this environment is the windows, which can cause high contrast in the video data even with the curtains closed. This contrast due to the incoming sunlight causes the landmarks to be less recognisable, as can be seen in figure 3.5.

Figure 3.1: The setup containing the poles and the Naov6 robot.

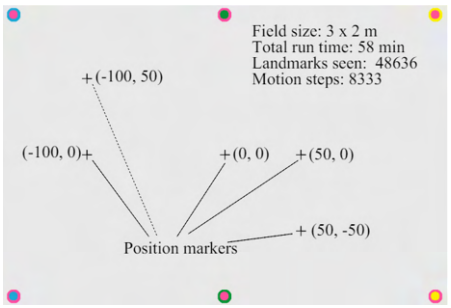


Figure 3.2: The robot football field of 3 by 2 meters with the position of the landmarks as described by Gutmann and Fox [7].

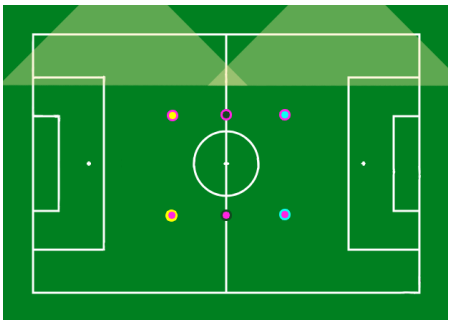


Figure 3.3: The robot football field of 9 by 6 meters with the position of the landmarks. Compared to the setup of Gutmann and Fox [7], the landmarks are moved in the field because this field is larger. As shown, the upper side of the field has sunlight coming in through the windows.

3.1.3 The data from the robot

The Naov6 robot was programmed using Choregraphe version 2.8.8 and NAOqi 2.8. Due to unstable turns, the robot walked in a square, making the maximum angle the robot should turn 90°. The robot retrieved the believed locations over time in world coordinates, the head position, and all data from the IMU and collected them in a log file. The robot recorded a video next to the log file to get the measurement data. The videos are 15 fps with an aspect ratio of 640 by 480 pixels and are in AVI format.

3.1.4 The ground-truth data

The ground truth data is recorded using OptiTrack[2], a motion capture system using infrared cameras and reflective markers to capture position. After calibrating the system, it found that the mean error was only 0.261mm, deeming it suitable as a ground truth. Optitrack measured the robot's location using the hat the robot was wearing. The hat, shown in figure 3.4, has reflectors. Some of these reflectors are spheres lifted away from the robot so that multiple cameras can always track them, regardless of the rotation of the robot. Using those markers in OptiTrack, a rigid body was created in which the centre was the middle of the robot's head. The X, Y and Z coordinates, along with the rotation of the rigid body, were recorded from OptiTrack.

Figure 3.4: OptiTrack markers mounted on the robot's head.

3.1.5 Combining the data

The odometry, measurement, and ground truth data must be combined to execute and test EKF SLAM. This can be achieved by sorting the data on time. The odometry and ground truth already had timestamps attached. The measurement times were inferred from the logged start and end times. The time per frame was distilled by starting from the recorded start time and adding a certain time interval between each frame. The time interval was calculated by the length of the video divided by the number of frames. This method was chosen because the difference between the logged start and end times did not match the length of the video. Hence, there could be a small delay between the odometry and measurement data. In the final experiments, 0.3 seconds were added to the timestamps of the measurement data to correct this delay.
Since OptiTrack and the Nao robot use different coordinate systems, both were combined into one coordinate frame: the Z axis points up, the X axis corresponds with the broad side of the field, and the Y axis corresponds with the short side of the field. The origin is the centre dot of the field, making it easy to see which quarter of the field the robot is in.

[2]https://optitrack.com/

3.1.6 The dataset

After processing the dataset, a video, logfile and ground truth file are created. The timestamps for each frame are available as well. This dataset will be used in the final tests.
Another dataset was created next to this one. This dataset was recorded on a robot with a relatively unstable walk, which required the robot to be held while recording. The robot could not turn without falling. Thus, the turns were simulated by only turning the head. The turns were manually integrated into the odometry data. Because these workarounds prevented the actual odometry error from being displayed, the decision to create the previously discussed dataset was made. The front-end tests use data from the videos from the old dataset.

3.2 Creating the EKF SLAM back end

A vital part of the SLAM algorithm is the back end, where the features get processed to update the map and the location [2]. This thesis will use the extended Kalman filter since this filter forms the basis of many state-of-the-art SLAM back ends [2]. This filter will be based on Thrun, Burgard, and Fox [3], as described in section 2.1.2. This code will be based on an online repository which implements some algorithms from this book[3]. Due to the Nao dataset having odometry information, the code was first rewritten to work with the odometry motion model. Then, some additions to the filter are tested to see how much this improves the accuracy. The back end will also be tested with a more stable dataset than the Nao datasets to see how reliable the back end is.

3.2.1 Different versions of the EKF filter

As discussed, multiple versions of the filter are tested. The first version is as described in section 2.1.2. Note that this code contains many sparse matrices. The original code already implemented some improvements: each feature in the state vector only has two components; the signature is left out. Furthermore, the creation of relatively large matrices is not done by multiplying selection matrices but by accessing the index of the matrix, represented as a 2D array. The focus of this thesis thus lies in other improvements. The first improvement aims to fix the overconfidence. This filter is inherently overconfident, as described in section 2.1.3. Huang, Mourikis, and Roumeliotis [8] propose the first-estimates Jacobian extended Kalman filter, described in section 2.1.3, to prevent this. This thesis will test the FEJ-EKF as well.
Other than improving the method of updating the state and covariance, filtering measurements used to update the state and covariance can also have a huge impact. Filtering measurements is vital since the colour filter may produce false positive results. Two methods of filtering results are used: one is to prevent the first landmark measurement from being a false positive. This filter checks whether a landmark has been observed in the same area n times. If so, the potential landmark can be added to the map; hence, this filter will be referred to as the *minimal view EKF*. The second method is to prevent false positive detections once a landmark has been observed. The filter discards the measurement when a detected landmark differs too much from the expected landmark position or bearing. The maximum range difference is 0.4 metres, and the maximum bearing difference is $\frac{\pi}{3}$ radians.

3.2.2 Testing the back end

Because the Nao robot has legs instead of wheels, a monocular camera, and walks on an unstable field, the front end of the algorithm might be unstable. Therefore, testing the back end on the Nao dataset will not clarify how much of the error comes from the back end. Thus, the back end must be tested on a more stable dataset. The original code uses the dataset created by Leung et al. [15], a multi-robot dataset with unique landmarks. However, this dataset contains velocity information instead of odometry information, and the robots move over an area significantly larger than an SPL football field. Because the final model will use odometry information and be

[3]https://github.com/ChengeYang/Probabilistic-Robotics-Algorithms

tested on a dataset with distances within a few metres, another dataset was used to test the back end. Gutmann and Fox [7] propose a dataset with the same landmarks used in this thesis and with the same context of robot football. Their setup is previously discussed and shown in figure 3.2. The dataset is 58 minutes in length. The ground truth is recorded by logging when the robot walks over certain positions on the field, also shown in figure 3.2. This leads to the ground truth being less reliable. However, the dataset is in the same context as the final dataset, and EKF localisation performs rather well, with an error of approximately 110mm [7]. Therefore, this dataset will be used to test the back end.

The result of testing the back end can be found in section 4.1.

3.3 Creating the EKF SLAM front end using colour filters

A baseline must be established to visualise how much of an impact YOLO can have on SLAM. The landmarks all consist of unique combinations of colours, making a colour filter a logical approach for a baseline.
This section will discuss how the colour filter was created and how it will be tested.

3.3.1 Creating the colour filter

There are several ways of implementing a colour filter. What is essential to keep in mind is that this filter has to run quite quickly, in real-time, on a robot. As discussed in section 2.2, the RGB format in which the Nao robot records its videos is less intuitive than, for example, the HSV format. Luijten [16] therefore defines the colour filters in HSV format and then transforms those filters to RGB format. Most applications transform each image to HSV format. However, this is a costly operation that has to be performed in real-time. With the solution from Luijten, the colour format transformation can be done beforehand. This solution will be used in this thesis to detect where the colours of the landmarks are. A colour filter is created for the colours pink, blue, yellow, and green based on the hues, saturation, and value of the sample images from the dataset. To check whether two colours are attached to a white landmark, a white colour filter was created as well.

These detected two colours must be connected to form the landmarks. Due to the robot often not standing straight when walking, many landmarks are recorded at an angle. Thus, checking whether a colour is directly below or above another colour might not always work. Hence why, the following technique was used to determine if a coloured landmark was present;

1. For each pink colour segment, calculate the weighted centre, the highest point and the lowest point.
2. Calculate the distance from the centre to the top and from the centre to the bottom. Take the longest distance, multiply this by 1.2 and call it d.
3. From the centre of the pink colour segment, go straight down (270°) by d and check whether the bottom contains a colour. If not, do the same for the top (90°).
4. If no colour is found, repeat the step above. However, instead of checking 270°and 90°, check $270° \pm \alpha$ and $90° \pm \alpha$. Let α start at two and increase by two each iteration until α is 60°.
5. If no colour surrounding the pink is found, the pink segment is not part of a landmark. Start over with the next pink segment. If a colour is found, repeat steps 3 and 4. However, instead of starting from the pink segment centre, start from the centre of the lowest colour. Scan only the bottom, and check only for the colour white.
6. If the white colour is found, the three segments together form a landmark. If not, the two colours are not part of a landmark.

As a final result, the colour filter outputs a list of the landmarks detected for each image. The type of landmark is recorded for each landmark, together with the centres of the pink segment, the colour segment, and the middle between these two. The length between the pink and colour segments is also given to calculate the range of the landmark.

3.3.2 From landmark to range and bearing

Calculating the range and bearing from the landmarks requires knowledge of both the length of the landmark on the image and the actual length of the landmark. The length was estimated by the distance between the centre of the pink colour segment and the other colour segment. There are two reasons that this was taken as the length measurement. The first reason corresponds with the reason why white cannot be included in the pole range estimation; the white colour filter is prone to selecting not only the bottom of the landmark but also large sections of the background. The background is selected because the landmarks often reflect the green from the field or are darker in the shade, which caused the white filter to have many values, making it less accurate. The white section may cause the background to become part of the landmark. The second reason corresponds with the reason why height instead of width is considered; one part is often darker due to the landmark's shape. This causes one part of the landmark to hardly be detected due to it being too dark. Therefore, the width of the colour detection is less reliable than the height.

Figure 3.5: Field landmarks placed at a certain distance at different angles from the robot. From left to right: 150cm at ±20°, ±10°and 0°, 100cm at ±20°, ±10°and 0°, 50cm at ±20°and 0°.

The Nao top camera has a horizontal field of view of 56.3°. The hypothesis was that the bearing could be calculated by simply dividing the horizontal field of view by the image's width and multiplying it by the distance the landmark was from the centre. Taking several pictures from the Nao robot of the landmarks confirmed this. Figure 3.5 shows these pictures. Five landmarks were placed at ±20°, ±10°and 0°. In the first picture, all landmarks were 150cm from the robot. In the second picture, all landmarks were 100cm from the robot. In the third picture, only the landmarks at ±20°and 0°were placed at a distance of 50cm. The final bearing formula can be seen in equation 3.1. The x component of the centre of the landmark is denoted by f_x. The equation is multiplied by -1 because the algorithm expects markings to the right of the robot to have a negative bearing.

$$bearing_{deg} = -1 * (f_x - 320) * \frac{56.3°}{640} \tag{3.1}$$

The range calculation was performed by taking the distance between the two colour segments in pixels at different distances to the robot. The pixel-to-centimetre ratios at various distances were calculated for each colour at each distance. Every landmark had a slight difference in the length of the colour segments. Thus, each pixel length was divided by the actual length of the landmark, depending on which landmark was detected. The distances at which the pixel-to-centimetre ratios were calculated were 75cm, 100cm, 150cm, and 200cm. After this, each average pixel-to-centimetre ratio for every distance was taken, and interpolation over these points created the formula for the range, which is shown in equation 3.2. The variable r denotes the pixels/cm

range of the detected landmark.

$$range_m = 3.0964 * r^{-0.952} \tag{3.2}$$

3.3.3 Testing the colour filter

The accuracy of the colour filter will be determined by its capability to detect both the correct landmark and bearing. These two factors will be tested on 150 images randomly taken from the first dataset recorded on the Nao robot. The number of true positives, false positives, true negatives and false negatives will be reported per image. The localisation will be tested by comparing the estimated and actual centres. The exact centre is defined as the middle of the dividing line between the two colours. This ground truth measurement was chosen as the centre because, ideally, the centre between the two colours should also be the centre of the dividing line. Since the first and final datasets were recorded at different points in time, their lighting condition differed; the second dataset had more cool-toned light, whereas the first had more warm tones. This difference led to the HSV colour masks having to be redefined between these tests. This points out one of the flaws of this colour filter: it is very situation-specific.
The results for testing the colour filter can be found in section 4.2.

