FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of master's thesis

Title:	Estimation of detection probability in multitarget filters using object advanced image processing techniques
Student:	Bc. Michal Seibert
Supervisor:	doc. Ing. Kamil Dedecius, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Instructions

Abstract: The subject of the thesis is the dynamic adjustment of the detection probability in random finite set (RFS)-based filters. These filters can track objects in noisy environments from imprecise measurements. However, they rely on the knowledge of the object (mis)detection probability. If this quantity is set inappropriately, the filters are oversensitive and prone to track loss. However, there is no convenient methodology for a consistent estimation of the detection probability. The present thesis aims to focus on its inference using algorithms for object detection and image segmentation. For object detection, the YOLO model should be used, for segmentation, the Segment anything from Meta AI is a possible way towards a solution. These models can recognize various objects in the image. It is conjectured that the combination of these algorithms can yield a filter with good robustness to target misdetections.

The goals are as follows:
- study the principles of the multitarget tracking algorithms
- study the principles of image segmentation and object detection
- propose a technique for estimation of object detection probability
- perform assessment of the proposed algorithm and discuss the obtained results

Literature:
[1] A. F. Garcia-Fernandez, A. S. Rahmathullah, and L. Svensson, "A Metric on the Space of Finite Sets of Trajectories for Evaluation of Multi-Target Tracking Algorithms," IEEE Trans.

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 31 January 2024 in Prague.

Signal Process., vol. 68, pp. 3917–3928, 2020, doi: 10.1109/TSP.2020.3005309.
[2] R. Mahler, Advances in Statistical Multisource-Multitarget Information Fusion. Artech house, 2014.
[3] B. N. Vo and W. K. Ma, "The Gaussian mixture probability hypothesis density filter," IEEE Transactions on Signal Processing, vol. 54, no. 11, pp. 4091–4104, 2006, doi: 10.1109/TSP.2006.881190.
[4] L. Stone, R. Streit, T. Corwin, and K. Bell, Bayesian Multiple Target Tracking. Artech house, 2013.
[5] R. R. Sanaga, "Multi-target tracking with uncertainty in the probability of detection," MSc. Thesis, Purdue Univ., 2019.

Master's thesis

ESTIMATION OF DETECTION PROBABILITY IN MULTITARGET FILTERS USING OBJECT ADVANCED IMAGE PROCESSING TECHNIQUES

Bc. Michal Seibert

Faculty of Information Technology
Katedra aplikované matematiky
Supervisor: doc. Ing. Dedecius Kamil, Ph.D.
April 18, 2024

Contents

List of Figures

List of Tables

Chtěl bych poděkovat především sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Declaration

FILL IN ACCORDING TO THE INSTRUCTIONS. VYPLŇTE V SOULADU S POKYNY. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. In sem justo, commodo ut, suscipit at, pharetra vitae, orci. Pellentesque pretium lectus id turpis.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur sagittis hendrerit ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue. Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed, est. In sem justo, commodo ut, suscipit at, pharetra vitae, orci. Pellentesque pretium lectus id turpis.

In Praze on April 18, 2024

Abstract

Fill in abstract of this thesis in English language. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Keywords enter, comma, separated, list, of, keywords, in, ENGLISH

Abstrakt

Fill in abstract of this thesis in Czech language. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Cras pede libero, dapibus nec, pretium sit amet, tempor quis. Sed vel lectus. Donec odio tempus molestie, porttitor ut, iaculis quis, sem. Suspendisse sagittis ultrices augue.

Klíčová slova enter, comma, separated, list, of, keywords, in, CZECH

Seznam zkratek

DFA	Deterministic Finite Automaton
FA	Finite Automaton
LPS	Labelled Prüfer Sequence
NFA	Nondeterministic Finite Automaton
NPS	Numbered Prüfer Sequence
XML	Extensible Markup Language
XPath	XML Path Language
XSLT	eXtensible Stylesheet Language Transformations
W3C	World Wide Web Consortium

Introduction

Chapter 1

Thesis establishment

Multi-target tracking (MTT), a fundamental aspect of surveillance and monitoring systems, has undergone significant advancements in recent years, transforming it into a critical field with diverse applications across various domains. The scope of MTT extends beyond mere tracking, encompassing tasks such as object detection, identification, and trajectory prediction. The primary goal is to maintain a comprehensive situational awareness, providing invaluable information for decision-making processes in various applications, ranging from defense and surveillance to autonomous systems and robotics. This section provides an overview of the evolution, applications, significance, and current research focus of multi-target tracking.

1.1 Evolution and development

The roots of MTT can be traced back to the early 20th century when radar technology emerged during World War II [1]. Initially developed for single-target detection, radar systems laid the groundwork for subsequent advancements in multi-target tracking. As scenarios evolved and became more complex, the need for advanced tracking capabilities grew, prompting the development of more sophisticated algorithms.

The 1970s and 1980s witnessed the emergence of basic tracking algorithms, marking the initial forays into the field. Subsequent decades saw the integration of probabilistic techniques, such as the Kalman filter [2], which significantly enhanced tracking accuracy. The 2000s marked the transition to data-driven approaches, with particle filters gaining popularity due to their ability to handle non-linear and non-Gaussian tracking scenarios [3]. However, due to the high computational complexity of particle filters, much attention is paid to data association filters such as JPDA [4] or RFS based filters [5].

1.2 Applications of multi-target tracking

The versatility of MTT is reflected in its diverse applications across various domains. In defense, MTT plays a pivotal role in monitoring and tracking multiple targets simultaneously, aiding in threat assessment, target prioritization, and distinguishing friend from foe.

The advent of autonomous systems, particularly in vehicles, has heightened the importance of MTT in predicting and tracking the movements of pedestrians, vehicles, and other obstacles. This application enhances the safety and efficiency of autonomous vehicles by providing real-time awareness of the surrounding environment [6].

Surveillance systems rely on multi-target tracking for monitoring activities in crowded environments and identifying suspicious behavior

Moreover, in fields such as robotics, defense, healthcare monitoring, and wildlife conservation, multi-target tracking systems contribute significantly to enhancing situational awareness, enabling real-time decision-making, improving resource allocation efficiency, and supporting various mission-critical tasks.

1.3 Multi target algorithms

The field of multi-target tracking is marked by a rich and diverse landscape of algorithms, each tailored to address specific challenges inherent in tracking multiple objects. These algorithms can be broadly categorized into association-based methods and Random Finite Set (RFS) based methods, each offering unique advantages and trade-offs.

Association-based methods form a foundational category in multi-target tracking, emphasizing the linking of measurements to existing tracks or the creation of new tracks. The well-established Kalman filter and its variants, such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), fall under this category. More advanced methods considering clutter such as Probabilistic Association (PDA) filter, Joint Probabilistic Association (JPDA) filter or Multiple Hypothesis Tracker (MHT) filter are another examples of filters from this category.

In contrast to association-based methods, Random Finite Set (RFS) based methods provide a probabilistic framework to model multiple target states simultaneously. These methods operate on sets of possible target states, allowing for a more comprehensive representation of uncertainty and variability in tracking scenarios. The Probability Hypothesis Density (PHD) filter, Cardinalized Probability Hypothesis Density (CPHD) or Poisson multi-Bernoulli mixture (PMBM) filter are prominent examples of an RFS-based approach.

1.4 Research interests

Contemporary research in multi-target tracking is characterized by a strong emphasis on addressing key challenges to further enhance the performance and robustness of tracking systems. Current research focus deald with problems such as:

- **Data Association in complex environment:** Handling scenarios with high target density, occlusions, and clutter remains a complex issue. Robust methods for accurate and efficient data association in crowded and dynamic environments are still actively researched.
- **Handling non-linear and non-Gaussian dynamics:** Real-world scenarios often exhibit non-linear and non-Gaussian characteristics. Improving tracking algorithms

to effectively handle these complexities, possibly through the integration of advanced probabilistic models is an ongoing area of research.

- **Real-time processing and computational efficiency:** Many MTT algorithms, especially those with high computational demands, face challenges in meeting real-time processing requirements. Efficient algorithms that strike a balance between computational complexity and tracking accuracy are continuously sought.
- **Sensor fusion and heterogeneous data integration:** Integrating information from various sensors, each with its own characteristics and limitations, is an ongoing challenge. Developing robust methods for sensor fusion to improve tracking accuracy and reliability is an active research area.
- **Handling variability in target behavior:** Targets in real-world scenarios may exhibit diverse and unpredictable behaviors. Adapting tracking algorithms to handle varying target speeds, accelerations, and maneuvers remains an unsolved problem.
- **Online learning and adaptive algorithms:** Designing algorithms that can adapt and learn online as they encounter new scenarios or dynamic changes in the environment is a topic of interest. Adaptive tracking systems that can continuously improve their performance without extensive retraining are sought after.

As we will see later in this work, all Gaussian filters considering clutter rely on accurate setting of certain parameters like motion model or measurement model. Moreover, recursion requires estimation of parameters survival probability and detection probability. This thesis focuses on tracking targets in video data and takes advantage of image processing model YOLO (You Only Look Once). Due to this combination, we are able to estimate detection probability in time and enhance the tracking capability of the filter.

The probabilistic formulation of the RFS-based filters and the inherent Bayesian processing of available information allow to accommodate the uncertainty arising from the presence of false detections, missed detections, and data association ambiguities. Nevertheless, a fundamental challenge persists: The performance of the filters is highly sensitive to the accurate setting of the target detection probability. This quantity, representing the likelihood of correctly identifying and associating observations with actual targets, is a critical parameter. It has a substantial impact on the Bayesian updating of the prior information. However, in real-world scenarios, the sensor performance is susceptible to various environmental conditions. Adverse weather, occlusions, or just the nature of the current scenario can lead to variations in detection probabilities. A mismatch between assumed and actual detection probabilities can result in a suboptimal tracking performance, leading to missed detections, false alarms, or inaccurate target state estimates [7].

The RFS-based formulation of the PHD or (P)MBM filters naturally takes the uncertainty about the detection probability into account [7]. The update formulae involve it as a function of the target state. In the figurative sense, this allows to model it as a function of the spatial and temporal properties of the environment. Still, two difficulties arise. First, the (Gaussian) filters are analytically tractable only if the detection probabilities are scalar numbers. Second, the nature of the detection probabilities differs from scenario to scenario.

In general, several methods have been proposed to deal with unknown detection probabilities. A Gaussian-beta modeling of a slowly-varying detection profile in the Cardinalized PHD filter is reported in [8]; its alternative for MBM filters follows in [9], and for PMBM filter in [10]. In [11], the authors propose to overcome some deficiencies in the CPHD filter [8] by different clutter/detection probability models. Another variant was recently proposed in [12]. A track-state augmentation with an amplitude offset-based prediction of the detection probability appeared in [13], however, this method suffers difficulties in multistatic fields. An automatic identification system-based sensor performance assessment for clutter-free environments is developed in [14]. A recent paper [15] deals with the unknown detection profile in the trajectory PHD/CPHD filters. There, the algorithm learns from the history of the unknown target detection probability.

As we track object in video data, it allows us to avoid the generic solutions and focus on the peculiarities associated with this specific data type. In particular, advantage of YOLO is taken, offering high-performance real-time object detection with high accuracy and efficiency [16]. Its ability to simultaneously predict multiple bounding boxes and class probabilities within an image is used, providing a streamlined and efficient approach to object detection. However, YOLO-based multi-target tracking systems can still be compromised under conditions such as adverse weather, low lighting, or scenarios with occlusions. Sensors may encounter difficulties in accurately detecting and localizing targets, leading to gaps or errors in the tracking process.

1.5 Structure

The structure of this thesis is divided as follows:

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 2

Theoretical Background

Multi-target tracking relies heavily on mathematical concepts from probability theory, statistics, and linear algebra to model and infer the state of multiple objects over time. Key mathematical concepts involved in MTT include Bayesian inference, which provides a framework for updating estimates based on observed evidence, and the use of probabilistic models such as the Gaussian distribution and its extensions, such as Gaussian mixture models. In addition, MTT commonly employs state space models to represent the dynamics of object motion, with popular models including the constant velocity and constant acceleration models. These models are integrated into Bayesian filters, such as the Kalman filter, which recursively estimates the state of multiple targets given noisy measurements.

2.1 Bayesian inference

Bayesian inference stands as a foundational pillar within the realm of probabilistic reasoning, offering a fundamental methodology for systematically updating beliefs in response to observed evidence. At its heart stands Bayes' rule, a fundamental theorem in probability theory, that formalizes the process of revising prior beliefs in light of new data. The essence of Bayesian inference transcends mere statistical calculations; it embodies a philosophical stance towards uncertainty, emphasizing the incorporation of prior knowledge and the iterative refinement of beliefs through the assimilation of empirical observations. Within the context of multi-target tracking, Bayesian inference assumes a paramount role, providing a principled framework for joining information from disparate sources, such as sensor measurements, historical data, and domain expertise. By embracing Bayesian principles, MTT algorithms gain the capacity to model and quantify uncertainty inherent in tracking scenarios, thereby fostering robustness and adaptability in the face of dynamic and complex environments. Moreover, Bayesian inference empowers MTT systems to exploit contextual cues and domain-specific knowledge, enhancing their ability to discern meaningful patterns among noise and uncertainty. Due to the Bayesian inference, MTT researchers and practitioners are equipped with a powerful tool for navigating the intricacies of multi-target tracking, facilitating informed decision-making and advancing the frontier of robust systems. Thus, Bayesian inference emerges

not merely as a mathematical construct but as a guiding philosophy underpinning the quest for understanding and reasoning in the face of uncertainty.

2.2 Bayes' rule

Bayes' rule, a cornerstone of Bayesian inference, embodies a fundamental principle in probability theory that underpins the systematic revision of beliefs in the face of new evidence. Mathematically expressed as a simple formula, Bayes' rule encapsulates the process of updating prior probabilities based on observed data, thereby yielding posterior probabilities that reflect the incorporation of new information. At its basis, Bayes' rule provides a formal mechanism for quantifying the impact of new evidence on the likelihood of various hypotheses or states of nature. The rule states that the posterior probability of a hypothesis given observed data is proportional to the product of the likelihood of the data given the hypothesis and the prior probability of the hypothesis, divided by the marginal likelihood of the data. In essence, Bayes' rule facilitates a principled approach to inference, allowing practitioners to integrate prior knowledge with empirical observations to arrive at more informed and reliable conclusions. In the context of multi-target tracking, Bayes' rule serves as the base upon which tracking algorithms are built, enabling the continuous refinement of estimates about the state of multiple targets based on sensor measurements and historical data. By adhering to the principles of Bayes' rule, MTT systems can effectively navigate the inherent uncertainty and complexity of tracking scenarios, thereby enhancing their robustness and accuracy.

▶ **Definition 2.1** (Conditional distribution)**.** *Let X and Y be jointly continuous random variables, f_Y continuous at y and $f_Y(y) > 0$. Then the conditional distribution function of X, given condition $Y = y$ is defined by*

$$F_{X|Y}(x|y) := \lim_{\epsilon \to 0} \mathbf{P}\{X \leq x | Y \in (y, y+\epsilon)\} = \frac{\partial F(x,y)/\partial y}{f_Y(y)}. \tag{2.1}$$

Differentiating this, the conditional density function of X, given the condition $Y = y$ is

$$f_{X|Y}(x|y) := \frac{\frac{\partial^2 F(x,y)}{\partial x \partial y}}{f_Y(y)} = \frac{f(x,y)}{f_Y(y)}. \tag{2.2}$$

Naturally, fixing y, $F_{X|Y}(\cdot|y)$ and $f_{X|Y}(\cdot|y)$ are proper distribution and density functions, respectively. It is also clear that X and Y are independent if and only if $F_{X|Y}(x|y) = F_X(x)$ and $f_{X|Y}(x|y) = f_X(x)$ for all x and y for which these quantities are defined.

▶ **Definition 2.2** (Bayes' theorem)**.** *Let x and x be random variables with densities $f(y|x)$ and $f(x)$. Then Bayes' theorem is defined as*

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)}, \quad f(y) > 0, \tag{2.3}$$

where

- $f(x|y)$ is the conditional posterior density of x,
- $f(x)$ is prior density,
- $f(y|x)$ is likelihood and $f(x)$ is marginal density of X, also called *evidence*, and is given by

$$f(y) = \int f(x, y)\partial x. \tag{2.4}$$

Combining Equations (2.1), (2.2) and (2.4) to Formula (2.3) we get complete formula for Bayes' theorem

$$f(x|y) = \frac{f(y|x)f(x)}{f(y)} = \frac{f(y|x)f(x)}{\int f(y|x)f(x)\partial x}. \tag{2.5}$$

The denominator is the normalizing constant independent of x, often only proportionality is used

$$f(x|y) \propto f(y|x)f(x). \tag{2.6}$$

2.3 Multivariate Gaussian distribution

In the realm of multi-target tracking, various probability distributions are employed to model the uncertainty associated with target states and measurements. However, among these distributions, the multivariate Gaussian distribution holds a preeminent position due to its versatility, mathematical tractability, and empirical relevance. While other distributions may capture specific aspects of target behavior or measurement noise, the Gaussian distribution emerges as the cornerstone of MTT due to its ability to characterize complex probability distributions in multi-dimensional spaces. As a result common assumption among MTT filters is, that the targets follows linear Gaussian dynamic and measuremets models, as in [1], [17], or [18]. The Gaussian distribution finds ubiquitous application across various components of tracking algorithms, including

- **State representation:** Gaussian distributions are used to model the probability distributions of target states, allowing for efficient representation and propagation of uncertainty over time.
- **Measurement model:** Gaussian distributions are employed to model the likelihood of sensor measurements given the true target state, facilitating the incorporation of sensor data into the tracking process.
- **Filtering algorithms:** Gaussian-based filters leverage the Gaussian assumption to derive recursive estimation algorithms for tracking multiple targets with optimal efficiency and accuracy.
- **Data association:** Gaussian mixture models (GMMs), which represent mixtures of Gaussian distributions, are utilized for probabilistic data association in MTT, enabling robust handling of measurement uncertainty and target ambiguity.

▶ **Note 2.3.** The multivariate Gaussian distribution is a generalization of the univariate Gaussian distribution, but instead of a scalar mean and variace, there are mean vector and covariance matrix, that describes correlations between variables. The distribution of a $k-$dimensional random vector $\mathbf{X} = (X_1, \ldots, X_k)^T$ and can be written in the notation

$$\mathbf{X} \sim \mathcal{N}(\mu, \mathbf{\Sigma}),$$

with $k-$dimensional mean vector $\mu = \mathsf{E}[\mathbf{X}] = (\mathsf{E}[X_1], \mathsf{E}[X_1], \ldots, \mathsf{E}[X_k])^T$ and $k \times k$ positive semi-definite covariance matrix Σ, with elements

$$\mathbf{\Sigma}_{ij} = \mathsf{E}[(X_i - \mu_i)(X_j - \mu_j)] = Cov[X_i, X_j]. \tag{2.7}$$

The probability density function (PDF) is given by formula

$$\mathcal{N}(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)\right), \tag{2.8}$$

where k is the length of the vector $\mathbf{x}$ and $|\cdot|$ denotes the determinant of a matrix. It should be noted that the exponent is known as Mahalanobis distance.

▶ **Definition 2.4** (Mahalanobis distance)**.** *For vectors x and y and a positive semi-definite matrix S, the Mahalanobis distance between two objects is defined as*

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T S^{-1} (\mathbf{x} - \mathbf{y})} \tag{2.9}$$

The Mahalanobis distance (MD) is the distance between two points in multivariate space and unlike Eucledian distance, it measures distances even between correlated points for multiple variables.

Target tracking filters regularly use conditional probabilities and joint distributions, especially Gaussian.

▶ **Theorem 2.5** (Conditional joint Gaussian distribution)**.** *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *are Gaussian random variables with distributions* $\mathcal{N}(\mathbf{x}; \mu_\mathbf{x}, \mathbf{\Sigma_{xx}})$ *and* $\mathcal{N}(\mathbf{y}; \mu_\mathbf{y}, \mathbf{\Sigma_y})$*, respectively. Let their joint probability is given by*

$$p(\mathbf{x}, \mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}\mu_\mathbf{x}\\ \mu_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma_{xx}} & \mathbf{\Sigma_{xy}}\\ \mathbf{\Sigma_{xy}} & \mathbf{\Sigma_{yy}}\end{bmatrix}\right). \tag{2.10}$$

Then the conditional distribution of $\mathbf{x}$ *given by* $\mathbf{y}$ *is defined as*

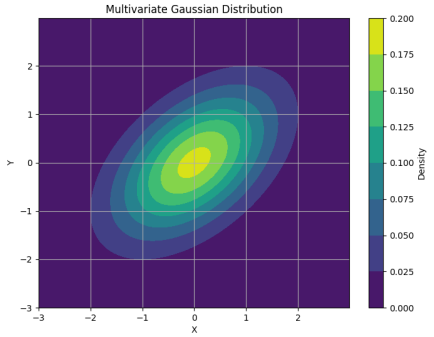
$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; \mu_{\mathbf{x}|\mathbf{y}}, \mathbf{\Sigma}_{\mathbf{x}|\mathbf{y}}), \tag{2.11}$$

where

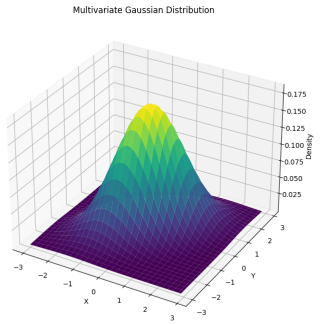
$$\mu_{\mathbf{x}|\mathbf{y}} = \mu_\mathbf{x} + \mathbf{\Sigma_{xy}} \mathbf{\Sigma_{yy}}^{-1} (\mathbf{y} - \mu_\mathbf{y}), \tag{2.12}$$

$$\mathbf{\Sigma}_{\mathbf{x}|\mathbf{y}} = \mathbf{\Sigma_{xx}} - \mathbf{\Sigma_{xy}} \mathbf{\Sigma_{yy}^{-1}} \mathbf{\Sigma_{xy}^T}. \tag{2.13}$$

Examples of multivariate Gaussian distribution are shown in Figure 2.1 and marginal distributions for multivariate Gaussian distribution in Figure 2.2.



(a) Example of multivariate Gaussian distribution - contour plot.



(b) Example of multivariate Gaussian distribution - 3D plot.

Figure 2.1 Examples of plots of multivariate Gaussian distribution.

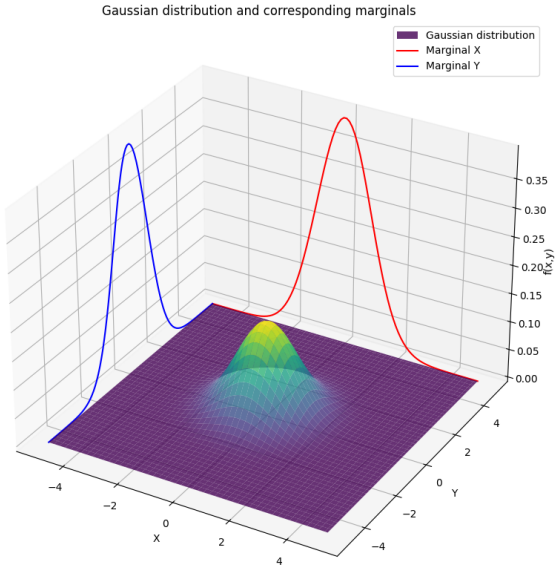


Figure 2.2 Gaussian distribution with corresponding marginals.

2.4 Gaussian mixture

Gaussian Mixture Models offer a flexible and powerful framework for representing complex probability distributions by combining multiple Gaussian components. In a Gaussian mixture model, a vector of parameters (e.g., observations of a signal) is modeled using a mixture distribution comprising several Gaussian components.

$$p(\Theta) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu_i, \mathbf{\Sigma}_i), \tag{2.14}$$

where the i^{th} vector component is characterized by Gaussian distributions with weights w_i, means μ_i and covariance matrices $\mathbf{\Sigma}_i$. These parameters are encapsulated to the parameter Θ in $p(\cdot)$.

In the context of multi-target tracking, Gaussian mixture models find utility in capturing the complex nature of target states and measurements. MTT often involves dealing with linear Gaussian models, where the posterior density is represented as a mixture of one or more Gaussian components. This representation enables MTT algorithms to probabilistically model uncertainties associated with target dynamics, sensor measurements, and data association.

The probability density function (pdf) of a Gaussian mixture distribution is a simple sum of each Gaussian component, expressed as

$$p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}; \mu_\mathbf{i}, \mathbf{\Sigma_i}), \tag{2.15}$$

where k is the number of Gaussian component, $w_i > 0$ is the weight of i^{th} component and $\sum_{i=1}^{k} w_i = 1$.

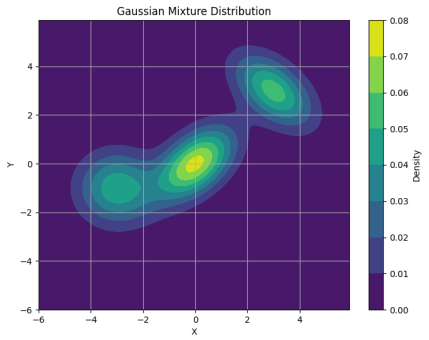
In MTT, Gaussian mixture models are particularly useful for tasks such as data association, where they probabilistically assign measurements to existing tracks or create new tracks based on the likelihood of observations given the target states. By modeling complex distributions of target states and measurements, Gaussian mixture models enable MTT algorithms to handle uncertainties and ambiguities inherent in real-world tracking scenarios, including occlusions, clutter, and target interactions.

Examples of multivariate Gaussian mixture distribution are shown in Figure 2.3. Gaussian mixture distribution with marginal distribution for each dimension in Figure 2.4.

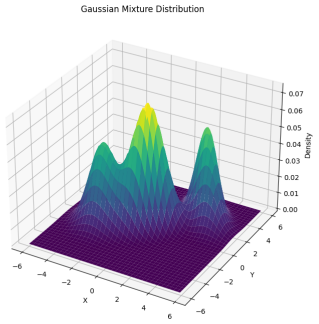
2.5 State space model

State-space models serve as a fundamental framework for describing the evolution of dynamic systems over time. In the context of multi-target tracking, state-space models provide a formalism for representing motion dynamics of targets and the measurement process, enabling efficient and accurate inference of target states from sensor data.

At the core of a state-space model lies a set of latent variables, known as the state vector, which encapsulates the unobservable quantities of interest, such as the position,



(a) Example of multivariate Gaussian mixture distribution - contour plot.



(b) Example of multivariate Gaussian mixture distribution - 3D plot.

Figure 2.3 Examples of plots of multivariate Gaussian mixture distribution. There are three components in figures with means $\{[0,0],[3,3],[-3,-1]\}$. Note that the density of peaks is very low, because the sum has to be equal 1.

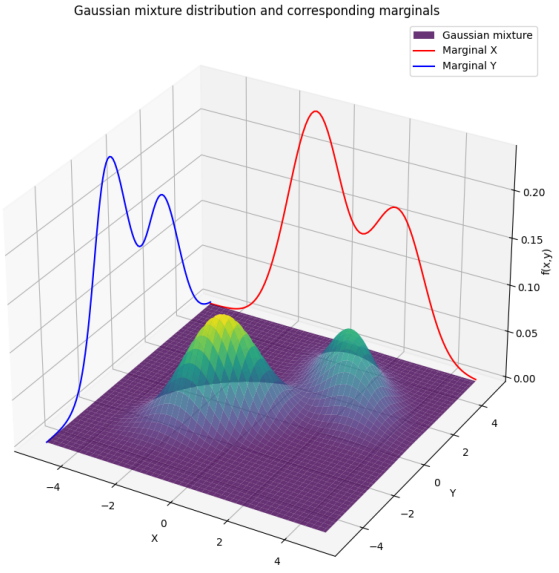


Figure 2.4 Gaussian mixture distribution with means $\{[0,0],[2,2]\}$, weights $\{[0.6,0.4]\}$ and corresponding marginals.

velocity, and acceleration of targets in MTT. The dynamics governing the evolution of the state vector are typically described by a transition model, which specifies how the state evolves over time according to a probabilistic process. This transition model can take various forms depending on the nature of the system dynamics, ranging from simple linear or non-linear models to more complex stochastic processes.

In addition to the transition model, state-space models incorporate an observation model that describes the relationship between the observed measurements and the underlying state variables. This observation model accounts for the uncertainties and noise inherent in the measurement process, allowing for the probabilistic mapping of observed data to the latent state space.

State-space model can be represented as a pair of stochastic equations.

1. **State transition equation:**

$$\mathbf{x_k} = f(\mathbf{x_{k-1}}, \mathbf{u_k}, \mathbf{w_k}), \tag{2.16}$$

   where $\mathbf{x_k}$ represents the state vector at time k, f denotes the transition function describing the evolution of the state, $\mathbf{u_k}$ represents optional control inputs, and $\mathbf{w_k}$ denotes process noise.

2. **Observation equation:**

$$\mathbf{z_k} = h(\mathbf{x_k}, \mathbf{v_k}), \tag{2.17}$$

   where $\mathbf{z_k}$ represents the observed measurements at time k, h denotes the observation function mapping the state to measurements, and v_k represents the measurement noise.

In MTT, state-space models provide a natural framework for representing the motion dynamics of multiple targets and the sensor measurements associated with each target. Each target is typically associated with its own state vector, allowing for simultaneous tracking of multiple objects within the same probabilistic framework.