3.4 Creating the EKF SLAM front-end using YOLO

YOLO, you only look once, is a state-of-the-art object detection model. This model will be used to detect the coloured landmarks. As discussed in section 2.3, in YOLOv8, it is possible to detect objects and perform image segmentation. Due to the size of the landmarks being important in estimating the distance, this thesis will focus on implementing image segmentation in YOLOv8 to detect coloured landmarks. The following steps are needed to implement YOLOv8: extending the dataset by image augmentation, training YOLO on the dataset, and transforming the YOLO output to the range and bearing measurements. These steps will be discussed in this section.

3.4.1 The YOLO dataset

To create the image segmentation dataset, it is vital to first look at the pitfalls of creating a dataset. Diwan, Anirudh, and Tembhurne [17] state important factors often overlooked in image detection datasets: multi-scale training, foreground-background class imbalance, and detection of relatively smaller objects. Therefore, the dataset should contain different image sizes, with the landmarks at different distances from the camera. Kaur and Singh [18] emphasise the importance of a dataset with variety. They state that common pitfalls also include a lack of variety in the pose and orientation of an object and a lack of inclusion of objects easily mistaken for the detected object. According to Kaur and Singh [18], it is common in data collection to include different lighting conditions in the dataset as well. Therefore, the dataset must contain different angles and poses of the landmarks and different lighting conditions. Objects easily mistaken for coloured landmarks, such as objects with bright colours, white parts, or different poles, must also be included in the dataset.

The dataset includes images from both the Nao robot and different football fields to ensure variety. The landmarks have been placed in several positions, such as lying on the ground, standing up, or at an angle, in different locations. Although the landmarks are indoors, different lighting conditions are included, such as near the window with open or closed curtains. Images such as coloured paper and paper tubes are also included to prevent false positives. Examples of these images are shown in figure 3.6. In the leftmost image, foreground and background objects are combined. The middle image shows objects easily mistaken for landmarks, and the rightmost image shows the landmarks on another football field.

Image augmentation

A large dataset is essential for image detection and segmentation models to perform as well as possible. Because of time constraints, creating a significant dataset with as much variety as

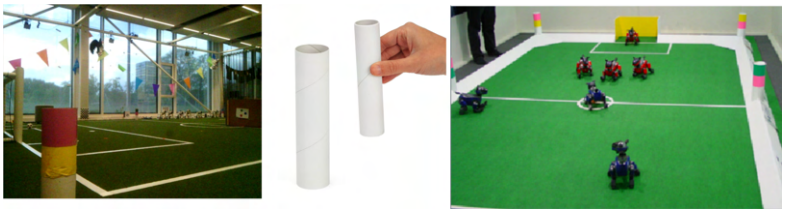


Figure 3.6: Some examples of images from the dataset before the data augmentation. The leftmost image contains the landmarks on the same Nao robot football field on which the final dataset is recorded in the foreground and background. The image in the centre shows white paper tubes to prevent the model from classifying every white tube as a landmark. The image on the right contains a smaller field with Aibo robots playing football with the landmarks on the edges of the field.

needed is difficult. Therefore, image augmentation in the dataset can help expand the dataset and add more variety. Image augmentation in this context would mean editing the images to generate new images. The most common augmentations can be divided into three categories: colour operations, such as brightness and contrast; geometric operations, such as rotations and translations; and bounding box operations, where only the pixel content within the bounding box is modified [19]. Zoph et al. [19] research what data augmentation strategies are most effective in improving the mean average precision of object detection models and if that set of augmentations is specific to the type of object detection model being used. They found that the most effective image augmentations were bounding box operations, such as replacing and scaling the objects, and geometric operations, specifically rotations. They emphasise that combining different types of data augmentation, such as the three types discussed above, was especially important. They also found that the mean average precision improvements were highest for small datasets, deeming this strategy useful for this thesis.

The dataset of 226 images was annotated in Roboflow[4] and split between 70% to train, 20% to test and 10% to validate. Roboflow offers several image augmentations, of which the following have been used to ensure a balance between colour operations, geometric operations and bounding box operations.

- **Rotation**: A rotation of the image between ±15°.
- **Shear**: A shear applied to the image within the range of ±11°, both vertically and horizontally.
- **Saturation**: An adjustment in the saturation of the image between ±10%.
- **Brightness**: An adjustment in the brightness of the image between ±20%.
- **Bounding box rotation**: A rotation of the part of the image within a bounding box between ±5°.
- **Bounding box brightness**: An adjustment in the brightness of the part of the image within the bounding box between ±15°.
- **Bounding box blur**: The part of the image in the bounding box was blurred up to 1.5px.

From each image in the training set, seven more images were created by randomly selecting augmentations from the list above. This resulted in 1180 total images. Figure 3.7 shows multiple examples of these augmentations, including image and bounding box augmentations.

[4]https://universe.roboflow.com/madelon-bernardy/guttman-colored-poles-dataset-spl-league

Figure 3.7: Some examples of images from the dataset after image augmentation. The leftmost and rightmost images contain the landmarks on the same Nao robot football field on which the final dataset is recorded with a rotation, bounding box brightness, and a minimal bounding box rotation. The centre image contains a picture of a smaller Aibo football field with bounding box brightness.

3.4.2 Training YOLOv8

YOLOv8 offers five sizes of their model: nano, small, medium, large, and extra large [6]. The smaller model is faster than the larger models whilst still being more accurate than the nano model. Due to SLAM being a problem that has to be solved in real-time, the small model appears to be the logical choice. To check whether this assumption is correct, both the small and medium models have been trained on the coloured landmarks dataset.
For the amount of epochs to use, Roboflow recommends 300. However, the small model was initially trained at only 100 epochs due to the relatively small dataset size. The training results can be seen in figure 3.8.

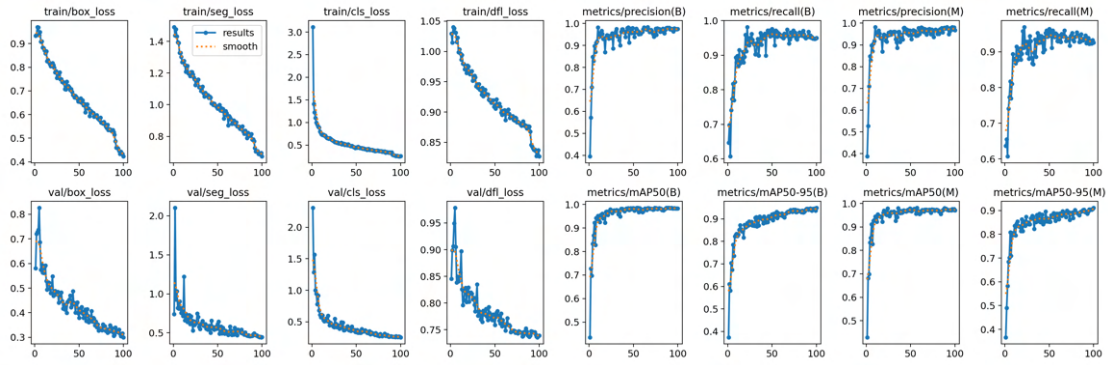


Figure 3.8: The training graphs of the YOLOv8 small model over 100 epochs

As can be seen in figure 3.8, the box loss and segmentation loss were still declining. Therefore, the decision to train the model 200 epochs was made. The training results can be seen in figure 3.9. As seen in the validation segmentation loss, the model starts to over-fit the data as it reaches epoch 200. Epoch 130 was the most optimal.
The medium-size model was initially trained at 200 epochs due to the model being larger. The training graph for this model can be seen in figure 3.10. Although the segmentation and box training losses are decreasing, the validation segmentation loss is starting to increase, and the validation box loss is not improving. That is why the decision was made not to train this model further.

3.4.3 From detected landmarks to range and bearing

YOLO results must be transformed to range and bearing measurements like the colour filter. How this is done for the colour filter is described in section 3.3.2. The formula for the bearing used in the colour filter is also used for the YOLO models, using the centre of the segmentation mask as the centre of the landmark. The range is calculated in a similar way; the same pictures for each pole at different distances were used. Instead of the distance between the two colour segments, the entire width and the entire height of each landmark were recorded. Then, interpolation was performed over the average measurements per distance to achieve two formulas for the range:

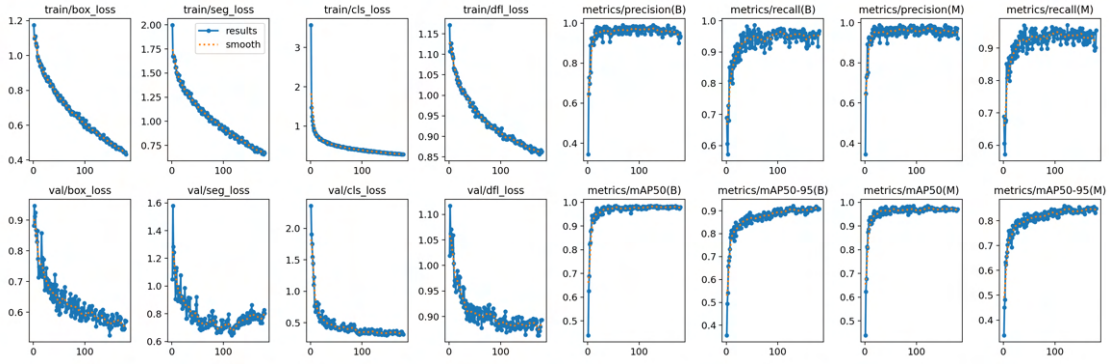


Figure 3.9: The training graphs of the YOLOv8 small model over 200 epochs. The best result was achieved at epoch 130.

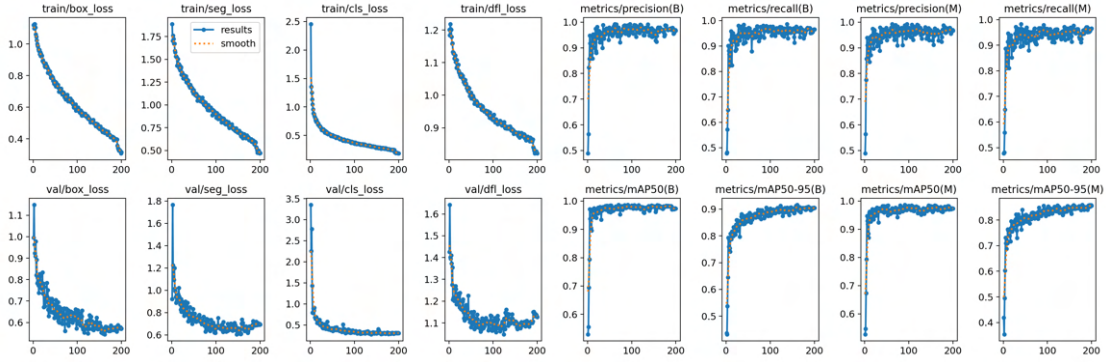


Figure 3.10: The training graphs of the YOLOv8 medium model over 200 epochs.

one based on the width, shown in equation 3.3, and one based on the height, shown in equation 3.4.

$$range_m = 70.057 width_{px}^{-1.109} \tag{3.3}$$

$$range_m = 209.16 height_{px}^{-0.965} \tag{3.4}$$

The width and the height are in pixels for this equation, instead of pixels-per-centimetre as in the colour filter range equation given in 3.2. The measurements are in pixels because all landmark widths and heights are the same when considering the white part. The width and height of the landmarks are determined by fitting a rotated bounding box on the segmented areas. To determine whether the width or height should be used to calculate the range is based on the width-to-height ratio of the landmark. If this is larger than expected, the width is taken. If it is smaller than expected, the height is taken.

3.4.4 Testing the YOLO models

This thesis uses an image segmentation model for landmark detection. Intersection over Union is a metric commonly used for image segmentation, whereas the mean average precision is often linked to object detection [20]. Hence, the IOU metric will be used to determine how accurate the localisation of the segmentation is.

Because the goal is to compare the YOLO method with the more traditional method of colour detection, the same test will be used. That is, 150 images are randomly taken from the first recorded datasets on the Nao, and the amount of true or false positives and true or false negatives are compared per image. A ground truth segmentation for each pole was created by hand. The IoU metric will be used to determine the accuracy of the localisation. For the results of these tests, see section 4.3.

3.5 Overview of the tests

The previous sections covered how to create the individual elements needed to answer the research question and sub-questions and how to test those elements. This section will give a short overview of how each element will be tested and what the tests will be to answer the research question. The following tests will be performed to test the individual elements that together will perform SLAM:

- Analysing the dataset: the dataset will already be used, and thus tested, in the final test. However, the dataset will first be analysed by comparing the odometry and ground truth information. The video aspect of the dataset is tested in the front-end tests.
- Testing the back end: the extended Kalman filter versions will be executed on the dataset by Gutmann and Fox [7]. The average error in location and landmark position will be recorded to estimate how reliable this filter is.
- Testing the front-end colour filter: 150 images from the first dataset recorded on the Nao will be taken. The number of true positives, false positives, true negatives and false negatives will be recorded for each image.
- Testing the YOLO filters on the front end: on the same 150 images from the colour filter test, both YOLO filters will be tested, and the number of true positives, false positives, true negatives and false negatives will be recorded. The IoU for each detected landmark will also be calculated for the YOLO segmentation model.

The final experiment will combine the front and back ends into one final test on data from the Nao robot. Different back-end and front-end combinations will perform SLAM on this dataset. The average error in location and landmark position will be recorded, showing how different front and back ends compare in accuracy.

CHAPTER 4

Experiments

This section will describe the results of each experiment. The previous chapter described the results of the training of the front end; thus, here, the focus lies on the validation based on the recorded dataset. These experiments include testing the colour filter and YOLO on 150 images and testing the back-end on the dataset by Gutmann and Fox [7]. Finally, these elements are combined to produce EKF SLAM on a Nao robot.

4.1 Testing the back end

The Gutmann dataset is used to test different versions of the back end [7]. Only the first ten minutes are considered for the tests since the Nao dataset lasts around one minute; ten minutes is thus enough to determine how well the back ends would perform on the Nao dataset. As stated by Gutmann and Fox [7], the ground truth of this dataset was recorded by logging when the robot walked over marks on the field. Therefore, there can be a certain error in the ground truth of the robot localisation since there can be a small delay between the robot standing on the marker and that event being logged. This possible inconsistency is why the error in landmark position may be a more accurate performance measure.

All individual tests have three graphs: one containing the robot localisation over time in metres, one containing the landmark localisation error over time, and one containing a map of the environment. This map includes the robot's believed path, the ground, and all the landmark positions and their believed positions. All these tests used the R matrix described in equation 4.1 and the Q matrix described in equation 4.2. Changing the third diagonal element of Q did not impact the results. The lack of impact is possible because the signature of each feature is removed from the state and covariance in the code.

$$R = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 130 \end{bmatrix}^2 \tag{4.1}$$

$$Q = \begin{bmatrix} 130 & 0 & 0 \\ 0 & 130 & 0 \\ 0 & 0 & 10^{16} \end{bmatrix}^2 \tag{4.2}$$

The average landmark position was updated every ten times, and its standard deviation every twenty times the state was updated. Thus, gaps in the landmark location broadly correspond to rejected measurements or a lack of measurements.
Between the different graphs, there are some correspondences: first, note how each robot localisation graph follows the same structure: around 100, 200 and 300 seconds, almost all the robot localisation show peaks. Visser, Bos, and Molen [21] show that the covariance of the robot increases when the robot turns and decreases when the robot walks in a straight line. Although

the recording method of the ground truth makes it difficult to confirm whether the turns cause the peaks, the error drops again after the robot walks in a straight line. This makes it more likely that turns do cause localisation errors.

The results of the classic EKF SLAM algorithm without modifications can be seen in figure 4.1. The landmark localisation is quite high. However, the landmark locations appear to have the correct distance from one another; the map appears to be at an angle of the ground truth.

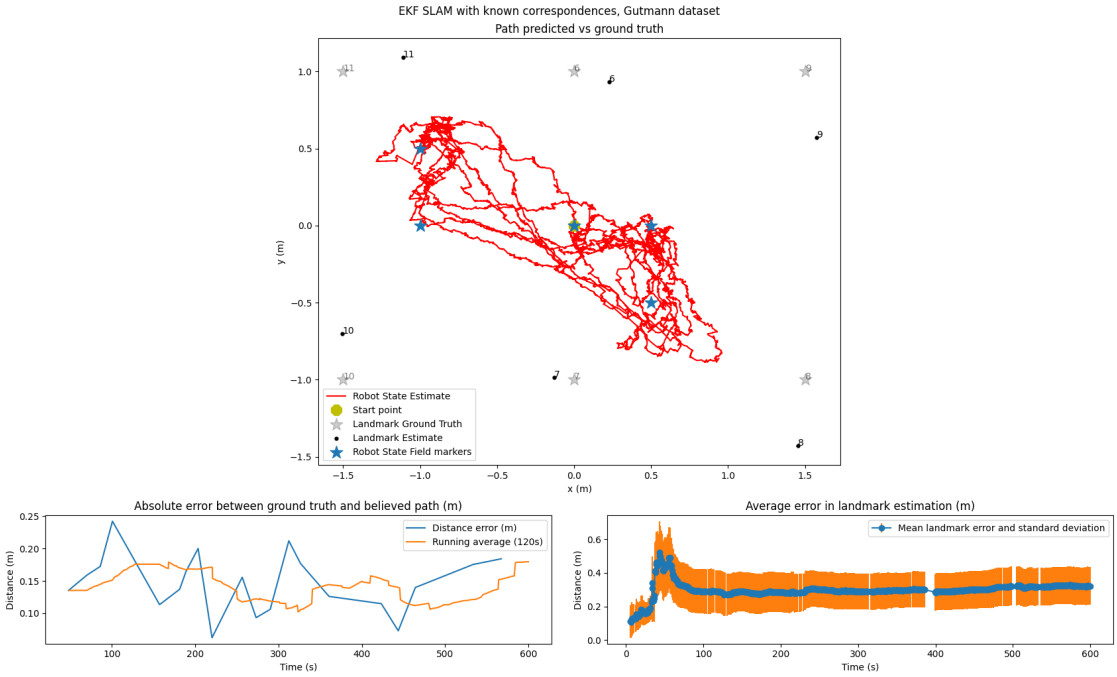


Figure 4.1: The performance of the classic EKF filter on the Gutmann dataset. The ground truth of the robot's location is measured each time the robot walks on the field markers, denoted with blue stars.

In this paper, multiple additions are tested to improve the EKF filter. The first addition tested is outlier rejection; if the difference between a measurement's bearing or range exceeds a threshold, it is discarded. The reader can find the results in figure 4.2. Although the landmark error peaks are initially smaller, eventually, the robot lacks the needed measurements to correct its position, causing the robot localisation error to increase. This can be seen in the map and in the robot localisation error at approximately 380 seconds. It appears that after the robot localisation error, a landmark localisation error occurs at approximately 400 seconds. When the location error is larger, more measurements exceed the threshold, resulting in the robot rejecting correct measurements. This causes localisation errors to grow, resulting in the peak of the robot and landmark localisation errors.

The second addition to EKF SLAM is another filter, the minimal view EKF, where a landmark is only considered when it is already seen n times. Because the EKF back end does not know whether a measurement is a false positive or not if it is the first time the landmark is seen, the previous filter would accept some false positives. When a landmark is only considered if seen in roughly the same location n times, fewer false positives are used as measurements. However, this also implies that the first few measurements will not be considered; hence, the odometry data in the beginning needs to be reliable. Choosing n too large causes the odometry data to be more prevalent, while choosing n too small can cause some false positives to be seen as landmarks. This dataset found the best results when n is three. These results are shown in figure 4.3. Although the two rightmost landmarks are still lower than the ground truth, this difference is less than with the classic EKF filter. The bottom-left landmark is also located too

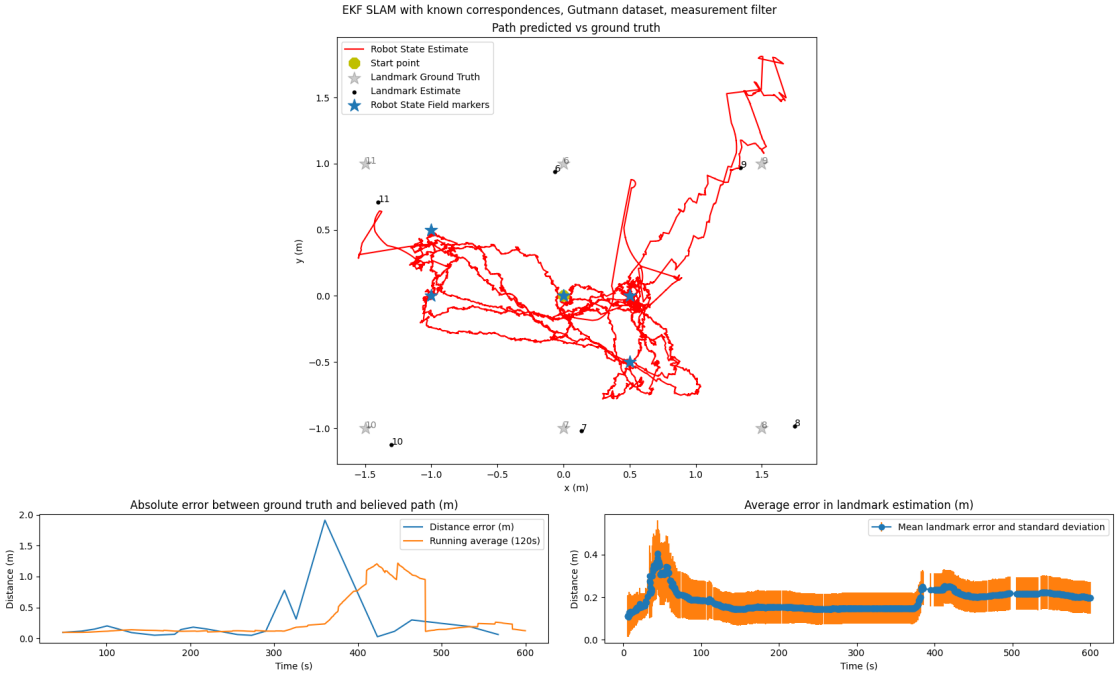


Figure 4.2: The performance of the classic EKF filter on the Gutmann dataset, with outlier rejection. The ground truth of the robot's location is measured each time the robot walks on the field markers, denoted with blue stars.

high: this indicates that the minimal view EKF lessened the rotation between the map and the ground truth map in comparison with the classical EKF, which is visible in figure 4.1.

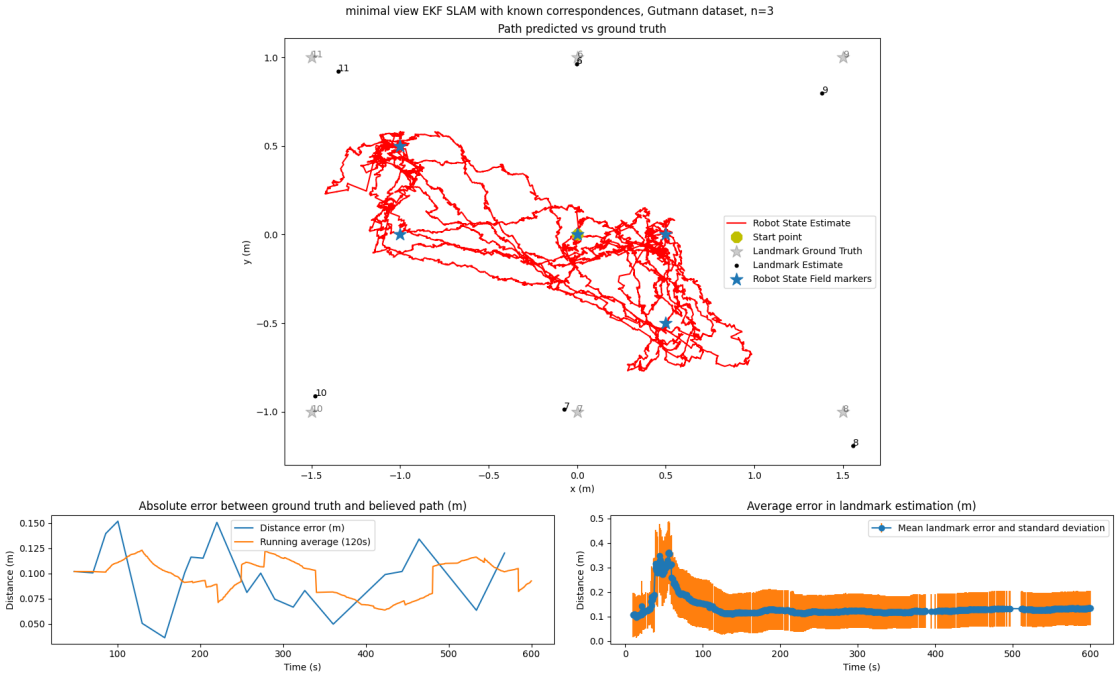


Figure 4.3: The performance of the classic EKF filter on the Gutmann dataset, with a minimal view filter and $n = 3$. The ground truth of the robot's location is measured each time the robot walks on the field markers, denoted with blue stars.