State-space models enable MTT algorithms to perform a range of tasks, including target prediction, data association, and state estimation by propagating the state forward in time using the transition model and updating the state based on observed measurements using the observation model. By incorporating uncertainty explicitly into the tracking process, state-space models facilitate robust inference of target states in complex and dynamic tracking scenarios.

While state-space models offer a powerful framework for multi-target tracking, they also present several challenges and considerations.

- **Model complexity:** Designing an appropriate state-space model requires careful consideration of the underlying dynamics and measurement process, which can be challenging in complex tracking scenarios with non-linearities and uncertainties.
- **Parameter estimation:** Estimating the parameters of a state-space model from data, such as the transition and observation matrices, can be computationally demanding and prone to issues such as overfitting or underfitting.

- **Computational Complexity:** Performing inference in state-space models often involves recursive algorithms such as the Kalman filter or particle filter, which can be computationally intensive, especially in high-dimensional or real-time tracking applications.

Despite these challenges, state-space models remain a cornerstone of multi-target tracking, offering a principled and flexible framework for representing and reasoning about dynamic systems in the presence of uncertainty. There are many state space models used in MTT. Before we define two of the most common ones in next sections, it should be noted, that definitions in (2.16) and (2.17) are too general, because f_k and h_k could by any functions.

To get a closed-form solution in the Bayesian inference framework, we need to choose conjugate distributions for the likelihood and the prior. The Kalman filter (see Section 2.8 for more) works for the Gaussian-linear case, where the functions f_t and h_t are linear and noisy variables and are distributed as Gaussians with zero mean. The Gaussian linear state space model has the formulation:

$$p(x_k|x_{k-1}) = Fx_{k-1} + Bu_k + w_k \qquad w_k \sim \mathcal{N}(0, Q), \tag{2.18}$$
$$p(z_k|x_k) = Hx_k + v_k \qquad v_k \sim \mathcal{N}(0, R), \tag{2.19}$$
$$p(x_0) \sim \mathcal{N}(\hat{x}_0, P_0), \tag{2.20}$$

where

- F is the transition matrix of appropriate dimension,
- B is the input matrix of appropriate dimension,
- H is the measurement matrix of appropriate dimension,
- Q is the symmetric positive semi-definite covariance matrix of motion noise w_k,
- R is the symmetric positive semi-definite covariance matrix of measurement noise v_k,
- $\hat{x}_0$ and P_0 are the mean and the covariance matrix of the prior state.

The control variable u_k, in general, represents some input signal from the environment, and B specifies how the input signal affects the dynamic system. This variable is ususally not considered in MTT scenarios. In multi-target tracking, most often with Gaussian linear models is worked, thus following formulation is instead used.

$$p(x_k|x_{k-1}) = \mathcal{N}(x_k; Fx_{k-1}, Q), \tag{2.21}$$
$$p(z_k|x_k) = \mathcal{N}(z_k; Hx_k, R), \tag{2.22}$$
$$p(x_0) \sim \mathcal{N}(x_0; \hat{x}_o, P_0). \tag{2.23}$$

2.5.1 Constant velocity model

The constant velocity model (CVM) is a fundamental component of multi-target tracking systems, providing a simplified yet effective representation of target motion dynamics

over time. This model assumes that the target's velocity remains constant between consecutive time steps, making it particularly suitable for tracking objects with relatively smooth and predictable motion patterns. In this section, we explore the conceptual basis, mathematical formulation, and practical implications of the constant velocity model in the context of MTT.

At its core, the constant velocity model embodies the notion of inertia, where a target maintains a constant velocity unless acted upon by external forces. This conceptual simplicity allows for a straightforward representation of target motion, making the constant velocity model a popular choice for MTT applications where targets exhibit relatively uniform and predictable movement behaviors.

Mathematically, the constant velocity model describes the evolution of a target's state vector over time in terms of its position and velocity. At each time step k, the state vector $\mathbf{x}_k$ comprises the position $[x_{1,k}, x_{2,k}]$ and velocity s of the target

$$\mathbf{x}_k = [x_{1,k}, x_{2,k}, s_{1,k}, s_{2,k}]^T. \tag{2.24}$$

The dynamics of the constant velocity model can be represented using a state transition matrix F and a process noise vector w_k, where the state transition matrix captures the relationship between the target's state at consecutive time steps

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w}_k. \tag{2.25}$$

All together, vector $\mathbf{x}_k$, matrices $\mathbf{F}$ and $\mathbf{Q}$, might, for example, be represented as

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ s_{1,k} \\ s_{2,k} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = q^2 \begin{bmatrix} \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 \\ 0 & \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} \\ \frac{\Delta^2}{2} & 0 & \Delta & 0 \\ 0 & \frac{\Delta^2}{2} & 0 & \Delta \end{bmatrix}, \tag{2.26}$$

where Δ stands for delta time, i.e., elapsed time between the last estimation and the current one, q is the motion noise parameter, which represents the uncertainty in the state transition. The measurement model transforms a state vector from the state space into the measurement space. Since filters derived from Kalman filter assumes only linear models, the measurement model for CVM assumes the same space as in the state vectors. As a result, the observation matrix $\mathbf{H}$ and the noise matrix $\mathbf{R}$ can be, with respect to (2.26), formulated

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{R} = r^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.27}$$

where r determines the variance of the measurement noise.

To obtain optimal results given by filters, it is essential to set parameters r and q appropriately. There are three main ways to do it. The first one requires the knowledge of motion noise given by tracked object and the knowledge of sensor's error when detecting targets, i.e. the noise measurement. This method is often used in simulations and experiments. The second one is simple trial and error method with reasonable choices. The last one is using automated methods, like in [19].

In the context of MTT, the constant velocity model serves as a fundamental building block for tracking algorithms, providing a simplified yet effective representation of a target motion. By assuming the constant velocity, MTT algorithms can predict the future positions of targets based on their current state and velocity, facilitating trajectory estimation and target prediction.

One common application of the constant velocity model in MTT is in the design of prediction algorithms, where future positions of targets are estimated based on their current state and velocity information. These predictions are essential for anticipating target movements and facilitating data association, enabling MTT algorithms to maintain track continuity and adapt to changes in target behavior over time.

Moreover, the constant velocity model can be seamlessly integrated into recursive Bayesian filters, such as the Kalman filter, for state estimation in MTT. By incorporating the constant velocity model into the state-space representation of the tracking problem, Kalman filter-based algorithms can effectively fuse measurement information with dynamic predictions to estimate the most likely trajectories of targets over time.

2.5.2 Constant acceleration model

The Constant Acceleration Model (CAM) stands as a sophisticated extension of the state-space model in multi-target tracking, offering a more comprehensive representation of target motion dynamics. Unlike simpler models such as the Constant Velocity Model, which assumes a constant velocity for targets over time, the CAM acknowledges the potential for changes in target acceleration, allowing for more accurate and flexible trajectory estimation. This section delves into the conceptual basis, mathematical formulation, and practical implications of the Constant Acceleration Model in the context of MTT.

In MTT scenarios, targets often exhibit varying degrees of acceleration due to factors such as changes in speed, direction, or environmental influences. Ignoring these accelerative effects can lead to biased trajectory estimates and diminished tracking accuracy. The CAM addresses this limitation by incorporating an additional acceleration component into the state-space model, enabling more faithful representation of target motion dynamics. By accounting for changes in acceleration, the CAM provides MTT algorithms with greater flexibility and adaptability in tracking targets with non-uniform motion profiles.

Mathematically, the Constant Acceleration Model extends the state transition function of the state-space model to accommodate changes in acceleration over time. At each time step k, the evolution of the target's state vector x_k is governed by a set of dynamic equations that describe the position, velocity, and acceleration of the target

$$\mathbf{x}_k = [x_{1,k}, x_{2,k}, s_{1,k}, s_{2,k}, a_{1,k}, a_{2,k}]^T, \tag{2.28}$$

where, as in CVM, $x_{1,k}$, $x_{2,k}$ is the position of an target in two-dimensional space, $s_{1,k}, s_{2,k}$ is the velocity and $a_{1,k}, a_{2,k}$ are the accelerations in both directions. Matrices

F and Q can then be expressed

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ s_{1,k} \\ s_{2,k} \\ a_{1,k} \\ a_{2,k} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta & 0 & \frac{1}{2}\Delta^2 & 0 \\ 0 & 1 & 0 & \Delta & 0 & \frac{1}{2}\Delta^2 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = q^2 \begin{bmatrix} \frac{\Delta^4}{4} & 0 & \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 \\ 0 & \frac{\Delta^4}{4} & 0 & \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} \\ \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 & \Delta & 0 \\ 0 & \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 & \Delta \\ \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 & \Delta & 0 \\ 0 & \frac{\Delta^3}{3} & 0 & \frac{\Delta^2}{2} & 0 & \Delta \end{bmatrix}. \tag{2.29}$$

The measurement model remains unchanged from the standard state-space model, relating the observed measurements $\mathbf{z}_k$ to the target's true state $\mathbf{x}_k$ through a measurement function $h(\cdot)$ corrupted by measurement noise v_t

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{R} = r^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{2.30}$$

The Constant Acceleration Model offers several practical advantages in MTT applications:

- Improved trajectory estimation: By accounting for changes in acceleration, the CAM enables more accurate and realistic trajectory estimation, particularly for targets exhibiting non-uniform motion patterns or sudden changes in velocity.
- Enhanced predictive capability: The inclusion of acceleration dynamics allows MTT algorithms to make more informed predictions about future target states, improving tracking performance and reducing prediction errors.
- Robustness to dynamic environments: In dynamic environments with varying levels of congestion, obstacles, or traffic patterns, the CAM provides MTT algorithms with greater robustness and adaptability, enabling effective tracking even in challenging scenarios.
- Applications in autonomous systems: In autonomous systems such as self-driving cars, drones, or robotics, the CAM plays a crucial role in motion planning, obstacle avoidance, and trajectory prediction, enhancing the safety and efficiency of autonomous navigation.

In summary, the Constant Acceleration Model represents a significant advancement in multi-target tracking, offering a more nuanced and comprehensive approach to modeling target motion dynamics. By incorporating acceleration effects into the tracking process, the CAM enables MTT algorithms to achieve higher accuracy, robustness, and predictive capability, making it an indispensable tool for a wide range of applications.

2.6 Hidden Markov Model

In the domain of multi-target tracking, the Hidden Markov Model (HMM) serves as a framework for capturing the temporal dynamics of target behavior in complex environments. Building upon the principles of Markov processes, the HMM extends the

state-space model by introducing hidden states that govern the evolution of observable measurements over time. Markov process of the first order is a model, in which current state depends only on the previous state

$$p(x_k|x_1,\ldots,x_{k-2},x_{k-1},z_1,\ldots,z_{k-2},z_{k-1}) = p(x_k|x_{k-1}) \quad \text{(transition probability)}, \tag{2.31}$$

$$p(z_k|x_1,\ldots,x_{k-2},x_{k-1},x_k,z_1,\ldots,z_{k-2},z_{k-1}) = p(z_k|x_k) \quad \text{(observation likelihood)}, \tag{2.32}$$

$$p(x_0) \quad \text{(initial state)}. \tag{2.33}$$

In models of higher order, the transition probability is

$$p(x_k|x_{k-1},\ldots,x_{k-n}), \tag{2.34}$$

where n is the number of the order. Details of this model is not described in this work and can be seen in [20].

In cases where the process is not didectly observable, but can be observed through another observable variable z_k, we often talk about Hidden Markov process.

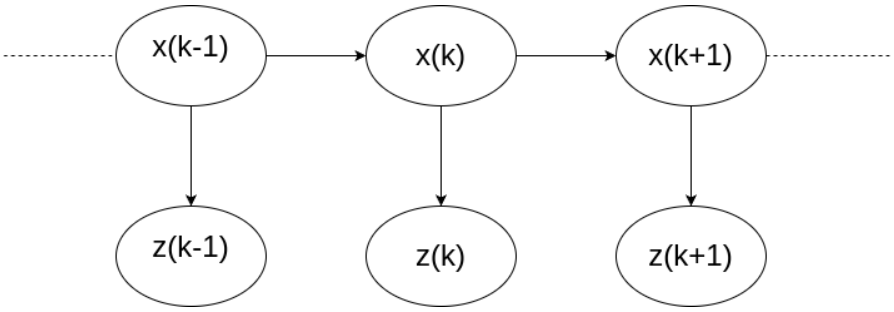


Figure 2.5 Demonstration of the course of the hidden Markov process.

At its core, the Hidden Markov Process represents a stochastic process characterized by a set of hidden states that transition probabilistically over time. While these hidden states are unobservable, they influence the generation of observable measurements, which are assumed to be conditionally independent given the hidden states. In the context of MTT, the hidden states may represent latent attributes of targets, such as their locations, velocities, or motion patterns, while the observable measurements correspond to sensor readings or detections obtained from surveillance systems.

2.7 Bayes' filter

The Bayes filter is a recursive framework that estimates an internal state of the system over time using measurements. The Bayes filter leverages both motion and measurement models, operating under the assumption that these models accurately describe the system's behavior. Operating within a recursive framework, the Bayes filter continually estimates the internal state of the system over time using available measurements. Each iteration of the filter entails two fundamental steps: prediction and update. During the prediction step, the filter anticipates the internal state x_k based on the preceding state

x_{k-1} and the motion model $p(x_k|x_{k-1})$. This prediction step is commonly referred to as the Chapman-Kolmogorov equation [21].

▶ **Theorem 2.6** (The Chapman-Kologorov equation prediction step)**.** *Given the set of measurements $z_{0:k-1} = z_0, \ldots, z_{k-1}$, set of control variables $u_{0:k-1} = u_0, \ldots, u_{k-1}$ and current state x_{k-1} and the motion model $p(x_k|x_{k-1})$, the prediction step is as follows:*

$$p(x_k|z_{0:t-1}, u_{0:k}) = \int p(x_k|x_{k-1}, u_k)p(x_{k-1}|z_{0:k-1}, u_{0:k-1})dx_{k-1}, \tag{2.35}$$

where the integral is taken over the entire state space of x_{k-1}, and $p(x_{k-1}|z_{0:k-1}, u_{0:k-1})$ is the posterior density of the state at time $k-1$.

On the update step, the Bayes' filter corrects (updates) the prediction step with measurements z_k using the measurement model $p(z_k|x_k)$. This step uses common Bayes' rule.

▶ **Theorem 2.7** (The update step of Bayes' filter)**.** *Given the output of Bayes' filter's prediction step, the measurements z_k observed at time k and the measurement model $p(z_k|x_k)$, the update step of Bayes' filter is formulated as follows:*

$$p(x_k|z_{0:t}, u_{0:t}) = \frac{p(z_k|x_k)p(x_k|z_{0:k-1}, u_{0:k})}{p(z_k|z_{0:k-1})} \propto p(z_k|x_k)p(x_k|z_{0:k-1}, u_{0:k}) \tag{2.36}$$

These two steps work in an iterative method, where, in each iteration, first prediction step is used to predict the state x_k and then update step is used to correct our guess with provided measurement.

2.8 Kalman filter

The Kalman filter, named after its developer Rudolf E. Kalman, has a rich history dating back to the early 1960s when Kalman first introduced the algorithm in a series of landmark papers. Initially developed for aerospace applications, the Kalman filter gained prominence for its ability to provide optimal state estimation in the presence of noisy measurements and uncertain dynamics.

Over the years, the Kalman filter has found widespread usage across diverse fields and industries, including aerospace, robotics, finance, and telecommunications. Its applications range from tracking spacecraft trajectories and guiding missiles to monitoring financial markets and controlling autonomous vehicles. The filter's versatility, efficiency, and effectiveness have made it a cornerstone of modern estimation and control systems.

The Kalman filter operates on the principles of recursive Bayesian estimation, utilizing a system dynamics model and noisy measurements to predict and update the state of a dynamic system over time. Its mathematical elegance and simplicity make it well-suited for real-time applications, enabling accurate and efficient state estimation even in complex and uncertain environments.

Despite its remarkable success, the Kalman filter continues to evolve, with extensions and variations such as the extended Kalman filter (EKF), unscented Kalman filter (UKF), and particle filter (PF) addressing more challenging scenarios involving non-linear dynamics and non-Gaussian noise.

In summary, the Kalman filter stands as a seminal contribution to the field of estimation and control, with a rich history of development and widespread usage across diverse domains. Its continued relevance and versatility underscore its status as a foundational tool for state estimation and tracking in modern technological applications.

2.8.1 Kalman filter inference

In Section 2.5 we talked about state space model, which is used by Kalman filter. Thus we work with equations

$$x_k = Fx_{k-1} + w_t \tag{2.37}$$
$$z_t = Hx_k + v_k, \tag{2.38}$$

where both noise variables are independent and with zero mean

$$w_k \sim \mathcal{N}(0, Q), \tag{2.39}$$
$$v_k \sim \mathcal{N}(0, R). \tag{2.40}$$

Then the state and measurement distributions follows

$$x_k \sim \mathcal{N}(Fx_{k-1}, Q) \qquad \text{with density} \quad p(x_k|x_{k-1}), \tag{2.41}$$
$$z_k \sim \mathcal{N}(Hx_k, R) \qquad \text{with density} \quad p(z_k|x_k). \tag{2.42}$$

Because Kalman filter is Bayesian, we need prior distribution for x_k. Model z_k is Gaussian, conjugate prior and is then also Gaussian with mean x^+_{k-1} and covariance matrix P^+_{k-1},

$$p(x_k|z_{0:k-1}) = \mathcal{N}(x^+_{k-1}, P^+_{k-1}). \tag{2.43}$$

The prediction step is derived from Formula (2.35). Note, that by multiplicating two Gaussian distributions we again get Gaussian distribution $\mathcal{N}(x^-_k, P^-_k)$ with hyperparemeters

$$x^-_k = Fx^+_{k-1} \tag{2.44}$$
$$P^-_k = FP^+_{k-1}F^T + Q. \tag{2.45}$$

By multiplicating two Gaussian distributions followed by marginalization, we enumerate the state equation. The estimation of state x^-_k is just applying model variables into appropriate equations. The estimated covariance P^-_k expresses the degree of uncertainty of the estimation. By applying this step, the uncertainty grows.

The update step corrects the prediction step by applying new observed measurements z_k. For that, formula (2.36) is used. The model is transformed into exponential form

$$\begin{aligned} p(z_k|x_k) &\propto \exp\left\{-\frac{1}{2}(z_k - Hx_k)^T R^{-1}(z_k - Hx_k)\right\} \\ &= \exp\left\{Tr\left(\underbrace{-\frac{1}{2}\begin{bmatrix}-1\\ x_k\end{bmatrix}\begin{bmatrix}-1\\ x_k\end{bmatrix}^T}_{\eta}\underbrace{\begin{bmatrix}z_k^T\\ H^T\end{bmatrix}R^{-1}\begin{bmatrix}z_k^T\\ H^T\end{bmatrix}^T}_{T(z_k)}\right)\right\}. \end{aligned} \tag{2.46}$$

The conjugated shape has the form

$$p(x_k|z_{0:k-1}) \propto \exp\left\{-\frac{1}{2}(x_k - x_k^-)^T (P_k^-)^{-1}(x_k - x_k^-)\right\}$$
$$= \exp\left\{Tr\left(\underbrace{-\frac{1}{2}\begin{bmatrix}-1\\ x_k\end{bmatrix}\begin{bmatrix}-1\\ x_k\end{bmatrix}^T}_{\eta}\underbrace{\begin{bmatrix}(x_k^-)^T\\ I\end{bmatrix}(P_k^-)^{-1}\begin{bmatrix}(x_k^-)^T\\ I\end{bmatrix}^T}_{\xi_k}\right)\right\}, \tag{2.47}$$

where I is identity matrix of appropriate shape.

The bayesian update is a sum of the hyperparameter and sufficient statistic,

$$\begin{aligned}\xi_k &= \xi_{k-1} + T(z_k)\\ &= \begin{bmatrix}(x_k^-)^T(P_k^-)^{-1}x_k^- + z_k^T R^{-1} z_k, & (x_k^-)^T(P_k^-)^{-1} + z_k^T R^{-1} H\\ (P_k^-)^{-1}(x_k^-)^T + H^T R^{-1} z_k, & (P_k^-)^{-1} + H^T R^{-1} H.\end{bmatrix}\end{aligned} \tag{2.48}$$

The posterior parameters are then derived

$$\begin{aligned}P_k^+ &= (\xi_{k;[2,2]})^{-1}\\ &= \left[(P_k^-)^{-1} + H^T R^{-1} H\right]^{-1}\\ &= (I - K_k H)P_k^- \end{aligned} \tag{2.49}$$

$$\begin{aligned}x_k^+ &= (\xi_{k;[2,2]})^{-1}\xi_{k;[2,1]}\\ &= P_k^+\left[(P_k^-)^{-1}(x_k^-)^T + H^T R^{-1} z_k\right]\\ &= x_k^- + P_k^+ H^T R^{-1}(z_k - H x_k^-),\end{aligned} \tag{2.50}$$

where

$$K_k = P_k^- H^T (R + H P_k^- H^T)^{-1} \tag{2.51}$$

is the Kalman gain. This form is the optimal Kalman gain, as it minimizes the root mean squared error. In general, the greater the gain, the greater the emphasis of the new measurements. The filter is then more sensitive.

There are several possible formulas that express the optimal Kalman filter. One of the most popular formulations is as follows:

$$\hat{z}_k^- = H\hat{x}_k^-, \qquad \text{(measurement prediction)} \tag{2.52}$$
$$\nu_t = z_t - \hat{z}_k^-, \qquad \text{(innovation or prediction error } z_t\text{)} \tag{2.53}$$
$$S_t = H P_k^- H^T + R, \qquad \text{(covariance of innovation } \nu_t\text{)} \tag{2.54}$$
$$K_t = P_k^- H^T S_t^{-1}, \qquad \text{(Kalman gain)} \tag{2.55}$$
$$x_k^+ = \hat{x}_k^- + K_t \nu_t, \qquad \text{(posterior state estimation } x_t\text{)} \tag{2.56}$$
$$P_k^+ = (I - K_t H)P_k^-. \qquad \text{(posterior covariance)} \tag{2.57}$$

The pseudo-algorithm of Kalman filter is shown in Algorithm 1.

Algorithm 1 Kalman Filter Algorithm

Inputs: Initial state estimate $\hat{x}_0$, initial covariance matrix P_0, system dynamics matrix F, process noise covariance matrix Q, measurement matrix H, measurement noise covariance matrix R, and measurements Z.
Outputs: Updated state estimate $\hat{x}_k$ and updated covariance matrix P_k.

procedure INITIALIZATION(x_0, P)
    $\hat{x}_0 = x_0$ ▷ Initialize state estimate
    $P_0 = P$ ▷ Initialize error covariance matrix
end procedure

procedure PREDICTION STEP
    $\hat{x}_{k|k-1} = F\hat{x}_{k-1}$ ▷ Predict state estimate
    $P_{k|k-1} = FP_{k-1}F^T + Q$ ▷ Predict covariance:
end procedure

procedure UPDATE STEP(z_k)
    $K_k = P_{k|k-1}H^T (HP_{k|k-1}H^T + R)^{-1}$ ▷ Kalman gain
    $\hat{x}_k = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1})$ ▷ Update state estimate
    $P_k = (I - K_k H)P_{k|k-1}$ ▷ Update covariance
end procedure

procedure KALMAN FILTER RECURSION
    Initialization
    for all $z_k \in Z$ **do**
        Perform Prediction Step
        Perform Update Step with z_k
    end for
end procedure

Chapter 3

Target tracking

Target tracking plays a pivotal role in radar systems, where the primary objective is to estimate the trajectory, position and other characteristics of moving objects within a surveillance area. Derived from the foundational principles of the Kalman filter, target tracking algorithms are tailored to the specific requirements and challenges inherent in radar applications.

Radar-based target tracking encounters various complexities, including survivability and the presence of multiple targets in the same neighbour. Survivability concerns the algorithm's ability to maintain accurate estimates despite target maneuvers, occlusions, or potential target loss. Furthermore, the phenomenon of multiple targets in the same neighbour introduces significant ambiguities and challenges in trajectory estimation and association.

In the realm of single-target tracking, algorithms such as the Probabilistic Data Association (PDA) filter and its derivatives, such as the Integrated Probabilistic Data Association (IPDA) filter or multi-target tracking variation Joint Probabilistic Data Association (JPDA) filter, are commonly employed. These algorithms provide robust solutions for associating measurements with existing tracks and updating target state estimates in dynamic scenarios.

Beyond single-target tracking, radar systems often necessitate multi-target tracking approaches. These approaches, particularly those based on Random Finite Sets (RFS), offer advanced capabilities for tracking multiple targets simultaneously. RFS-based filters, including the Probability Hypothesis Density (PHD) filter and the Multi-Bernoulli filter, excel in scenarios where the number of targets is uncertain or dynamic.

In this chapter, we delve into the intricacies of radar-based target tracking algorithms, exploring their theoretical foundations, practical implementations, and performance characteristics. By understanding the diverse range of tracking algorithms available and their respective strengths, we can design and deploy effective tracking systems to meet the demands of modern surveillance and reconnaissance applications.

3.1 Data association

Data association uncertainty arises in remote sensing systems, such as radar, sonar, or electro-optical devices, when measurements are obtained from sources that may not necessarily be the target of interest [22]. This uncertainty occurs particularly in situations where the target signal is weak, necessitating a lower detection threshold, which may result in the detection of background signals, sensor noise or clutter. Additionally, data association uncertainty can occur when multiple targets are present in close proximity. Utilizing spurious measurements in a tracking filter can lead to divergence of the estimation error and, consequently, track loss.

Addressing this challenge involves two primary problems. The first is the selection of appropriate measurements to update the state of the target of interest in the tracking filter, which can be a Kalman filter or an extended Kalman filter. The second problem involves determining whether the filter needs modifications to account for data association uncertainty. The objective is to obtain the minimum mean square error (MMSE) estimate of the target state and associated uncertainty.

The optimal estimator involves the recursive computation of the conditional probability density function of the state, with detailed conditions provided under which this pdf serves as a sufficient statistic in the presence of data association uncertainty.

3.2 Clutter

When it comes to clutter, two scenarios may occur. The first one is a single target in clutter. This problem of tracking a single target in clutter appears when several measurements appear in the validation region. The validated measurements comprise the accurate measurement, if detected within this region, along with spurious measurements originating from clutter or false alarms. In air traffic control systems, where cooperative targets are involved, each measurement includes a target identifier known as the squawk number. If this identifier is entirely reliable, data association uncertainty is eliminated. However, in cases where a potentially hostile target is non-cooperative, data association uncertainty becomes a significant challenge.

3.2.1 Validation region

In target tracking scenarios, the process of signal detection provides measurements, from which the appropriate ones for inclusion in the target state estimator are chosen. In radar systems, for instance, the signal reflected from the target of interest is sought within a specific time interval, determined by the expected range of the target when it reflects the transmitted energy. A range gate is established, and detections falling within this gate can be associated with the target of interest. These measurements could include informations such as range, azimuth, elevation, or direction cosines. For radar or active sonar, we might also measure how fast something is moving toward or away from us. For passive sonar, we might look at the direction something's coming from, when it arrives, and the difference in sounds it makes. And for optical sensors, we might measure the angle it is seen from. By setting up a multidimensional gate, the signal from the target

is detected efficiently, avoiding the need to search for it across the entire measurement space.

However, while a measurement within the gate is a candidate for association with the target, it is not guaranteed to have originated from the target itself. Thus, the establishment of a validation region becomes necessary. The validation region is designed to ensure that the target measurement falls within it with a high probability, known as the gate probability, based on the statistical characterization of the predicted measurement. In the event, where more than one detection appears within the gate, association uncertainty arises. With this uncertainty, it is important to figure out which measurement really comes from the target and should be used to update our tracking information. This includes things like the target's estimated position and its variability, or more broadly, the key information we need about the target. Measurements outside the validation region can be disregarded, as they are too distant from the predicted measurement and are unlikely to have originated from the target of interest. This scenario typically arises when the gate probability is close to unity, and the statistical model used to define the gate is accurate.

Figure 3.1 illustrates a scenario involving multiple validated measurements. The validation region depicted in the figure is two-dimensional and takes the form of an ellipse centered at the predicted measurement $\hat{z}$. The elliptical shape of the validation region arises from the assumption that the error in the target's predicted measurement, known as the innovation, follows a Gaussian distribution. The parameters defining the ellipse are determined by the covariance matrix S of the innovation.

All measurements within the validation region have the potential to originate from the target of interest, although only one of them is the true measurement. As a result, the possible association events include: z_1 originating from the target, with z_2 and z_3 being from clutter; z_2 originating from the target, with z_1 and z_3 being from clutter; z_3 originating from the target, with z_2 and z_1 being from clutter; or all measurements being from clutter. These association events are mutually exclusive and exhaustive, enabling the application of the total probability theorem to obtain the state estimate in the presence of data association uncertainty.

Under the assumption of a single target, the spurious measurements are considered random interference. A common model for such false measurements assumes that they are uniformly spatially distributed and independent across time, corresponding to residual clutter. Any constant clutter is assumed to have already been removed.