The final addition to EKF SLAM is to rewrite the filter to a first-estimates Jacobian EKF, as described by Huang, Mourikis, and Roumeliotis [8]. The results are shown in figure 4.4. Although the initial peak in landmark localisation did not improve significantly, the filter restores itself much better. This is most likely due to the FEJ-EKF filter not becoming overconfident. Although most landmarks are estimated to be below the ground truth, the classic filter's rotation does occur in this graph. These findings suggest that the FEJ-EKF back end might perform the best on longer datasets.

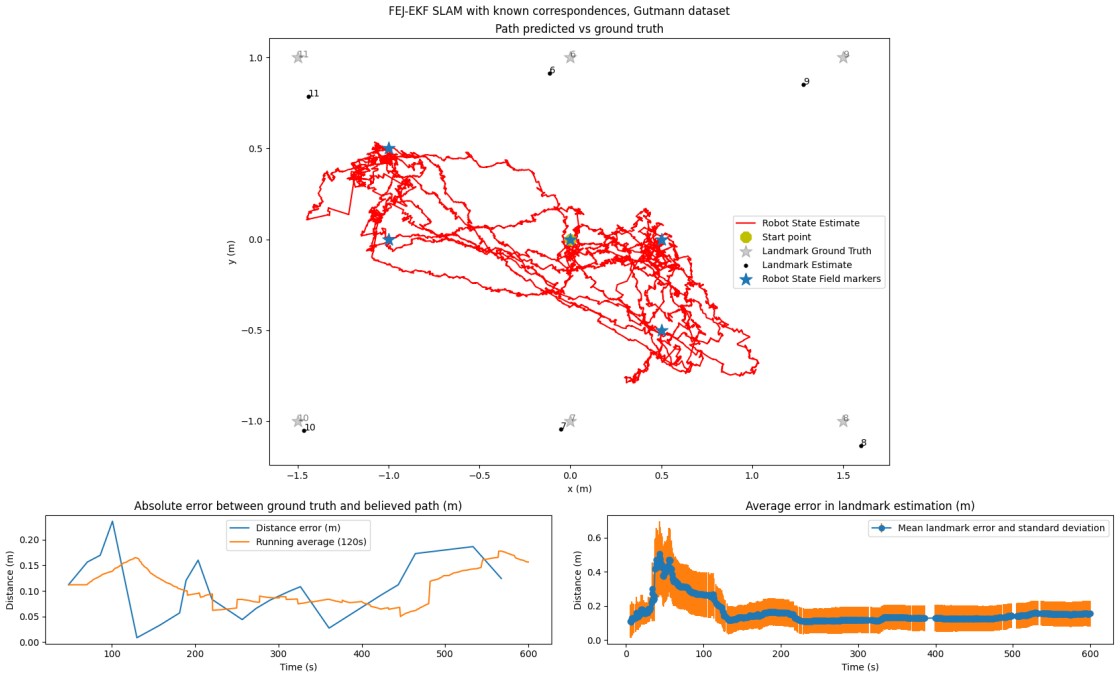


Figure 4.4: The performance of the FEJ-EKF filter on the Gutmann dataset. The ground truth of the robot's location is measured each time the robot walks on the field markers, denoted with blue stars.

These versions of EKF SLAM are compared in table 4.1. The best-performing filters are the minimal view EKF with $n = 3$ and the FEJ-EKF. The minimal view EKF with $n = 3$ produces better results when the average is taken. However, as discussed previously, the FEJ-EKF stabilises itself better and prevents the map from being rotated from the ground truth. **This would make FEJ-EKF the more stable algorithm over time**.

	Landmark localisation error (m)		Robot localisation error (m)	
	Mean	Standard deviation	Mean	Standard deviation
classic EKF SLAM	0.2990	0.05240	0.1486	0.04544
filter EKF SLAM	0.1870	0.04787	0.2451	0.4142
minimal view EKF SLAM, n=5	0.2755	0.05088	0.1458	0.08423
minimal view EKF SLAM, n=3	**0.1369**	0.04122	**0.09702**	0.03284
FEJ-EKF SLAM	0.1662	0.07848	0.1072	0.05832

Table 4.1: Comparison of different versions of EKF SLAM based on landmark and robot localisation errors.

4.2 Testing the colour filter

Figure 4.5 gives the results for the colour filter tested on 150 random images taken from a moving robot. The localisation error in pixels and correct and incorrect detections are given for each image. Out of the 150 images, there are 75 true positives, 56 true negatives, two false positives and 39 false negatives. This makes the precision $\frac{75}{75+2} * 100 = 97.4\%$. The recall is $\frac{75}{75+39} * 100 = 65.8\%$. The average localisation error is 8.20 pixels, with a standard deviation of 7.17 pixels.

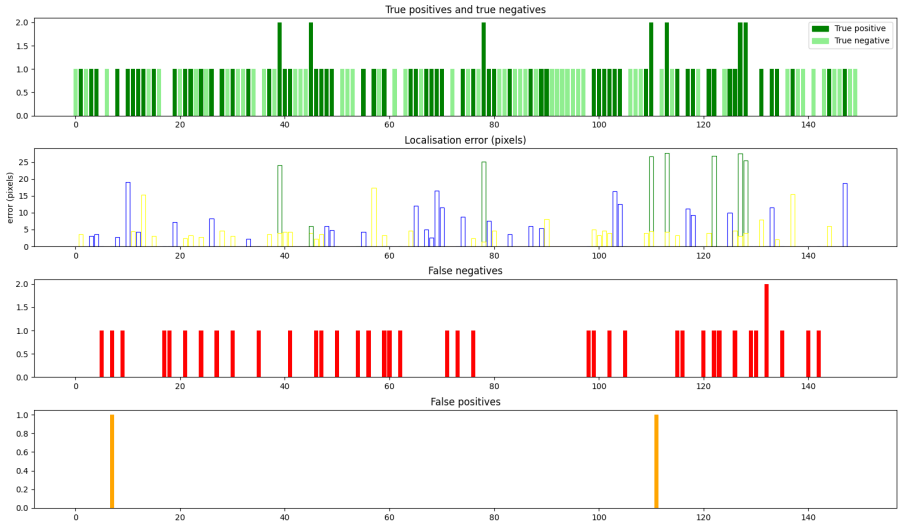


Figure 4.5: The results for the colour filter test. The topmost graph represents the amount of true positives detected or if a true negative is detected. The localisation graph shows the localisation error in pixels for each detected colour. The final two graphs show the false positives and false negatives.

Figure 4.6 shows the two false positives. One detection is in the pink flag, and a building in the background, and the other is in the pink, yellow, and white part of the poster in the background. Both falsely detected landmarks are relatively small and in the background, causing them to be easy to filter.

Figure 4.7 gives two examples of false negatives. In figure 4.7a, the contrast between the background and the landmark leads to the landmark being quite dark, causing the filters not to detect the colour. Because the field on which the datasets are recorded is close to a window, many images in datasets recorded on this field will have high contrast. Given that the colour filter does not succeed in detecting dark landmarks, this renders the colour filter less usable in this environment. Both image 4.7b and 4.7a have motion blur because the pictures were randomly selected from a dataset recorded on a moving robot. This could also decrease the likelihood of landmarks being detected since the colours might blend. In figure 4.7b, two landmarks are present, but only the left landmark is detected. The reason why the landmark is not detected is not entirely apparent. The undetected landmark is closer to the window. This contrast may cause one side of the landmark to be brighter, making one side too white to detect and the other too dark. This would leave a relatively thin stripe in the proper colour range, which the filter might not detect. Both false negative examples show that this version of a colour filter struggles in environments with high-contrast lighting conditions.

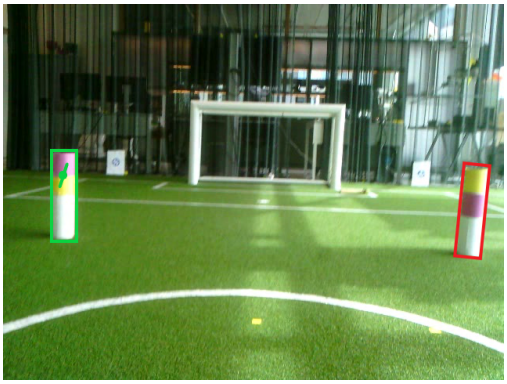
Two examples of where the localisation error is quite large are given in figure 4.8. In figure 4.8a, the believed centre is far to the right, while in figure 4.8b, the believed centre is far to the left. Both examples have lighting from one side. This leads to the lighter side being more often recognised as a landmark, whilst the colour filters do not recognise the darker side. As discussed, the colour filter uses colour masks defined in the HSV space to determine where a certain colour

(a) Example of a false positive detection in the pink flag and the building, as well as a false negative detection of the green landmark

(b) Example of a false positive detection in the poster.

Figure 4.6: The two false positives in the colour filter test. The correctly recognised landmarks are highlighted in green, and the incorrectly recognised landmarks are highlighted in red. All detections recognised as landmarks have a circle in the believed centre and a line between the two colours representing the believed length.

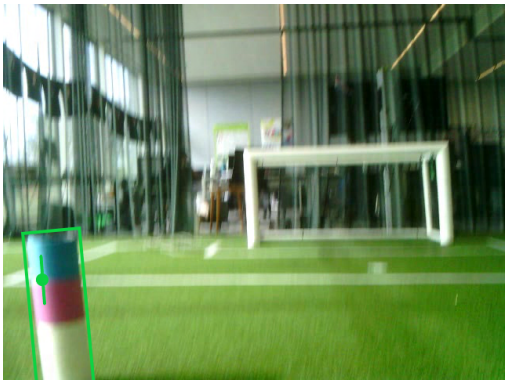
(a) Example of false negative detection of the green-pink landmark.



(b) Example of a false negative detection of the yellow-pink landmark.

Figure 4.7: Two examples of false negatives in the colour filter test. The correctly recognised landmarks are highlighted in green, and the incorrectly recognised landmarks are highlighted in red. All detections recognised as landmarks have a circle in the believed centre and a line between the two colours representing the believed length.

(a) Example of a relatively large localisation error on the green-pink landmark.

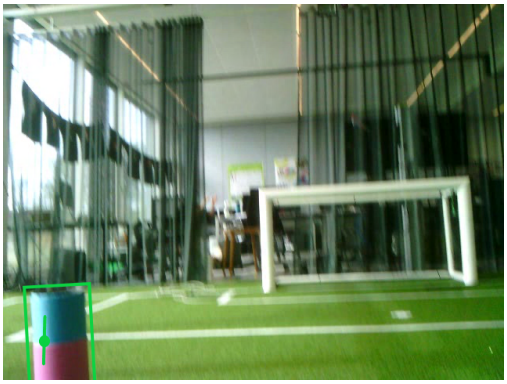


(b) Example of a relatively large localisation error on the blue-pink landmark.

Figure 4.8: Two examples where the localisation error is large in the colour filter test. The correctly recognised landmarks are highlighted in green, and the incorrectly recognised landmarks are highlighted in red. All detections recognised as landmarks have a circle in the believed centre and a line between the two colours representing the believed length.

is present. These masks also have a minimum value, meaning darker colours go undetected.

With the algorithm described in section 3.3, some edge cases are expected to fail. For example, when the white part of the landmark is invisible, the final test fails, causing the landmark not to be recognised. Two peculiar cases of true positives are shown in figure 4.9. In figure 4.9a, there is no clear white bottom of the landmark visible, yet the landmark is still recognised. The landmark is so light that the white filter also recognised some pink and blue parts as white. In figure 4.9b, the pink-yellow landmark is slightly visible and bright due to the lighting. Yet the landmark is still recognised, possibly because the pink and yellow filters accept bright colours. Another reason this landmark was recognised might be that the landmark stood straight, causing the algorithm to detect the other colours more easily.



(a) Example of a true positive even though the white part is invisible.

(b) Example of a true positive even though a large part of the landmark is not visible.

Figure 4.9: Two examples of true positives in the colour filter test. The correctly recognised landmarks are highlighted in green, and the incorrectly recognised landmarks are highlighted in red. All detections recognised as landmarks have a circle in the believed centre and a line between the two colours representing the believed length.

	Precision	Recall	Average localisation error
Colour filter	**97.4%**	65.8%	8.20 pixels
YOLOv8s	92.9%	98.3%	IoU: 0.8714
YOLOv8m	94.4%	**99.2%**	IoU: **0.8783**

Table 4.2: Comparison of the different front ends by precision and recall. The precision and recall are calculated using 150 random images from a walking NAO robot.

4.3 Testing YOLO

This section discusses the performance of both YOLO models. In section 4.4, the results from combining the front-end and back-end can be seen.
The results for YOLOv8s on 130 epochs can be seen in figure 4.10. The results for YOLOv8m on 200 epochs can be seen in figure 4.11. In the test of the small-sized model, there are 117 true positives, 54 true negatives, nine false positives and two false negatives. This makes the precision $\frac{117}{117+9} * 100 = 92.9\%$ and the recall $\frac{117}{117+2} * 100 = 98.3\%$. The average IoU of the medium-sized model is 0.87. In the test of the medium-sized model, there are 119 true positives, 51 true negatives, seven false positives and one false negative. This makes the precision $\frac{119}{119+7} * 100 = 94.4\%$, and the recall $\frac{119}{119+1} * 100 = 99.2\%$. The average IoU of the medium-sized model is 0.88. Since these models will ideally be used to perform SLAM, precision and recall are important: the higher the recall, the more information the robot has. The higher the precision, the lower the error on the localisation may be. Both results look promising for performing EKF-SLAM. Table 4.2 compares the different front-ends. Although the precision of the colour filter is slightly higher, the recall indicates that the YOLO models provide more data, giving the robot more data to improve its map.

In figure 4.12, six images with the YOLO detections are shown, both for the small and medium-sized models. The first two figures, 4.12a and 4.12b, detect a piece of yellow tape as a landmark. The medium model is less confident that this is a landmark than the small model. The small-sized model also recognises the goalpost as a landmark. Both models had high confidence scores for all the wrong segmentations since all were higher than 49%. The second two figures, 4.12c and 4.12d, fail to detect the green-pink landmark, possibly due to the high contrast in the pictures. However, the small model fails to detect this landmark and classifies the yellow-pink landmark as two separate landmarks, leading to another false positive detection. The medium-sized model is less likely to create these mistakes. In the final two figures, 4.12e and 4.12f, the second false negative result of the small model is compared with the medium model. Again, the false negative is caused by a wrong classification. The medium model outperforms the small model based on classification, and the segmentation is also more accurately fitted on the blue-pink landmark.

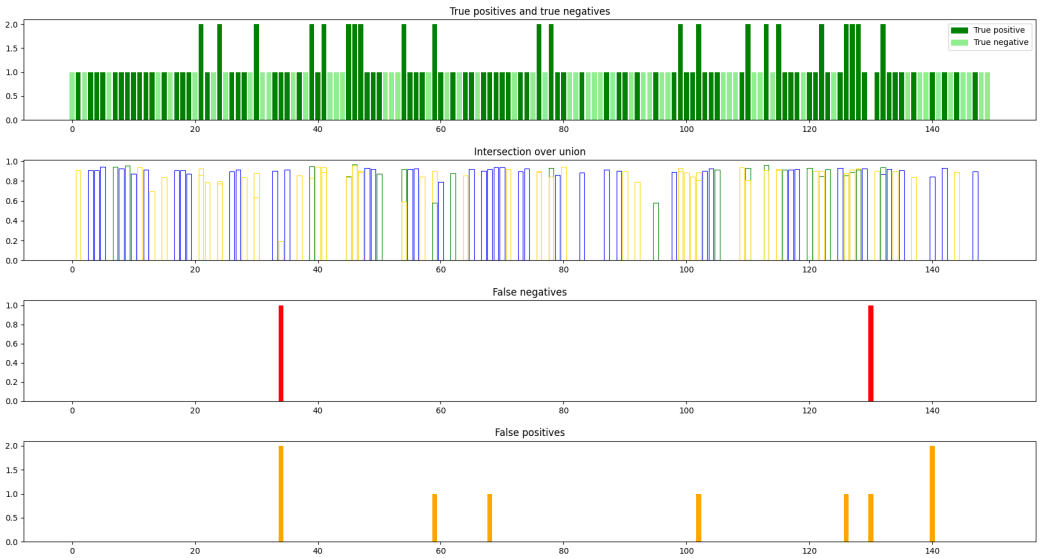


Figure 4.10: Results of the yolov8s model trained for 200 epochs. Tested on 150 images

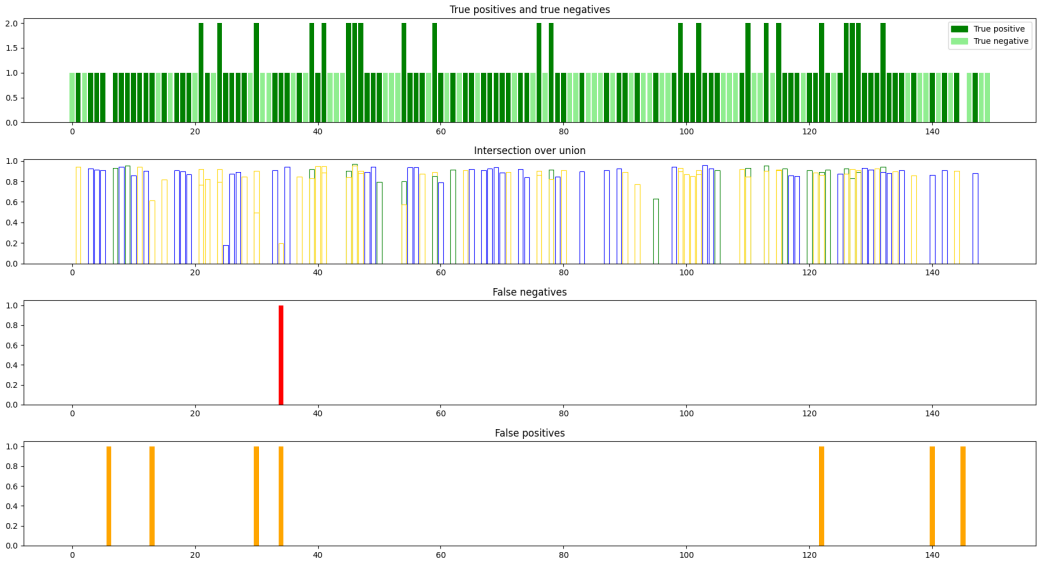
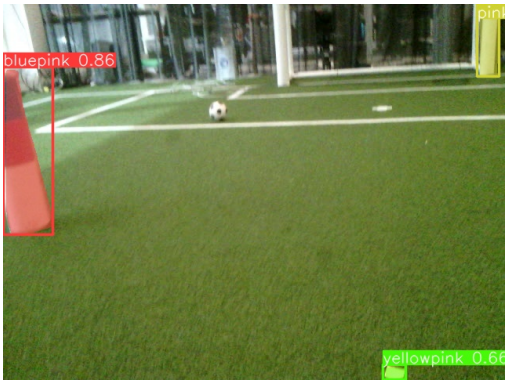
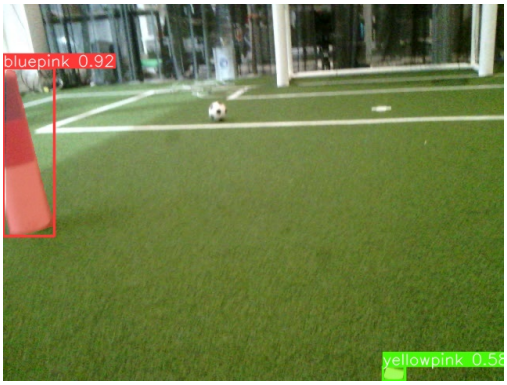


Figure 4.11: Results of the yolov8m model trained for 200 epochs. Tested on 150 images



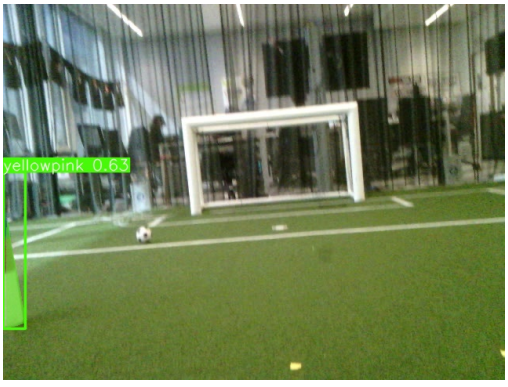
(a) Example of false positives detected with the small model. Blue-pink detection: true positive, confidence: 85.7%. Yellow-pink detection in the tape on the field: false positive, confidence: 66.3%. Pink-yellow detection in the goal post: false positive, confidence: 49.1%.



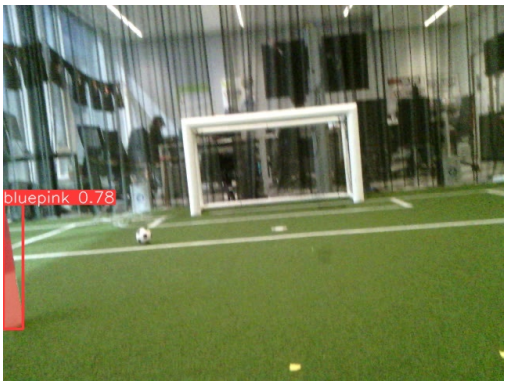
(b) Example of false positives detected with the medium model. Blue-pink detection: true positive, confidence: 92.1%. Yellow-pink detection in the tape on the field: false positive, confidence: 58.5%.

(c) Example of false positives detected with the small model. Yellow-pink was detected three times, two times in the left pole, with confidences 48.6% (marked as true positive) and 34.0% (marked as false positive), one time on the right side, with confidence 27.5% (marked as false positive). The wrongly classified green-pink landmark is marked as a false negative.

(d) Example of false positives detected with the medium model. Yellow-pink was correctly detected one time with a confidence of 25.9%. The green-pink landmark is classified as a blue-pink landmark with confidence 49.5%, marked as a false positive. The wrongly classified green-pink landmark is marked as a false negative.



(e) Example of a false negative and false positive due to an error in classification with the small model. The false positive yellow-pink landmark has a confidence of 63.6%. The false negative is the wrongly classified blue-pink landmark.



(f) Example of a true positive detection in the blue landmark with confidence: 77.8% with the medium model.

Figure 4.12: Examples of false positives and false negatives in both the small and medium model.

In conclusion, both YOLO models from this test appear to perform better in terms of recall from these tests. When comparing the medium-sized model with the small-sized model, **the medium-sized model is often more accurate in terms of localisation, less likely to wrongly classify a landmark, but more confident in wrong classifications**.

4.4 Testing SLAM on the Nao dataset

4.4.1 The dataset

The dataset is seen in figure 4.13, including both odometry and ground-truth information. It can be seen that the bearing of the robot is increasingly inaccurate. The average robot localisation error is 0.5039 metres, with a standard deviation of 0.2030 metres.

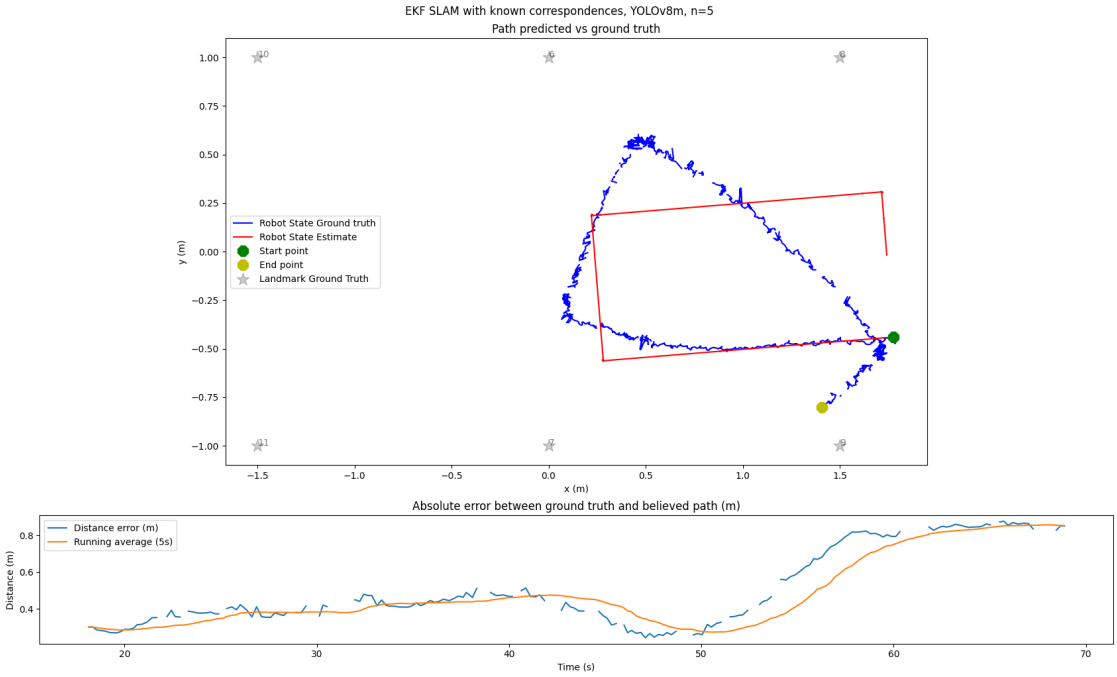


Figure 4.13: The dataset recorded on a Nao robot, with ground-truth data from OptiTrack and the robot's path considering only the odometry information.