When several targets as well as clutter or false alarms appear in the same neighbour, the data association becomes even more challenging. Figure 3.2 shows this scenario, where predicted measurement for the targets are close to each other. These predicted points are labeled as $\hat{z_1}$ and $\hat{z_2}$. In this figure many association combinations are possible; z_1 from target $\hat{z_1}$ or clutter; z_2 from target $\hat{z_1}$ or clutter; z_4 from target $\hat{z_2}$ or clutter; z_3 from target $\hat{z_1}$ or target $\hat{z_2}$ or clutter. However, if z_3 originated from target $\hat{z_1}$, then it is probable, that z_4 originated from target $\hat{z_2}$.

This scenario demonstrates the intricate relationships among associations when persistent interference from neighboring targets coexists with random interference or clutter. In such cases, joint association events become necessary to properly account for these dependencies.

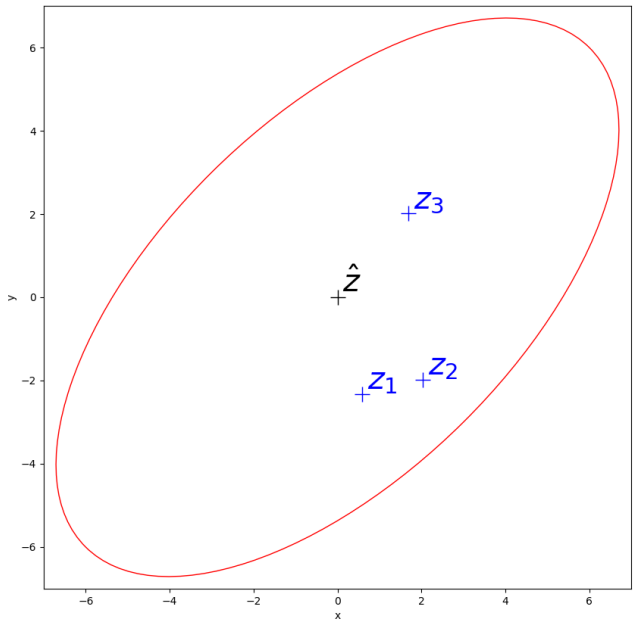


Figure 3.1 Several measurements z_i appeared in the validation region of a single target. $\hat{z}$ is a predicted measurement and none or any of the measurement $z_1 - z_3$ may have originated from the target.

A more complex scenario may arise due to the inherent finite resolution capability of signal processing systems. While each measurement is typically assumed to originate from either a target or clutter, an additional possibility must be considered: the merging of detections from multiple targets. Specifically, measurement z_3 could potentially result from such merging, representing an unresolved measurement. This introduces a fourth origin hypothesis for a measurement lying within the intersection of two validation regions.

The discussion highlights the challenges associated with associating measurements to tracks. The complete problem involves associating measurements at each time step, updating the track's sufficient statistic, and propagating it to the subsequent time step.

3.3 Single target tracking

Single Target Tracking algorithms are specifically designed to track a single target in presence of clutter and other sources of interference. Unlike conventional Kalman filters, which is not inherently equipped to handle clutter. Single Target Tracking algorithms incorporate mechanisms to enhance target survivability in cluttered environments.

One prominent algorithm in the data association category of target tracking is the Probabilistic Data Association filter. The PDA filter addresses the challenge of associating measurements with the correct target while accounting for data association uncertainty. This uncertainty arises from the presence of clutter and the possibility of

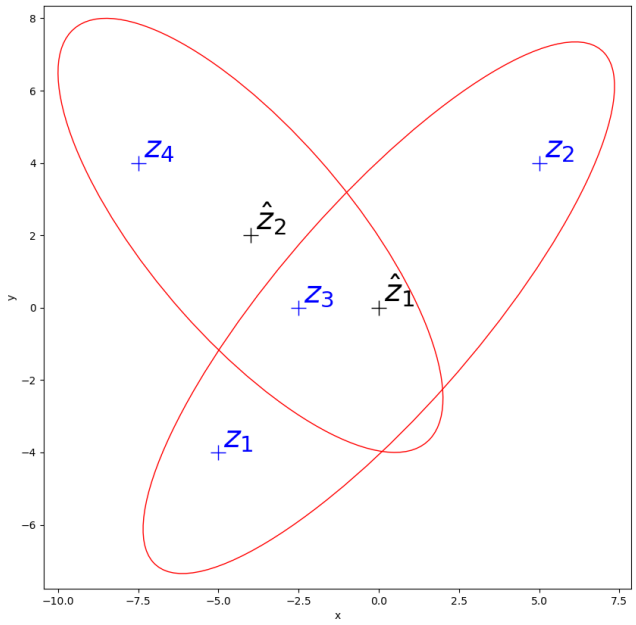


Figure 3.2 Several measurements z_i appeared in the validation region of one of targets $\hat{z_1}$ or $\hat{z_2}$. $\hat{z_1}$ and $\hat{z_2}$ are predicted measurements and none or any of the measurement $z_1 - z_3$ may have originated from the target $\hat{z_1}$ and none or any of the measurement $z_3 - z_4$ may have originated from the target $\hat{z_2}$.

spurious measurements.

3.3.1 PDA filter

The Probabilistic Data Association algorithm computes the association probabilities for each validated measurement at the current time with respect to the target being tracked. This probabilistic or Bayesian information serves as a foundation for the Probabilistic Data Association Filter tracking algorithm, which effectively addresses the uncertainty regarding the origin of measurements. In scenarios where the state and measurement equations are linear, the resulting PDAF algorithm operates based on the Kalman Filter. However, if the state or measurement equations are non-linear, the PDAF algorithm is instead based on the Extended Kalman Filter. This adaptation allows the PDAF algorithm to accommodate non-linearity in the system dynamics or measurement processes, ensuring robust performance in diverse tracking scenarios.

The PDAF algorithm comes with many assumptions:

1. Only one target of interest is present, whose state $x \in R^{n_x}$ is assumed to evolve in time according to the equation

$$x_k = F_{k-1}x_{k-1} + w_{k-1}, \tag{3.1}$$

with the true measurement $z_k \in R^{n_z}$ given by

$$z_k = H_k x_k + v_k, \tag{3.2}$$

where w_{k-1} and v_k are zero mean mutually independent, white Gaussian noise variables with covariances Q_{k-1} and R_k, respectively.

2. The track has been initialized.

3. The past information through time $k-1$ about the target is summarized approximately by a suffifient statistic in the form of the Gaussian posterior

$$p(x_{k-1}|z_{k-1}) = \mathcal{N}(x_{k-1}; x_{k-1|k-1}, P_{k-1|k-1}). \tag{3.3}$$

4. At each time a measurement validation region is set up around the predicted measurement to select the candidate measurements for association to the target of interest.

5. If the target was detected and the corresponding measurement fell into the validation region, then, according to (3.2), at most one of the validated measurements can be target originated.

6. The remaining measurements are assumed to be due to false alarms or clutter and are modeled as independent and identically distributed with uniform spatial distribution, and the number of false alarms or clutter points obeys either a Poisson distribution, that is, a spatial Poisson process with known spatial density λ, or a diffuse prior.

7. The target detections occur independently over time with known probability p_D.

All these assumptions make a filter for state estimation, that is almost as simple as Kalman filter, but more effective in the presence of clutter.

As well as the Kalman Filter, the PDAF also consists of prediction and update step. Before the update step, two additional steps occur – measurement validation and data association.

3.3.1.1 PDAF predict step

The prediction step of PDA filter from time $k-1$ to k is as in the standard KF,

$$x_{k|k-1} = F_{k-1} x_{k-1|k-1}, \tag{3.4}$$

$$\eta_{k|k-1} = H_k x_{k|k-1}, \tag{3.5}$$

$$P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_{k-1}, \tag{3.6}$$

where $P_{k-1|k-1}$ is from the Equation (3.3). The innovation covariance matrix S_k is computed as

$$S_k = H_k P_{k|k-1} H_k^T + R_k. \tag{3.7}$$

3.3.1.2 PDAF measurement validation step

As discussed in Section 3.2.1, measurements considered to be originated from a given target should fall into validation region. This elliptical region is formulated as

$$\nu(k,\gamma) = \bigcup_{z \in Z_k} [(z - \hat{z}_{k|k-1})^T S_k^{-1} (z - \hat{z}_{k|k-1})) \leq \gamma], \tag{3.8}$$

where Z_k are all measurements at time k, $\gamma > 0$ is the gate threshold corresponding to the gate probability p_G, which is the probability, that the region contains the correct measurement if detected and S_k given in (3.7) is the covariance of the innovation.

3.3.1.3 PDAF data association step

The clutter in target tracking is assumed to be Poisson model with spatial density λ. In PDAF it yields to the association probability for $z_{i,k}$ being the correct measurement as

$$\beta_{i,k} = \begin{cases} \dfrac{\mathcal{L}_{i,k}}{1 - p_D p_G + \sum_{j=1}^{m_k} \mathcal{L}_{j,k}}, & i = 1, \ldots, m_k, \\ \dfrac{1 - p_D p_G}{1 - p_D p_G + \sum_{j=1}^{m_k} \mathcal{L}_{j,k}}, & i = 0, \end{cases} \tag{3.9}$$

where $i = 0$ stands for case, where none of the measurements is correct, i.e. there is not any measurement in validation region or none of them originate from the target, p_D is the detection probability, p_G is the gate probability analogous to (3.8), and

$$\mathcal{L}_{i,k} = \frac{\mathcal{N}(z_{i,k}; \hat{z}_{k|k-1}, S_k) p_D}{\lambda} \tag{3.10}$$

is the likelihood ratio of the measurement $z_{i,k}$ from the validation region. The parameter λ (the density of the spatial Poisson clutter process) in the denominator of (3.10) models the clutter density of the uniform pdf of the location of false measurement. The Probabilistic Data Association algorithm produces association probabilities that are influenced by the position of the respective innovation on the Gaussian exponential of the likelihood ratio (3.10), in comparison to other innovations.

3.3.1.4 PDAF update step

The update step equation of update step is formulated as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \nu_k, \tag{3.11}$$

where the combined innovation is

$$\nu_k = \sum_{i=1}^{m_k} \beta_{i,k} \nu_{i,k}, \tag{3.12}$$

and the Kalman gain is the same as in (2.51)

$$K_k = P_{k|k-1} H_k^T S_k^{-1}. \tag{3.13}$$

The covariance matrix associated with the update state (3.11) is expressed as

$$P_{k|k} = \beta_{0,k} P_{k|k-1} + (1 - \beta_{0,k}) P_{k|k}^c + \tilde{P}_k, \tag{3.14}$$

where the covariance of the updated state with the corrected measurement is

$$P_{k|k}^c = P_{k|k-1} - K_k S_k K_k^T, \tag{3.15}$$

and of the innovation term spreads according to

$$\tilde{P}_k = K_k \left(\sum_{i=1}^{m_k} \beta_{i.k} \nu_{i,k} \nu_{i,k}^T - \nu_k \nu_k^T \right) K_k^T. \tag{3.16}$$

With probability $\beta_{0,k}$ none of the measurement is correct. In such case, there is no update step of the state estimation. The prediction covariance $P_{k|k-1}$ in (3.14) has weight $\beta_{0,k}$. There is probability $1 - \beta_{0,k}$ that correct measurement appears and the update covariance $P_{k|k}^c$ has weight $1 - \beta_{0,k}$. However, there is no certainty, which of the validated measurement z_k is correct, so the positive semidefinite covariance matrix increases. This matrix is the result of the measurement origin uncertainty. Note, that in (3.16) the dependence of the estimation is non-linear. The PDAF is non-linear estimator, while the estimate update in (3.11) appears linear, but the association probabilities $\beta_{i,k}$ depend on the innovation in (3.9).

3.4 Multi target tracking

In the field of multi-target tracking, there are many different algorithms and approaches. One of the classes is the particle filters approach ([3], [23]–[25]). These filters are able to handle non-linear motion, but the computational demands are really high. The second class is bayesian filtering, which is usually more effective when it comes to computational demands. Moreover there are two different approaches – data association filters and random finite sets filters . The example of data association, which we talked about in Section 3.1, is the PDA filter (Section 3.3.1) or the integrated probabilistic data association filter. The updated version of these filters, that can handle more targets, especially when two or more targets' validation region is in the same neighbourhood are the joint probabilistic data association filter and the joint integrated probabilistic data association filter. The other family is filters based on random finite sets. Random finite sets statistics are explained in the following section.

3.4.1 RFS statistics

Multi-target tracking presents challenges in estimating the states of multiple dynamic objects over time, complicated by varying target counts, cluttered sensor measurements,

and data association ambiguity. Traditional tracking frameworks rely on explicit associations between measurements and targets, leading to computational complexities, particularly in scenarios with high target densities and clutter rates.

Recent advancements introduce Random Finite Set theory as an alternative approach to multi-target tracking. RFS theory represents sets with random cardinalities and values, offering a flexible framework for modeling uncertain multi-target scenarios. By treating both the multitarget state and sensor measurements as RFS, tracking algorithms can capture uncertainties effectively.

In scenario with more targets, i.e., multitarget scenario, let $M(k-1)$ be the number of targets at time k. At time $k-1$ the target states are $x_{k-1,1}, \ldots, x_{k-1,M(k-1)} \in \mathcal{X}$. At the following time step k, some of these targets may disappear, some may evolve to their new states and also some new targets may be borned. As a result we have $M(k)$ targets with states $x_{k,1}, \ldots, x_{k,M(k)} \in \mathcal{X}$. The RFS model formulation ensures, that the order in which the states are listed has no significance. Also, at time k, $N(k)$ measurements $z_{k,1}, \ldots, z_{k,N(k)} \in \mathcal{Z}$ are received by the sensors, but the origins of these measurements are not known and the RFS model again ensures, that the order in which the measurements came, has no significance. Some of these measurements are generated by the targets, some are only false measurements, i.e., clutter.

The goal of multiple-target tracking is to collectively estimate both the quantity of targets and their respective states using measurements of uncertain origin. Even under ideal circumstances where the sensor detects all targets without clutter, single-target filtering techniques are inadequate due to the absence of information regarding the origin of each observation.

Given the absence of inherent ordering within the sets of target states and measurements at a specific time, they can be naturally depicted as finite sets, denoted as

$$X_k = \{x_{k,1}, \ldots, x_{k,M(k)}\} \in \mathcal{F}(\mathcal{X}) \tag{3.17}$$

$$Z_k = \{z_{k,1}, \ldots, z_{k,N(k)}\} \in \mathcal{F}(\mathcal{Z}), \tag{3.18}$$

where $\mathcal{F}(\mathcal{X})$ and $\mathcal{F}(\mathcal{Z})$ are collections of all finite subsets of $\mathcal{X}$ and $\mathcal{Z}$, respectively. In the random finite set approach, we consider the sets of targets and measurements, denoted as X_k and Z_k respectively, as the state and observation. This perspective allows us to frame the multi-target tracking problem as a filtering task with a state space $\mathcal{F}(\mathcal{X})$ and an observation space $\mathcal{F}(\mathcal{Z})$.

In a scenario with a single target, uncertainty is typically represented using random vectors to model the state x_k and the measurement z_k. Similarly, in a multi-target scenario, we represent uncertainty by using random finite sets to model the multie-target state X_k and the multi-target measurement Z_k. An RFS X is essentially a random variable that takes on finite sets of values, described by a discrete probability distribution and a set of joint probability densities. The distribution specifies the number of elements in X, while the densities describe the distribution of these elements.

We now delve into an RFS-based model for the evolution of the multi-target state, taking into account target motion, birth and death. Given a multi-target state X_{k-1} at time $k-1$, each individual target $x_{k-1} \in X_{k-1}$ either remains present at time k with probability $p_{S,k}(x_{k-1})$ or disappears with probability $1 - p_{S,k}(x_{k-1})$. If the target continues to exist, the probability density of transitioning from state x_{k-1} to x_k is denoted

by $f_{k|k-1}(x_k|x_{k-1})$. Consequently, for each state $x_{k-1} \in X_{k-1}$ at time $k-1$, its behavior in the subsequent time step is modeled as the RFS

$$S_{k|k-1}(x_{k-1}), \tag{3.19}$$

that can represent two scenarios: either the survival of a target, indicated by the set $\{x_k\}$, or the absence of a target, denoted by $\emptyset$, implying the target's demise.

The occurrence of a new target at time k can originate from two possibilities: spontaneous births, which are independent of any existing target, or the spawning from a target present at time $k-1$.

Given a multi-target state X_{k-1} at time $k-1$, the state X_k at time k is formulated as the union of surviving targets, newly spawned targets, and spontaneously borned targets

$$X_k = \left[\bigcup_{\zeta \in X_{k-1}} S_{k|k-1}(\zeta)\right] \cup \left[\bigcup_{\zeta \in X_{k-1}} B_{k|k-1}(\zeta)\right] \cup \Gamma_k, \tag{3.20}$$

where

- Γ_k is a RFS of spontaneous births at time k,
- $B_{k|k-1}(\zeta)$ is a RFS of targets spawned at time k from a target with previous state ζ.

It is also assumed that RFSs in (3.20) are independent of each other, but the forms of Γ_k and $B_{k|k-1}(\cdot)$ are problem dependent.

The measurement sets Z_k are also random finite sets including detection uncertainty and clutter. A target $x_k \in X_k$ can be either detected with probability $p_{D,k}(x_k)$ or misdetected with probability $1 - p_{D,k}(x_k)$. Moreover, at time k, each state x_k produces an RFS

$$\Theta_k(x_k) \tag{3.21}$$

that is z_k in a case when the target is detected or $\emptyset$ otherwise. Furthermore, the sensor not only receives measurements originated from targets, but also a set K_k, which are false measurements (clutter). Thus, at time k, the multi-target measurement set Z_k observed by the sensor is also formulated as a union of measurements originated from targets and clutter

$$Z_k = \left[\bigcup_{x \in X_k} \Theta_k(x)\right] \cup \mathcal{K}_k. \tag{3.22}$$

The RFSs are assumed to be independent of each other and the actual form of K_k is problem dependent.

Finite Set Statistics (FISST) is fundamental within the RFS framework, providing a systematic way to apply RFS theory to multi-target tracking. FISST extends Bayesian filtering techniques to multi-target scenarios, enabling rigorous estimation of multi-target states in the presence of clutter and data association uncertainties. Datailed descriptions, informations and equations can be found in [26]. In this book, for sake of interest , we

can find out that for a function $f = \partial F_{(\cdot)}(T)$ the set integral over a subset $S \subset E$ is defined as

$$\int_S f(X)\partial X \equiv \sum_{i=0}^{\infty} \frac{1}{i!} \int_{S^i} f(x_1, \ldots, x_i)\lambda_K^i(dx_1 \ldots dx_i), \tag{3.23}$$

where $\lambda_K(S)$ denote the hyper-volume of S in units of K, which is usually in MTT modeled as RFS Poisson process with uniform rate of K^{-1} with intensity $\lambda = \lambda_K/K$, i.e.,

$$\mu(\mathcal{T}) = \sum_{i=0}^{\infty} \frac{\lambda^i(\mathcal{T} \cap E^i)}{i!}, \tag{3.24}$$

where E and $\mathcal{T}$ are closed and bounded subsets of R^n [27].

3.4.2 PHD filter

The PHD filter serves as an approximation devised to tackle the computational complexity inherent in the multi-target Bayes filter. Instead of tracking the complete posterior density of multi-target states over time, the PHD filter focuses on propagating the posterior intensity, which represents the first-order statistical moment of the posterior multi-target state [5]. This operational strategy draws parallels with the constant gain Kalman filter, which tracks the first moment (mean) of the single-target state [18].

For a Random finite set X on $\mathcal{X}$ with probability distribution P, its first-order moment is a non-negative function v on $\mathcal{X}$, known as the intensity. This function satisfies the property that for each region $S \subseteq \mathcal{X}$ [28]

$$\int |X \cap S| P(dX) = \int_S v(x)dx \tag{3.25}$$

In essence, the integral of v over any region S yields the expected number of elements of X within S. Consequently, the total mass $\hat{N} = \int v(x)dx$ represents the expected number of elements of X. The local maxima of the intensity v correspond to points in $\mathcal{X}$ with the highest local concentration of expected number of elements, which can then be utilized to generate estimates for the elements of X. The simplest approach involves rounding $\hat{N}$ and selecting the resulting number of highest peaks from the intensity. In the tracking domain, this intensity is also referred to as the probability hypothesis density.

An important subclass of RFSs, known as Poisson RFSs, is characterized entirely by their intensities. An RFS X is considered Poisson if its cardinality distribution $Pr(|X| = n)$ follows a Poisson distribution with mean $\hat{N}$, and for any finite cardinality, the elements $x \in X$ are independently and identically distributed according to the probability density $v(\cdot)/N$. In the context of the multi-target tracking problem, it is common practice to model the clutter RFS [denoted as $\mathcal{K}_k$ in (3.22)] and the birth RFSs [Γ_k and $B_{k|k-1}(x_{k-1})$ in (3.20)] as Poisson RFSs.

To formulate PHD filter, let first denote multi-target state and measurement models

with

$\gamma_k(\cdot)$ intensity of the births RFS Γ_k at time k; (3.26)

$\beta_{k|k-1}(\cdot|\zeta)$ intensity of the RFS $B_{k|k-1}(\zeta)$ spawned at time k by a target with previous state ζ; (3.27)

$p_{S,k}(\zeta)$ probability that a target still exists at time k given that its previous state is ζ; (3.28)

$p_{D,k}(x)$ probability of detection given a state x at time k; (3.29)

$\kappa_k(\cdot)$ intensity of clutter RFS $\mathcal{K}_k$ at time k. (3.30)

PHD filter naturally comes with certain assumptions.

1. Each target evolves and generates observations independently of each other.
2. Clutter is Poisson and independent of target-originated measurements.
3. The predicted multi-target RFS governed by $p_{k|k-1}$ is Poisson.

Assumptions 1 and 2 are very common in several tracking applications. The assumption 3 is acceptable approximation in situations with small correlation between targets and is satisfied when there is no spawning and RFSs X_{k-1} and Γ_k are Poisson.

3.4.2.1 PHD filter recursion

Let assume a target state x described by an intensity function $v(x)$. At time k, the prediction of the prior intensity $v_{k-1}(x)$ is given by

$$\begin{aligned} v_{k|k-1}(x) = &\int p_{S,k}(\zeta)\phi_{k|k-1}(x|\zeta)v_{k-1}(\zeta)d\zeta \\ &+ \int \beta_{k|k-1}(x|\zeta)v_{k-1}(\zeta)d\zeta + v_{\gamma,k}(x), \end{aligned} \tag{3.31}$$

where $p_{S,k}(\cdot)$ is the probability of target survival, $\phi_{k|k-1}(\cdot|\cdot)$ is the target state transition density, and $v_{\gamma,k}(\cdot)$ denotes the prior PHD of the target's birth at time k. The predicted intensity $v_{k|k-1}$ is then updated by the measurement set Z_k given by sensors at time k according to the Bayesian update

$$\begin{aligned} v_k(x) = &[1 - p_{D,k}(x)]v_{k|k-1}(x) \\ &+ \sum_{z\in Z_k} \frac{p_{D,k}(x)g_k(z|x)v_{k|k-1}(x)}{\kappa_k(z) + \int p_{D,k}(\zeta)g_k(z|\zeta)v_{k|k-1}(\zeta)d\zeta}, \end{aligned} \tag{3.32}$$

where $g_k(\cdot|\cdot)$ is the likelihood function, $p_{D,k}(\cdot)$ is the probability of detection, and $\kappa_k(\cdot)$ is the clutter density.

Equations (3.31) and (3.32) show that the PHD filter avoids the combinatorial ineffectivity awakening from the unknown association of measurements with correct targets. The posterior intensity is a function of a single-target state space $\mathcal{X}$, thus the recursion is more computationally effective than (2.35) and (2.36) that in the case of the PHD filter operates on $\mathcal{F}(\mathcal{X})$. But in general it is not possible to come up with a closed-form solution for the PHD recursion.

3.4.2.2 GM-PHD for linear models

In this section we delve into details of multi-target linear PHD Gaussian models. With PHD recursions (3.31) and (3.32) it is achievable to get an efficient multi-target closed-form solution for multi-target algorithms. This solution proposed by Vo and Ma in [18] requires additional assumptions to the already proposed ones. Not only standard linear Gaussian model for each targets are assumed, but also the linear Gausssian multi-target model includes assumptions on the death, birth and detections.

4. Each target follows a linear Gaussian dynamical model and the sensor has a linear Gaussian measurement model, i.e.,

$$f_{k|k-1}(x|\zeta) = \mathcal{N}(x; F_{k-1}\zeta, Q_{k-1}), \tag{3.33}$$

$$g_k(z|x) = \mathcal{N}(z; H_k x, R_k), \tag{3.34}$$

where $\mathcal{N}(\cdot; m, P)$ is a Gaussian density with mean m and covariance P, F_{k-1} is the state transition matrix, Q_{k-1} is the process noise covariance, H_k is the observation matrix and R_k is the observation noise covariance.

5. The survival and detection probabilities are state independent, i.e.,

$$p_{S,k}(x) = p_{S,k}, \tag{3.35}$$

$$p_{D,k}(x) = p_{D,k}. \tag{3.36}$$

6. The intesities of the birth and spawn RFSs are Gaussian mixtures of the form

$$\gamma_k(x) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(x; m_{\gamma.k}^{(i)}, P_{\gamma,k}^{(i)}), \tag{3.37}$$

$$\beta_{k|k-1}(x|\zeta) = \sum_{j=1}^{J_{\beta,k}} w_{\beta,k}^{(j)} \mathcal{N}(x; F_{\beta,k-1}^{(j)}\zeta + d_{\beta,k-1}^{(j)}, Q_{\beta,k-1}^{(j)}), \tag{3.38}$$

where $J_{\gamma,k}, w_{\gamma,k}^{(i)}, m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}, i = 1, \ldots, J_{\gamma,k}$ are given model parameters that actuate the shape of the birth intensity and $J_{\beta,k}, w_{\beta,k}^{(j)}, F_{\beta,k-1}^{(j)}, d_{\beta,k-1}^{(j)}$ and $Q_{\beta,k-1}^{(j)}, j = 1, \ldots, J_{\beta,k}$ determine the shape of the spawning intensity of a target with previous state ζ.

Assumptions 4 and 5 are very common and used in various tracking algorithms [29], but the closed-form solution allows for state- and time-dependent $p_{S,k}$ and $p_{D,k}$. This convenience is used in the practical part of this thesis.

The means $m_{\gamma,k}^{(i)}$ in assumption 6 are the peaks of the spontaneous birth intensity in (3.37). These are the points with highest concentrations of the expected number of spontaneous births, meaning they represent places like airports or other places with highest probability target appearance. $P_{\gamma,k}^{(i)}$ is the covariance matrix that regulate the spread of the birth intensity. The weight $w_{\gamma,k}^{(i)}$ gives the expected number of new targets originating from $m_{\gamma,k}^{(i)}$. Similary in (3.38) with addition, that the peak $F_{\beta,k-1}^{(j)}\zeta + d_{\beta,k-1}^{(j)}$ is an affine function of ζ. This spawned target can be pictured as a new target spawned in the neighborhood of another target with previous state ζ. This can be, for example, a smaller ship detached from its mother ship.

3.4.2.3 GM-PHD recursion for linear models

In this section we delve into Gaussian mixture probability hypothesis density filter recursion for the linear Gaussian multi-target model and its closed-form solution to the (3.31) and (3.32). Suppose, that with assumptions 4-6 the posterior intensity at time $k-1$ is a Gaussian mixture of the form

$$\nu_{k-1}(x) = \sum_{i=1}^{J_{k-1}} w_{k-1}^{(i)} \mathcal{N}(x; , m_{k-1}^{(i)}, P_{k-1}^{(i)}). \tag{3.39}$$

Then, the predicted intensity for time k is also a Gaussian mixture and is given by

$$\nu_{k|k-1}(x) = \nu_{S,k|k-1}(x) + \nu_{\beta,k|k-1}(x) + \gamma_k(x), \tag{3.40}$$

where $\gamma_k(x)$ is given in (3.37). This yields

$$\nu_{S,k|k-1}(x) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(x; m_{S,k|k-1}^{(j)}, P_{S,k|k-1}^{(j)}), \tag{3.41}$$

$$m_{S,k|k-1}^{(j)} = F_{k-1} m_{k-1}^{(j)}, \tag{3.42}$$

$$P_{S,k|k-1}^{(j)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^T, \tag{3.43}$$

$$\nu_{\beta,k|k-1}(x) = \sum_{j=1}^{J_{k-1}} \sum_{l=1}^{J_{\beta,k}} w_{k-1}^{(j)} w_{\beta,k}^{(l)} \mathcal{N}(x; m_{\beta,k|k-1}^{(j,l)}, P_{\beta,k|k-1}^{(j,l)}), \tag{3.44}$$

$$m_{\beta,k|k-1}^{(j,l)} = F_{\beta,k-1}^{(l)} m_{k-1}^{(j)} + d_{\beta,k-1}^{(l)}, \tag{3.45}$$

$$P_{\beta,k|k-1}^{(j,l)} = Q_{\beta,k-1}^{(l)} + F_{\beta,k-1}^{(l)} P_{\beta,k-1}^{(l)} (F_{\beta,k-1}^{(l)})^T. \tag{3.46}$$

Thus the predicted intensity at the time k is a Gaussian mixture

$$\nu_{k|k-1}(x) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}), \tag{3.47}$$

and the posterior intensity at time k is also a Gaussian mixture

$$\nu_k(x) = (1 - p_{D,k}) \nu_{k|k-1}(x) + \sum_{z \in Z_k} \nu_{D,k}(x; z), \tag{3.48}$$

where

$$\nu_{D,k}(x;z) = \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z)\mathcal{N}(x;m_{k|k}^{(j)}(z),P_{k|k}^{(j)}), \tag{3.49}$$

$$w_k^{(j)}(z) = \frac{p_{D,k}^{(j)}(x;z)w_{k|k-1}^{(j)}q_k^{(j)}(z)}{\kappa_k(z)+p_{D,k}^{(j)}(x;z)\sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)}q_k^{(l)}(z)}, \tag{3.50}$$

$$m_{k|k}^{(j)}(z) = m_{k|k-1}^{(j)} + K_k^{(j)}(z - H_k m_{k|k-1}^{(j)}), \tag{3.51}$$

$$P_{k|k}^{(j)}(z) = [I - K_k^{(j)}H_k]P_{k|k-1}^{(j)}, \tag{3.52}$$

$$K_k^{(j)}(z) = P_k^{(j)}H_k^T(H_k P_{k|k-1}^{(j)}H_k^T + R_k)^{-1}. \tag{3.53}$$

Formulations (3.39), (3.40) and (3.47), (3.48) are the result of applying standard Gaussian properties, i.e., given F, d, Q, m and P of appropriate dimensions and Q, P are positive definite matrices, this yields

$$\int \mathcal{N}(x;F\zeta+d,Q)\mathcal{N}(\zeta;m,P)d\zeta = \mathcal{N}(x;Fm+d,Q+FPF^T) \tag{3.54}$$

and given H, R, m and P of appropriate dimensions assuming R and P are positive definite matrices

$$\mathcal{N}(z;Hx,R)\mathcal{N}(x;m,P) = q(z)\mathcal{N}(x;\tilde{m},\tilde{P}), \tag{3.55}$$

where

$$q(z) = \mathcal{N}(z;Hm,R+HPH^T), \tag{3.56}$$

$$\tilde{m} = m + K(z - Hm), \tag{3.57}$$

$$\tilde{P} = (I - KH)P, \tag{3.58}$$

$$K = PH^T(HPH^T + R)^{-1}. \tag{3.59}$$

Equations (3.33), (3.35), and (3.37) - (**??**) are inserted into the PHD prediction Equation (3.31), and then replacing integrals of the type (3.54) with suitable Gaussian functions. Similarly, Formulations (3.47), (3.48) are established by substituting Equations (3.34), (3.36), and (3.47) into the PHD update Equation (3.32), and subsequently replacing integrals resembling Form (3.54) and products of Gaussians resembling Form (3.55) with appropriate Gaussian functions.

If we comply prior intensity ν_0 as a Gaussian mixture, then all following predicted intensities $\nu_{k|k-1}$ and posterior intensities ν_k are also Gaussian mixtures and the recursion holds. Formulations (3.39) and (3.40) give a closed-form guidance for computing the means, covariances and weights of ν_k from $\nu_{k|k-1}$ after a new set of measurements is available.

The intensity $\nu_{k|k-1}$ in (3.40) includes three intensities $\nu_{S,k|k-1}$, $\nu_{,k|k-1}$ and γ_k. These intensities represent existing targets, spawned targets and spontaneous births. The updated intensity ν_k consists a misdetection case $(1 - p_{D,k})\nu_{k|k-1}$ and $|Z_k|$ detection

cases $\nu_{D,k}(;z)$ for every $z \in Z_k$. The GM-PHD recursion prediction step thus follows Kalman prediction step and the GM-PHD update step follows Kalman update step.

By summing the weights of $\nu_{k|k-1}$ and ν_k we can get the expected number of targets $\tilde{N}_{k|k-1}$ and $\tilde{N}_k$, respectively,

$$\tilde{N}_{k|k-1} = \tilde{N}_{k-1}\left(p_{S,k} + \sum_{j=1}^{J_{\beta,k}} w_{\beta,k}^{(j)}\right) + \sum_{j=1}^{J_{\gamma,k}} w_{\gamma,k}^{(j)} \tag{3.60}$$

$$\tilde{N}_k = \tilde{N}_{k|k-1}(1 - p_{D,k}) + \sum_{z \in Z_k} \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z), \tag{3.61}$$

where $\tilde{N}_{k|k-1}$ is the expected number of targets after the prediction step, which is the sum of surviving , spawning and born targets. Similary, $\tilde{N}_k$ is the expected number of targets after the update step, which is the sum of the updated number of targets plus the mean number of targets that are not detected.

The GM-PHD filter pseudo-algorithm is provided in the Algorithm 2.

3.4.2.4 Pruning and merging in GM-PHD filter

Due to the propagation of the Gaussian mixture in time, the PHD filter suffers from computation issues resulting from exponential increase of Gaussian components

$$(J_{k-1}(1 + J_{\beta,k}) + J_{\gamma,k})(1 + |Z_k|) = O(J_{k-1}|Z_k|), \tag{3.62}$$

where J_{k-1} is the number of Gaussian components of the intensity ν_{k-1}.

To resolve this issue, many techniques can be applied. The simplest one is to remove components with a low weight to the next timestep. This can be done by predefining a threshold. Only targets with a higher weight than this threshold continue to survive.

The next possibility is keeping a certain number of targets and removing the ones with the lowest weight. However, some of the Gausssian components can appear so close together, that they can be merged into a single component. Such procedure is a good approximation with appropriate merging threshold. Pseudo-algorithm for this procedure is shown in the Algorithm 3.

After computing the posterior intensity ν_k, the subsequent objective is to derive estimates for multi-target states. In the context of the Gaussian mixture representation of ν_k, extracting estimates for the multi-target states becomes relatively straightforward, as the means of the individual Gaussian components typically correspond to the local maxima of ν_k, given that they are adequately separated. However, it is essential to note that closely spaced Gaussian components may have been merged after pruning.

Since the magnitude of each peak is influenced by both its weight and covariance, merely selecting the $\tilde{N}_k$ highest peaks of ν_k may yield to state estimates associated with Gaussians to have weak weights. This scenario is suboptimal, as the expected number of targets attributed to these peaks might be small despite their peak magnitudes being substantial. A more effective approach involves choosing the means of the Gaussians with weights exceeding a certain threshold, such as 0.5. This method ensures a more robust selection of state estimates.

The outlined procedure for the state estimation in the Gaussian mixture PHD filter is summarized in the Algorithm 4.

Algorithm 2 Pseudo-algorithm for the GM-PHD filter

Require: $\{w_{k-1}^{(i)}, m_{k-1}^{(i)}, P_{k-1}^{(i)}\}_{i=1}^{J_{k-1}}$, and the measurement set Z_k.

procedure PREDICTION STEP FOR BIRTH TARGETS:
  for $j = 1, \ldots, J_{\gamma,k}$ **do**
    $i := i + 1$,
    $w_{k|k-1}^{(i)} = w_{\gamma,k}^{(i)}, \quad m_{k|k-1}^{(i)} = m_{\gamma,k}^{(i)}, \quad P_{k|k-1}^{(i)} = P_{\gamma,k}^{(i)}$,
  end for
  for $j = 1, \ldots, J_{\beta,k}$ **do**
    for $l = 1, \ldots, J_{k-1}$ **do**
      $i := i + 1$,
      $w_{k|k-1}^{(i)} = w_{k-1}^{(l)} w_{\beta,k}^{(j)}$,
      $m_{k|k-1}^{(i)} = d_{\beta,k-1}^{(j)} + F_{\beta,k-1}^{(j)} m_{k-1}^{(l)}$,
      $P_{k|k-1}^{(i)} = Q_{\beta|k-1}^{(j)} + F_{\beta,k-1}^{(j)} P_{k-1}^{(l)} (F_{\beta,k-1}^{(j)})^T$.
    end for
  end for
end procedure

procedure PREDICTION STEP FOR EXISTING TARGETS:
  for $j = 1, \ldots, J_{k-1}$ **do**
    $i := i + 1$,
    $w_{k|k-1}^{(i)} = p_{S,k} w_{k-1}^{(j)}$,
    $m_{k|k-1}^{(i)} = F_{k-1} m_{k-1}^{(j)}$,
    $P_{k|k-1}^{(i)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^T$,
  end for
  $J_{k|k-1} = i$
end procedure

procedure COMPUTATION OF PHD UPDATE COMPONENTS:
  for $j = 1, \ldots, J_{k|k-1}$ **do**
    $\eta_{k|k-1}^{(j)} = H_k m_{k|k-1}^{(j)}, \quad S_k^{(j)} = R_k + H_k P_{k|k-1}^{(j)} H_k^T$,
    $K_k^{(j)} = P_{k|k-1}^{(j)} H_k^T [S_k^{(j)}]^{-1}, \quad P_{k|k}^{(j)} = [I - K_k^{(j)} H_k] P_{k|k-1}^{(j)}$.
  end for
end procedure

procedure PHD UPDATE STEP:(Z_k)
  for $j = 1, \ldots, J_{k|k-1}$ **do** ▷ Misdetection
    $w_k^{(j)} = (1 - p_{D,k}) w_{k|k-1}^{(j)}, \quad m_k^{(j)} = m_{k|k-1}^{(j)}, \quad P_k^{(j)} = P_{k|k-1}^{(j)}$
  end for
  $l := 0$
  for all $z \in Z_k$ **do** ▷ Detection
    $l := l + 1$,
    for $j = 1, \ldots, J_{k|k-1}$ **do**
      $w_k^{(lJ_{k|k-1}+j)} = p_{D,k} w_{k|k-1}^{(j)} \mathcal{N}(z; \eta_{k|k-1}^{(j)}, S_k^{(j)})$,
      $m_k^{(lJ_{k|k-1}+j)} = m_{k|k-1}^{(j)} + K_k^{(j)} (z - \eta_{k|k-1}^{(j)})$,
      $P_k^{(lJ_{k|k-1}+j)} = P_{k|k}^{(j)}$,
    end for
    $w_k^{(lJ_{k|k-1}+j)} := \frac{w_k^{(lJ_{k|k-1}+j)}}{\kappa_k(z) + \sum_{i=1}^{J_{k|k-1}} w_k^{(lJ_{k|k-1}+i)}}$, for $j = 1, \ldots, J_{k|k-1}$,
  end for
  $J_k = lJ_{k|k-1} + J_{k|k-1}$.
end procedure

Output: $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\}_{i=1}^{J_k}$.

Algorithm 3 Pseudo-algorithm for pruning in the GM-PHD filter

Require: $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\}_{i=1}^{J_k}$, a truncation threshold T, a merging threshold U and a maximum number of allowed Gaussian terms J_{max}. Set $l = 0$, and $I = \{i = 1, \ldots, J_k | w_k^{(i)} > T\}$.

procedure MERGING OF TARGETS
  while $I \neq \emptyset$ **do**
    $l := l + 1$
    $j := \text{argmax}_{i \in I} w_k(i)$,
    $L := \left\{ i \in I | (m_k^{(i)} - m_k^{(j)})^T (P_k(i))^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U \right\}$,
    $\tilde{w}_k^{(l)} = \sum_{i \in L} w_k^{(i)}$,
    $\tilde{m}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} w_k(i) x_k^{(i)}$,
    $\tilde{P}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} w_k^{(i)} (P_k^{(i)} + (m_k^{(l)} - m_k^{(i)})(m_k^{(l)} - m_k^{(i)})^T$,
    $I := I/L$.
  end while
end procedure

If $l > J_{max}$ then replace $\{\tilde{w}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)}\}_{i=1}^{l}$ by those of the J_{max} Gaussians with largest weights.

Output: $\{\tilde{w}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)}\}_{i=1}^{l}$ as pruned Gaussian components.

Algorithm 4 Pseudo-algorithm for state extraction in the GM-PHD filter

Require: $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}\}_{i=1}^{J_k}$.

Set $\hat{X}_k = \emptyset$
for $i = 1, \ldots, j_k$ **do**
  if $w_k^{(i)} > 0.5$, **then**
    for $j = 1, \ldots, round(w_k^{(i)})$ **do**
      update $\hat{X}_k := [\hat{X}_k, m_k^{(i)}]$
    end for
  end if
end for
Output: $\hat{X}_k$ as the multi-target state estimate.

Chapter 4

Object detection and segmentation

In this thesis we focus on tracking objects in video data using target tracking algorithms with dynamic detection probability due to the existence of obstacles that cause sensors to not detect objects in certain situations. But first, we need to have a sensor to detect objects in a frame.

4.1 Object detection

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within images or video frames, respectively. Unlike image classification, which assigns a single label to an entire image, object detection algorithms aim to detect multiple objects of various classes and localize them with bounding boxes. Alongside image classification and object detections, there are other tasks, such as semantic segmentation and instance segmentation. Figure 4.1 shows differences between these types of image processing tasks.

The development of object detection algorithms has evolved significantly over the years, driven by advances in deep learning, dataset availability and computational power. Traditional object detection methods relied on handcrafted features and machine learning algorithms, such as sliding window-based classifiers, histogram of oriented gradients (HOG) [30], and Haar cascades [31]. While effective in certain scenarios, these methods often lacked robustness and scalability, particularly in complex and cluttered scenes.

With the advent of deep learning, convolutional neural networks (CNNs) improved the field of object detection. CNNs are capable of automatically learning hierarchical representations of data, making them well suited for image analysis tasks. The rise of CNN-based approaches has led to significant improvements in object detection accuracy and efficiency. At its core, a convolutional neural network is comprised of multiple layers, each designed to perform specific operations on input data, typically images. The fundamental layers in a CNN include convolutional layers, pooling layers, and fully connected layers.

1. **Convolutional layers:** These layers are responsible for extracting features from the input data. They consist of filters (also called kernels) that slide across the input image, performing a mathematical operation known as convolution. Each filter

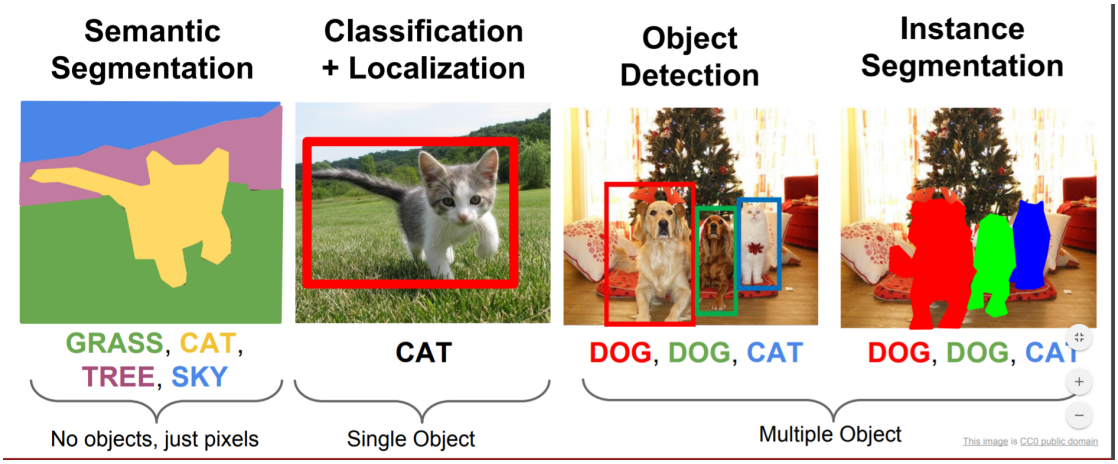


Figure 4.1 Image detection and segmentation task types. (Source: techvidvan.com.)

detects certain patterns or features, such as edges, textures, or shapes. By convolving the filters with the input image, the CNN can capture hierarchical representations of features, starting from simple edges and gradients to more complex structures.

2. **Pooling layers:** Pooling layers are interspersed between convolutional layers to reduce the spatial dimensions of the feature maps while retaining important information. Common pooling operations include max pooling and average pooling, which downsample the feature maps by taking the maximum or average value within each pooling region. This downsampling helps in reducing computational complexity and controlling overfitting by enforcing spatial invariance.

3. **Fully connected layers:** These layers are typically placed at the end of the CNN and serve to classify the features extracted by the convolutional layers. Each neuron in a fully connected layer is connected to every activation in the previous layer, forming a dense network. These layers use techniques like softmax activation to produce probability distributions over the classes in a classification task or regression outputs in a regression task.

It is important to note, that it is common to combine more convolutional, pooling and fully connected layers with differently sized kernels in the architecture to improve the performance. Example of such architecture is shown in Figure 4.2 During training, CNNs use a process called backpropagation to adjust the parameters (weights and biases) of the network based on the disparity between the predicted outputs and the ground truth labels. This optimization process aims to minimize a predefined loss function, such as cross-entropy loss for classification tasks or mean squared error for regression tasks. By iteratively updating the parameters using optimization algorithms like stochastic gradient descent (SGD) or its variants, CNNs gradually learn to recognize and classify patterns within the input data, ultimately improving their performance on various visual tasks such as object detection and segmentation.

Object detection poses several challenges, including variations in object appearance, scale, orientation, occlusion, and cluttered backgrounds. Additionally, real-world images

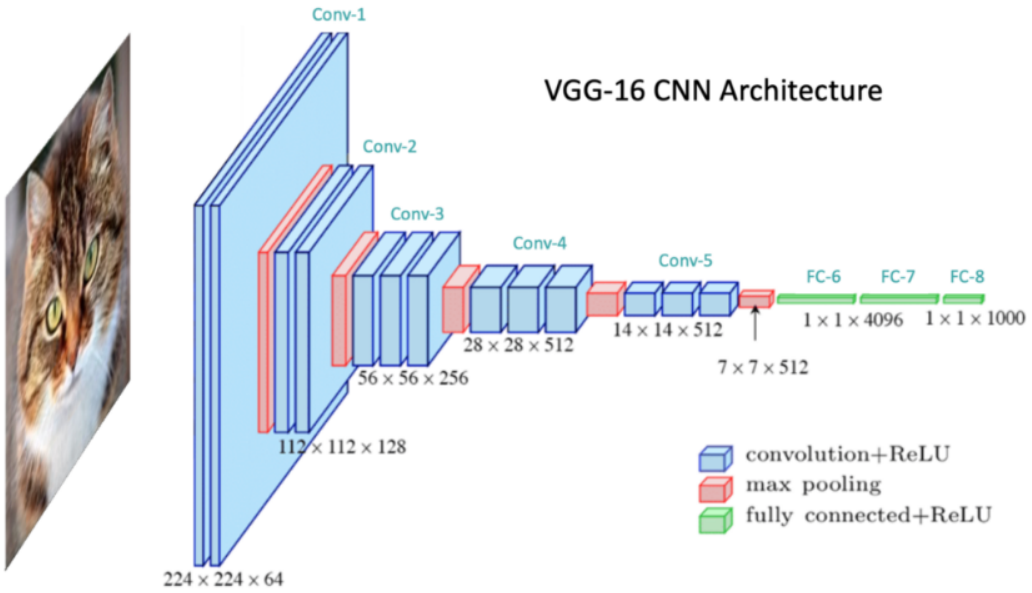


Figure 4.2 CNN architecture example. (Source learnopencv.com)

often contain multiple objects of different classes, making it essential for detection algorithms to handle overlapping and partially visible objects.

Furthermore, object detection systems often have to balance accuracy and speed to meet the demands of real-time applications. Achieving high detection accuracy while maintaining fast inference times is a fundamental challenge, especially for devices with low hardware resources and applications requiring low-latency processing.

Object detection algorithms typically consist of several components:

- **Input processing:** Images or video frames are preprocessed to standardize their format and size, often involving resizing, normalization, and data augmentation to enhance model generalization.
- **Feature extraction:** Feature extraction is performed to capture relevant information from the input data. In deep learning-based approaches, convolutional neural networks are commonly used to extract hierarchical features that encode object appearance and spatial relationships.
- **Localization:** Localization involves predicting the spatial extent of objects within the image using bounding boxes. This step requires regression or classification to estimate bounding box coordinates and confidence scores for object presence.
- **Classification:** Object classification assigns class labels to detected objects based on their visual appearance. Classification models are trained to distinguish between different object categories, enabling accurate identification of objects within the scene.
- **Post-processing:** Post-processing techniques, such as non-maximum suppression

(NMS), are applied to refine detection results, suppress duplicate detections, and improve localization accuracy.

4.1.1 YOLO

In the realm of object detection, the emergence of You Only Look Once (YOLO) represents a paradigm shift, propelled by the fusion of deep learning and innovative architectural design. YOLO stands as a testament to the transformative power of convolutional neural networks (CNNs) in redefining the landscape of computer vision applications, particularly in real-time object detection scenarios. This object detector was proposed by Developed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in [32].

At its core, YOLO introduces a groundbreaking concept: the ability to predict bounding boxes and class probabilities for objects within an image in a single pass through the neural network. This departure from traditional object detection methods, which often involve multi-stage processes and post-processing steps, marks a significant leap forward in efficiency and speed. By consolidating object localization and classification into a unified framework, YOLO streamlines the detection pipeline, enabling seamless integration into various applications requiring rapid decision-making based on visual data.

A distinguishing feature of YOLO lies in its holistic approach to image analysis. Unlike conventional methods that segment images into regions of interest for further processing, YOLO takes a global perspective by dividing the input image into a grid and processing it as a whole. This global context consideration not only enhances the model's understanding of spatial relationships but also reduces the risk of misclassification, particularly in complex scenes with multiple objects. Each bounding box prediction includes coordinates (x, y) for the center of the box, width, height, and confidence score indicating the likelihood of containing an object. Additionally, class probabilities are estimated for each bounding box to determine the object category. YOLO employs a loss function that penalizes localization errors, confidence errors, and classification errors. This loss function is optimized during training using labeled datasets to learn accurate object representations. Moreover, YOLO adopts anchor boxes to improve localization accuracy and handle scale variations of objects within the image.

The evolution of YOLO over successive versions underscores its adaptability to diverse use cases and computational environments. While early iterations prioritized speed, subsequent iterations like YOLOv4 and YOLOv5 have aimed to enhance accuracy without compromising real-time performance [33]. This flexibility in model selection empowers users to tailor YOLO to their specific requirements, whether it be for surveillance systems demanding rapid detection or autonomous vehicles requiring precise object localization.

Compared to traditional object detection methodologies such as region-based convolutional neural networks (R-CNN) [34] and single-shot detectors (SSD), YOLO stands out for its simplicity and efficiency. By eliminating the need for complex post-processing steps and leveraging a unified architecture for object detection (see Figure 4.3 for YOLO architecture), YOLO achieves a fine balance between speed and accuracy. This versatility

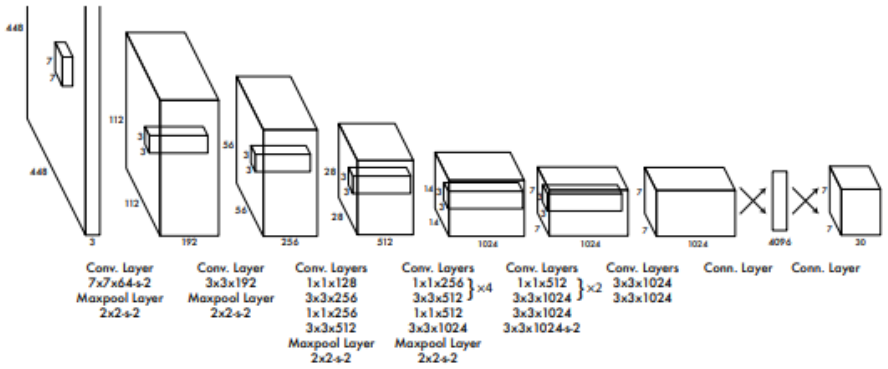


Figure 4.3 Yolo architecture proposed in [32] has 24 convolutinal layers followed by 2 fully connected layers. Convolutional layers were pretrained on the ImageNet classificator at half resolution (224 x 224 input image) and the doubled the resolution for detection. (Source [32].)

extends YOLO's applicability across a myriad of domains, including but not limited to autonomous driving, industrial automation, healthcare imaging, and augmented reality.

4.2 Image segmentation

Image segmentation, a fundamental technique in computer vision, involves dividing a digital image into distinct pixel groups called segments. By breaking down the visual data into manageable segments, image segmentation facilitates more efficient and sophisticated image processing.

The methods employed for image segmentation vary from straightforward heuristic approaches to cutting-edge deep learning techniques. Traditional algorithms typically analyze basic visual attributes such as color and brightness to delineate object boundaries and background regions. Machine learning plays a pivotal role in this domain by training models on labeled datasets, allowing them to accurately classify objects and regions within images.

With its versatility and practical applications, image segmentation finds widespread use across artificial intelligence scenarios. Its applications span from assisting medical diagnoses through imaging to enabling automation in robotics and self-driving vehicles. Furthermore, it plays a crucial role in tasks such as identifying objects within satellite imagery.

There are three main task types in image segmentation: semantic segmentation, instance segmentation and panoptic segmentation [35]. The distinction between various types of image segmentation tasks hinges on how they handle semantic classes, which are specific categories assigned to individual pixels within an image.

In the realm of computer vision, semantic classes are broadly categorized into two types, each requiring distinct segmentation techniques for accurate results.

- *Things* represent classes of objects characterized by identifiable shapes, such as *car, tree,* or *person*. These classes typically consist of discrete instances with consistent

sizes and distinguishable constituent parts. For instance, all cars have wheels, which are distinct from the car itself.

- *Stuff* refers to semantic classes with amorphous shapes and variable sizes, like sky, water, or grass . Unlike *things*, *stuff* lacks clearly defined, countable individual instances and does not possess distinct parts. For example, both a single blade of grass and an entire field of grass fall under the category of *grass*.

In certain image contexts, some classes can straddle the line between being considered *things* or *stuff*. For instance, a large gathering of people might be interpreted either as multiple people – each being a clearly shaped, countable thing, or as a singular, indistinct person.

While much of the focus in object detection typically centers on *thing* classes, it is crucial to recognize that *stuff*, such as sky, walls, floors, and ground constitutes the bulk of our visual environment. *Stuff* serves as crucial contextual information for identifying *things*. For instance, a metallic object on the road is likely a car, while a blue background behind an object is indicative of water if it is a boat or sky if it is a plane. This interplay between *stuff* and *things* holds particular significance for deep learning models.

4.2.1 Semantic segmentation

Semantic segmentation represents the most straightforward form of image segmentation. In this approach, a semantic segmentation model assigns a semantic class to every pixel in an image, without providing additional context or information, such as the identification of specific objects.

Unlike more nuanced segmentation techniques, semantic segmentation treats all pixels uniformly as *stuff* without distinguishing between *stuff* and *things*. This means that it does not differentiate between different types of objects within the same class.

For instance, imagine a semantic segmentation model trained to analyze urban scenes. It would generate segmentation masks outlining the boundaries of various classes of *things* and *stuff*. However, it would not differentiate between individual instances of the same class. For example, if there are multiple trees in a forest, the model would likely treat them as one continuous tree segment rather than recognizing each individual tree separately.

4.2.2 Instance segmentation

Instance segmentation represents a departure from the approach of semantic segmentation. While semantic segmentation focuses solely on assigning semantic classes to each pixel without distinguishing between individual instances, instance segmentation precisely outlines the shape of each separate object instance within an image.

In essence, instance segmentation segregates *things* from *stuff*, disregarding the latter, and can be viewed as a more sophisticated version of object detection. Instead of providing rough bounding boxes, instance segmentation furnishes precise segmentation masks for each object instance.

This task poses greater challenges compared to semantic segmentation. Even when instances of the same class overlap or touch each other, instance segmentation models

must accurately separate and delineate the shape of each instance. In contrast, semantic segmentation models may simply group them together.

Instance segmentation algorithms typically adopt either a two-stage or one-shot methodology to address the task.

Two-stage models, exemplified by Region-based Convolutional Neural Networks [36], initially conduct conventional object detection to produce bounding boxes for each potential instance. Subsequently, they refine segmentation and classification within these bounding boxes to precisely delineate each object instance.

Conversely, one-shot models, such as YOLO, streamline the process by simultaneously performing object detection, classification, and segmentation. This approach enables real-time instance segmentation.

While one-shot methods boast faster processing speeds, they often sacrifice some accuracy. On the other hand, two-stage approaches prioritize accuracy but may be slower due to the additional refinement steps.

4.2.3 Panoptic segmentation

Panoptic segmentation represents a synthesis of semantic and instance segmentation methodologies, offering a comprehensive understanding of an image.

In panoptic segmentation, each pixel receives both a semantic label and an instance ID. Pixels with the same label and ID are assigned to the same object instance. For *stuff* pixels, the instance ID is disregarded.

This approach provides computer vision systems with a complete and cohesive interpretation of an image, combining the benefits of both semantic and instance segmentation. However, achieving panoptic segmentation consistently and efficiently presents significant computational challenges. Despite its evident appeal, realizing panoptic segmentation in a manner that balances accuracy and computational efficiency remains a formidable task.

The challenge in achieving panoptic segmentation lies in reconciling the conflicting approaches of semantic and instance segmentation models. Semantic segmentation treats all pixels uniformly as *stuff*, disregarding individual instances of objects, while instance segmentation focuses solely on isolating individual objects, ignoring *stuff*. Integrating these methodologies proves difficult as neither model can effectively take on the responsibilities of the other.

Early attempts at panoptic segmentation involved combining separate semantic and instance segmentation models, followed by a post-processing phase to merge their outputs. However, this approach encountered two significant drawbacks: it demanded substantial computational resources and struggled with inconsistencies between the data produced by the semantic segmentation network and that produced by the instance segmentation network.