4.4.2 The results

As discussed in section 3.5, the previously tested back ends and front ends are combined in this experiment. Section 4.1 shows that all three tested additions can potentially improve EKF SLAM, either in robot localisation, landmark localisation, or both. Because all additions can improve SLAM, in the final experiment, all these improvements of the back end are combined. Due to the Nao dataset having false positive detections, all methods tested have a filter. However, if the first time a landmark is detected is from a false positive measurement, the filter does not work; hence, the minimal view method is incorporated in each test. Since false positives may occur in multiple frames, the n value is larger than in section 4.1. FEJ-EKF SLAM is only tested on the lower value of n because the first estimates are important in this filter. When n increases, the more the few landmark estimates depend on the odometry. Thus, the low n value will ensure the first estimates are not false positives and that their position does not mostly depend on the odometry.
For the back-ends of these tests, the values of the Q and R matrices have been varied per test to ensure the best results. The range per element of the R matrix is shown in equation 4.3 and

is shown in equation 4.4 for the Q matrix.

$$R = \begin{bmatrix} [5,25] & 0 & 0 \\ 0 & [5,25] & 0 \\ 0 & 0 & [75,130] \end{bmatrix}^2 \tag{4.3}$$

$$Q = \begin{bmatrix} [100,150] & 0 & 0 \\ 0 & [100,150] & 0 \\ 0 & 0 & 10^{16} \end{bmatrix}^2 \tag{4.4}$$

As shown in section 4.3, YOLOv8m often estimates higher percentages for false positives than YOLOv8s. However, the model is less prone to detecting false positives. Hence, for these tests, the YOLOv8m front end only used measurements with a confidence higher than $87,5\%$, while the YOLOv8s front end only used measurements higher than 85%.
All previously discussed versions of the front end are compared. Although YOLO appeared quite resilient to false positives in section 4.3, the minimal view method still proved useful in preventing wrong landmark locations.
For the results of the different SLAM versions on the Nao dataset, see table 4.3.

			Colour filter	YOLOv8s	YOLOv8m
Classic EKF	Landmark localisation	Mean error	1.758	0.3005	0.7320
		Standard deviation	0.2704	0.07372	0.02652
	Robot localisation	Mean error	0.4345	0.4150	0.2812
		Standard deviation	0.2092	0.2593	0.1256
minimal view EKF, n= 5	Landmark localisation	Mean error	**0.8162**	0.2722	0.2416
		Standard deviation	0.1102	0.02452	0.02507
	Robot localisation	Mean error	**0.2606**	0.2127	0.2502
		Standard deviation	0.09184	0.1303	0.1238
minimal view EKF, n = 10	Landmark localisation	Mean error	1.028	0.2954	0.2601
		Standard deviation	0.06283	0.009539	0.02408
	Robot localisation	Mean error	0.7314	0.1758	0.2181
		Standard deviation	0.2218	0.07410	0.1225
FEJ-EKF, n=5	Landmark localisation	Mean error	0.9297	**0.2656**	**0.2212**
		Standard deviation	0.03833	0.01788	0.002273
	Robot localisation	Mean error	0.4615	**0.1738**	**0.2079**
		Standard deviation	0.1833	0.06988	0.08213
Average of modifications	Landmark localisation	Mean error	0.9246	0.2777	0.2410
	Robot localisation	Mean error	0.4845	0.1874	0.2254

Table 4.3: This table gives the error in landmark and robot locations for different combinations of back and front ends in metres.

When comparing the front ends, these results give two findings. First, the YOLO models prove more robust than the colour filter. Second, in every test, the YOLOv8s model outperforms the YOLOv8m model in robot localisation. However, YOLOv8m outperforms YOLOv8s on landmark localisation. The best colour filter result is shown in figure 4.14 to clarify why the colour filter performs worse than YOLO. Surprisingly, the minimal view EKF with $n = 5$ is the best back end. Please note that the method of creating this graph is the same as for the results of the Gutmann dataset. Thus, the standard deviation of the landmark error is plotted at every other data point. As can be seen from the figure, a false positive landmark is detected. This detection impacts both landmark localisation and robot localisation. In every test for the colour filter, this landmark is detected at the wrong location; the false positive landmark is detected for multiple seconds, causing it to be challenging to filter. As the robot turns away from the false positive landmark, the robot follows the same shape of the ground truth path, implying that without false positives, the colour filter might perform quite accurately.
Next to this, the robot localisation error also has a different shape than would be expected. In section 4.1, the results of the Gutmann dataset all had clear peaks in the robot location. A plateau is formed here, which drops as the predicted location crosses the ground truth between the second and third turn and inlines after the lines have crossed. **This indicates that when the**

robot's bearing is incorrect between the second and third turn, the robot localisation might appear better than if the bearing were correct.

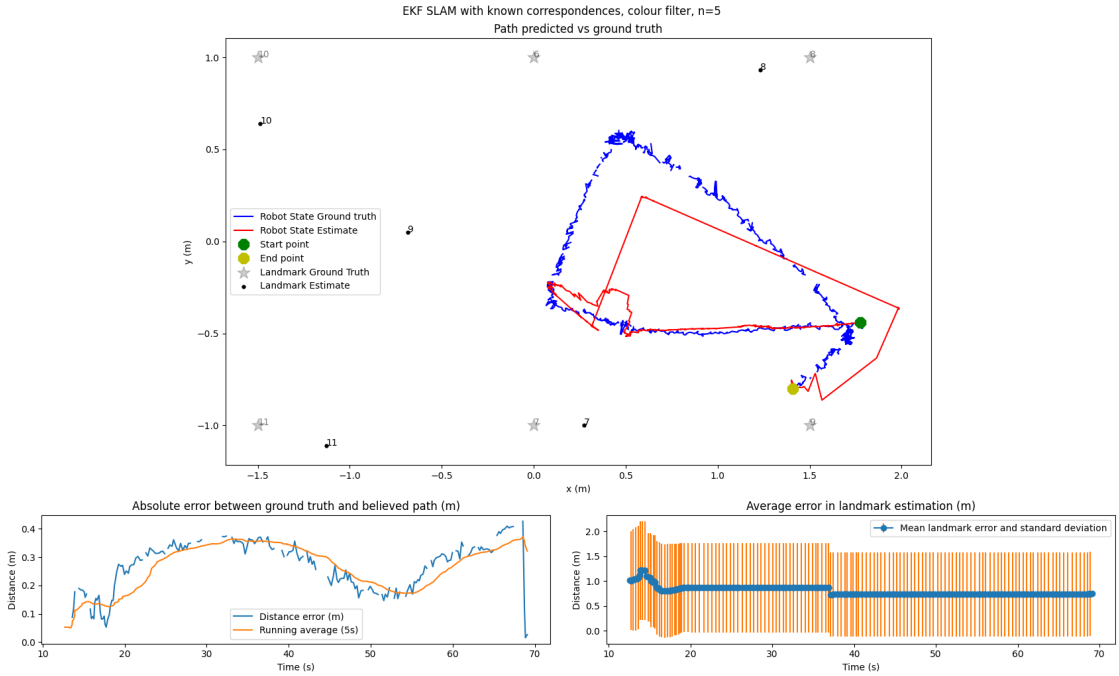


Figure 4.14: The performance of SLAM on the Nao dataset, with the colour filter as the front end and the minimal view EKF filter with $n = 5$ as the back end.

The best YOLOv8s and YOLOv8m based SLAM models are shown in figures 4.16 and 4.15, respectively, to clarify why YOLOv8s is better in robot localisation and YOLOv8m is better in landmark localisation. Note that figure 4.15 shows the same pattern in robot localisation as figure 4.14, further confirming that the robot localisation graph might be misleading when analysing how accurate robot localisation is.
The turns play a vital role in analysing the differences between YOLOv8m and YOLOv8s regarding landmark localisation. As can be seen, YOLOv8m appears to be more accurate in the first turn, as it turns towards the window. In the second turn, away from the window, the medium-sized model already detects landmark 9, which increases the location accuracy but decreases the accuracy of the robot's angle. Detecting landmark 9 in the second turn, whilst the small-sized model only detects the landmark in the final turn, is even more significant due to the incorporated minimal view. In the final turn, the medium model already has seen the landmark n times and thus can instantly use the measurements to update its location and map. At the same time, the small model still has to confirm the measurements as valid before they can be incorporated into the map. Thus, the earlier detection of landmarks may explain why the medium-sized model is better at mapping the environment.

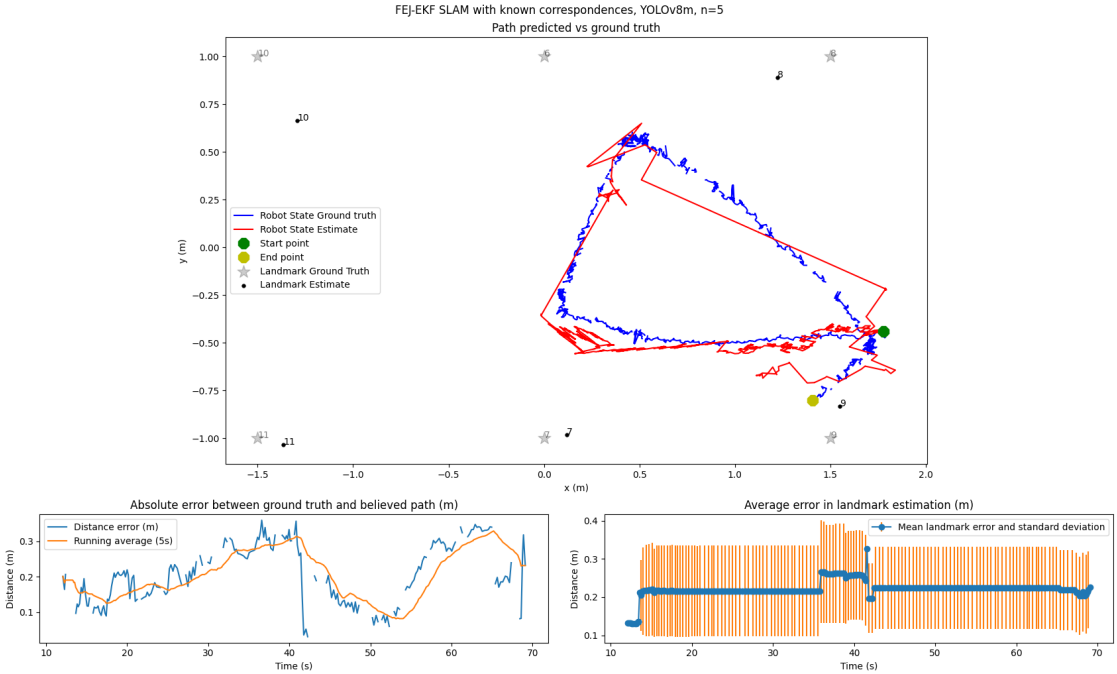


Figure 4.15: The performance of SLAM on the Nao dataset, with YOLOv8m as the front end and the minimal view FEJ-EKF filter with $n = 5$ as the back end.

The question of why the YOLOv8s is better at robot localisation remains. As discussed before, the first and second turns are crucial; between the first and second turns, there are hardly any measurements due to the contrast of the window. No landmarks are in view between the second and third turns. Consequently, the robot cannot use measurements to update its bearing at these time intervals. The medium-sized model detects the 9th landmark in the second turn; the first five frames are discarded due to the minimal view addition. The discarded frames may lead to a slight offset in the landmark detection, causing the bearing to be less accurate. The less accurate bearing may result in a less precise robot position. This reasoning would imply that the medium-sized model would outperform the small-sized model in environments with less contrast. Better localisation may also suggest that over time, the robot position of the medium-sized model will be more accurate.

Next to comparing front ends, comparing back ends gives essential insights as well. The results of section 4.1 showed that a higher n value would result in worse localisation results. Although this applies to landmark localisation, robot localisation improves when n is larger. As shown when analysing the front ends, the robot's bearing after the second turn can improve robot localisation more if it is incorrect than if the bearing were correct. Especially since the tests in section 4.1 over a longer period shows that robot localisation should be worse with a higher n, **this further confirms that the landmark localisation might be a more reliable metric when analysing the performance of SLAM.**

Performing SLAM on this dataset produces the best results when the FEJ-EKF back end is combined with a YOLO front end. Because of the assumption that landmark localisation is a more accurate performance metric, **in the long term, the medium-sized model will most likely outperform the small-sized model in both robot and landmark localisation**. In case of false positives, the FEJ-EKF back end might produce worse results than other back ends, as shown by the results of the colour filter combined with the FEJ-EKF. However, all tested back ends have relatively large mapping and localisation errors in such a case.