Recent advancements in panoptic segmentation architectures aim to overcome these limitations through a more unified deep learning approach. Many of these architectures utilize a common backbone network, such as a feature pyramid network (FPN) [37], to extract features from the input image. These features are then fed into parallel branches, such as a foreground branch and a background branch, or a semantic head and an instance

head. The outputs from each branch are merged using a weighted system to produce the final segmentation. Notable proposed architectures include EfficientPS [38], OANet [39], PanopticFPN [40], UPSNet [41], SOGNet [42], BGRNet [43], AUNet [44] and others.

4.2.4 Traditional image segmentation

Traditional image segmentation techniques leverage pixel color values and related characteristics, such as brightness, contrast, or intensity, for feature extraction. These methods are often trained with simple machine learning algorithms and are particularly useful for tasks like semantic classification. Despite their limitations in precision compared to deep learning-based approaches, traditional methods offer advantages in terms of lower cost and computational demands, allowing them to efficiently solve certain problems.

Some common traditional image segmentation techniques include:

1. **Thresholding:** Thresholding methods create binary images by classifying pixels based on whether their intensity exceeds or falls below a predetermined threshold value. Otsu's method is frequently used to identify the threshold value that minimizes intra-class variation [45].

2. **Histograms:** Histograms plot the frequency of specific pixel values in an image and are often employed to define thresholds. For instance, histograms can infer background pixel values, aiding in the isolation of object pixels.

3. **Edge detection:** Edge detection techniques identify object boundaries or classes by detecting abrupt changes in brightness or contrast [46].

4. **Watersheds:** Watershed algorithms [47] convert images to grayscale and generate a topographic map where each pixel's elevation is determined by its brightness. Regions, boundaries, and objects can be inferred from the formation of valleys, ridges, and catchment basins.

5. **Region-based segmentation:** Region-growing algorithms [48] initiate segmentation with one or more seed pixels, progressively grouping neighboring pixels with similar characteristics. These algorithms can be either agglomerative or divisive.

6. **Clustering-based segmentation:** Clustering algorithms [49], an unsupervised learning technique, partition visual data into clusters of pixels with similar values. One popular variant is K-means clustering, where k represents the number of clusters. In this method, pixel values are treated as data points, and k random points are selected as the centroids of clusters. Each pixel is then assigned to the nearest centroid based on similarity. Centroids are iteratively relocated to the mean of each cluster until convergence, resulting in stabilized clusters.

4.2.5 Deep learning image segmentation

Trained on meticulously annotated datasets, deep learning image segmentation models harness the power of neural networks to uncover underlying patterns within visual data.

These models discern salient features crucial for tasks such as classification, detection, and segmentation.

Despite their higher computational requirements and longer training times, deep learning models consistently outperform traditional approaches, serving as the cornerstone for ongoing advancements in computer vision.

Some prominent deep learning models utilized in image segmentation include:

1. **Fully Convolutional Networks:** FCNs [50], commonly employed for semantic segmentation, are a variant of convolutional neural networks characterized by flexible layers. In FCNs, an encoder network processes visual input through convolutional layers to extract pertinent features for segmentation or classification. The compressed feature data is then passed through decoder layers to upsample and reconstruct the input image along with segmentation masks.

2. **U-Nets:** U-Nets [51] adapt the FCN architecture to mitigate data loss during downsampling by incorporating skip connections. These connections enable the preservation of finer details by selectively bypassing certain convolutional layers as information propagates through the network. The name U-Net derives from the shape of diagrams illustrating its layered arrangement (see Figure 4.4).

3. **Deeplab:** Similar to U-Nets, Deeplab modifies the FCN architecture [52]. In addition to utilizing skip connections, Deeplab employs dilated (or "atrous") convolution to produce larger output maps without requiring additional computational resources.

4. **Mask R-CNNs:** Mask R-CNNs stand out as a leading model for instance segmentation [34]. These models integrate a region proposal network (RPN), responsible for generating bounding boxes for potential instances, with an FCN-based "mask head" that produces segmentation masks within each confirmed bounding box.

5. **Transformers:** Inspired by the success of transformer models [53] in natural language processing, newer models like the Vision Transformer (ViT) employ attention mechanisms in lieu of convolutional layers. These models have demonstrated comparable or superior performance to CNNs for various computer vision tasks.

4.2.5.1 Transformers

As we use Transformer model in our work, we briefly demonstrate Transformers' architecture and how they work. Transformers consist of a stack of identical layers, typically comprising an encoder and a decoder (Figure 4.5). The encoder processes the input sequence, while the decoder generates the output sequence.

- **Encoder:** The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism (Figure 4.6), and the second is positionwise fully connected feed-forward network. There is a residual connection [54] around each of two sub-layers, followed by normalization [55]: the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer itself.

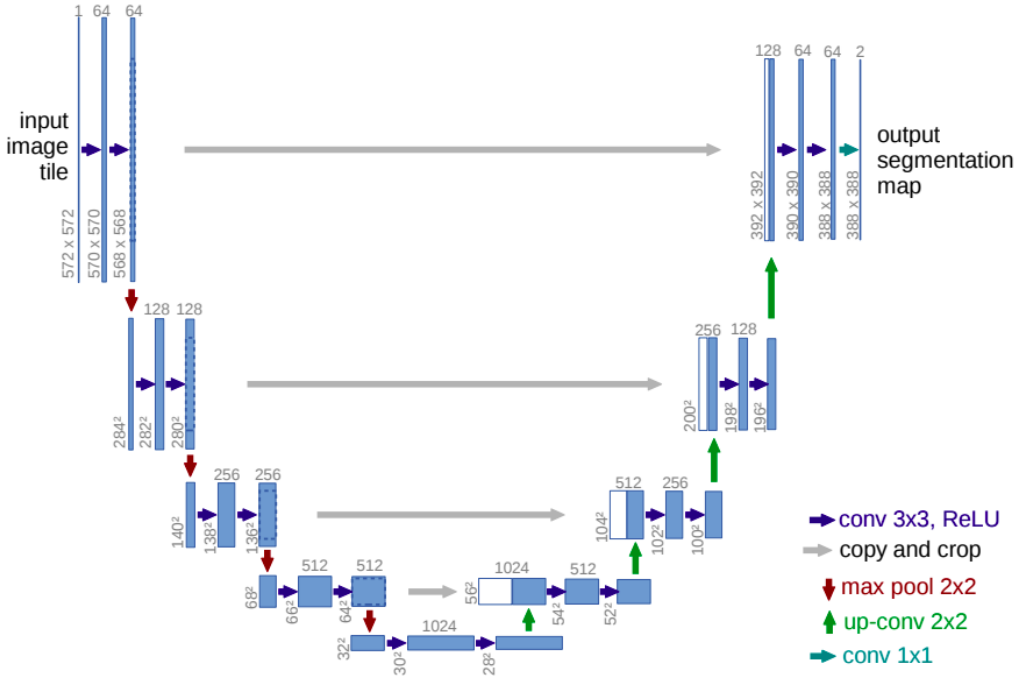


Figure 4.4 U-net architecture (example for 32x32 pixels in the lowest resolution). (Source [51])

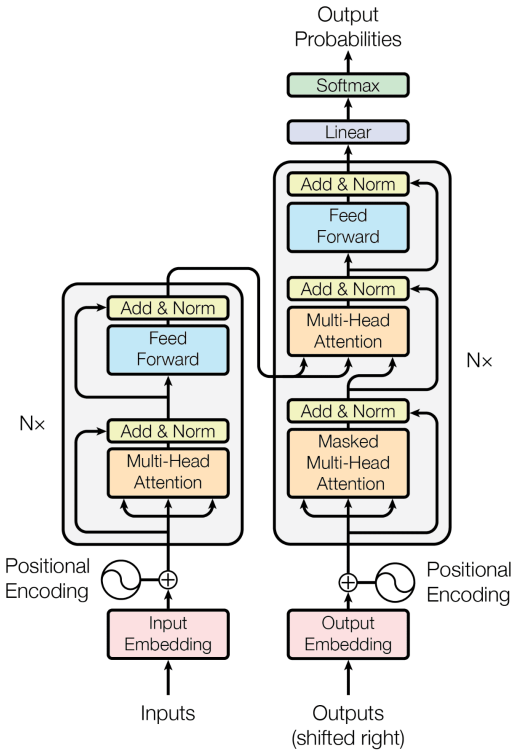


Figure 4.5 Transformer architecture. (Source [53])

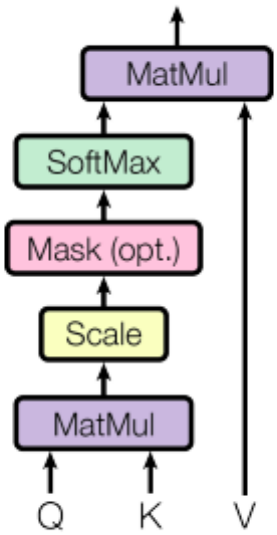


Figure 4.6 Multi-head attention. (Source [53])

- **Decoder:** The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, there is residual connections around each of the sub-layers, followed by layer normalization.

The core of the Transformer architecture lies in its self-attention mechanism [53]. This mechanism enables each token in the input sequence to attend to every other token, learning contextual relationships and dependencies. By computing attention scores between all pairs of tokens and taking weighted sums of their embeddings, the Transformer can effectively capture long-range dependencies and dependencies between distant tokens.

To enhance the expressiveness of self-attention, Transformers employ multi-head attention mechanisms. Instead of computing attention once, multi-head attention computes attention multiple times in parallel, each with its set of learnable parameters. This allows the model to attend to different aspects of the input sequence simultaneously, facilitating richer representations and improved performance.

Unlike sequential models, Transformers do not inherently encode the positional information of tokens. To address this limitation, positional encoding is added to the input embeddings to convey the position of each token in the sequence. This positional information is crucial for the Transformer to distinguish between tokens with the same content but different positions within the sequence.

In addition to self-attention layers, Transformers incorporate feedforward neural networks within each layer. These FNNs consist of multiple fully connected layers with non-linear activation functions, enabling the model to capture complex interactions and

transformations within the input data

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2, \tag{4.1}$$

where W are weights and b stands for bias.

To stabilize training and facilitate gradient flow, Transformers employ layer normalization and residual connections within each layer. Layer normalization normalizes the activations across feature dimensions, while residual connections enable the direct flow of gradients through the network, mitigating the vanishing gradient problem.

During training, Transformers are optimized using backpropagation and gradient-based optimization algorithms, such as Adam [56] or SGD [57], to minimize a predefined loss function. By iteratively updating the parameters of the model based on the disparity between the predicted outputs and the ground truth labels, Transformers learn to encode and generate meaningful representations of input sequences, enabling them to excel in various NLP tasks and more recently image processing tasks.

4.2.6 Segment Anything

To get the best possible results in practical part of this thesis, it is necessary to use powerful, yet fast image segmentation model. The Segment Anything (SAM) model represents a significant advancement in image segmentation, introducing a novel approach to tackling the challenges in this field [58]. SAM is designed to address the need for efficient and effective segmentation models that can generalize to diverse tasks and data distributions, even in zero-shot scenarios.

The SAM project introduces a new task, model, and dataset for image segmentation. It leverages a promptable model architecture to achieve impressive zero-shot performance on diverse segmentation tasks. This task involves generating valid segmentation masks given any segmentation prompt. A prompt can include spatial or textual information indicating what to segment in an image. SAM ensures the output mask is reasonable for at least one interpretation of the prompt, handling ambiguity effectively.

SAM's architecture comprises three main components: an image encoder, a prompt encoder, and a mask decoder (Figure 4.7). The image encoder processes input images using a Vision Transformer, while the prompt encoder handles various types of prompts, including points, boxes, and text. The mask decoder efficiently generates segmentation masks based on the input image and prompt embeddings. SAM is designed for real-time performance, enabling interactive use with low latency.

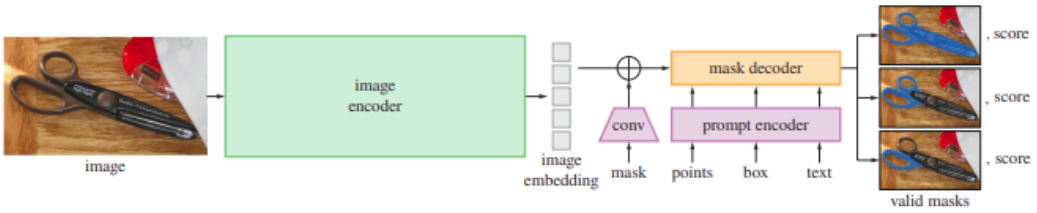


Figure 4.7 SAM architecture (Source [58])

SAM addresses ambiguity by predicting multiple output masks for a single prompt. It efficiently ranks these masks based on confidence scores, allowing it to produce accurate segmentations even in complex scenarios.

Efficiency is a core consideration in SAM's design. The model's components are optimized for fast execution, enabling it to run in a web browser on CPU with a runtime of approximately 50ms. This real-time performance facilitates seamless interaction with the model for prompt-based segmentation tasks.

SAM is trained using the linear combination of focal loss [59] and dice loss [60], tailored for the promptable segmentation task. It simulates an interactive setup during training, sampling prompts to ensure robustness and adaptability.

SAM's contributions include the development of a promptable segmentation task, a novel model architecture optimized for real-time performance, and a large-scale dataset (SA-1B) comprising over 1 billion masks and 11 million images. The model's zero-shot capabilities and efficient design make it a valuable tool for various segmentation tasks, with potential applications across diverse domains. The example of segmented images are shown in Figure 4.8.

Figure 4.8 The demonstration of SAM's strength. These masks were annotated fully automatically by SAM and are part of the SA-1B dataset. (Source [58])

4.2.7 Grounded Segment Anything

Even though the SAM model with text prompt was not released yet, many other projects occur recently and combines other models with SAM. One of such project is Grounded Segment Anything [61]. This project connects image processing models Grounding DINO [62] as an open-set object detector and SAM [58] as an object segmentation model. Moreover, to enhance to performance and usability of Grounded SAM, other powerful models are used.

- **BLIP:** BLIP is Vision-Language Pre-training (VLP) framework which transfers flexibly to both vision -language understanding and generation tasks [63]. VLP frameworks aim to improve performance of downstream vision and language tasks by pretraining the model on large-scale image-text pairs.
- **Recognize Anything:** Recognize Anything Model (RAM) is a string foundation model for image tagging [64]. RAM makes a substantial step for large models in computer vision, demonstrating the zero-shot ability to recognize any common category with high accuracy.
- **Stable-Diffusion:** Stable-Diffusion is latent text-to-image diffusion model (LDMs)[65]. LDMs enhance the efficiency and flexibility of diffusion models (DMs) by training them in the latent space of pretrained autoencoders. By leveraging powerful denoising autoencoders, LDMs achieve state-of-the-art results in tasks such as image inpainting, class-conditional image synthesis, and super-resolution while significantly reducing computational requirements.
- **OSX:** OSX is a one-stage pipeline for expressive whole-body mesh recovery from single images [66]. OSX employs a Component Aware Transformer (CAT) consisting of a global body encoder and a local face/hand decoder. By leveraging a feature-level upsample-crop scheme and keypoint-guided deformable attention, OSX achieves precise estimation of face and hand parameters while maintaining connection coherence.

Grounded DINO also offers cooperation of SAM and other models: Visual ChatGPT [67], RAM++ [68], VoxelNeXt [69] and more.

Chapter 5

Dynamic time and state varying detection probability

As stated in the Introduction, this work focuses on tracking objects in frame sequence using target tracking methods and object detection and segmentation algorithms. Object detectors, which were discussed in the previous section, are essential for getting measurements from an image.

Note, that in all target tracking algorithms for cluttered environments and targets birth and dead possibilities, there is a parameter called detection probability (p_D). As its name suggests, this parameter expresses a probability that the target is detected in particular time. It is also commonly used as a time and state independent scalar. The desired outcome is to enhance the performance of multi-target algorithms in video data in scenarios, where the targets may be hidden by an obstacle, and reduce impacts of a misdetection of a sensor. Therefore, it is appropriate to have a dynamic time- and state-varying detection probability.

5.1 Problem definition

In the practical part of this thesis, three different settings of object detection and segmentation can be found. We choose Probabilist Hypothesis Density filter as a multi-target tracking algorithm, as it is on the simpler side of RFS-based methods. Furthermore, it is computationally very effective in comparison with filters such as the CPHD or the PMBM filter, which may seem as a more appropriate option for this task. However, these more complicated filters have high computational demands, which are not appropriate when used in video streams. Provided settings are:

1. **S1: YOLO + PHD** – Yolo itself can produce an object segmentation mask with lower accuracy, but with greater speed. This setting is suitable when using CPU only.
2. **S2: YOLO + SAM + PHD** – As SAM cannot annotate labels to objects, it requires another object detector before segmentation. In this setting, we use YOLO as an object detector and SAM for mask segmentation of detected objects. This procedure requires GPU for faster computation.

3. **S3: Grounded SAM + PHD** – Grounded SAM is the most universal approach, as it can detect objects based on a text input and uses SAM for segmentation. This setting also requires GPU for faster computation.

5.1.1 The modified GM-PHD filter

To modify the GM-PHD filter for our use case, a few assumptions have to be defined. These assumptions are very similar to the assumptions of the PHD filter and the GM-PHD filter in Section 3.4.2 and 3.4.2.2.

1. Each target evolves and generates observations independently of each other.
2. Clutter is Poisson and independent of target-originated measurements.
3. The predicted multi-target RFS governed by $p_{k|k-1}$ is Poisson.
4. Each target follows a linear Gaussian dynamical model and the sensor has a linear Gaussian measurement model, i.e.,

$$f_{k|k-1}(x|\xi) = \mathcal{N}(x; F_{k-1}\xi, Q_{k-1}), \tag{5.1}$$
$$g_k(z|x) = \mathcal{N}(z; H_k x, R_k), \tag{5.2}$$

   where $\mathcal{N}(\cdot; m, P)$ is a Gaussian density with mean m, covariance P, F_{k-1} is the state transition matrix, Q_{k-1} is the process noise covariance, H_k is the observation matrix and R_k is the observation noise covariance.

5. The survival probability is state independent, i.e.,

$$p_{S,k}(x) = p_{S,k}. \tag{5.3}$$

6. The detection probability is state dependent, i.e.,

$$p_{D,k}(x) = \begin{cases} p_{D,k} & \text{if detected for the first time,} \\ p_{D,k}(x) & \text{otherwise.} \end{cases} \tag{5.4}$$

7. The intensity of the birth RFS is Gaussian mixture of the form

$$\gamma_k(x) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(x; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}). \tag{5.5}$$

Note, that the assumption 1 may be ambiguous, as the states of each of the targets are often dependently evolving. Imagine, for example, a common traffic scenario. If one of the cars immediatelly stops, the following vehicles have to stop as well. Meaning, they are affected by another target. The assumption 2 is reasonable, especially if we consider that the clutter rate should be very low in our scenarios. The survival probability is state

independent as in the GM-PHD filter and should be set high, as the targets are expected to survive to the next time step. The detection probability is state dependent and equal to $p_{D,k}$, i.e., predefined threshold, in situation when the target is detected for the first time. Otherwise, if the target evolved from the previous state ξ, the detection probability is calculated according to Section 5.2. Assumption 7 contains only intensity of the birth RFS, the spawn intensity RFS is left out. Nevertheless it still can fluently be added to the assumption and equations afterwards.

5.1.1.1 The modified PHD recursions

Let us assume a target state x described by an intensity function $v(x)$. At time k, the prediction of the prior intensity $v_{k-1}(x)$ is given by

$$v_{k|k-1}(x) = \int p_{S,k}(\xi)\phi_{k|k-1}(x|\xi)v_{k-1}(\xi)d\xi + v_{\gamma,k}(x), \tag{5.6}$$

where $p_{S,k}(\cdot)$ is the probability of target survival, $\phi_{k|k-1}(\cdot|\cdot)$ is the target state transition density, and $v_{\gamma,k}(\cdot)$ denotes the prior PHD of the targets birth at time k. The predicted intensity $v_{k|k-1}$ is then updated by the measurement set Z_k given by sensors at time k according to the Bayesian update

$$\begin{aligned} v_k(x) = {} & [1 - p_{D,k}(x)]v_{k|k-1}(x) \\ & + \sum_{z\in Z_k} \frac{p_{D,k}(x)g_k(z|x)v_{k|k-1}(x)}{\kappa_k(z) + \int p_{D,k}(\xi)g_k(z|\xi)v_{k|k-1}(\xi)d\xi}, \end{aligned} \tag{5.7}$$

where $g_k(\cdot|\cdot)$ is the likelihood function, $p_{D,k}(\cdot)$ is the probability of detection, and $\kappa_k(\cdot)$ is the clutter density.

5.1.1.2 The modified GM-PHD recursion

In the context of the linear Gaussian multiple-target model, the PHD recursion equations (5.6) and (5.7) attain analytical solution [18]. Suppose that the posterior intensity at time $k-1$ is a Gaussian mixture of the form

$$v(x) = \sum_{i=1}^{J_{\gamma,k}} w_{\gamma,k}^{(i)} \mathcal{N}(x; m_{\gamma,k}^{(i)}, P_{\gamma,k}^{(i)}). \tag{5.8}$$

The predicted intensity for time k is also a Gaussian mixture of the form

$$v_{k|k-1}(x) = v_{S,k|k-1}(x) + \gamma_k(x), \tag{5.9}$$

where $\gamma_k(x)$ is given in Formula (5.5). This yields

$$v_{S,k|k-1}(x) = p_{S,k} \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(x; m_{S,k|k-1}^{(j)}, P_{S,k|k-1}^{(j)}), \tag{5.10}$$

$$m_{S,k|k-1}^{(j)} = F_{k-1} m_{k-1}^{(j)}, \tag{5.11}$$

$$P_{S,k|k-1}^{(j)} = Q_{k-1} + F_{k-1} P_{k-1}^{(j)} F_{k-1}^T. \tag{5.12}$$

Thus the predicted intensity at the time k is a Gaussian mixture

$$v_{k|k-1}(x) = \sum_{i=1}^{J_{k|k-1}} w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}), \tag{5.13}$$

and the posterior intensity at time k is also Gaussian mixture,

$$\begin{aligned} v_k(x) = &\sum_{i=1}^{J_{k|k-1}} [1 - p_{D,k}^{(i)}(x)] w_{k|k-1}^{(i)} \mathcal{N}(x; m_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}) \\ &+ \sum_{z \in Z_k} v_{D,k}(x; z), \end{aligned} \tag{5.14}$$

where

$$v_{D,k}(x; z) = \sum_{j=1}^{J_{k|k-1}} w_k^{(j)}(z) \mathcal{N}(x; m_{k|k}^{(j)}(z), P_{k|k}^{(j)}), \tag{5.15}$$

$$w_k^{(j)}(z) = \frac{p_{D,k}^{(j)}(x; z) w_{k|k-1}^{(j)} q_k^{(j)}(z)}{\kappa_k(z) + p_{D,k}^{(j)}(x; z) \sum_{l=1}^{J_{k|k-1}} w_{k|k-1}^{(l)} q_k^{(l)}(z)}, \tag{5.16}$$

$$m_{k|k}^{(j)}(z) = m_{k|k-1}^{(j)} + K_k^{(j)}(z - H_k m_{k|k-1}^{(j)}), \tag{5.17}$$

$$P_{k|k}^{(j)}(z) = [I - K_k^{(j)} H_k] P_{k|k-1}^{(j)}, \tag{5.18}$$

$$\kappa_k^{(j)}(z) = P_k^{(j)} H_k^T (H_k P_{k|k-1}^{(j)} H_k^T + R_k)^{-1}. \tag{5.19}$$

The dynamically estimated detection probability in 5.14 follows from one of two possible scenarios described in Section 5.2.

5.1.2 S1: YOLO + PHD

It is possible to fine tune YOLO model on personalised datasets to either improve the performance in the detection of pretrained classes, or to train the model for detecting other classes. With over 70 pretrained class instances, it is not necessary to fine tune our YOLO model for the experiments in this work. The model also detects all class instances it is trained for, that appear in a scene. For testing purposes, we filter only objects we want to detect and get measurements from. The YOLO itself is able to produce a segmentation mask in real-time, but with less accuracy and precision. This is caused by a reduced image resolution which is 640x640 pixels. To fit the YOLO output to our frame, this output needs to be resized back to our desired frame size. This resizing causes the imperfections. However, the precision to pixel level is not always necessary in our scenarios, thus this setting is, in most cases, sufficient enough. The advantage of this approach is that it runs quickly enough with CPU only. The example of object segmentation YOLO model is shown in the Figure 5.1.

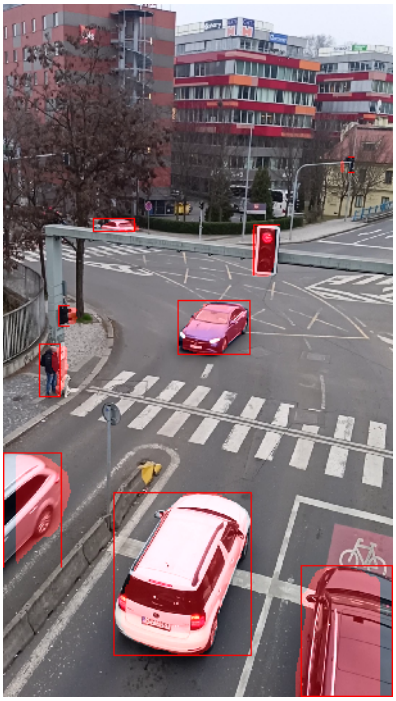


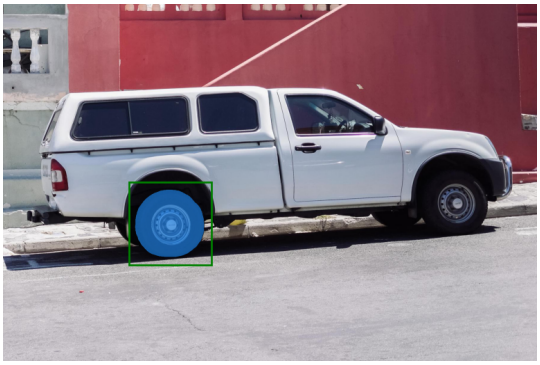
Figure 5.1 YOLO segmentation example. This picture shows all detected objects the YOLO model is trained for and also the segmented objects' masks. These masks are imperfect, but often sufficient enough.

5.1.3 S2: YOLO + SAM + PHD

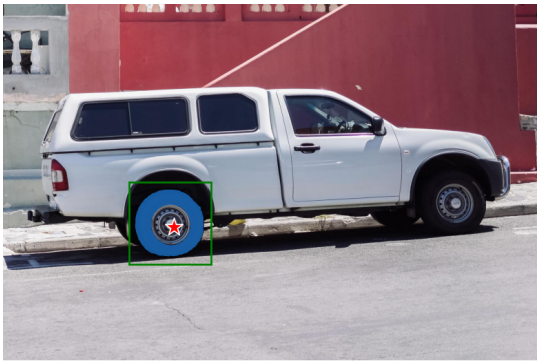
The SAM model architecture in [58] is prepared to first process a text input and then segment objects based on the input. However, this feature is not available yet in the original implementation, and, for our purposes, we need an object detector before the segmentation process can be performed. The SAM model is able to segment a desired object based on one of two provided inputs with high precision and accuracy. The first possible input is a bounding box around the required object, which can be provided by an output of the YOLO model. The second option is to provide a rough center point of an object, which can also be the center of a bounding box provided by the YOLO model.

However, for our purposes, it is not recommended to use the combination of both inputs, as SAM is able to produce multiple masks for an object. For example, if we denote, that we want to segment a whole person by a bounding box, one of the output masks can be just his jacket, as it makes a valid segmentation mask. Combining these two inputs can, therefore, produce this undesirable output. Another example of such behaviour is shown in the Figure 5.2

The combination of the SAM model with an object detector can segment an object with high precision and accuracy. The cooperation of these two models is demonstrated in Figure 5.3.



(a) By defining an object by a bounding box, SAM is able to make a segmented masks of this object and choose the most probable one.



(b) If we use a bounding box together with a point, we receive a different result. Here, just the trucks's tire, instead of the entire wheel, is selected.

Figure 5.2 Comparison of using only a bounding box vs combination of a bounding box and a point as an input for SAM.

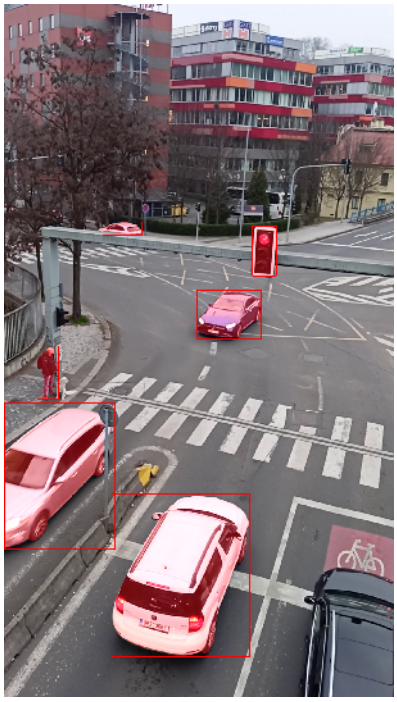


Figure 5.3 The cooperation of YOLO and SAM models. The YOLO provides bounding boxes of objects, which are inputs to SAM. The SAM model then makes segmented masks of these objects.

5.1.4 S3: Grounded SAM + PHD

Grounding DINO is an object detector that denotes objects in an image, based on a given text prompt. This feature allows us to potentially track all the required objects without the need to fine tune every class instance to a model. Moreover, this model is able to mark objects based on an ambiguous text input. Despite this universality, in cases where the model is not confident enough that the desired object appears in a scene, different results can be received with the same input every time the model is executed. This

brings uncertainty in situations with ambiguous text inputs and we should be aware of it in practical scenarios.

For a segmentation task, Grounding DINO's output serves as an input to the SAM model, as it produces bounding boxes of the detected objects. SAM uses these bounding boxes to segment objects and creates segmentation binary masks. The cooperation of these models is demonstrated in Figure 5.4.

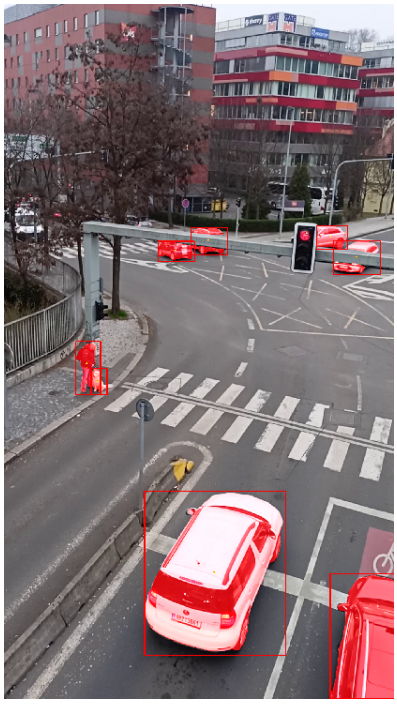


Figure 5.4 The result of the Grounded SAM model. Grounding DINO marks objects with bounding boxes and SAM segments objects inside these bboxes. Marked objects are founded by Grounding DINO with text input *person, car*.

5.2 Dynamic detection probability in video data

To model the dynamic state- and time-dependent detection probability $p_{D,k}(x)$, we propose to base the current point estimate of $p_D(x)$ at time k on the similarity of the subsequent frame properties,

$$p_{D,k}(x) = \frac{\sum_{\sigma_j \in S_{sim}} \sum_{i=1}^{|H|} \sigma_j \left[h_i \left(M(x;k|k-1) \circ D_k \right), h_i \left(M(x;k-1) \circ D_k \right) \right]}{\|S_c\| \cdot \|H\|}, \quad (5.20)$$

where D_k is the frame in the given color spectrum, $h_i(\cdot)$ is a color histogram made from given color spectrum , $A \circ B$ is the Hadamard product, $M(\cdot,\cdot)$ is the object binary mask, and $\sigma_j[\cdot,\cdot]$ is the given similarity of two vectors. The $\|\cdot\|$ values in the denominator represent the set size. The whole fraction is, in fact, the mean value across all color spectra and similarity functions.

There are many color spectra to consider, each providing certain benefits and downsides. The list of color spectra used in this work is as follows.

- **RGB** – The RGB (Red, Green, Blue) color model is ubiquitous in electronic displays, digital cameras, and computer graphics. It defines colors by their intensities of red, green, and blue components. One of its primary advantages lies in its widespread use and an intuitive representation for additive color mixing. However, RGB lacks direct perceptual relevance to human vision, as it does not inherently represent attributes like brightness or hue.
- **XYZ** – In contrast, the XYZ color space, established by the International Commission on Illumination (CIE), serves as a standard for quantifying colors in scientific and industrial applications. Even though it offers a device-independent color representation and facilitates color matching and conversion, it is not as perceptually intuitive and can be complex to work with practically.
- **HSV** – HSV (Hue, Saturation, Value) is favored in graphics software and image editing for its intuitive representation of color. It aligns better with human perception compared to RGB, allowing independent adjustment of hue, saturation, and brightness. It lacks the widespread hardware and software support enjoyed by RGB and may not be as straightforward for certain color manipulation tasks.
- **LAB** – LAB (CIELAB color space), another CIE-defined model. Unlike XYZ, it aims for perceptual uniformity. Widely used in color correction, image editing, and color management, LAB offers uniform changes in perceived color with uniform changes in LAB values. Despite its perceptual accuracy, it can be complex to comprehend and lacks universal software and hardware support compared to simpler models like RGB or HSV.
- **HLS** – HLS (Hue, Lightness, Saturation) finds its place in computer graphics and image editing applications. Similar to HSV but with lightness instead of value, HLS provides an intuitive representation for color adjustment tasks. Unfortunatelly, its support may not be as widespread as RGB or HSV, and it may lack the perceptual accuracy of LAB in certain color correction scenarios.

We apply a range of similarity functions, each with its unique advantages and limitations. These functions include cosine similarity, intersection, and correlation.

- **Cosine similarity** – Cosine similarity, a widely used metric, determines the cosine of the angle between two vectors in a multi-dimensional space. It is insensitive to vector magnitudes, focusing more on their orientation, which proves valuable when the absolute magnitude of vectors is less crucial than their relative orientations. However, cosine similarity does not take into account the distribution of data points. Cosine similarity is defined as

$$S_{cos}[A,B] = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{5.21}$$

  where the sums run over all elements of the arguments.

- **Intersection** – Intersection similarity, on the other hand, calculates the overlap between two sets. This metric is commonly employed in applications like document retrieval and collaborative filtering. Its simplicity and intuitiveness make it ideal for comparing binary data or sets, where the presence or absence of elements is more significant than their values. Additionally, it treats all elements equally, disregarding their magnitudes, which can be a limitation in certain contexts. The intersection is formulated

$$S_{inter}[A,B] = \frac{\sum_{i=1}^{\|A\|} min(A_i, B_i)}{\sum_{i=1}^{\|B\|} B_i}, \tag{5.22}$$

  where $\|\cdot\|$ denotes the set size.

- **Pearson correlation coefficient** – Correlation measures the linear relationship between two variables and is frequently used in statistics, finance, and signal processing. It captures both the strength and direction of the relationship between variables, making it valuable for identifying patterns and dependencies in data. Moreover, correlation assumes a linear relationship between variables, which may not always hold true, and it is susceptible to outliers and non-linear relationships, potentially affecting its reliability in certain scenarios. In this work, we calculate the correlation between corresponding elements in sets A, B as

$$S_{corr}[A,B] = \frac{cov(A,B)}{\sigma_A \sigma_B}, \tag{5.23}$$

  where cov is the covariance and $\sigma_{(\cdot)}$ is the standard deviation,

For simplicity, let us denote the fraction in (5.20) as S_c. This value represents the "average" similarity across all used color spectra and similarity functions.

As mentioned in Section 5.1.1.2, the dynamically estimated detection probability in (5.14) follows from one of two possible scenarios. First, there is no current measurement at time k, hence the current mask results from its predicted position,

$$p_{D,k}^{(i)}(x) = \begin{cases} S_c^{(i)} \left[h(M_{k|k-1}^{(i)}(x) \circ D_k), h_{0,k}^{(i)} \right] & \text{if } h_{0,k}^{(i)} \text{ exists,} \\ p_{D,k}^{(i)} & \text{otherwise,} \end{cases} \tag{5.24}$$

and where the histogram from the last detection $h_{0,k}^{(i)} \equiv h_{0,k-1}^{(i)}$ is used in the first scenario. A user-preset value $p_{D,k}^{(i)}$ is used for targets undetected so far. The mask is obtained via

$$M_{k|k-1}^{(i)}(x) = \begin{cases} M_{k-1}^{(i)}(x)[m - v_x, n - v_y] & \text{if } m \geq v_x, n \geq v_y, \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{5.25}$$

where m, n are indices of the binary mask M and v_x, v_y are x- and y-direction velocities of the target. $\mathbf{0}$ is a zero vector of the same size as the mask. The second scenario takes the current measurement z into account. The detection probability is modified according to

$$p_{D,k}^{(j)}(x; z) = S_c \left[h(M_k^{(j)}(z) \circ D_k), h_{0,k}^{(j)} \right], \tag{5.26}$$

where $h_{0,k}^{(j)}$ is the histogram resulting from the last detection,

$$h_{0,k}^{(j)} = \begin{cases} h(M_{k-1}^{(j)}(z) \circ D_{k-1}) & \text{if } h(M_{k-1}^{(j)}(z)) \text{ exists,} \\ h_{0,k-1} & \text{otherwise.} \end{cases} \tag{5.27}$$

The histogram $h(M_{k-1}^{(j)}(z) \circ D_{k-1})$ stands for the previously detected target, and $h_{0,k-1}$ is the histogram from the time step when the target was detected the last time.

5.3 Modified pruning for GM-PHD filter

As the PHD filter propagates the posterior intensity, the number of potential hypotheses exponentially increases. This leads to an enormous increase in memory and computational demands. Pruning techniques become indispensable in mitigating this computational load by selectively discarding less likely hypotheses, allowing for a more focused and efficient tracking process. These pruning mechanisms, guided by predefined thresholds or heuristics, ensure that the computational resources are allocated judiciously, striking a balance between accuracy and computational efficiency in multi-target tracking applications.

Recall that sensors in our work are represented by cameras. Naturally, the detecting algorithms are unable to recognize an object hidden by an obstacle. These obstacles may differ not only in size but also in color. In situations where the color of an obstacle closely matches the surrounding scene, the likelihood of detection remains sufficiently high, increasing the risk that the target may not survive even though it is only hidden. In order to suppress this phenomenon, we introduce a method for modified pruning along with standard pruning and merging techniques [18].

The most deployed generic method for pruning consists of removing targets with weights below some predefined threshold. Nonetheless, as mentioned before, targets with some history may only be hidden, and it is not desired to remove these targets from the scene. To overcome this issue, we assign each target a tag that represents the state in which the target is likely to occur,

$$S = \{\text{detected, hidden, dead}\}.$$

This tag is modeled by a discrete-time Markov chain with the transition matrix

$$P = \begin{bmatrix} p_{D,k} & 1 - p_{D,k} & 0 \\ p_{D,k} & (1 - p_{H,k}^{T_H}) \cdot (1 - p_{D,k}) & p_{H,k}^{T_H} \cdot (1 - p_{D,k}) \\ p_{D,k} & (1 - p_{H,k}^{T_H}) \cdot (1 - p_{D,k}) & p_{H,k}^{T_H} \cdot (1 - p_{D,k}) \end{bmatrix}, \tag{5.28}$$

where $p_{D,k}$ is a generic detection probability, and T_H is an exponent for controlling probabilities that is set higher in scenarios, where the target color mask is similar to the color mask of the obstacle, or may be set lower otherwise. $p_{H,k}$ is the probability that the target is removed. This probability is the result of (5.31), where the bounding boxes of the object detector are used. If we denote by $B_{P,k}^{(j)}$ the bounding box predicted T_c steps

ahead and by $B^{(j)}_{S,k}$ the bounding box of the last step k when the object was detected, then

$$B^{(j)}_{P,k} = B^{(j)}_{P,k-1} + n^{(j)}_k \cdot [v^{(j)}_{x,k}, v^{(j)}_{y,k}{}^{(j)}_{x,k}, v^{(j)}_{y,k}], \tag{5.29}$$

$$B^{(j)}_{S,k} = B^{(j)}_{S,k-1}, \tag{5.30}$$

$$p_{H,k} = S_c \left[D_k(B^{(j)}_{P,k}), D_k(B^{(j)}_{S,k}) \right], \tag{5.31}$$

where

$$n^{(j)}_k = \begin{cases} T_c & \text{if } k \; mod \; T_c = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{5.32}$$

$D_k(\cdot)$ is a part of a frame within the bounding box given by $B(\cdot)$. With this approach, when a target is most likely in the *hidden* state, the pruning threshold is heuristically lowered to prevent the target removal.

In summary, first we define a birth place in a scene. We call this birth place a spawn point (SP). It is possible to define more spawn points in a scene, depending on the situation. If a target crosses this spawn point, it is initialized, and since it is the first time the target has been seen, the detection probability is a predefined constant. In the next time step, suppose that the target is detected and there is one measurement (originating from this target) in the validation region. Due to the GM-PHD recursion, two new targets arise from the previous target with the state ξ. One originates from the misdetection possibility and one from the measurement. As we first predict the target's position and covariance matrix, we move the target's mask detected at time $k-1$ to k by its velocity (in case of the CVM model). Then the predicted mask color properties are compared to the mask made from the new measurement from time k. If nothing unpredictable happens, both masks should occur on a very similar position, thus the color masks' properties should be very similar. It such case, the detection probability is very likely to be high, lowering the weight of the misdetection-born target. As the weight is small, the target is likely to be removed during the pruning step.

If the previously detected target is hidden behind an obstacle and the object detector does not provide any measurement, only misdetection-born target arises. In such case, the mask from the time when the target was lastly detected is compared to the predicted mask that now also contains the obstacle's color properties. Due to using cameras as a sensor, frames at high frame rate are produced. It is not desired to lower fps to something like 1 fps, as a lot of information are lost. We lower the fps to 10-20 fps to enhance the dynamics of the scene. However, in a such short time period, the target, e.g. a car, does not move enough. The predicted mask then intersects with the lastly detected target's mask by a considerably huge amount. That is why we define a hidden probability $p_{H,k}$ and modify the pruning of components. As the masks are likely to intersect, the detection probability is likely to be high in the first couple of frames, where the target is hidden. It is essential for the target to survive through this time period. The hidden probability is based on the color properties of bounding boxes from the time the target was lastly detected and from the bounding box that is moved T_c time steps ahead in the direction of the target's velocity. The moved bounding box reveals the color properties

of the obstacle, that are likely to be different from those assigned to the target when it was lastly detected. Then, the hidden probability should be low and the target's state should be in a hidden state, based on 5.28, therefore the pruning threshold is lowered. After the target overtakes the obstacle, it should appear in a scene and be detected again. If it does not and the color properties of a scene given by the moving bounding box differ from the ones captured when the target was lastly detected, the hidden probability is high and the target is in a dead state. The pruning threshold is not lowered and the detection probability is low, so the target is removed during the pruning step.

5.4 Merging in GM-PHD filter with dynamic detection probability

Another way to lower the computational demands and to potentially increase the robustness of a filter is through merging of the targets that appear in the same neighbourhood. This procedure is especially useful in situations when many measurements occur in a validation region of a target. If at least some of these measurements originate from clutter, the target count increases rapidly due to the GM-PHD recursion, increasing the computational requirements. To prevent this issue, the target merging technique is necessary. In the GM-PHD filter, each target carries three variables: the target's weight w, mean m and a covariance matrix P. Merging these values of multiple filters is straightforward and is described in [18] and Section 3.4.2.4. However, in our modified GM-PHD filter, each target also holds these information:

- **Current bounding box position:** The bbox in $xyxy$ format determines current position of an object. This bbox is determined by a bbox given by a measurement z in the current time step k in an update step. The target's bbox is shifted in the prediction step in (5.29), thus the target must have been initialized in the past in order to have the current bbox.
- **Current mask:** The current binary mask is given by a measurement z in the current time step k in the update step. In order to inialize current mask, it must have been initialized by a measurement in the past. The mask is shifted according to (5.25).
- Data structure **Object Stats:** Every new target born in the update step of the GM-PHD filter by a measurement z from a target with a previous state ξ, contains a data structure *Object Statss*. This structure is composed of many variables determining the target's history. These variable arise in the update step from a measurement and are useful for calculating the dynamic detection probability in Equations (5.24) - (5.31).
  - **Bounding box position:** This variable in Object stats represents a bounding box from the time the target was lastly detected. To calculate $p_{H,k}$ in (5.31), this and the current bbox are compared.
  - **Mask:** This variable represents the target's mask from the time the target was lastly detected. In (5.24) and (5.26) this mask is compared to the current mask resulting from the prediction step of a target with the previous state ξ.
  - **Frame:** To compare the color histograms in (5.24) and (5.26), each target contains a frame from the time k in which it was born.

- **Time stamp:** This is a variable to save the time k the target was born.

- Data structure **Markov Chain:** This data structure determines the target's state. As the distribution determined by a transition matrix in (5.28) is not stationary, it is necessary to calculate the resulting distribution in every time step.
  - **Initial distribution:** To get the target's state, an initial distribution needs to be defined first.
  - **Result matrix:** The result matrix is an ongoing outcome of a previous result matrix and a transition matrix, i.e, result matrix $R_k = R_{k-1}P^{k-k_0}$, where k_0 is the time the target was born, $R_0 = I_3$ and I_n is the identity matrix of the shape $n \times n$.

The merging procedure of the previous information is inspired by the original GM-PHD merging procedure in [18].

The current bounding box merging is straightforward as (x_1, y_1) and (x_2, y_2) coordinates defining the bbox are averaged using the targets' weight. Let us define a set L, which represents a subset of targets to merge into a single target. Then a resulting target's bbox

$$\tilde{B}_k^{(l)} = \frac{\sum_{i \in L} B_k^{(i)} * w_k^{(i)}}{\sum_{i \in L} w_k^{(i)}}, \tag{5.33}$$

where $w_k^{(i)}$, is the weight of a target i is the weighted mean of targets in subset L. The same procedure is applied to the bounding box in *Object Stats*, i.e., the previous bbox.

The merging of masks is slightly complicated, as we merge targets' masks into a single "average" mask. First, binary masks of targets in the subset L are converted into *float* data type, then multiplied with targets' weight and divided by the sum of these weights. To get the resulting binary mask, each element of mask

$$\tilde{M}_k^{(l)} = \frac{\sum_{i \in L} M_k^{(i)} * w_k^{(i)}}{\sum_{i \in L} w_k^{(i)}} \tag{5.34}$$

is rounded either to zero or one and the mask is converted to the binary data type.

We have not experimented with complicated merging of frames and time stamps, only argument maxima is used for both values, i.e,

$$\tilde{t}_k^{(l)} = max_{i \in L} t_k^{(i)}, \tag{5.35}$$

$$\tilde{D}_k^{(l)} = D_k^{(\tilde{t}_k^{(l)})}. \tag{5.36}$$

As the initial distribution is the same for every target, the resulting merged target's initial distribution is only a copy of any target in L.

The result matrix merging procedure is similar to the bounding box distribution. The result matrix

$$\tilde{N}_k^{(l)} = \frac{\sum_{i \in L} N_k^{(i)} * w_k^{(i)}}{\sum_{i \in L} w_k^{(i)}} \tag{5.37}$$

is the "average" result matrix of targets in subset L.

The pseudo-code algorithm of modified pruning in GM-PHD filter is shown in Algorithm 5.

Algorithm 5 Pseudo-algorithm for pruning in the GM-PHD filter

Require: $\{w_k^{(i)}, m_k^{(i)}, P_k^{(i)}, B_k^{(i)}, M_k^{(i)}, B_{OS,k}^{(i)}, M_{OS,k}^{(i)}, t_{OS,k}^{(i)}, D_k^{(i)}, G_{OS,k}^{(i)}, N_{OS,k}^{(i)}\}_{i=1}^{J_k}$, a truncation threshold T, a merging threshold U and a maximum number of allowed Gaussian terms J_{max}. Set $l = 0$, and $I = \{i = 1, \ldots, J_k | w_k^{(i)} > T\}$.

procedure MERGING OF TARGETS
  while $I \neq \emptyset$ **do**
    $l := l + 1$
    $j := \text{argmax}_{i \in I} w_k(i)$,
    $L := \left\{ i \in I | (m_k^{(i)} - m_k^{(j)})^T (P_k(i))^{-1} (m_k^{(i)} - m_k^{(j)}) \leq U \right\}$,
    $\tilde{w}_k^{(l)} = \sum_{i \in L} w_k^{(i)}$,
    $\tilde{m}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} w_k(i) x_k^{(i)}$,
    $\tilde{P}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} w_k^{(i)} (P_k^{(i)} + (m_k^{(l)} - m_k^{(i)})(m_k^{(l)} - m_k^{(i)})^T$,
    $\tilde{B}_k^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} B_k^{(i)} * w_k^{(i)}$
    $\tilde{M}_k^{(l)} = round(\frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} M_k^{(i)} * w_k^{(i)})$
    $\tilde{B}_{OS,k}^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} B_{OS,k}^{(i)} * w_k^{(i)}$ ▷ OS stands for Object Stats
    $\tilde{M}_{OS,k}^{(l)} = round(\frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} M_{OS,k}^{(i)} * w_k^{(i)})$
    $\tilde{t}_{OS,k}^{(l)} = max_{i \in L} t_{OS,k}^{(i)}$
    $\tilde{D}_{OS,k}^{(l)} = D_k^{(\tilde{t}_{OS,k}^{(l)})}$
    $\tilde{G}_{OS,k}^{(l)} = G_{OS,k}^{(j)}$ ▷ Initial distribution of Markov process
    $\tilde{N}_{OS,k}^{(l)} = \frac{1}{\tilde{w}_k^{(l)}} \sum_{i \in L} N_{OS,k}^{(i)} * w_k^{(i)}$
    $I := I/L$.
  end while
end procedure

If $l > J_{max}$ then replace $\{\tilde{w}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)}, \tilde{B}_k^{(i)}, \tilde{M}_k^{(i)}, \tilde{B}_{OS,k}^{(i)}, \tilde{M}_{OS,k}^{(i)}, \tilde{t}_{OS,k}^{(i)}, \tilde{D}_{OS,k}^{(i)}, \tilde{G}_{OS,k}^{(i)}, \tilde{N}_{OS,k}^{(i)}\}_{i=1}^{l}$ by those of the J_{max} Gaussians with largest weights.

Output: $\{\tilde{w}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{P}_k^{(i)}, \tilde{B}_k^{(i)}, \tilde{M}_k^{(i)}, \tilde{B}_{OS,k}^{(i)}, \tilde{M}_{OS,k}^{(i)}, \tilde{t}_{OS,k}^{(i)}, \tilde{D}_{OS,k}^{(i)}, \tilde{G}_{OS,k}^{(i)}, \tilde{N}_{OS,k}^{(i)}\}_{i=1}^{l}$ as pruned Gaussian components.

Chapter 6

Experiments

In order to demonstrate the efficacy of the proposed method for estimating detection probability, numerous experiments were conducted. This section meticulously examines each of these experiments, delineating the selected parameters and conducting a comparative analysis against the GM-PHD filter with static detection probability. A mixture of video was made to examine the experiments.

- **V1:** This video shot by a camera from a highway was downloaded from YouTube. The link of this video: youtube.com
- **V2:** The next video comes from YouTube as well. It captures a common traffic. The link of this video: youtube.com
- **V3:** To get a video of a traffic with an obstacle in the view of the camera, we recorded a video on our own. This video is one of them.
- **V4:** Another video, we recorded, with a slightly different scenario.

In all experiments as a state space model, CVM model is used, i.e., the state $x_k = [p_{x,k}, p_{y,k}, v_{x,k}, v_{y,k}]^T$ of each target consists of two-dimensional position $(p_{x,k}, p_{y,k})$ and velocity $(v_{x,k}, v_{y,k})$. The measurement is the center of the mask of the detected object, which is typically noisy. The survival probability of the targets $p_{S,k} = 0.99$. A state evolution model (3.33) is employed with

$$F_k = \begin{bmatrix} I_2 & \Delta I_2 \\ 0_2 & I_2 \end{bmatrix}, \quad Q_k = \sigma_v^2 \left[\Delta I_4\right], \tag{6.1}$$

where I_n and 0_n denote the $n \times n$ identity and zero matrices respectively and $\Delta = 1$ is the sampling period.

For every experiment, different parameters for the model have to be applied. Parameters like P, σ_v are displayed in corresponding tables.

The measurement follows the observation model (3.34) with $H_k = [I_2, 0_2]$, $R_k = \sigma_\epsilon^2 I_2$, where σ_ϵ is the standard deviation of the measurement noise. I_2 and 0_2 stand for the 2×2 identity and zero matrices, respectively. The standard deviation σ_ϵ is displayed in tables as well.

Since the experiments compare the proposed method for estimating the dynamic detection probability with constant detection probability, each table contains additional information about the necessary parameters. For constant detection probability, it is the value of the probability. For dynamic detection probability, values of T_H, T_C from Equation (5.24) and initial $p_{D,k}(x)$ is provided.

The pruning thresholds are the same for both GM-PHD filters, except filter with dynamic detection probability includes second pruning threshold for pruning targets with state *detected* or *hidden*. The basic pruning threshold is covered in tables as T_p, the lowered pruning threshold as T_l.

In addition, object detectors' results are dependent on thresholds as well. The Yolo model requires confidence threshold T_{YOLO} for objects visualization. The Grounding DINO model requires two thresholds as it detects objects based on text input. The text input threshold T_{text} and bounding box threshold T_{bbox} for given experiment are included in corresponding parameter settings tables.

The labels above the targets are the numerical representations of the state targets: 0 = detected, 1 = hidden, 2 = dead.

The means of the birth places are roughly in the middle of the traffic lanes and its confidence ellipses of size of three standard deviations are bounded by blue circles. The red bounding boxes in figures represent bounding boxes given by an object detection model. The red ellipses are the covariance matrices of the targets.

6.1 E1: Traffic without any obstacle

The first experiment focuses on comparison of GM-PHD filter with constant detection probability and GM-PHD filter with dynamic detection probability with different settings explained in Section 5.1. To analyze, if our proposed method works on common scenarios, the video do not include any obstacle, thus the targets are constantly visible.

The video is recorded at 29 fps. For simplicity only detections of the right side of the traffic are taken into account in this video. In this experiments frames 36-79 of video V1 is shown, as it includes some curios situations.

6.1.1 V1 – GM-PHD with constant detection probability

The measurements for GM-PHD filter with constant detection probability are obtained by the YOLO object detection model. The parameters' values are displayed in Table 6.1.

P_D	P	σ_v	σ_ϵ	T_p	T_{YOLO}
0.9	$diag(600, 600, 600, 600)$	150	0.1	0.1	0.3

Table 6.1 The parameter settings for experiment E1-V1 with constant detection probability.

Figure 6.2 shows some highlights of the GM-PHD filter with the constant detection probability.

- **6.2a:** This is the starting frame, where four cars were previously detected, thus they became our observed targets. In the distance we see that one more car was detected

by YOLO, but it did not cross any spawning point, so it does not count into observed targets.

- **6.2b:** Another car is approaching to the scene and should be initialized soon. The YOLO model is not able to always detect all desired objects, as it happened here with the left distanced car. As the detection probality is 0.9, this target did not survive the pruning step and is lost now.
- **6.2c:** The previously lost car is detected again, but does not belong to the set of observed targets furthermore.
- **6.2d:** The new car crossed the spawning point, but the YOLO model was not able to detect this kind of car for many frames in a row, thus this car is not tracked at all.
- **6.2e:** Two cars on the right were not detected again. But this time onw of the targets was able to survive, so we continue to track at least one of the cars.
- **6.2f:** The previously undetected cars are detected again and both cars are tracked again.
- **6.2g:** Another car arrives to the scene and this time it is successfully detected and initialized.
- **6.2h:** Even though all six cars are detected by YOLO, only three targets appear in the scene. One of the targets covers two cars at once so we can guess that four out of six targets are covered by the GM-PHD filter.

In Graph 6.1 it is clearly seen that even though the number of targets is increasing, the misdetection of the YOLO model causes the targets loss. The "All targets in set" line is fully covered by the blue line, thus the targets are absolutely lost, thus they can not be reborn by a measurement.

This experiment shows us that the GM-PHD filter is able to accurately track the position of objects. With detection probablity $p_D = 0.9$ and without modified pruning technique, the filter is very sensitive to the output of object detector and if YOLO is not able to detect all desired objects, targets could be easily lost.