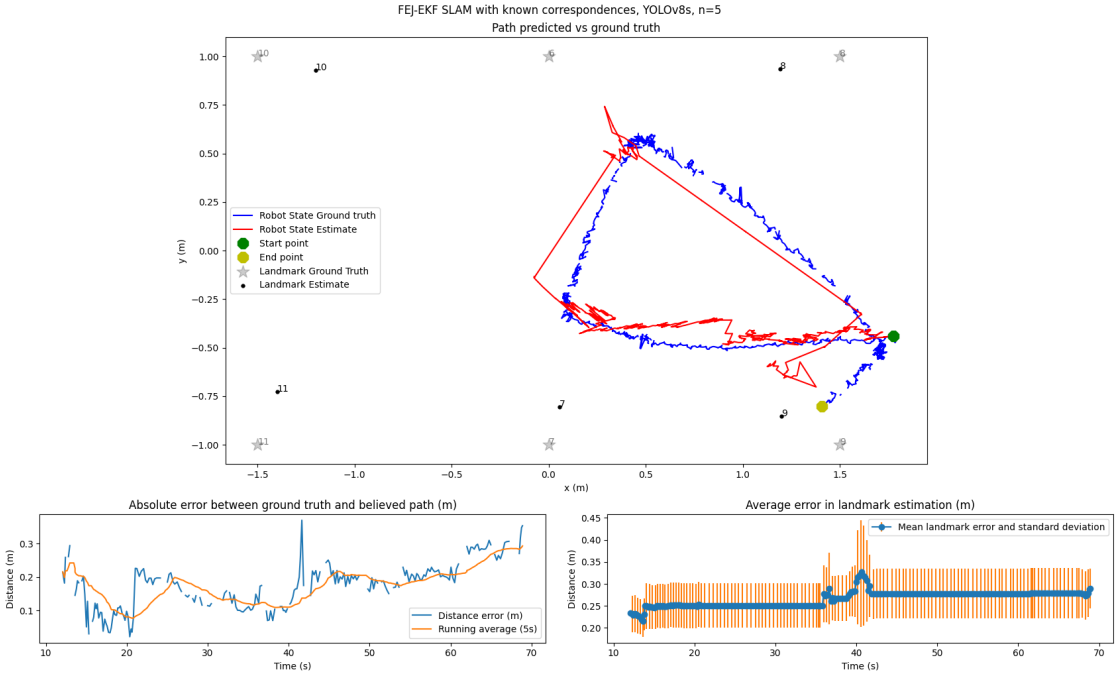


Figure 4.16: The performance of SLAM on the Nao dataset, with YOLOv8s as the front end and the minimal view FEJ-EKF filter with $n = 5$ as the back end.

4.5 Overview of the results

This section discussed the results of all the experiments conducted in this thesis. First, different versions of the back end were tested on the Gutmann dataset, and the FEJ-EKF back end performed the best. See table 4.1 for the results of this test. Secondly, the colour filter and the YOLO models were tested. Table 4.2 compares these front ends. The colour filter performs best on precision but worse on recall. The medium-sized YOLO model performs the best on recall, with a high precision. Finally, the different versions were tested on a dataset recorded on a Nao robot. These results are shown in table 4.3. On the front end, the YOLOv8s model is more accurate in estimating the robot position, while the YOLOv8m model is more accurate in mapping the environment. Of the back ends, the FEJ-EKF back end produces the most accurate results.

The next chapter will combine these results to answer the research questions.

CHAPTER 5

Conclusions

The research question of this thesis is: *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*. This research question is split into two sub-questions: *How do different versions of the Extended Kalman filter impact the performance of SLAM in the context of the robot football standard platform league?*, and *How do YOLOv8s and YOLOv8m improve the performance of EKF-based SLAM compared to a classic colour filter?*. This section aims to answer these questions.

To answer the first sub-question, different versions of the back end were tested on the Gutmann dataset in section 4.1. Table 4.1 shows the output for these tests. Both versions of the minimal view EKF and FEJ-EKF outperform the classic EKF in landmark and robot localisation. Adding a filter to EKF SLAM improves landmark localisation. However, the robot localisation worsened compared to the classic EKF. From these results, it can be concluded that both FEJ-EKF and minimal view EKF can improve the classic extended Kalman filter. Adding a filter based on already mapped landmarks can improve the map but may also lead to a larger robot localisation error.

The second sub-question compares YOLOv8s with YOLOv8m and a more classical feature extraction algorithm, namely a colour filter. Section 4.2 tests this colour filter. Section 4.3 tests two versions of the YOLOv8 model: the small and medium versions. Table 4.2 compares the performance of all three feature extraction methods on the same 150 images. Over the 150 images, all front ends had a precision of over 92%, with the colour filter having the highest precision but the lowest recall. Both YOLO models scored high on the recall, with the medium model scoring slightly higher. From these results, it can be concluded that the YOLO models provide SLAM with more measurements. Although the results from the colour filter are slightly more precise, table 4.3 shows that the colour filter in the final algorithm always has a more significant landmark localisation error. Thus, it can be concluded from these results that the YOLO models both provide more measurements and more precise information.

To answer the research question, the front and back-end versions are combined and executed on a dataset recorded on a Nao robot. These results are shown in figure 4.3. When comparing the YOLO models to the colour filter on classic EKF SLAM, the small model improves landmark localisation by 1.45 metres, and the medium model improves landmark localisation by 1.2 metres. The medium model improves the robot localisation most, with an improvement of 0.153 metres. Even when comparing the front ends with their overall performance in average modifications, both YOLO models improve both localisation errors by more than half when compared to the colour filter. When analysing the impact of additions to the back end, the conclusion can be drawn that for every front end, the lowest errors in robot and landmark localisation are found with additions to the back end. For both YOLO models, FEJ-EKF SLAM has the most impact on errors in both robot and landmark positions.

The sub-questions thus conclude that FEJ-EKF is the best back end out of the tested methods and that YOLOv8m is the best front end. Overall, this thesis thus concludes that SLAM can be significantly improved in localisation and mapping errors by incorporating YOLOv8 into the front end and modifying the back end to filter measurements and be less overconfident.

CHAPTER 6

Discussion

This research has formulated an answer to the research question: *How do localisation and mapping in EKF SLAM improve by using YOLOv8 on the front end and adding modifications to the back end in the context of the robot football standard platform league?*. This section reflects on the research's validity, limitations, and implications. Furthermore, this section interprets the results and suggests further research.

6.1 Validity and reproducibility of the research

This thesis took measurements to ensure the research was valid and the results were reproducible, both on the front and back end of the algorithm. The back end is tested not only on the Nao dataset but also on the Gutmann dataset. In both instances, the modified versions produce better results. Hence, the conclusion that adding outlier detection to and reducing overconfidence in EKF SLAM can improve the localisation is reproducible. The front end was tested over 150 images in different lighting conditions than the final dataset. Although this led to the colour masks having to be redefined, the ability of the algorithm to still detect landmarks in various settings was shown. The test on 150 images and the final test also showed that YOLO performs well on multiple datasets. Because the tests of the front end and back end are reproducible, it follows that the final test results are also most likely reproducible despite being tested on one dataset.

6.2 Interpretation of the results

The results in section 4.4 show that both front-end and back-end results significantly improve localisation and mapping errors. These results are not unexpected because YOLO is well-known for its excellent performance in object detection. Furthermore, Huang, Mourikis, and Roumeliotis [8] show that the back-end improvements should also have an impact. However, two aspects of the results are relatively unexpected. The first is that the FEJ-EKF SLAM performs the best on a small dataset out of the tested back ends, whilst the first Gutmann test implicated that the FEJ-EKF back end might lead to less accurate localisation on a small dataset. As discussed by Visser, Bos, and Molen [21], the Gutmann dataset contains many large bearing offsets, thus not all measurement data is reliable. Most likely, the FEJ-EKF back end did not perform the best on the Gutmann dataset, with no measurement filters. Still, it did perform well with filters for measurements on the final dataset.
The second unexpected result is that the medium-sized YOLO model does not outperform the small model in terms of both localisation and mapping accuracy, but only in terms of mapping accuracy. There are two explanations for this. The first has to do with the lack of measurements when facing the window, which heavily influences the bearing, which in turn causes the localisation on sections without visible landmarks to decrease. The second ties into this reasoning, stating that average robot pose error is not as reliable as a landmark localisation when it comes

to analysing the performance. Both reasonings are further explained in section 4.4. These explanations imply that the medium model will perform better on both the robot and landmark localisation, as was expected.

When comparing the results to other papers, the fact that the results are not unexpected is highlighted further. Visser, Bos, and Molen [21] implement both EKF SLAM and FastSLAM on the Gutmann dataset. They find that EKF SLAM performs the best on this dataset because there are only six landmarks. They apply a validation gate to filter invalid measurements. Their extended Kalman filter reaches a landmark localisation error below 0.1 metres. This corresponds with the best results from the test on the Gutmann dataset in this thesis; the localisation error of the minimal view EKF with $n = 3$ and FEJ-EKF stabilise around 0.1 metre from $t = 250s$ onwards. Although the filtering with a validation gate might work better, the results for this thesis lie in the same range of values.
Huang, Mourikis, and Roumeliotis [8] test FEJ-EKF SLAM against classic EKF SLAM and other versions of SLAM. They test these algorithms with Monte Carlo comparison studies, one where the robot moves in a straight line and one with loop closures. The FEJ-EKF SLAM always outperforms the classic EKF SLAM regarding landmark localisation, with an improvement of 30% on the RMS. In robot localisation, FEJ-EKF SLAM leads to an improvement of 17% on the RMS. When comparing FEJ-EKF $n = 3$ with minimal view EKF with the same n, there are improvements in the landmark and robot localisation. Still, these improvements are not in the same range as those from Huang, Mourikis, and Roumeliotis [8]. This difference might be because the authors state that the amount of noise they have added for their tests is more than would typically be encountered to highlight the performance of FEJ-EKF SLAM.

The results from this paper show that additions such as filters and improvements in consistencies lead to optimisations in EKF SLAM. More mathematical filtering approaches, such as a validation gate, are slightly more effective than the intuitive filtering approaches tested in this thesis. This thesis shows that EKF SLAM, primarily when improved, can be used to estimate a robot's position broadly and to estimate a map of the environment.

6.3 Limitations of the research

Although care has been taken to ensure these results are reliable and valid, the reader should bear in mind that this thesis mostly focuses on a single environment. The YOLOv8 models and the colour filter have only been tested on one football field. Combined with the fact that the dataset for YOLO was relatively small due to time constraints, this might result in the YOLO models being overfitted for this environment, even though images from other contexts have been included. They might produce many false positives in other environments, such as outdoor environments. Future research could prevent this by either expanding an existing dataset or taking enough time to ensure the dataset is large and diverse. The principles of the colour filter should work to detect the landmarks in multiple environments. However, as discussed in section 3.3, the colour masks rapidly became inaccurate as the lighting conditions changed. Thus, although the colour filter can be used as a baseline for a single dataset, it might fail when using the same colour filter on multiple datasets.
Furthermore, this thesis does not consider how well all front ends would compare on a dataset with less contrast. This would confirm whether the reasoning in 6.2 is correct. Future research could prevent this by testing the algorithm on multiple datasets on different fields.

6.4 Implications

This paper concludes that implementing YOLOv8 into the front end can significantly impact SLAM accuracy and that editing the EKF back end to incorporate filters and lessen overconfidence will also improve accuracy. This implies that as YOLO develops, robots' mapping and localisation algorithms can even further develop and become increasingly accurate. This does

not just apply to robot football but to other robotics applications as well, such as self-driving cars and rescue robots, for both of which localisation and mapping play a vital role [22][23].

6.4.1 Ethical remark

Improving localisation and mapping for Nao robots has limited ethical aspects because these robots are mainly used for research and entertainment[1]. However, as stated earlier, improvements in localisation and mapping impact all mobile robots. Thus, the ethical aspects of improving SLAM in the context of robot football correspond with the ethical aspects of improving autonomous robotics as a whole. Kopacek and Hersh [24] describe several robotics applications and their ethical implications. Next to rescue robots and autonomous vehicles, autonomous robots can also have harmful applications, such as armed security force robots and military robots. From a utilitarian point of view, it is difficult to determine whether improving localisation and mapping is ethical, for it is uncertain whether improved robotics will have a positive effect. From a deontological point of view, it could be argued that this thesis is ethical since improving robot localisation and mapping for non-harmful applications could fit into a rule-based system that applies to everyone [25].