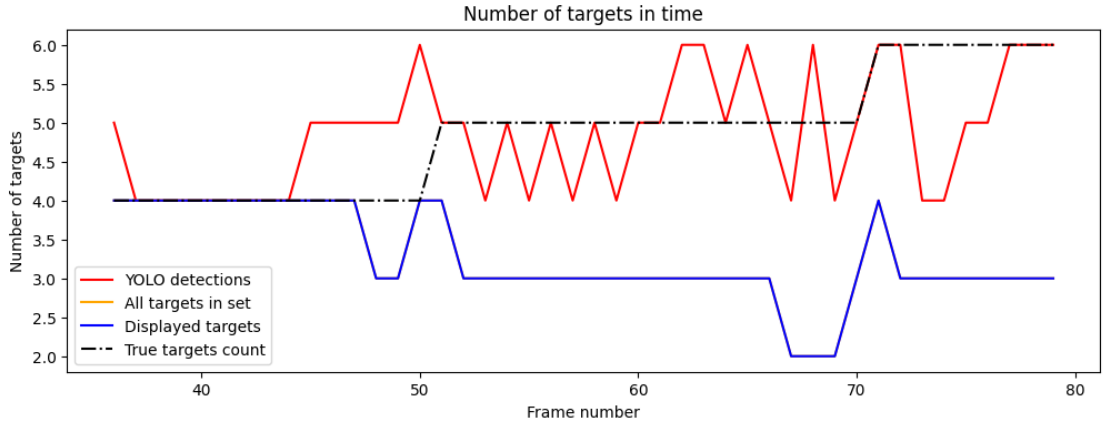
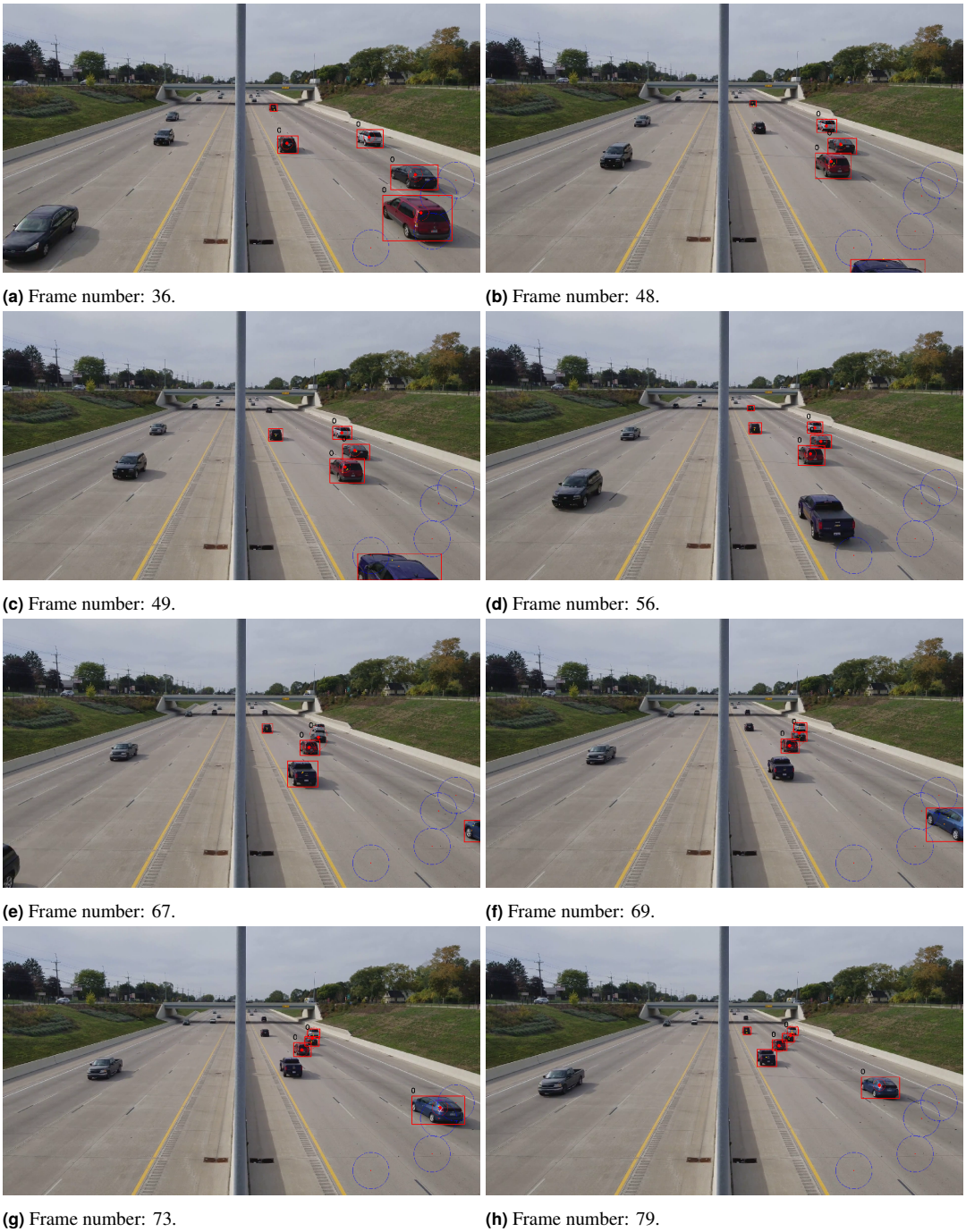


Figure 6.1 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 49. **(d)** Frame number: 56.

(e) Frame number: 67. **(f)** Frame number: 69.

(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.2 Image sequence of tracked objects using GM-PHD filter with constant detection probability.

6.1.2 V1 – GM-PHD with dynamic detection probability

Following experiments test GM-PHD filter with the dynamic detection probability and the modified pruning on the video *V1*.

6.1.2.1 S1 – YOLO + YOLO

This experiment uses settings *S1*, i.e, the YOLO model provides with both object detection bboxes and segmentation masks. The parameter settings are shown in Table 6.2.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{YOLO}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3

Table 6.2 The parameter settings for experiment E1-V1-S1 with dynamic detection probability.

Figure 6.4 shows the performance of the GM-PHD filter with the dynamic detection probability with settings *S1*.

- **6.4a:** As in previous experiment the first frame starts with four cars that were previously detected. The car that is far away is detected, but not initialized, as it have not crossed any spawning point.
- **6.4b:** In the frame 48, the left distanced car was not detected as in previous experiment. But due to the high detection probability and modified pruning, the target is able to survive.
- **6.4c:** The previously undetected car is detected again and the target survives even with misdetection in previous frames.
- **6.4d:** The new car crossed the spawning point, but the YOLO model is not able to detect this vehicle for many consecutive frames, thus this car is not initialized.
- **6.4e:** Two cars on the right are not detected again but targets survive.
- **6.4f:** The previously undetected cars are detected again and both targets get their measurements increasing their weight.
- **6.4g:** Another car arrives to the scene, and it is successfully initialized.
- **6.4h:** The undetected car on the left caught up other targets. As a result of this is initializing this target, even though it is misdetected most of the time. And so we have six true targets and also six tracked targets.

The graph 6.3 presents better performance of the GM-PHD filter. Even though not all the targets are displayed when they should, most of the time, there are more targets is a queue that do not have weight high enough to be displayed. As soon as these targets get their measurements, their weight increases and are back on the scene.

Not only the number of displayed targets is closer to the true targets count, but also targets waiting in the queue exceeds the true count, so we are aware of the potential targets, that can appear in the scene.

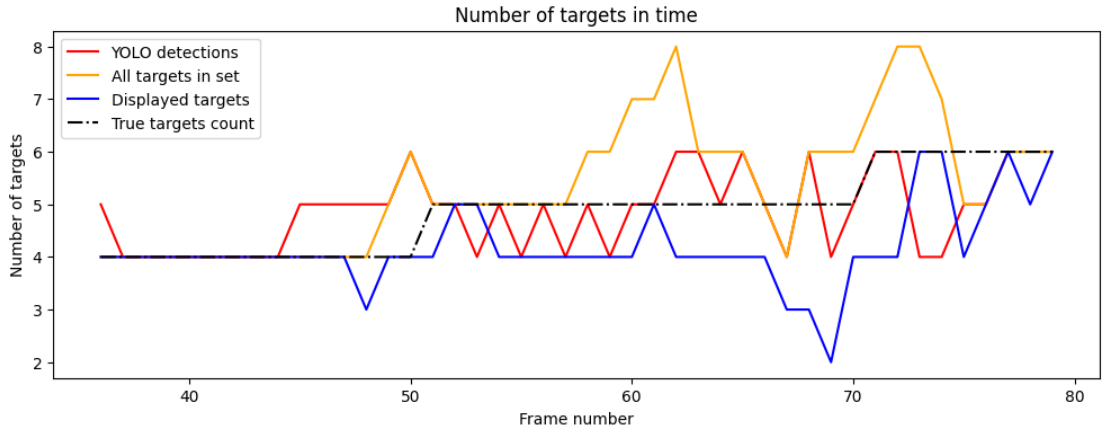
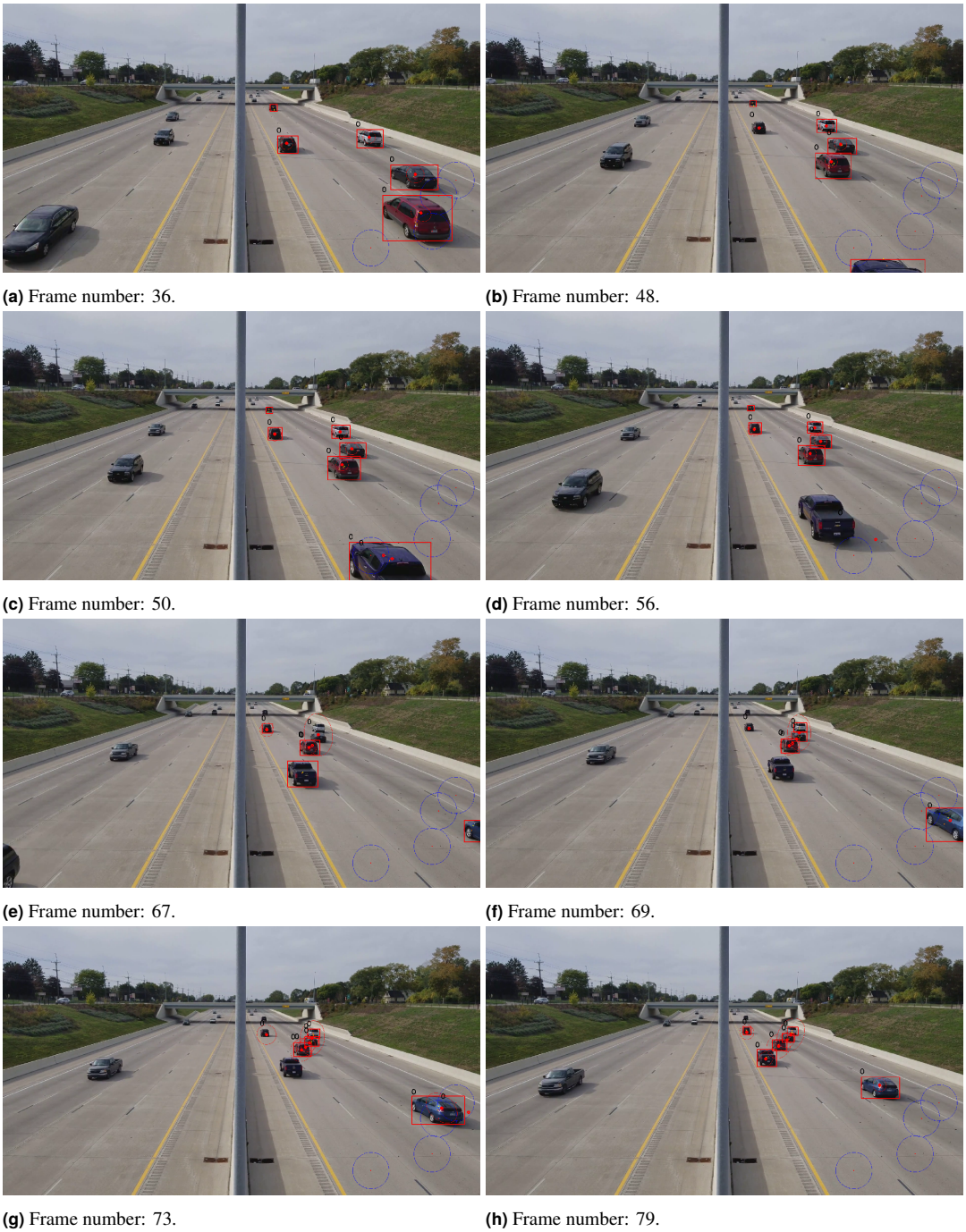


Figure 6.3 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 50. **(d)** Frame number: 56.

(e) Frame number: 67. **(f)** Frame number: 69.

(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.4 Image sequence of tracked objects using GM-PHD filter with dynamic detection probability and YOLO only.

6.1.2.2 S2 – YOLO + SAM

This experiment employs configuration *S2*, wherein the YOLO model furnishes objects' bounding boxes and the SAM model furnishes segmentation masks. The parameter configurations can be found in Table 6.3.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{YOLO}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3

Table 6.3 The parameter settings for experiment E1-V1-S2 with dynamic detection probability.

Figure 6.6 illustrates the GM-PHD filter's performance with dynamic detection probability, employing settings *S2*. This sequence is very similar to the previous experiment. There are four targets at the beginning and they are tracked successfully the whole time. At frame 6.6f two of the targets were not detected, but both survived. The YOLO model is not able to detect the fifth car, but it is initialized lated due to the other target.

The graph 6.5 shows a better stability of number of tracked targets. This might be cased by the fact, that object detection YOLO model gives slightly different results than the object detection YOLO model with segmentation. The "All targets in set" orange line representing the number of targets in filter's queue is also more accurate to the true count, deflecting only be one target at maximum.

Settings *S2* perform a little better than settings *S1*. The number of tracked objects is closer to the true value.

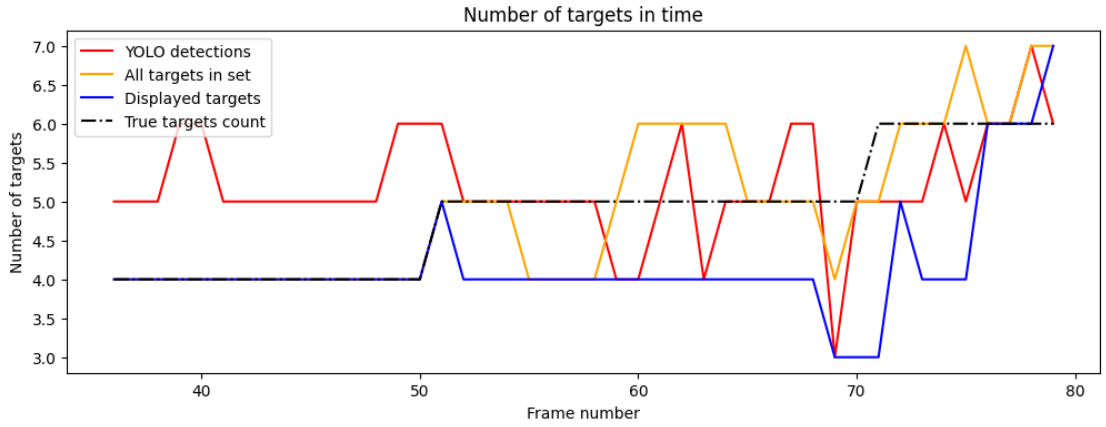
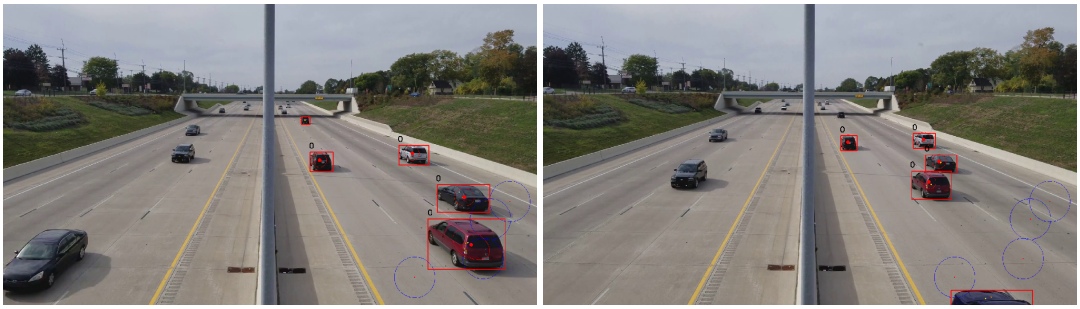


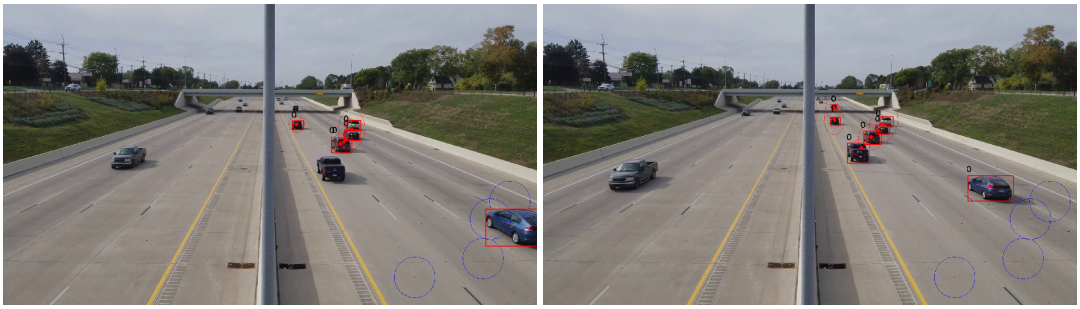
Figure 6.5 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 53. **(d)** Frame number: 57.

(e) Frame number: 62. **(f)** Frame number: 69.



(g) Frame number: 70. **(h)** Frame number: 78.

Figure 6.6 Image sequence of tracked objects using GM-PHD filter with dynamic detection probability, the YOLO object detector and the SAM image segmentation model.

6.1.2.3 S3 – Grounded DINO

The experiment with settings *S3* uses Grounding DINO object detector and the SAM image segmentation model. All used parameters are included in Table 6.4.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{text}	T_{bbox}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3	0.3

Table 6.4 The parameter settings for experiment E1-V1-S3 with dynamic detection probability.

Figure 6.8 shows the performance of the GM-PHD filter with the dynamic detection probability with settings *S3*.

- **6.8a:** Due to the different object detection model, there are more cars detected in the scene. Only four cars drove through the spawn points till this moment, so only there objects are counted to the true count.
- **6.8b:** All targets are tracked successfully.
- **6.8c:** Unlike YOLO, Grounding DINO is able to detect the arriving car.
- **6.8g:** The next arriving car is detected and initialized as well.
- **6.8h:** In this frame new problems arises. As cars are far away, closer to each other and the model is still able to detect all cars, we gain too many new targets. For this model, different observation noise is neccessary. Moreover, for these kind of situations, dynamic model and observation noise would be appropriate ssolution.

In graph 6.7 we see, that the number of detected objects is far beyond the true count. But the number of displayed targets is almost perfect till frame number 66. After that as targets are closer to each other, the number of targets grows rapidly, causing errors to the number of tracked objects.

This settings outperforms the other settings by far. The great performance of Grounding DINO causes another problems arising from characteristics of the video. The video is taken from an angle, which makes cars smaller as they go. This dynamics do go together well with static measurement and observation noise.

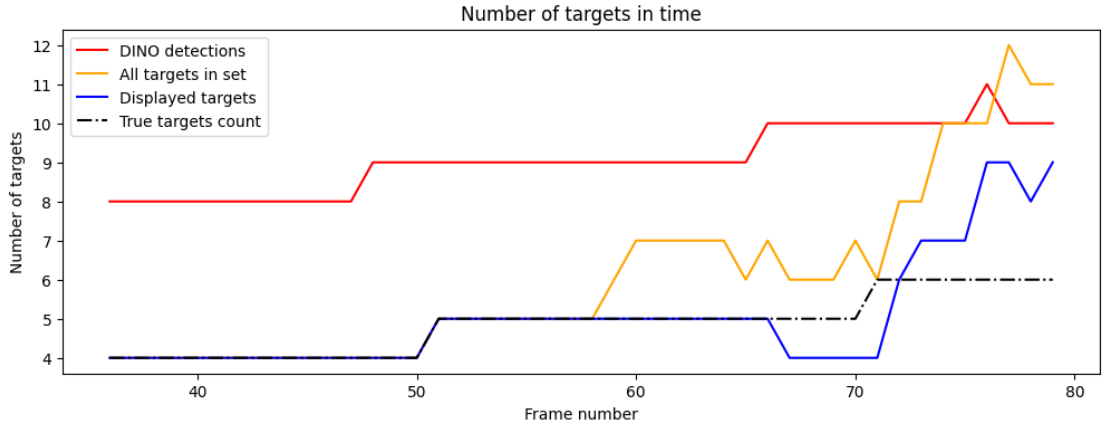
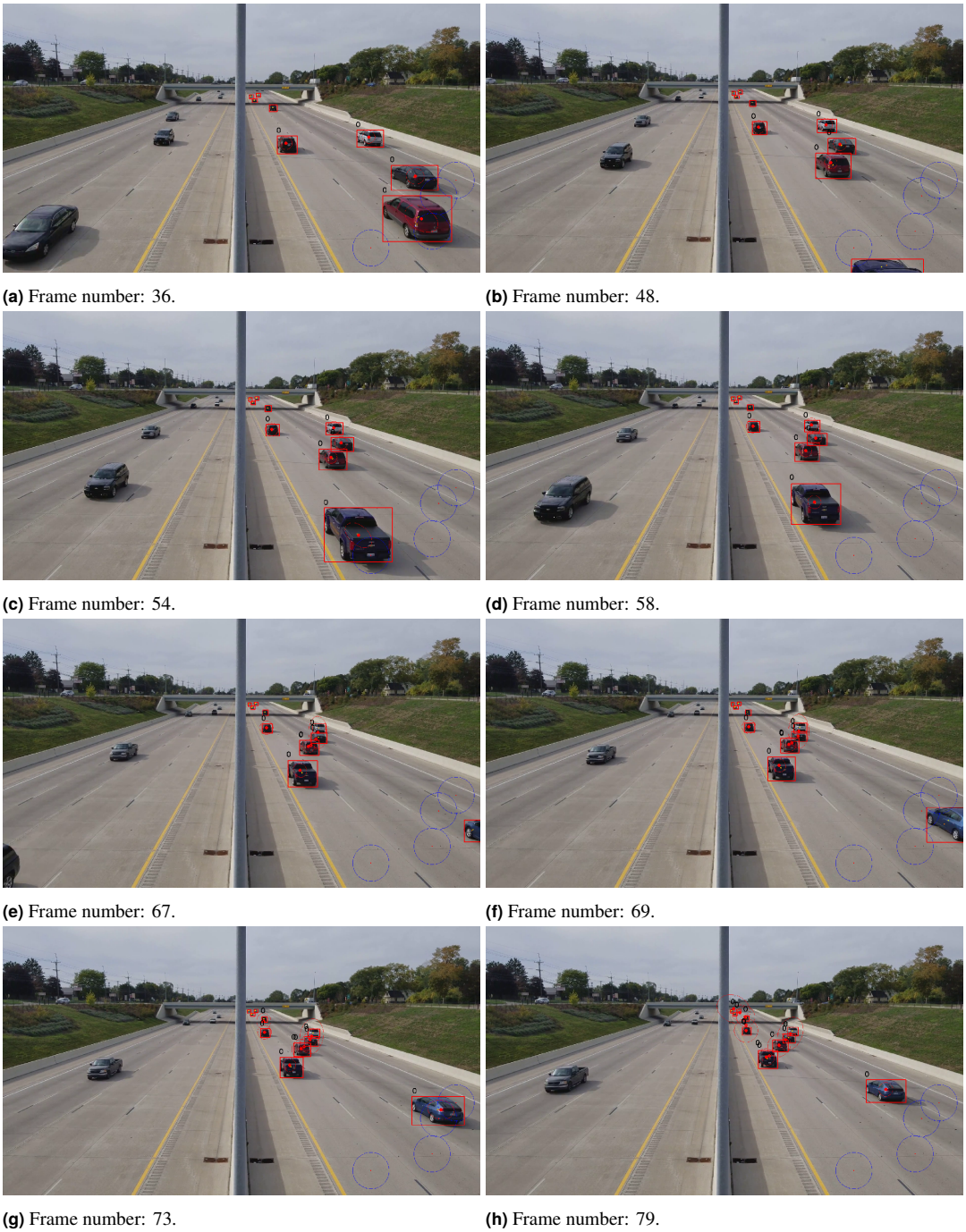


Figure 6.7 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 54. **(d)** Frame number: 58.

(e) Frame number: 67. **(f)** Frame number: 69.

(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.8 Image sequence of tracked objects using the GM-PHD filter with the dynamic detection probability and the Grounded DINO model.

6.1.3 V2 – GM-PHD with constant detection probability

The measurements for GM-PHD filter with constant detection probability are obtained by the YOLO object detection model. The parameters' values are displayed in Table 6.1.

P_D	P	σ_v	σ_ϵ	T_p	T_{YOLO}
0.9	$diag(600, 600, 600, 600)$	150	0.1	0.1	0.3

Table 6.5 The parameter settings for experiment E1-V1 with constant detection probability.

Figure 6.2 shows some highlights of the GM-PHD filter with the constant detection probability.

- **6.2a:** This is the starting frame, where four cars were previously detected, thus they became our observed targets. In the distance we see that one more car was detected by YOLO, but it did not cross any spawning point, so it does not count into observed targets.
- **6.2b:** Another car is approaching to the scene and should be initialized soon. The YOLO model is not able to always detect all desired objects, as it happened here with the left distanced car. As the detection probality is 0.9, this target did not survive the pruning step and is lost now.
- **6.2c:** The previously lost car is detected again, but does not belong to the set of observed targets furthermore.
- **6.2d:** The new car crossed the spawning point, but the YOLO model was not able to detect this kind of car for many frames in a row, thus this car is not tracked at all.
- **6.2e:** Two cars on the right were not detected again. But this time onw of the targets was able to survive, so we continue to track at least one of the cars.
- **6.2f:** The previously undetected cars are detected again and both cars are tracked again.
- **6.2g:** Another car arrives to the scene and this time it is successfully detected and initialized.
- **6.2h:** Even though all six cars are detected by YOLO, only three targets appear in the scene. One of the targets covers two cars at once so we can guess that four out of six targets are covered by the GM-PHD filter.

In Graph 6.1 it is clearly seen that even though the number of targets is increasing, the misdetection of the YOLO model causes the targets loss. The "All targets in set" line is fully covered by the blue line, thus the targets are absolutely lost, thus they can not be reborn by a measurement.

This experiment shows us that the GM-PHD filter is able to accurately track the position of objects. With detection probablity $p_D = 0.9$ and without modified pruning technique, the filter is very sensitive to the output of object detector and if YOLO is not able to detect all desired objects, targets could be easily lost.

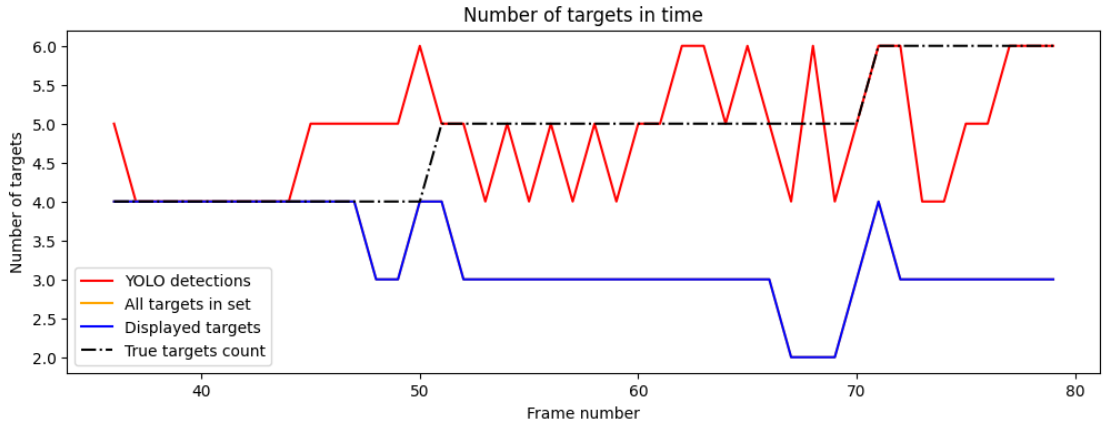
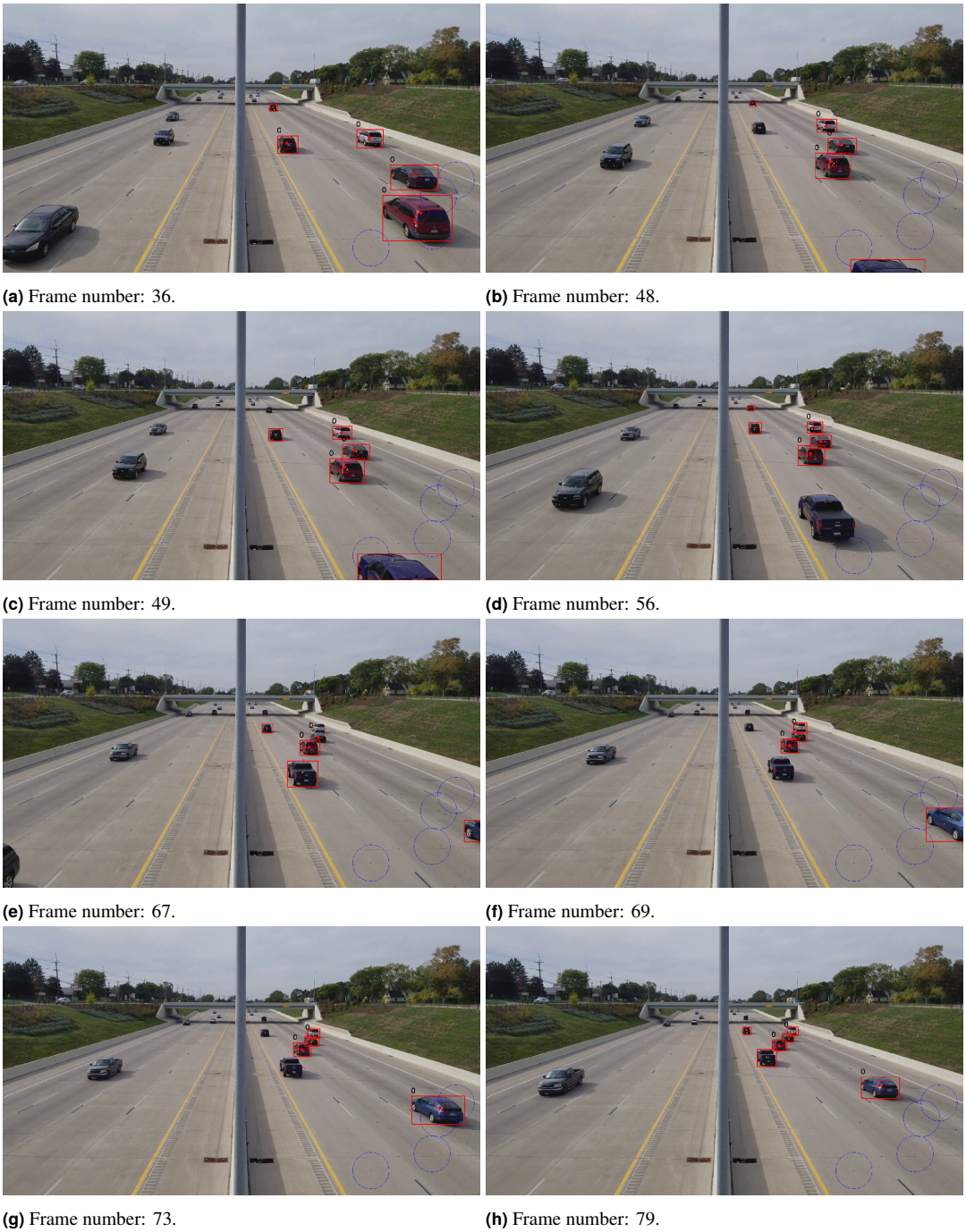


Figure 6.9 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 49. **(d)** Frame number: 56.