6.5 Further research

Based on this thesis's results, two research topics are proposed. The first is to extend these versions of SLAM to unknown correspondence and to replace the landmarks with field corners. The YOLOv8 model trained by Gijs de Jong[2] could be used for the front end. When multiple landmarks are visible, Visser and Oomes [26] show that these two points, together with the gravitational force of the robot, can estimate the range and the robot position quite accurately. If the field corners are used on the front end, this algorithm could be used in a robot football match.
Robot football is a team sport. Multiple robots have to localise themselves at the same time. If all robots construct a map of their environment, overlapping these maps should produce an even more accurate result. The second research topic proposed is thus to find efficient ways to combine the maps of multiple robots in real time. Efficient is defined as small messages and infrequent messages to other robots since this is required in the context of robot football in the SPL league. Cadena et al. [2] discuss how distributed multi-robot SLAM still has open challenges when it comes to improving the map estimate. Thus, testing how this can be applied to robot football might also give insight into how this can be implemented in other robotics applications.

[1] https://www.aldebaran.com/en/nao
[2] https://universe.roboflow.com/fieldmarks/splfieldmark

Bibliography

[1] Florian Vahl. "Active Vision for Humanoid Soccer Robots using Reinforcement Learning". Bachelor's Thesis. University of Hamburg, Department of Informatics, Jan. 2022. DOI: 10.13140/RG.2.2.16291.94248.

[2] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: 10.1109/TRO.2016.2624754.

[3] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Cambridge, MA: The MIT Press, 2005. ISBN: 0262201623.

[4] Andréa Macario Barros et al. "A Comprehensive Survey of Visual SLAM Algorithms". In: *Robotics* 11.1 (2022). ISSN: 2218-6581. DOI: 10.3390/robotics11010024.

[5] Ayman Beghdadi and Malik Mallem. "A comprehensive overview of dynamic visual SLAM and deep learning: concepts, methods and challenges". In: *Machine Vision and Applications* 33.4 (2022), p. 54. DOI: https://doi.org/10.1007/s00138-022-01306-w.

[6] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS". In: *Machine Learning and Knowledge Extraction* 5.4 (2023), pp. 1680–1716. ISSN: 2504-4990. DOI: 10.3390/make5040083.

[7] J.-S. Gutmann and D. Fox. "An experimental comparison of localization methods continued". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 2002, 454–459 vol.1. DOI: 10.1109/IRDS.2002.1041432.

[8] Guoquan P. Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. "Analysis and improvement of the consistency of extended Kalman filter based SLAM". In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 473–479. DOI: 10.1109/ROBOT.2008.4543252.

[9] George H. Joblove and Donald Greenberg. "Color spaces for computer graphics". In: *SIGGRAPH Comput. Graph.* 12.3 (Aug. 1978), pp. 20–25. ISSN: 0097-8930. DOI: 10.1145/965139.807362.

[10] Vencislav Popov, Markus Ostarek, and Caitlin Tenison. "Practices and pitfalls in inferring neural representations". In: *NeuroImage* 174 (Mar. 2018). DOI: 10.1016/j.neuroimage.2018.03.041.

[11] Di Wu and Da-Wen Sun. "Colour measurements by computer vision for food quality control A review". In: *Trends in Food Science Technology* 29.1 (2013), pp. 5–20. ISSN: 0924-2244. DOI: https://doi.org/10.1016/j.tifs.2012.08.004.

[12] Turgay Celik and Kai-Kuang Ma. "Computer vision based fire detection in color images". In: *2008 IEEE Conference on Soft Computing in Industrial Applications*. 2008, pp. 258–263. DOI: 10.1109/SMCIA.2008.5045970.

[13] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[14] Edmundo Casas et al. "A comparative study of YOLOv5 and YOLOv8 for corrosion segmentation tasks in metal surfaces". In: *Array* (2024), p. 100351. ISSN: 2590-0056. DOI: https://doi.org/10.1016/j.array.2024.100351.

[15] Keith YK Leung et al. "The UTIAS multi-robot cooperative localization and mapping dataset". In: *The International Journal of Robotics Research* 30.8 (2011), pp. 969–974. DOI: 10.1177/0278364911398404.

[16] H.J.C. Luijten. "Basics of color based computer vision implemented in Matlab". Dynamics and Control Technology Group, Eindhoven University of Technology, 2005. URL: https://pure.tue.nl/ws/portalfiles/portal/4373031/612499.pdf (visited on 05/21/2024).

[17] Tausif Diwan, G Anirudh, and Jitendra V Tembhurne. "Object detection using YOLO: Challenges, architectural successors, datasets and applications". In: *multimedia Tools and Applications* 82.6 (2023), pp. 9243–9275. DOI: https://doi.org/10.1007/s11042-022-13644-y.

[18] Jaskirat Kaur and Williamjeet Singh. "Tools, techniques, datasets and application areas for object detection in an image: a review". In: *Multimedia Tools and Applications* 81.27 (2022), pp. 38297–38351. DOI: https://doi.org/10.1007/s11042-022-13153-y.

[19] Barret Zoph et al. "Learning data augmentation strategies for object detection". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*. Springer. 2020, pp. 566–583. DOI: https://doi.org/10.1007/978-3-030-58583-9_34.

[20] R. Shanmugamani and S.M. Moore. *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*. Packt Publishing, 2018. ISBN: 9781788295628. URL: https://books.google.nl/books?id=dgdOswEACAAJ.

[21] Arnoud Visser, David de Bos, and Hessel van der Molen. "An Experimental Comparison of Mapping Methods, the Gutmann dataset". In: *Proceedings of the RoboCup IranOpen 2011 Symposium (RIOS11)*. Apr. 2011. URL: https://staff.fnwi.uva.nl/a.visser/publications/ComparisonMappingMethods.pdf (visited on 06/15/2024).

[22] Wenbang Deng et al. "Semantic RGB-D SLAM for Rescue Robot Navigation". In: *IEEE Access* 8 (2020), pp. 221320–221329. DOI: 10.1109/ACCESS.2020.3031867.

[23] Athanasios Chalvatzaras, Ioannis Pratikakis, and Angelos A. Amanatiadis. "A Survey on Map-Based Localization Techniques for Autonomous Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 8.2 (2023), pp. 1574–1596. DOI: 10.1109/TIV.2022.3192102.

[24] Peter Kopacek and Marion Hersh. "Roboethics". In: *Ethical Engineering for International Development and Environmental Sustainability*. Ed. by Marion Hersh. London: Springer London, 2015, pp. 65–102. ISBN: 978-1-4471-6618-4. DOI: 10.1007/978-1-4471-6618-4_3.

[25] Marcellinus Jozef Becker. "Mens en techniek". In: *Ethiek van de Digitale Media*. Uitgeverij Boom, 2015, pp. 162–172. ISBN: 9789089533296.

[26] Arnoud Visser and Stijn Oomes. "Position and Altitude of the Nao Camera Head from Two Points on the Soccer Field plus the Gravitational Direction". In: *Proceedings of the RoboCup 2024 symposium*. Eindhoven, July 2024. URL: https://staff.fnwi.uva.nl/a.visser/publications/TwoPointPoseEstimation.pdf (visited on 06/20/2024).

APPENDIX A

Appendix

A.1 The velocity motion model

In algorithm 6, a motion model using the equation discussed in section 2.1.2 is described. Because constant rotational and translational velocities are assumed, it can be stated that the robot moves around a circle. This circle has the centre (x^*, y^*) and has a radius of r^*. $\Delta\theta$ is the change in bearing. In lines 7 and 8, the actual velocities are calculated for which the initial and final pose are correct. The γ denotes one final rotation after u_t is executed. This is needed since the algorithm applies an *inverse motion model*. If γ was not implemented, for most poses x_t, the robot wouldn't move along a circle, and thus, the probability would be zero. Finally, the probability of all deviations from the expected velocities is calculated to return the probability that x_t is the actual next state. How much error is expected is implemented in the error parameters, denoted by α [3].

Algorithm 6 The velocity motion model [3]

1: **procedure** MOTION MODEL VELOCITY(x_t, u_t, x_{t-1})
2: $\mu = \frac{1}{2}\frac{(x-x')\cos\theta+(y-y')\sin\theta}{(y-y')\cos\theta-(x-x')\sin\theta}$
3: $x^* = \frac{x+x'}{2} + \mu(y'-y)$
4: $y^* = \frac{y+y'}{2} + \mu(x'-x)$
5: $r^* = \sqrt{(x-x^*)^2+(y-y^*)^2}$
6: $\Delta\theta = atan2(y'-y^*, x'-x^*) - atan2(y-y^*, x-x^*)$
7: $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
8: $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$
9: $\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$
10: **return prob**$(v-\hat{v}, \alpha_1|v| + \alpha_2|\omega|) * $ **prob**$(\omega-\hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) * $ **prob**$(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

A.2 YOLOv8 architecture

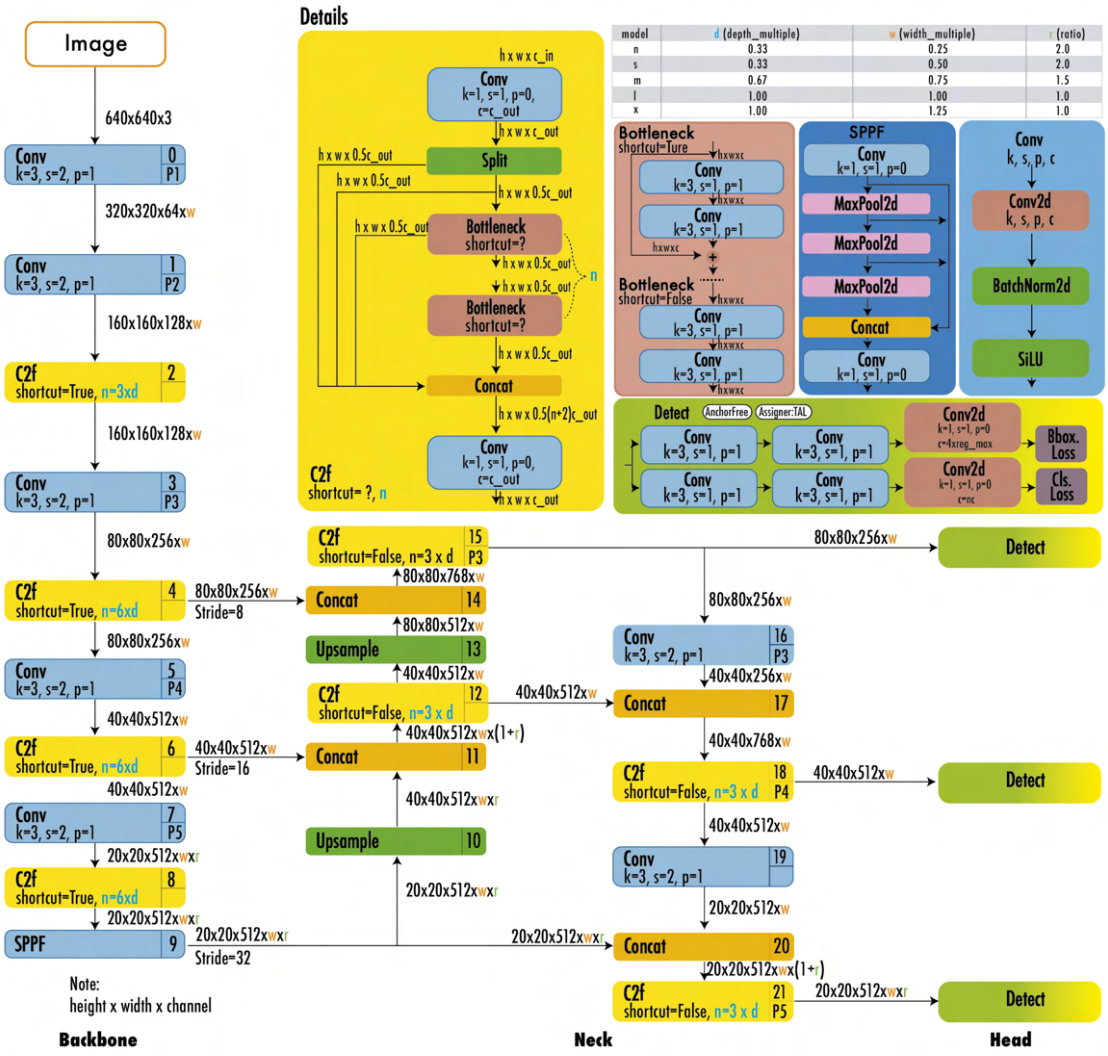


Figure A.1: The YOLOv8 architecture. Source: [6]