(e) Frame number: 67. **(f)** Frame number: 69.

(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.10 Image sequence of tracked objects using GM-PHD filter with constant detection probability.

6.1.4 V1 – GM-PHD with dynamic detection probability

Following experiments test GM-PHD filter with the dynamic detection probability and the modified pruning on the video *V1*.

6.1.4.1 S1 – YOLO + YOLO

This experiment uses settings *S1*, i.e, the YOLO model provides with both object detection bboxes and segmentation masks. The parameter settings are shown in Table 6.2.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{YOLO}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3

Table 6.6 The parameter settings for experiment E1-V1-S1 with dynamic detection probability.

Figure 6.4 shows the performance of the GM-PHD filter with the dynamic detection probability with settings *S1*.

- **6.4a:** As in previous experiment the first frame starts with four cars that were previously detected. The car that is far away is detected, but not initialized, as it have not crossed any spawning point.
- **6.4b:** In the frame 48, the left distanced car was not detected as in previous experiment. But due to the high detection probability and modified pruning, the target is able to survive.
- **6.4c:** The previously undetected car is detected again and the target survives even with misdetection in previous frames.
- **6.4d:** The new car crossed the spawning point, but the YOLO model is not able to detect this vehicle for many consecutive frames, thus this car is not initialized.
- **6.4e:** Two cars on the right are not detected again but targets survive.
- **6.4f:** The previously undetected cars are detected again and both targets get their measurements increasing their weight.
- **6.4g:** Another car arrives to the scene, and it is successfully initialized.
- **6.4h:** The undetected car on the left caught up other targets. As a result of this is initializing this target, even though it is misdetected most of the time. And so we have six true targets and also six tracked targets.

The graph 6.3 presents better performance of the GM-PHD filter. Even though not all the targets are displayed when they should, most of the time, there are more targets is a queue that do not have weight high enough to be displayed. As soon as these targets get their measurements, their weight increases and are back on the scene.

Not only the number of displayed targets is closer to the true targets count, but also targets waiting in the queue exceeds the true count, so we are aware of the potential targets, that can appear in the scene.

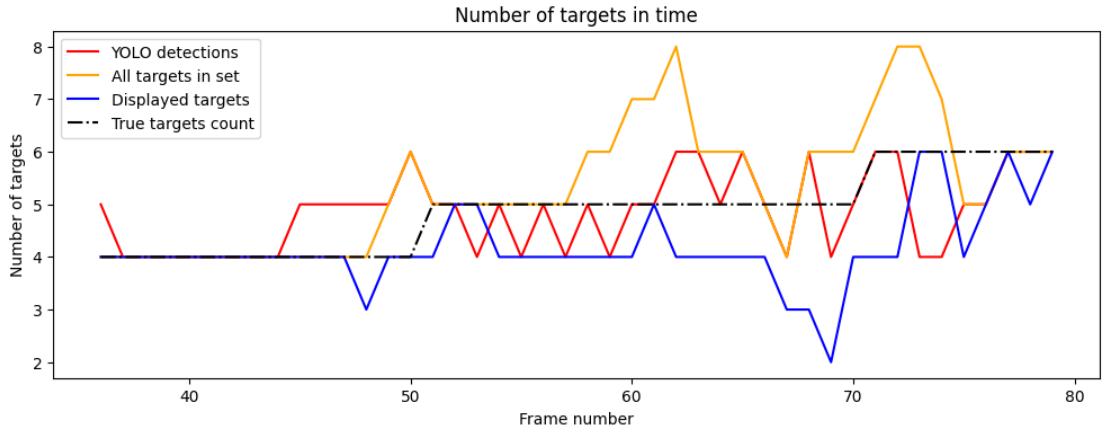
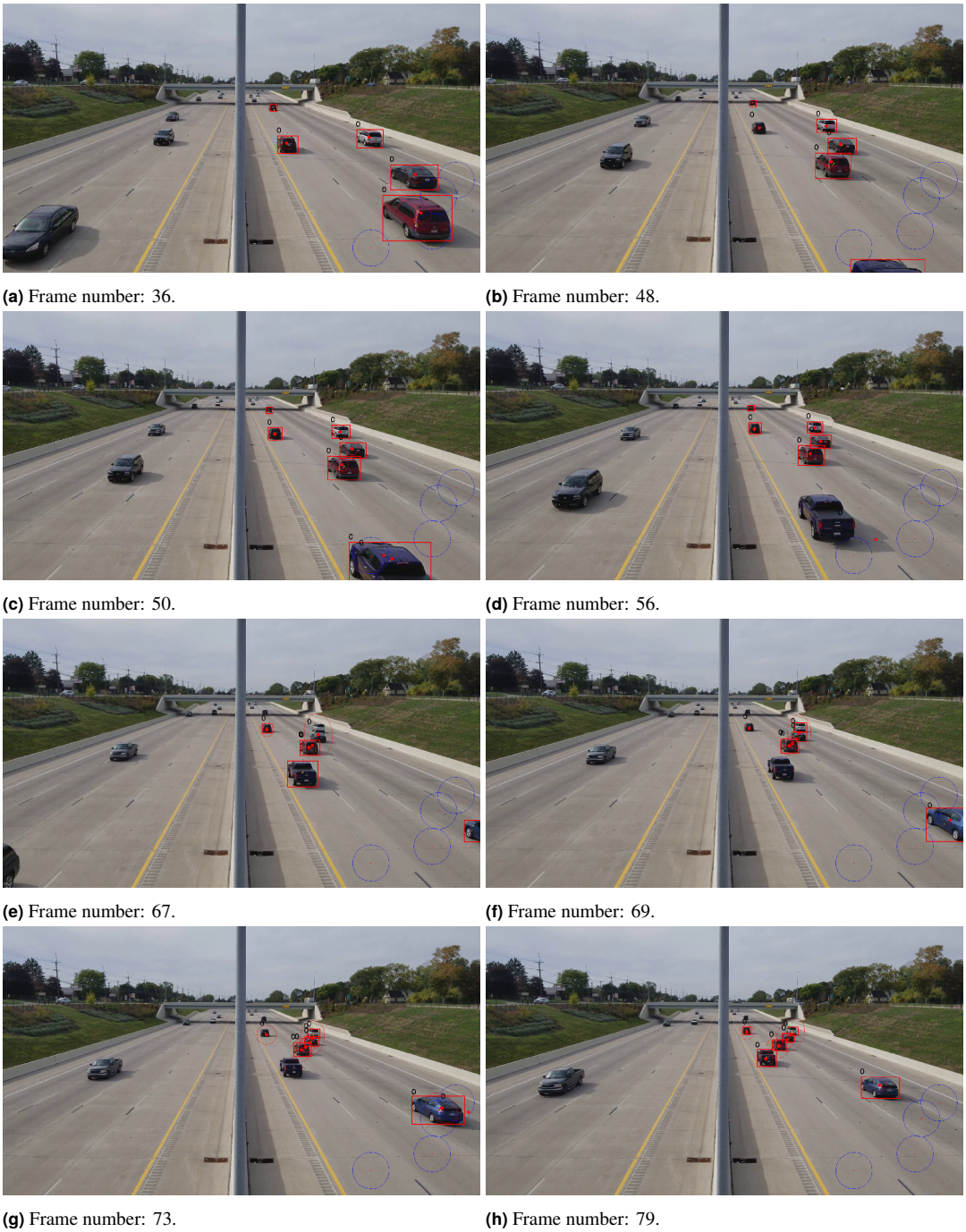


Figure 6.11 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 50. **(d)** Frame number: 56.

(e) Frame number: 67. **(f)** Frame number: 69.

(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.12 Image sequence of tracked objects using GM-PHD filter with dynamic detection probability and YOLO only.

6.1.4.2 S2 – YOLO + SAM

This experiment employs configuration *S2*, wherein the YOLO model furnishes objects' bounding boxes and the SAM model furnishes segmentation masks. The parameter configurations can be found in Table 6.3.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{YOLO}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3

Table 6.7 The parameter settings for experiment E1-V1-S2 with dynamic detection probability.

Figure 6.6 illustrates the GM-PHD filter's performance with dynamic detection probability, employing settings *S2*. This sequence is very similar to the previous experiment. There are four targets at the beginning and they are tracked successfully the whole time. At frame 6.6f two of the targets were not detected, but both survived. The YOLO model is not able to detect the fifth car, but it is initialized lated due to the other target.

The graph 6.5 shows a better stability of number of tracked targets. This might be cased by the fact, that object detection YOLO model gives slightly different results than the object detection YOLO model with segmentation. The "All targets in set" orange line representing the number of targets in filter's queue is also more accurate to the true count, deflecting only be one target at maximum.

Settings *S2* perform a little better than settings *S1*. The number of tracked objects is closer to the true value.

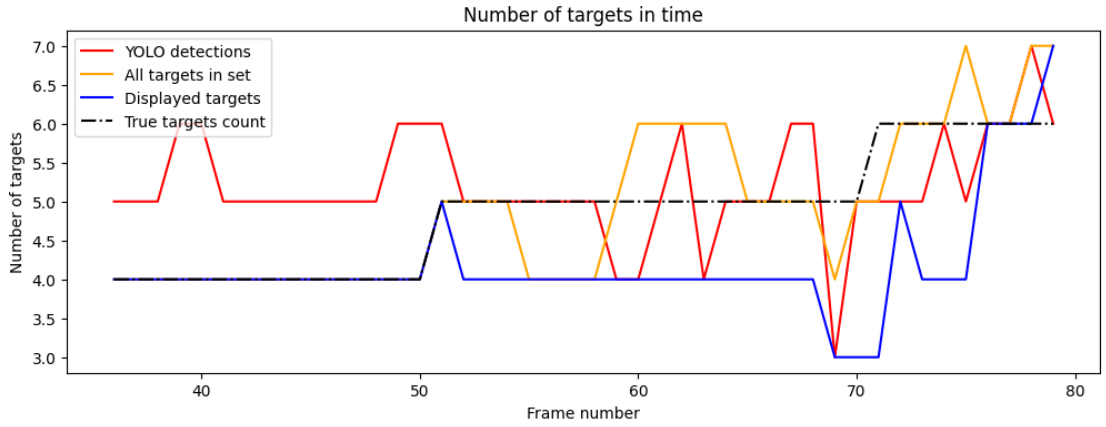
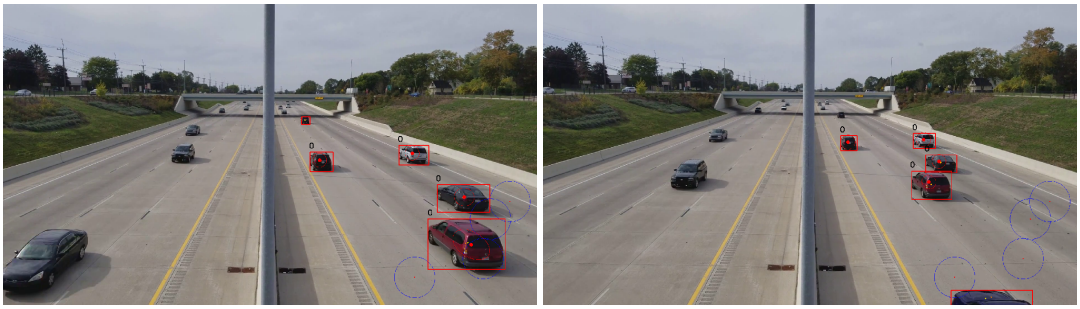


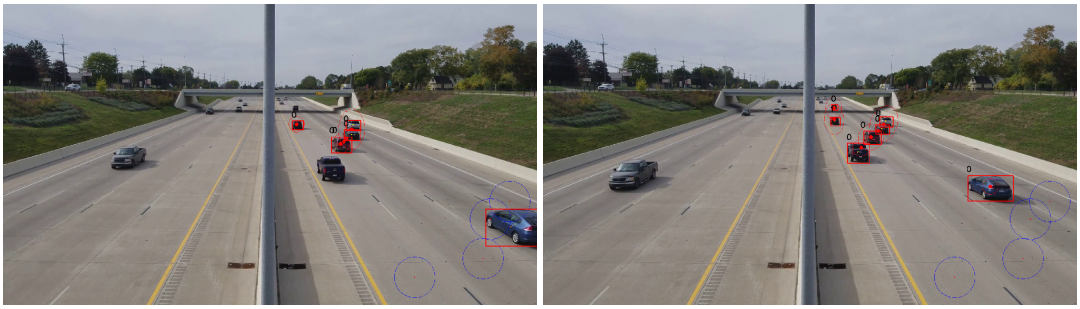
Figure 6.13 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 53. **(d)** Frame number: 57.

(e) Frame number: 62. **(f)** Frame number: 69.



(g) Frame number: 70. **(h)** Frame number: 78.

Figure 6.14 Image sequence of tracked objects using GM-PHD filter with dynamic detection probability, the YOLO object detector and the SAM image segmentation model.

6.1.4.3 S3 – Grounded DINO

The experiment with settings *S3* uses Grounding DINO object detector and the SAM image segmentation model. All used parameters are included in Table 6.4.

$P_{D,k}(x)$	P	σ_v	σ_ϵ	T_H	T_d	T_p	T_l	T_{text}	T_{bbox}
0.3	$diag(600, 600, 600, 600)$	150	0.1	1	3	0.1	0.01	0.3	0.3

Table 6.8 The parameter settings for experiment E1-V1-S3 with dynamic detection probability.

Figure 6.8 shows the performance of the GM-PHD filter with the dynamic detection probability with settings *S3*.

- **6.8a:** Due to the different object detection model, there are more cars detected in the scene. Only four cars drove through the spawn points till this moment, so only there objects are counted to the true count.
- **6.8b:** All targets are tracked successfully.
- **6.8c:** Unlike YOLO, Grounding DINO is able to detect the arriving car.
- **6.8g:** The next arriving car is detected and initialized as well.
- **6.8h:** In this frame new problems arises. As cars are far away, closer to each other and the model is still able to detect all cars, we gain too many new targets. For this model, different observation noise is neccessary. Moreover, for these kind of situations, dynamic model and observation noise would be appropriate ssolution.

In graph 6.7 we see, that the number of detected objects is far beyond the true count. But the number of displayed targets is almost perfect till frame number 66. After that as targets are closer to each other, the number of targets grows rapidly, causing errors to the number of tracked objects.

This settings outperforms the other settings by far. The great performance of Grounding DINO causes another problems arising from characteristics of the video. The video is taken from an angle, which makes cars smaller as they go. This dynamics do go together well with static measurement and observation noise.

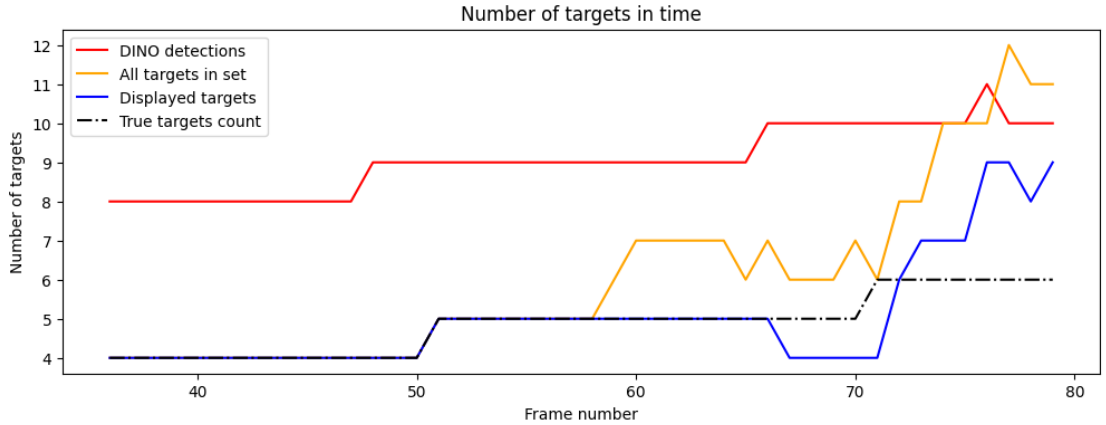
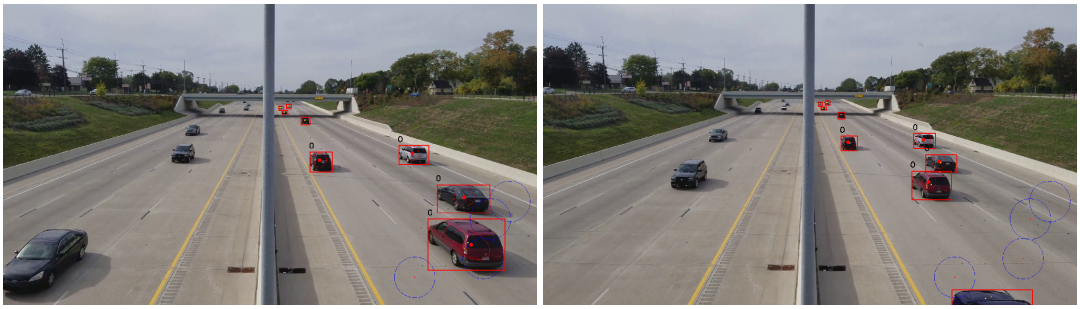


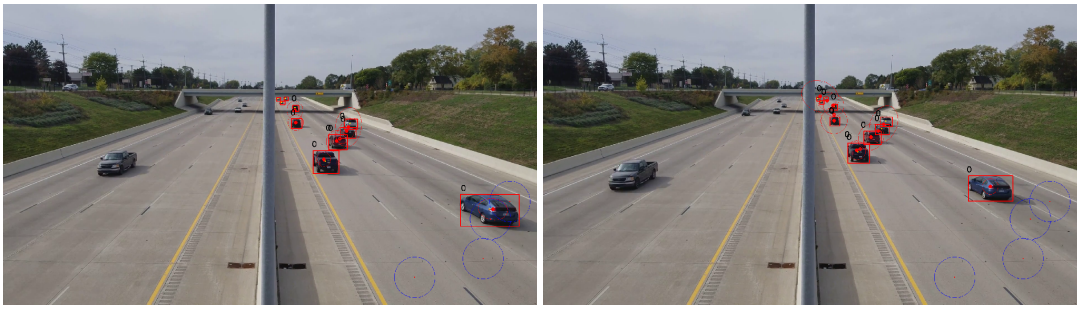
Figure 6.15 Development chart of number of detected targets, targets in filter's queue, displayed targets and true targets' count.



(a) Frame number: 36. **(b)** Frame number: 48.

(c) Frame number: 54. **(d)** Frame number: 58.

(e) Frame number: 67. **(f)** Frame number: 69.



(g) Frame number: 73. **(h)** Frame number: 79.

Figure 6.16 Image sequence of tracked objects using the GM-PHD filter with the dynamic detection probability and the Grounded DINO model.

6.2 E1: paper

6.3 E2: settings comparison

6.4 E3: add obstacle, compare with PHD

6.5 E4: comparison of mean vector

6.6 E5: comparison of covariance

Bibliography

[1] Y. Bar-Shalom and X. Li, *Multitarget-multisensor Tracking: Principles and Techniques*. Yaakov Bar-Shalom, 1995, ISBN: 9780964831209.

[2] B. A., *Kalman Filter from the ground up*. Alex Becker, 2023, ISBN: 978-965-93120-1-6.

[3] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002. DOI: 10.1109/78.978374.

[4] B. E., *Fundamentals of Sensor Fusion*. Norwegian University of Science and Technology, 2020.

[5] R. Mahler, "Multitarget bayes filtering via first-order multitarget moments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, 2003. DOI: 10.1109/TAES.2003.1261119.

[6] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[7] G. Hendeby and R. Karlsson, "Gaussian mixture phd filtering with variable probability of detection," in *17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1–7.

[8] R. P. S. Mahler, B.-T. Vo, and B.-N. Vo, "Cphd filtering with unknown clutter rate and detection profile," *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3497–3513, 2011. DOI: 10.1109/TSP.2011.2128316.

[9] B.-T. Vo, B.-N. Vo, R. Hoseinnezhad, and R. P. S. Mahler, "Robust multi-bernoulli filtering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 399–409, 2013. DOI: 10.1109/JSTSP.2013.2252325.

[10] G. Li, L. Kong, W. Yi, and X. Li, "Robust poisson multi-bernoulli mixture filter with unknown detection probability," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 886–899, 2021. DOI: 10.1109/TVT.2020.3047107.

[11] C. Li, W. Wang, T. Kirubarajan, J. Sun, and P. Lei, "Phd and cphd filtering with unknown detection probability," *IEEE Transactions on Signal Processing*, vol. 66, no. 14, pp. 3784–3798, 2018. DOI: 10.1109/TSP.2018.2835398.

[12] J. Wei, F. Luo, J. Qi, and L. Ruan, "A modified bgm-phd filter with unknown detection probability," in *2023 6th International Conference on Information Communication and Signal Processing (ICICSP)*, 2023, pp. 492–496. DOI: 10.1109/ICICSP59554.2023.10390615.

[13] E. Hanusa and D. W. Krout, "Track state augmentation for estimation of probability of detection in multistatic sonar data," in *2013 Asilomar Conference on Signals, Systems and Computers*, 2013, pp. 1733–1737. DOI: 10.1109/ACSSC.2013.6810598.

[14] S. Horn, "Near real time estimation of surveillance gaps," in *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 1871–1877.

[15] S. Wei, B. Zhang, and W. Yi, "Trajectory phd and cphd filters with unknown detection profile," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8042–8058, 2022. DOI: 10.1109/TVT.2022.3174055.

[16] S. Yan, Y. Fu, W. Zhang, W. Yang, R. Yu, and F. Zhang, "Multi-target instance segmentation and tracking using yolov8 and bot-sort for video sar," in *2023 5th International Conference on Electronic Engineering and Informatics (EEI)*, 2023, pp. 506–510. DOI: 10.1109/EEI59236.2023.10212903.

[17] Á. F. García-Fernández, J. L. Williams, K. Granström, and L. Svensson, "Poisson multi-bernoulli mixture filter: Direct derivation and implementation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 4, pp. 1883–1901, 2018. DOI: 10.1109/TAES.2018.2805153.

[18] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006. DOI: 10.1109/TSP.2006.881190.

[19] Y. Bulut, D. Vines-Cavanaugh, and D. Bernal, "Process and measurement noise estimation for kalman filtering," *Conference Proceedings of the Society for Experimental Mechanics Series*, vol. 3, May 2011. DOI: 10.1007/978-1-4419-9834-7_36.

[20] U. Hadar and H. messer, "High-order hidden markov models - estimation and implementation," in *2009 IEEE/SP 15th Workshop on Statistical Signal Processing*, 2009, pp. 249–252. DOI: 10.1109/SSP.2009.5278591.

[21] K. Dedecius and P. M. Djurić, "Sequential estimation and diffusion of information over networks: A bayesian approach with exponential family of distributions," *IEEE Transactions on Signal Processing*, vol. 65, no. 7, pp. 1795–1809, 2017. DOI: 10.1109/TSP.2016.2641380.

[22] Y. Bar-Shalom, F. Daum, and J. Huang, "The probabilistic data association filter," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 82–100, 2009.

[23] Z. Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1805–1819, 2005. DOI: 10.1109/TPAMI.2005.223.

[24] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002. DOI: 10.1109/78.978396.

[25] A. Doucet, N. Gordon, and V. Krishnamurthy, "Particle filters for state estimation of jump markov linear systems," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 613–624, 2001. DOI: 10.1109/78.905890.

[26] I. Goodman, R. Mahler, and H. Nguyen, *Mathematics of Data Fusion*, ser. Theory and Decision Library B. Springer Netherlands, 1997, ISBN: 9780792346746.

[27] B.-N. Vo, S. Singh, and A. Doucet, "Random finite sets and sequential monte carlo methods in multi-target tracking," in *2003 Proceedings of the International Conference on Radar (IEEE Cat. No.03EX695)*, 2003, pp. 486–491. DOI: 10.1109/RADAR.2003.1278790.

[28] D. Daley and D. Vere-Jones, *An introduction to the theory of point processes*. Springer, Nov. 2003.

[29] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, ser. Mathematics in science and engineering. Academic Press, 1988, ISBN: 9780120797608.

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

[31] Y. Li, X. Xu, N. Mu, and L. Chen, "Eye-gaze tracking system by haar cascade classifier," in *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, 2016, pp. 564–567. DOI: 10.1109/ICIEA.2016.7603648.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "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.

[33] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.01.135.

[34] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[35] *What is image segmentation?* [Online]. Available: https://www.ibm.com/topics/image-segmentation (visited on 05/04/2024).

[36] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[37] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[38] R. Mohan and A. Valada, "Efficientps: Efficient panoptic segmentation," *International Journal of Computer Vision (IJCV)*, 2021.

[39] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao, "Learning two-view correspondences and geometry using order-aware network," *International Conference on Computer Vision (ICCV)*, 2019.

[40] A. Kirillov, R. Girshick, K. He, and P. Dollár, *Panoptic feature pyramid networks*, 2019. arXiv: 1901.02446 [cs.CV].

[41] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, *Upsnet: A unified panoptic segmentation network*, 2019. arXiv: 1901.03784 [cs.CV].

[42] Y. Yang, H. Li, X. Li, Q. Zhao, J. Wu, and Z. Lin, *Sognet: Scene overlap graph network for panoptic segmentation*, 2019. arXiv: 1911.07527 [cs.CV].

[43] Y. Wu, G. Zhang, Y. Gao, X. Deng, K. Gong, X. Liang, and L. Lin, *Bidirectional graph reasoning network for panoptic segmentation*, 2020. arXiv: 2004.06272 [cs.CV].

[44] H. Sun, C. Li, B. Liu, H. Zheng, D. D. Feng, and S. Wang, *Aunet: Attention-guided dense-upsampling networks for breast mass segmentation in whole mammograms*, 2019. arXiv: 1810.10151 [cs.CV].

[45] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. DOI: 10.1109/TSMC.1979.4310076.

[46] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986. DOI: 10.1109/TPAMI.1986.4767851.

[47] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991. DOI: 10.1109/34.87344.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016. DOI: 10.1109/TPAMI.2015.2437384.

[49] G. Coleman and H. Andrews, "Image segmentation by clustering," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, 1979. DOI: 10.1109/PROC.1979.11327.

[50] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[51] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].

[52] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018. DOI: 10.1109/TPAMI.2017.2699184.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[55] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. arXiv: 1607.06450 [stat.ML].

[56] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[57] A. Rakhlin, O. Shamir, and K. Sridharan, *Making gradient descent optimal for strongly convex stochastic optimization*, 2012. arXiv: 1109.5647 [cs.LG].

[58] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, *Segment anything*, 2023. arXiv: 2304.02643 [cs.CV].

[59] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020. DOI: 10.1109/TPAMI.2018.2858826.

[60] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571. DOI: 10.1109/3DV.2016.79.

[61] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, *Grounded sam: Assembling open-world models for diverse visual tasks*, 2024. arXiv: 2401.14159 [cs.CV].

[62] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, *Grounding dino: Marrying dino with grounded pre-training for open-set object detection*, 2023. arXiv: 2303.05499 [cs.CV].

[63] J. Li, D. Li, C. Xiong, and S. Hoi, *Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation*, 2022. arXiv: 2201.12086 [cs.CV].

[64] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, Y. Guo, and L. Zhang, *Recognize anything: A strong image tagging model*, 2023. arXiv: 2306.03514 [cs.CV].

[65] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2022. arXiv: 2112.10752 [cs.CV].

[66] J. Lin, A. Zeng, H. Wang, L. Zhang, and Y. Li, "One-stage 3d whole-body mesh recovery with component aware transformer," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 21 159–21 168. DOI: 10.1109/CVPR52729.2023.02027.

[67] C. Wu, S. Yin, W. Qi, X. Wang, Z. Tang, and N. Duan, *Visual chatgpt: Talking, drawing and editing with visual foundation models*, 2023. arXiv: 2303.04671 [cs.CV].

[68] X. Huang, Y.-J. Huang, Y. Zhang, W. Tian, R. Feng, Y. Zhang, Y. Xie, Y. Li, and L. Zhang, *Open-set image tagging with multi-grained text supervision*, 2023. arXiv: 2310.15200 [cs.CV].

[69] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, *Voxelnext: Fully sparse voxelnet for 3d object detection and tracking*, 2023. arXiv: 2303.11301 [cs.CV].

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF