Vehicle tracking in single camera systems

Iljazović, Florijan

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:552541

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-06-02**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO–MATEMATIČKI FAKULTET
MATEMATIČKI ODSJEK

Florijan Iljazović

VEHICLE TRACKING IN SINGLE CAMERA SYSTEMS

Diplomski rad

Voditelji rada:
prof.dr.sc. Bojan Basrak
dr.sc. Drago Špoljarić

Zagreb, rujan 2020.

Ovaj diplomski rad obranjen je dana ____________________ pred ispitnim povjerenstvom u sastavu:

1. ____________________________, predsjednik
2. ____________________________, član
3. ____________________________, član

Povjerenstvo je rad ocijenilo ocjenom ________________.

Potpisi članova povjerenstva:

1. ____________________________
2. ____________________________
3. ____________________________

Contents

Introduction **1**

1 Multivariate Gaussian distribution **3**
  1.1 Elementary properties 3
  1.2 Bayes' theorem for Gaussian random vectors 4

2 Basics of Kalman filtering **9**
  2.1 Introduction to object tracking 9
  2.2 Recursive Bayesian solution to object tracking 12
  2.3 Kalman filter 15
  2.4 Derivation of Kalman filtering equations 17
  2.5 Extended Kalman filter 22
  2.6 Unscented Kalman filter 25

3 Monocular 3D object tracking **29**
  3.1 Introduction 29
  3.2 KITTI Vision Benchmark 31
  3.3 Bounding box representation 32
  3.4 Detection 33
  3.5 Tracking 37

4 Implementation and results **39**
  4.1 Detection module 39
  4.2 Tracking module 40
  4.3 KITTI tracking benchmark 42

Bibliography **47**

Introduction

To say that neural networks and deep learning have changed the landscape of many research areas in academia and industry would be an understatement. Applications as diverse as protein folding models ([24]), speech recognition ([12]), natural language processing ([28]) and recommender systems ([29]) are emerging.

However, most dramatic success was found in the field of computer vision with the advent of *convolutional neural networks* ([16]), architecture of which was inspired by visual cortex of living organisms. These networks were used to achieve superior performance in object detection tasks on large image datasets ([15]).

The exciting prospect is to leverage ability of deep learning in the context of autonomous driving. In the standard line of research, complex perception system with expensive hardware (e.g. lidar) is used to provide data (most notably positions of surrounding objects) to object tracking algorithms and finally to vehicle control systems. If deep learning can be used as a perception module via accurate analysis of images obtained from the *single* camera mounted on the vehicle, serious cost reduction of such a product would be achieved.

This work aims to examine such a possibility by combining state of the art monocular detection system with the proven tracking algorithm - **Kalman filter** ([14]) - into multiple object tracking pipeline, and reporting its performance on public KITTI benchmark video sequences.

In the first chapter we review some important properties of multivariate normal distribution, with the stress on the Bayes' theorem for Gaussian densities. We immediately remind ourselves of the version of Bayes' theorem for the case of continuous random variables (similar statement holds for vectors and/or discrete variables):

Proposition 0.0.1 (Bayes' theorem). *Let* $(X, Y) : \Omega \to \mathbb{R}^2$ *be a continuous random vector on the probability space* $(\Omega, \mathcal{F}, \mathbb{P})$*, and* $(x, y) \in \mathbb{R}^2$ *an element in its range.*

Then

$$f_{X|Y}(x|y) = \frac{f_X(x)f_{Y|X}(y|x)}{f_Y(y)} = \frac{f_X(x)f_{Y|X}(y|x)}{\int_x f_{Y|X}(y|x)f_X(x)dx}. \tag{1}$$

Tracking algorithms have firm grounding in recursive reasoning developed on top of Bayes' theorem, which we show in the second chapter after giving some introduction to object tracking problem and terminology. Then we give detailed derivation of Kalman filter and overview of its notable variations, extended Kalman filter and unscented Kalman filter.

In chapter 3 we shortly review state of the art object detection and tracking systems, and present our method. In the final chapter we give notes on the implementation and report results on the KITTI tracking benchmark.

I thank my mentor dr.sc. Drago Špoljarić for his guidance, effort and patience in writing this thesis. I also thank my mother Nevenka for her support during my education.

Chapter 1

Multivariate Gaussian distribution

1.1 Elementary properties

Continuous k-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_k)$ is said to have a multivariate normal distribution if its density at point $\boldsymbol{x} \in \mathbb{R}^k$ is given by:

$$\phi(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{k/2}\sqrt{\det \boldsymbol{\Sigma}}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\tau} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right\}, \tag{1.1}$$

for some $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}$ positive definite matrix and $\boldsymbol{\mu} \in \mathbb{R}^k$. Then it can be shown that $\mathbb{E}(\boldsymbol{X}) = \boldsymbol{\mu}$ and $\text{cov}(\boldsymbol{X}) = \mathbb{E}((\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^{\tau}) = \boldsymbol{\Sigma}$, i.e. matrix $\boldsymbol{\Sigma}$ and vector $\boldsymbol{\mu}$ are covariance matrix and expectation of the random vector $\boldsymbol{X}$, respectively. When $\boldsymbol{X}$ is distributed according to (1.1), we write:

$$\boldsymbol{X} \sim N_k(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

and call it *k-dimensional Gaussian random vector* $\boldsymbol{X}$. We often write $N(\boldsymbol{\mu}_{\boldsymbol{X}}, \boldsymbol{\Sigma}_{\boldsymbol{X}})$ to further specify the parameters and ignore dimensionality. This is so-called non-degenerate case of multivariate normal distribution, degenerate case with $\boldsymbol{\Sigma}$ being singular is of no importance in this work.

It is convenient to observe for future considerations the form of the logarithm of the density from (1.1):

$$\ln \phi(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\tau} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) + \text{const} =$$

$$= -\frac{1}{2}\boldsymbol{x}^{\tau} \boldsymbol{\Sigma}^{-1} \boldsymbol{x} + \boldsymbol{x}^{\tau} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \text{const}. \tag{1.2}$$

where we denote with *const* those terms that are independent of $\boldsymbol{x}$. Therefore, when the logarithm of the density of the given random vector has the form (1.2), one can safely conclude that vector is Gaussian and infer the parameters.

Well known property of the multivariate Gaussian distribution is the following:

Theorem 1.1.1 (**Conditional and marginal Gaussian distributions**). *Let* $\boldsymbol{W} : \Omega \to \mathbb{R}^m$ *and* $\boldsymbol{Z} : \Omega \to \mathbb{R}^n$ *be jointly Gaussian random vectors on the probability space* $(\Omega, \mathcal{F}, \mathbb{P})$*, with joint distribution given by:*

$$\begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{Z} \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \right). \tag{1.3}$$

Marginal distributions of vectors $\boldsymbol{W}$ *and* $\boldsymbol{Z}$ *are then given by:*

$$\boldsymbol{W} \sim N_m(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}), \tag{1.4}$$

$$\boldsymbol{Z} \sim N_n(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}). \tag{1.5}$$

Furthermore, let us denote with $\boldsymbol{\Lambda}$ *the partitioned matrix obtained by inverting the covariance matrix* $\boldsymbol{\Sigma}$ *from (1.3):*

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}^{-1}.$$

Conditional distribution of $\boldsymbol{W}$ *given* $\boldsymbol{Z} = \boldsymbol{z}$ *is then given by:*

$$\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}} \sim N_m\big(\boldsymbol{\mu}_1 - \boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}(\boldsymbol{z} - \boldsymbol{\mu}_2), \quad \boldsymbol{\Lambda}_{11}^{-1}\big). \tag{1.6}$$

Analogous result is obtained for conditional distribution of $\boldsymbol{Z}$ *given* $\boldsymbol{W} = \boldsymbol{w}$*.*

Complete derivation can be found in e.g. [3], but it is a standard topic in the M.S. level statistics courses as well.

1.2 Bayes' theorem for Gaussian random vectors

In this section we present the important result that will play the central role in our treatment of Kalman filter in the next chapter. It is in fact a variation of the Bayes' theorem 0.0.1, adapted for the special case of Gaussian densities.

First we state without a proof a formula for the inverse of partitioned matrix (see e.g. [2]), which we employ in the proof of the subsequent theorem:

Lemma 1.2.1 (Inverse of a partitioned matrix)**.** *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$*,* $\boldsymbol{B} \in \mathbb{R}^{n\times m}$*,* $\boldsymbol{C} \in \mathbb{R}^{m\times n}$ *and* $\boldsymbol{D} \in \mathbb{R}^{m\times m}$ *such that* $\boldsymbol{D}$ *and* $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ *are regular. Then*

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{M}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{M}\boldsymbol{B}\boldsymbol{D}^{-1} \end{pmatrix}, \tag{1.7}$$

where it was abbreviated:

$$\boldsymbol{M} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}. \tag{1.8}$$

Matrix $\boldsymbol{M}^{-1}$ *is known as the Schur complement of the partitioned matrix on the left-hand side in (1.7), with respect to the submatrix* $\boldsymbol{D}$*.*

Theorem 1.2.2 (**Bayes' theorem for Gaussian vectors**)**.** *Let* $\boldsymbol{W} : \Omega \to \mathbb{R}^m$ *and* $\boldsymbol{Z} : \Omega \to \mathbb{R}^n$ *be random vectors on probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ *such that:*

$$\boldsymbol{W} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1.9}$$

$$\boldsymbol{Z}|_{\boldsymbol{W}=\boldsymbol{w}} \sim N(\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}, \boldsymbol{\Lambda}), \tag{1.10}$$

where $\boldsymbol{\mu}, \boldsymbol{w} \in \mathbb{R}^m$*,* $\boldsymbol{b} \in \mathbb{R}^n$*,* $\boldsymbol{A} \in \mathbb{R}^{n\times m}$*,* $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times m}$ *and* $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$*. Then, distribution of* $\boldsymbol{Z}$ *is given by:*

$$\boldsymbol{Z} \sim N(\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}, \boldsymbol{\Lambda} + \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\tau}) \tag{1.11}$$

and conditional distribution of $\boldsymbol{W}$ *given* $\boldsymbol{z}$ *is:*

$$\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}} \sim N\big(\boldsymbol{\Pi}\left(\boldsymbol{A}^{\tau}\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{b}) + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right), \boldsymbol{\Pi}\big), \tag{1.12}$$

where the matrix $\boldsymbol{\Pi} \in \mathbb{R}^{m\times m}$ *is given by:*

$$\boldsymbol{\Pi} = \left(\boldsymbol{\Sigma}^{-1} + \boldsymbol{A}^{\tau}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\right)^{-1}. \tag{1.13}$$

Proof. Define random vector $\boldsymbol{U} : \Omega \to \mathbb{R}^{m+n}$

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{W} \\ \boldsymbol{Z} \end{pmatrix}$$

and notice that its density at some point $\boldsymbol{u} = \begin{pmatrix} \boldsymbol{w} & \boldsymbol{z} \end{pmatrix}^{\tau}$ can be written as:

$$f_{\boldsymbol{U}}(\boldsymbol{u}) = f_{\boldsymbol{W}}(\boldsymbol{w}) f_{\boldsymbol{Z}|\boldsymbol{W}}(\boldsymbol{z}|\boldsymbol{w}).$$

We now examine the logarithm of this density, taking into account the assumptions (1.9) and (1.10):

$$\begin{aligned}
\ln f_{\boldsymbol{U}}(\boldsymbol{u}) &= \ln f_{\boldsymbol{W}}(\boldsymbol{w}) + \ln f_{\boldsymbol{Z}|\boldsymbol{W}}(\boldsymbol{z}|\boldsymbol{w}) = \\
&= \ln \phi(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) + \ln \phi(\boldsymbol{z}|\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}, \boldsymbol{\Lambda}) \\
&= -\frac{1}{2}(\boldsymbol{w} - \boldsymbol{\mu})^{\tau}\boldsymbol{\Sigma}^{-1}(\boldsymbol{w} - \boldsymbol{\mu}) - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{w} - \boldsymbol{b})^{\tau}\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}) + \text{const} = \\
&= -\frac{1}{2}\boldsymbol{w}^{\tau}\boldsymbol{\Sigma}^{-1}\boldsymbol{w} + \boldsymbol{w}^{\tau}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{w})^{\tau}\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{w}) + \\
&\qquad + (\boldsymbol{z} - \boldsymbol{A}\boldsymbol{w})^{\tau}\boldsymbol{\Lambda}^{-1}\boldsymbol{b} + \text{const}.
\end{aligned} \tag{1.14}$$

Notice that in the last equality we used (1.2) to split the two exponents.

We proceed by grouping the expression (1.14) into linear and quadratic terms (with respect to variables $\boldsymbol{w}$ and $\boldsymbol{z}$). First order terms read as:

$$\begin{aligned}&\boldsymbol{w}^\tau\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}+\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}-\boldsymbol{w}^\tau\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}=\\ &\qquad=\boldsymbol{w}^\tau\big(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}-\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}\big)+\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}=\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{z}\end{pmatrix}^\tau\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}-\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}\\ \boldsymbol{\Lambda}^{-1}\boldsymbol{b}\end{pmatrix}.\end{aligned}$$

Second order terms are rearranged as:

$$\begin{aligned}&-\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{\Sigma}^{-1}\boldsymbol{w}-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{A}\boldsymbol{w})^\tau\boldsymbol{\Lambda}^{-1}(\boldsymbol{z}-\boldsymbol{A}\boldsymbol{w})\\ &=-\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{\Sigma}^{-1}\boldsymbol{w}-\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}+\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{w}+\frac{1}{2}(\boldsymbol{A}\boldsymbol{w})^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}-\frac{1}{2}(\boldsymbol{A}\boldsymbol{w})^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{w}=\\ &=-\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{\Sigma}^{-1}\boldsymbol{w}-\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}+\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{w}+\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}-\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{w}=\\ &=-\frac{1}{2}\boldsymbol{w}^\tau\big(\boldsymbol{\Sigma}^{-1}+\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\big)\boldsymbol{w}-\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}+\frac{1}{2}\boldsymbol{z}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{w}+\frac{1}{2}\boldsymbol{w}^\tau\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{z}=\\ &=-\frac{1}{2}\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{z}\end{pmatrix}^\tau\begin{pmatrix}\boldsymbol{\Sigma}^{-1}+\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A} & -\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\\ -\boldsymbol{\Lambda}^{-1}\boldsymbol{A} & \boldsymbol{\Lambda}^{-1}\end{pmatrix}\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{z}\end{pmatrix}.\end{aligned}$$

Substituting these findings back into (1.14) while introducing following notation:

$$\boldsymbol{C}=\begin{pmatrix}\boldsymbol{\Sigma}^{-1}+\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A} & -\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\\ -\boldsymbol{\Lambda}^{-1}\boldsymbol{A} & \boldsymbol{\Lambda}^{-1}\end{pmatrix}^{-1}, \tag{1.15}$$

$$\boldsymbol{\gamma}=\boldsymbol{C}\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}-\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b}\\ \boldsymbol{\Lambda}^{-1}\boldsymbol{b}\end{pmatrix}, \tag{1.16}$$

leads to:

$$\ln f_{\boldsymbol{U}}(\boldsymbol{u})=-\frac{1}{2}\boldsymbol{u}^\tau\boldsymbol{C}^{-1}\boldsymbol{u}+\boldsymbol{u}^\tau\boldsymbol{C}^{-1}\boldsymbol{\gamma}+\text{const.}$$

Comparing this with the earlier result (1.2), we find:

$$\boldsymbol{U}=\begin{pmatrix}\boldsymbol{W}\\ \boldsymbol{Z}\end{pmatrix}\sim N(\boldsymbol{\gamma},\boldsymbol{C}). \tag{1.17}$$

We should clarify expressions for $\boldsymbol{C}$ and $\boldsymbol{\gamma}$ further. To calculate $\boldsymbol{C}$, we use lemma 1.2.1. First evaluate the corresponding Schur complement by substituting the appropriate values in (1.8):

$$(\boldsymbol{\Sigma}^{-1}+\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}-\big(-\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\big)\boldsymbol{\Lambda}\big(-\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\big))^{-1}=\boldsymbol{\Sigma}.$$

Using (1.7) we then obtain:

$$\operatorname{cov}(\boldsymbol{U}) = \boldsymbol{C} = \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{A}^\tau \\ \boldsymbol{A}\boldsymbol{\Sigma} & \boldsymbol{\Lambda} + \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\tau \end{pmatrix}. \tag{1.18}$$

Next, performing the multiplication in (1.16) now gives:

$$\mathbb{E}(\boldsymbol{U}) = \boldsymbol{\gamma} = \begin{pmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\boldsymbol{A}^\tau \\ \boldsymbol{A}\boldsymbol{\Sigma} & \boldsymbol{\Lambda} + \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\tau \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - \boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{b} \\ \boldsymbol{\Lambda}^{-1}\boldsymbol{b} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b} \end{pmatrix}. \tag{1.19}$$

Having in mind (1.17), and the fact that we just calculated the parameters of the distribution, we see that we can use theorem 1.1.1 to reach conclusions on the distributions of marginal $\boldsymbol{Z}$ and conditional $\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}}$. Namely, statement (1.11) follows directly from (1.5) using (1.18) and (1.19) because submatrix $\boldsymbol{\Lambda}+\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^\tau$ corresponds to $\operatorname{cov}(\boldsymbol{Z})$) and vector $\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b}$ to $\mathbb{E}(\boldsymbol{Z})$.

Statement (1.12) is verified similarly. From (1.6) it follows that $\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}}$ is normally distributed, with the covariance matrix (denoted with $\boldsymbol{\Pi}$) being the inverse of the first diagonal block in the inverted matrix $\boldsymbol{C}$. The required inverse is then found in equation (1.15):

$$\boldsymbol{\Pi} := \operatorname{cov}(\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}}) = \big(\boldsymbol{\Sigma}^{-1} + \boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\big)^{-1}.$$

Hence (1.13) holds. It is left to justify the mean from (1.12). We use the expression for the mean from (1.6) in theorem 1.1.1, again substituting the required matrix terms from (1.15):

$$\begin{aligned}
\mathbb{E}(\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{z}}) &= \boldsymbol{\mu} - \boldsymbol{\Pi}\big(-\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\big)(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b}) = \\
&= \boldsymbol{\Pi}\boldsymbol{\Pi}^{-1}\boldsymbol{\mu} + \boldsymbol{\Pi}\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{A}\boldsymbol{\mu} - \boldsymbol{b}) = \\
&= \boldsymbol{\Pi}\left(\underbrace{\boldsymbol{\Pi}^{-1}\boldsymbol{\mu} - \boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}\boldsymbol{A}\boldsymbol{\mu}}_{=\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}} + \boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{b})\right) = \\
&= \boldsymbol{\Pi}\big(\boldsymbol{A}^\tau\boldsymbol{\Lambda}^{-1}(\boldsymbol{z} - \boldsymbol{b}) + \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\big),
\end{aligned}$$

which finishes the proof. □

Chapter 2

Basics of Kalman filtering

2.1 Introduction to object tracking

In the problem of object tracking, the aim is to estimate dynamic properties of the object, such as its position, speed etc., cumulatively denoted by a vector $\boldsymbol{x}_k \in \mathbb{R}^{d_x}$, as it evolves through time. Time is considered discrete, represented as $k \in \mathbb{N}$. Vector $\boldsymbol{x}_k$ is called **object state**.

The primary means in doing so are **measurements** or **observations**, denoted by $\boldsymbol{y}_k \in \mathbb{R}^{d_y}$, along with any information (or assumption) on the object dynamics by which its state propagates through time, and the relationship between measurements and the object state.

For example, in a very simplistic vehicle tracking scenario, we might declare the object state at time k to be $\boldsymbol{x}_k = [\boldsymbol{r}_k, \dot{\boldsymbol{r}}_k]^\tau$, where $\boldsymbol{r}_k = [x_k, y_k]^\tau \in \mathbb{R}^2$ is the vehicle center position at time k (assuming flat earth with all objects grounded at $z = 0$), and $\dot{\boldsymbol{r}}_k = [\dot{x}_k, \dot{y}_k]^\tau$ are velocities in respective directions. Then the object dynamics is captured as follows:

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} + T\dot{\boldsymbol{r}}_{k-1},$$

where $T = t_k - t_{k-1}$ is the time interval between measurements, assumed to be constant. Velocity is assumed constant, $\dot{\boldsymbol{r}}_k = \dot{\boldsymbol{r}}_{k-1}$. Furthermore, we imagine having a sensor of some kind, providing us with successive measurements of the object position $\boldsymbol{y}_k = [x_k, y_k]^\tau$. Assuming perfect sensor, we then expect that sensor measurements at time k correspond exactly to the object state component $\boldsymbol{r}_k$. All of this can be summarized in the form of the **object dynamics equation** and **measurement**

equation, as follows:

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1}, \tag{2.1}$$

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k, \tag{2.2}$$

where we set a **transition matrix**

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and **measurement matrix**

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

It is highly unlikely that the measurements of a given sensor are perfect. *Sensor noise* is a major source of uncertainty in the problem of object tracking. In this work, the sensor is in fact a camera, and the measurements (e.g. the coordinates of the center of the vehicle) are inferred from monocular images. Camera is mounted on the ego-vehicle, the movement of which could be very challenging. Furthermore, it is well known that inference of depth from a single monocular image is in itself already a demanding task. Thus it seems natural to introduce some randomness in equations (2.1) and (2.2), and to make object state $\boldsymbol{x}_k$ and observations $\boldsymbol{y}_k$ random variables, instead of deterministic quantities.

Therefore, we impose a probabilistic model on the problem as follows. The notation follows that of [6], with minor changes that hopefully contribute to clarity. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and $d_x, d_y \in \mathbb{N}$. We define a stochastic process of the object state $(\boldsymbol{X}_k)_{k\in\mathbb{N}}$ with $\boldsymbol{X}_k : \Omega \to \mathbb{R}^{d_x}$. Similarly, observation (measurement) process is defined as $(\boldsymbol{Y}_k)_{k\in\mathbb{N}}$, $\boldsymbol{Y}_k : \Omega \to \mathbb{R}^{d_y}$. We also introduce a serious restriction so that both the object state and measurement process random vectors are continuous, and that joint distribution of states and observations, until any moment in time k, is continuous. We denote this joint distribution density simply with f, and time moment k is implied from the arguments in the expression:

$$f(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{y}_1, \dots, \boldsymbol{y}_k). \tag{2.3}$$

Sometimes this is called the *complete-data density function.* Because of the complete-data distribution being continuous, any marginal distribution is continuous as well, and can be obtained by integrating over (2.3). We use f for any marginal distribution density with respect to the above complete-data distribution, when it contains at least one state and one observation vector. E.g. we would write:

$$f(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{y}_k) = \int_{\boldsymbol{y}_1} \cdots \int_{\boldsymbol{y}_{k-1}} f(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{y}_1, \dots, \boldsymbol{y}_k) d\boldsymbol{y}_1 \dots d\boldsymbol{y}_{k-1}.$$

Marginal distribution densities of state vectors are in turn denoted with p, and q is used for observation vector densities. We give two notable examples:

$$\begin{aligned} p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) &= \int_{\boldsymbol{y}_1} \cdots \int_{\boldsymbol{y}_k} f(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{y}_1, \dots, \boldsymbol{y}_k) d\boldsymbol{y}_1 \dots d\boldsymbol{y}_k, \\ q(\boldsymbol{y}_1, \dots, \boldsymbol{y}_k) &= \int_{\boldsymbol{x}_1} \cdots \int_{\boldsymbol{x}_k} f(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k, \boldsymbol{y}_1, \dots, \boldsymbol{y}_k) d\boldsymbol{x}_1 \dots d\boldsymbol{x}_k. \end{aligned}$$

A crucial task in probabilistic modelling of tracking problems is to translate the object dynamics equation (e.g. (2.1)) and measurement equation (e.g. (2.2)) into corresponding conditional densities:

- $\boldsymbol{x}_k \mapsto p(\boldsymbol{x}_k | \boldsymbol{x}_{k-1})$ for given $\boldsymbol{x}_{k-1} \in \mathbb{R}^{d_x}$ is called **transition density**,
- $\boldsymbol{y}_k \mapsto q(\boldsymbol{y}_k | \boldsymbol{x}_k)$ for given $\boldsymbol{x}_k \in \mathbb{R}^{d_x}$ is called **likelihood function**, also known as *emission density.*

This is done by analyzing the underlying dynamical system and its sources and types of uncertainty.

Another conditional density of central importance in future considerations is the following:

$$p(\boldsymbol{x}_k | \boldsymbol{y}_1, \dots, \boldsymbol{y}_k) = \frac{f(\boldsymbol{x}_k, \boldsymbol{y}_1, \dots, \boldsymbol{y}_k)}{q(\boldsymbol{y}_1, \dots, \boldsymbol{y}_k)}. \tag{2.4}$$

This is conditional density function of the k-th object state vector, conditioned on the first k observations $\boldsymbol{Y}_1 = \boldsymbol{y}_1, \dots, \boldsymbol{Y}_k = \boldsymbol{y}_k$. It captures our knowledge on the object state at time k, having received k-th observation $\boldsymbol{y}_k$. In *Bayesian* framework, it is called the **posterior state density**, and represents our primary analysis target. The corresponding random vector is denoted

$$\boldsymbol{X}_k |_{\boldsymbol{y}_1, \dots, \boldsymbol{y}_k}$$

and similar notational convention for random vectors is used throughout.

As we shall see, Bayesian reasoning lends itself nicely to the problem of tracking. The general objective is the following:

- suppose that at time $k-1$, the posterior density $p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1})$ is known,
- use this as a *prior*, and employ Bayes' theorem to calculate the *posterior*, $p(\boldsymbol{x}_k|\boldsymbol{y}_1,\dots,\boldsymbol{y}_k)$.

This recursive procedure, that we call *recursive Bayesian solution to object tracking*, will be presented in some detail in the following sections.

2.2 Recursive Bayesian solution to object tracking

In this section, we introduce certain assumptions on the complete-data joint distribution from (2.3) in order to develop a recursive solution for posterior state density from (2.4). These assumptions come in the form of conditional independence properties, and seem to reflect many real-world phenomena. They are based on the discussion from [6].

To simplify the notation for the expressions including historical data, we use $\mathbf{X}^k = (\boldsymbol{X}_1,\dots,\boldsymbol{X}_k)$ and $\mathbf{Y}^k = (\boldsymbol{Y}_1,\dots,\boldsymbol{Y}_k)$, for any $k \in \mathbb{N}$.

The assumptions read as follows:

C0 *Causality principle* - past cannot depend on the future:

$$q\big(\mathbf{y}^{k-1}|\mathbf{x}^{k+m}\big) = q\big(\mathbf{y}^{k-1}|\mathbf{x}^{k-1}\big),\ m \in \mathbb{N}_0. \tag{2.5}$$

C1 Current state is conditionally independent of the past data, given the most recent state:

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = p\big(\boldsymbol{x}_k|\boldsymbol{x}_{k-1},\mathbf{y}^{k-1}\big). \tag{2.6}$$

C2 Current observation is conditionally independent of the past data, given the current state:

$$q(\boldsymbol{y}_k|\boldsymbol{x}_k) = q(\boldsymbol{y}_k|\boldsymbol{x}_k,\mathbf{y}^{k-1}). \tag{2.7}$$

CM State process $(\boldsymbol{X}_k)_{k\in\mathbb{N}}$ has *Markov property*, meaning that state at time k does not depend on the previous states at $k-2,\dots,1$, given the last state (time $k-1$). For continuous random vectors, this means that for every $k \geq 2$:

$$p\big(\boldsymbol{x}_k|\mathbf{x}^{k-1}\big) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}). \tag{2.8}$$

As mentioned, interesting recursive relation is now derived for the posterior density $p(\boldsymbol{x}_k|\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ of the state process.

Theorem 2.2.1. *Let the object state process* $(\boldsymbol{X}_k)_{k\in\mathbb{N}}$ *and the measurement process* $(\mathbf{Y}_k)_{k\in\mathbb{N}}$ *for the tracking problem be assembled from continuous random vectors. Then, under assumptions* **C0 - C2** *and that state process is Markov, one has the following recursive relation for the posterior state density:*

$$p\big(\boldsymbol{x}_k|\mathbf{y}^k\big) = \frac{q(\boldsymbol{y}_k|\boldsymbol{x}_k)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1})} \int_{\boldsymbol{x}_{k-1}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p\big(\boldsymbol{x}_{k-1}|\mathbf{y}^{k-1}\big) d\boldsymbol{x}_{k-1}. \tag{2.9}$$

Proof. We take a step back from the posterior density required in (2.9) and first take a look at complete conditional density of the object state variables $p\big(\mathbf{x}^k|\mathbf{y}^k\big)$. Using Bayes' formula from proposition (0.0.1):

$$p\big(\mathbf{x}^k|\mathbf{y}^k\big) = \frac{q\big(\mathbf{y}^k|\mathbf{x}^k\big)p\big(\mathbf{x}^k\big)}{q\big(\mathbf{y}^k\big)}. \tag{2.10}$$

We now use causality principle to write the *likelihood* function $q\big(\mathbf{y}^k|\mathbf{x}^k\big)$ differently:

$$\begin{aligned} q\big(\mathbf{y}^k|\mathbf{x}^k\big) &= q\big(\boldsymbol{y}_k, \mathbf{y}^{k-1}|\mathbf{x}^k\big) \\ &= q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}, \mathbf{x}^k\big) q\big(\mathbf{y}^{k-1}|\mathbf{x}^k\big) \\ &= q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}, \mathbf{x}^k\big) q\big(\mathbf{y}^{k-1}|\mathbf{x}^{k-1}\big). \end{aligned}$$

In the last equality, (2.5) was used with $m = 0$. We substitute this into (2.10), together with

$$\begin{aligned} p\big(\mathbf{x}^k\big) &= p\big(\boldsymbol{x}_k, \mathbf{x}^{k-1}\big) = p\big(\boldsymbol{x}_k|\mathbf{x}^{k-1}\big)p\big(\mathbf{x}^{k-1}\big), \\ q\big(\mathbf{y}^k\big) &= q\big(\boldsymbol{y}_k, \mathbf{y}^{k-1}\big) = q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}\big)q\big(\mathbf{y}^{k-1}\big). \end{aligned}$$

to find:

$$\begin{aligned} p\big(\mathbf{x}^k|\mathbf{y}^k\big) &= \frac{q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}, \mathbf{x}^k\big) q\big(\mathbf{y}^{k-1}|\mathbf{x}^{k-1}\big) p\big(\boldsymbol{x}_k|\mathbf{x}^{k-1}\big) p\big(\mathbf{x}^{k-1}\big)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1}) q(\mathbf{y}^{k-1})} = \\ &= \frac{q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}, \mathbf{x}^k\big) p\big(\boldsymbol{x}_k|\mathbf{x}^{k-1}\big)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1})} \underbrace{\frac{q\big(\mathbf{y}^{k-1}|\mathbf{x}^{k-1}\big) p\big(\mathbf{x}^{k-1}\big)}{q(\mathbf{y}^{k-1})}}_{=p\left(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\right)} = \\ &= \frac{q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}, \mathbf{x}^k\big) p\big(\boldsymbol{x}_k|\mathbf{x}^{k-1}\big)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1})} p\big(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\big), \end{aligned}$$

where in the last equality we again used Bayes' formula from (1). Continue simplifying this expression by noticing that conditional independence property from (2.7) and Markov property (2.8) give:

$$p\big(\mathbf{x}^k|\mathbf{y}^k\big) = \frac{q(\boldsymbol{y}_k|\boldsymbol{x}_k)}{q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}\big)} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p\big(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\big). \tag{2.11}$$

To obtain the recursion for $p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)$ instead of $p\big(\mathbf{x}^k|\mathbf{y}^k\big)$, we integrate out variables $\boldsymbol{x}_1$ through $\boldsymbol{x}_{k-1}$ from (2.11):

$$\begin{aligned} p\big(\boldsymbol{x}_k|\mathbf{y}^k\big) &= \int_{\mathbf{x}^{k-1}} p\big(\mathbf{x}^k|\mathbf{y}^k\big) d\mathbf{x}^{k-1} = \\ &= \frac{q(\boldsymbol{y}_k|\boldsymbol{x}_k)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1})} \int_{\mathbf{x}^{k-1}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p\big(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\big) d\mathbf{x}^{k-1} = \\ &= \frac{q(\boldsymbol{y}_k|\boldsymbol{x}_k)}{q(\boldsymbol{y}_k|\mathbf{y}^{k-1})} \int_{\boldsymbol{x}_{k-1}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \left(\int_{\mathbf{x}^{k-2}} p\big(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\big) d\mathbf{x}^{k-2} \right) d\boldsymbol{x}_{k-1}. \end{aligned}$$

The inner integral is recognized as:

$$\int_{\mathbf{x}^{k-2}} p\big(\mathbf{x}^{k-1}|\mathbf{y}^{k-1}\big) d\mathbf{x}^{k-2} = \int_{\mathbf{x}^{k-2}} p\big(\boldsymbol{x}_{k-1}, \mathbf{x}^{k-2}|\mathbf{y}^{k-1}\big) d\mathbf{x}^{k-2} = p\big(\boldsymbol{x}_{k-1}|\mathbf{y}^{k-1}\big).$$

Substituting this back gives the recursion from (2.9). □

Observe again the integral in recursion (2.9), and notice that in fact:

$$\int_{\boldsymbol{x}_{k-1}} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p\big(\boldsymbol{x}_{k-1}|\mathbf{y}^{k-1}\big) d\boldsymbol{x}_{k-1} = p\big(\boldsymbol{x}_k|\mathbf{y}^{k-1}\big). \tag{2.12}$$

Now we can shortly summarize how recursion (2.9) is used for real-time tracking applications:

- **Prediction step**. Suppose that at time $k-1$ distribution of vector $\boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}$ is known, i.e. function $\boldsymbol{x}_{k-1} \mapsto p\big(\boldsymbol{x}_{k-1}|\mathbf{y}^{k-1}\big)$ is given. Determine the predicted distribution density $\boldsymbol{x}_k \mapsto p\big(\boldsymbol{x}_k|\mathbf{y}^{k-1}\big)$ by evaluating the integral in (2.12). This is the distribution density of the object state at future time k, given the data up until the time $k-1$.
- **Filtering step**. Upon arrival of new data $\boldsymbol{y}_k$ at time k, determine the posterior distribution density function $\boldsymbol{x}_k \mapsto p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)$ by multiplying $p\big(\boldsymbol{x}_k|\mathbf{y}^{k-1}\big)$ obtained in the prediction step with likelihood function $q(\boldsymbol{y}_k|\boldsymbol{x}_k)$. Quantity $q\big(\boldsymbol{y}_k|\mathbf{y}^{k-1}\big)$ is a normalization factor.

If one needs a point estimate for the state vector at time k, integrate to obtain the expected value of $\boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_k}$ denoted by $\hat{\boldsymbol{x}}_{k|k}$:

$$\hat{\boldsymbol{x}}_{k|k} := \mathbb{E}(\boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_k}) = \int_{\boldsymbol{x}_k} p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)\boldsymbol{x}_k d\boldsymbol{x}_k.$$

In order to gauge the uncertainty of the estimate, covariance matrix is calculated:

$$\boldsymbol{P}_{k|k} := \operatorname{cov}(\boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_k}) = \int_{\boldsymbol{x}_k} p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)\big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}\big)\big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}\big)^{\tau} d\boldsymbol{x}_k.$$

Note that first and second moment of the predicted distribution density $p\big(\boldsymbol{x}_k|\mathbf{y}^{k-1}\big)$ are denoted analogously:

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k-1} &:= \mathbb{E}\big(\boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}}\big),\\ \boldsymbol{P}_{k|k-1} &:= \operatorname{cov}\big(\boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}}\big).\end{aligned}$$

2.3 Kalman filter

In the previous section, we explored a somewhat general case of the tracking problem. Apart from a few natural conditional independence properties, not much was assumed about the distributions of state and observation processes. There are many tracking models, varying greatly in their assumptions on the transition distribution $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$, likelihood function $q(\boldsymbol{y}_k|\boldsymbol{x}_k)$ and treatment of recursion (2.9).

Possibly the most famous one is that of **Kalman filter**, named after one of its main contributors, Rudolf E. Kálmán (see [14]). In addition to the general assumptions from previous section, i.e. **C0**, **C2** and **CM**, Kalman filter makes precise statements on system dynamics and probability distributions.

The dynamical system in question is considered linear, with dynamics and measurement equations given as in section 2.1:

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1}, \tag{2.13}$$

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k. \tag{2.14}$$

Even though these were stated for quite a simple case of $2D$ vehicle tracking, they are in fact very common in practice (for varying choices of $\boldsymbol{x}_k$, $\boldsymbol{y}_k$, $\boldsymbol{F}$ and $\boldsymbol{H}$).

Their probabilistic counterparts are obtained by introducing additive Gaussian noise:

$$\boldsymbol{X}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{v}_k, \tag{2.15}$$

$$\boldsymbol{Y}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{w}_k, \tag{2.16}$$

where

$$\boldsymbol{v}_k \sim N(\boldsymbol{0}, \boldsymbol{Q}_k),$$
$$\boldsymbol{w}_k \sim N(\boldsymbol{0}, \boldsymbol{R}_k).$$

Matrices $\boldsymbol{Q}_k$, $\boldsymbol{R}_k$ are called *process covariance* and *measurement covariance* matrix, respectively. We interpret equations (2.15) and (2.16) conditionally, and introduce the initial state distribution assumption:

K0 Initial state $\boldsymbol{X}_1$ is normally distributed:

$$\boldsymbol{X}_1 \sim N(\hat{\boldsymbol{x}}_1, \boldsymbol{P}_1). \tag{2.17}$$

K1 Transition density of Kalman filter is Gaussian:

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \phi(\boldsymbol{x}_k|\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}_k). \tag{2.18}$$

K2 Likelihood function at time is Gaussian:

$$q(\boldsymbol{y}_k|\boldsymbol{x}_k) = \phi(\boldsymbol{y}_k|\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}_k). \tag{2.19}$$

In filtering literature (see e.g. [6]) one can encounter another assumption - that posterior state density is Gaussian:

KP Posterior state distribution at time k is Gaussian, and we denote the parameters of mean and covariance at time k with $\hat{\boldsymbol{x}}_{k|k}$ and $\boldsymbol{P}_{k|k}$:

$$p(\boldsymbol{x}_k|\boldsymbol{y}_1, \dots, \boldsymbol{y}_k) = \phi\big(\boldsymbol{x}_k|\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}\big). \tag{2.20}$$

However, we show in the next section that (2.20) is in fact emergent phenomenon from assumptions **K0** - **K2**, and therefore doesn't need to be assumed.

Under assumptions **K0** - **K2**, posterior distribution is Gaussian at every time step. Furthermore, predicted distribution (from equation (2.12)) is Gaussian as well (see proposition 2.4.1 from next section). Notice how this simplifies the program presented in previous section - all the distributions are Gaussian, and in essence there is no need for updating them by costly procedures such as integral evaluation in (2.12). One just has to keep track of the mean and covariance. The method of updating these parameters upon arrival of new data is subject of the next section.

2.4 Derivation of Kalman filtering equations

In this section the aim is to derive celebrated Kalman filtering equations. We begin by establishing the distribution of vector

$$\boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}},$$

the predicted object state at time k.

Proposition 2.4.1. *Suppose that at time $k-1$, posterior object state is Gaussian:*

$$\boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}} \sim N(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}). \tag{2.21}$$

Conditioned on the values of observation variables up until the time step $k-1$

$$\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1},$$

predicted object state is normally distributed according to

$$\begin{aligned} \boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}} &\sim N\big(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}\big), && (2.22)\\ \hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1|k-1}, && (2.23)\\ \boldsymbol{P}_{k|k-1} &= \boldsymbol{Q}_k + \boldsymbol{F}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}^{\tau}. && (2.24) \end{aligned}$$

Proof. Define random vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$ as follows:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}},\\ \boldsymbol{Z} &= \boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}. \end{aligned}$$

Vector $\boldsymbol{W}$ is assumed to be normally distributed in (2.21). We determine the density of vector $\boldsymbol{Z}$ conditioned on $\boldsymbol{W} = \boldsymbol{x}_{k-1}$:

$$f_{\boldsymbol{Z}|\boldsymbol{W}}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}). \tag{2.25}$$

First equality holds because both vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$ have conditioning on $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{k-1}$ in their definitions. In the second equality we notice these conditions can be omitted, by invoking a conditional independence property from (2.6).

From (2.25) it now follows that vector $\boldsymbol{Z}$ conditioned on $\boldsymbol{W} = \boldsymbol{x}_{k-1}$ has the same normal distribution as in (2.18), i.e. transition distribution.

Let's summarize our findings on properties of vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$:

$$\begin{aligned} \boldsymbol{W} &\sim N\big(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}\big),\\ \boldsymbol{Z}|_{\boldsymbol{W}=\boldsymbol{x}_{k-1}} &\sim N(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}_k). \end{aligned}$$

The predicted state distribution is now inferred directly from theorem 1.2.2. Namely, distribution of $\boldsymbol{Z} = \boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}$ is specified in (1.11), and we easily recover results (2.23) and (2.24). □

Couple of simple but useful results from matrix algebra will be needed to derive Kalman filtering equations.

Lemma 2.4.2. *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$*,* $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ *and* $\boldsymbol{C} \in \mathbb{R}^{m\times m}$*, such that* $\boldsymbol{A}$*,* $\boldsymbol{C}$*,* $\boldsymbol{A}^{-1} + \boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}$ *and* $\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C}$ *are all regular. Then following identity holds:*

$$\big(\boldsymbol{A}^{-1} + \boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}\big)^{-1}\boldsymbol{B}^{\tau}\boldsymbol{C}^{-1} = \boldsymbol{A}\boldsymbol{B}^{\tau}(\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C})^{-1}. \tag{2.26}$$

Proof. Multiplying left-hand side by $\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C}$ gives:

$$\begin{aligned}
&\big(\boldsymbol{A}^{-1} + \boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}\big)^{-1}\boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}(\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C}) = \\
&= \big(\boldsymbol{A}^{-1} + \boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}\big)^{-1}\big(\boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{B}^{\tau}\big) = \\
&= \big(\boldsymbol{A}^{-1} + \boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B}\big)^{-1}\big(\boldsymbol{B}^{\tau}\boldsymbol{C}^{-1}\boldsymbol{B} + \boldsymbol{A}^{-1}\big)\boldsymbol{A}\boldsymbol{B}^{\tau} = \\
&= \boldsymbol{A}\boldsymbol{B}^{\tau}.
\end{aligned}$$

Multiplying right-hand side by $\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C}$ evidently gives $\boldsymbol{A}\boldsymbol{B}^{\tau}$ as well. Identity (2.26) now holds because matrix $\boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^{\tau} + \boldsymbol{C}$ was assumed regular. □

Lemma 2.4.3 (Woodbury identity)**.** *Let* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$*,* $\boldsymbol{B} \in \mathbb{R}^{n\times m}$*,* $\boldsymbol{C} \in \mathbb{R}^{m\times n}$ *and* $\boldsymbol{D} \in \mathbb{R}^{m\times m}$*, such that* $\boldsymbol{A}$*,* $\boldsymbol{D}$*,* $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ *and* $\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$ *are all regular. Then following identity holds:*

$$\big(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\big)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}. \tag{2.27}$$

Proof. Multiplying right-hand side by $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ gives:

$$\begin{aligned}
&\Big(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\Big)\big(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\big) = \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\big(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\big) = \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)^{-1}\big(\boldsymbol{I} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\big)\boldsymbol{C} = \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{B}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)^{-1}\big(\boldsymbol{D} + \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}\big)\boldsymbol{D}^{-1}\boldsymbol{C} = \\
&= \boldsymbol{I} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} = \\
&= \boldsymbol{I},
\end{aligned}$$

which proves the identity. □

Finally, the following theorem states how to update the parameters of the posterior object state distribution at moment of time $k-1$ with the arrival of the new observation $\boldsymbol{y}_k$ at time k.

Theorem 2.4.4. *Suppose that at time $k-1$, posterior object state is Gaussian:*

$$\boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}} \sim N(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}). \tag{2.28}$$

Posterior object state at time k is then normally distributed as well:

$$\boldsymbol{X}_{k}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k}} \sim N(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}), \tag{2.29}$$

and its mean and covariance obey the following recursive relations:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\big(\boldsymbol{y}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}\big), \tag{2.30}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}, \tag{2.31}$$

where:

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^\tau\big(\boldsymbol{R}_k + \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^\tau\big)^{-1} \tag{2.32}$$

is called Kalman gain matrix. Furthermore, predicted observation vector $\boldsymbol{Y}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}$ is normally distributed as well:

$$q(\boldsymbol{y}_k|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}) = \phi(\boldsymbol{y}_k|\boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{R}_k + \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^\tau). \tag{2.33}$$

Proof. Define random vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$ as follows:

$$\boldsymbol{W} = \boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}},$$
$$\boldsymbol{Z} = \boldsymbol{Y}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}.$$

Vector $\boldsymbol{W}$ is the predicted object state at time k, and its distribution is described in proposition 2.4.1. $\boldsymbol{Z}$ is predicted observation vector at time k. We determine the density of vector $\boldsymbol{Z}$ conditioned on $\boldsymbol{W} = \boldsymbol{x}_k$:

$$f_{\boldsymbol{Z}|\boldsymbol{W}}(\boldsymbol{y}_k|\boldsymbol{x}_k) = q(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}) = q(\boldsymbol{y}_k|\boldsymbol{x}_k). \tag{2.34}$$

First equality holds because both vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$ have conditioning on $\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}$ in their definitions. In the second equality we notice these conditions can be omitted, by invoking a conditional independence property from (2.7).
Noticing that distribution found in (2.34) is exactly the emission distribution from (2.19), we summarize:

$$\boldsymbol{W} \sim N\big(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}\big), \tag{2.35}$$

$$\boldsymbol{Z}|_{\boldsymbol{W}=\boldsymbol{x}_k} \sim N(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}_k), \tag{2.36}$$

to conclude that we are under conditions of theorem 1.2.2. Mean and covariance of $\boldsymbol{Z} = \boldsymbol{Y}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}$, as stated in (2.33), are then easily verified via (1.11).

We now pursue the other consequence of theorem 1.2.2, i.e. results on

$$\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{y}_k} = \boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1},\boldsymbol{y}_k},$$

which is in fact a posterior object state vector at time k. It is normally distributed, as acknowledged in (2.29):

$$\boldsymbol{W}|_{\boldsymbol{Z}=\boldsymbol{y}_k} \sim N(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}).$$

Comparing this with (1.12) and (1.13) combined with (2.35) and (2.36) gives:

$$\boldsymbol{P}_{k|k} = \boldsymbol{\Pi} = \left(\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{H}\right)^{-1}, \tag{2.37}$$

$$\hat{\boldsymbol{x}}_{k|k} = \boldsymbol{P}_{k|k} \left(\boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{y}_k + \boldsymbol{P}_{k|k-1}^{-1} \hat{\boldsymbol{x}}_{k|k-1}\right). \tag{2.38}$$

To continue, first apply the matrix identity (2.27) from lemma 2.4.3 to (2.37) to derive (2.31):

$$\begin{aligned}
\boldsymbol{P}_{k|k} = \boldsymbol{\Pi} &= \left(\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{H}\right)^{-1} = \\
&\overset{(2.27)}{=} \boldsymbol{P}_{k|k-1} - \underbrace{\boldsymbol{P}_{k|k-1} \boldsymbol{H}^\tau \big(\boldsymbol{R}_k + \boldsymbol{H} \boldsymbol{P}_{k|k-1} \boldsymbol{H}^\tau\big)^{-1}}_{=\boldsymbol{K}_k} \boldsymbol{H} \boldsymbol{P}_{k|k-1} = \\
&= \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k \boldsymbol{H} \boldsymbol{P}_{k|k-1} \\
&= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_{k|k-1}.
\end{aligned}$$

Now expand (2.38) using both expressions for $\boldsymbol{P}_{k|k}$:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k|k} &= \boldsymbol{P}_{k|k} \left(\boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{y}_k + \boldsymbol{P}_{k|k-1}^{-1} \hat{\boldsymbol{x}}_{k|k-1}\right) = \\
&= \left(\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{y}_k + (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_{k|k-1} \boldsymbol{P}_{k|k-1}^{-1} \hat{\boldsymbol{x}}_{k|k-1} = \\
&= \left(\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{H}\right)^{-1} \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{y}_k + (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \hat{\boldsymbol{x}}_{k|k-1}.
\end{aligned}$$

For the second addend, the Kalman gain matrix expression from (2.31) was used, but for the first addend we employ the inverse form in (2.37) in order to apply matrix identity (2.26) from lemma 2.4.2:

$$\begin{aligned}
(\boldsymbol{P}_{k|k-1}^{-1} &+ \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} \boldsymbol{H})^{-1} \boldsymbol{H}^\tau \boldsymbol{R}_k^{-1} = \\
&\overset{(2.26)}{=} \boldsymbol{P}_{k|k-1} \boldsymbol{H}^\tau \big(\boldsymbol{H} \boldsymbol{P}_{k|k-1} \boldsymbol{H}^\tau + \boldsymbol{R}_k\big)^{-1} = \\
&\overset{(2.32)}{=} \boldsymbol{K}_k.
\end{aligned}$$

Now look back to conclude:

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k} &= \boldsymbol{K}_k \boldsymbol{y}_k + (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \hat{\boldsymbol{x}}_{k|k-1} = \\ &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \big(\boldsymbol{y}_k - \boldsymbol{H} \hat{\boldsymbol{x}}_{k|k-1} \big),\end{aligned}$$

as stands in (2.30) of the theorem statement. □

As noted in section 2.1, posterior object state density at time k, i.e. distribution of random vector $\boldsymbol{X}_k|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_k}$, is of the greatest importance in real-time tracking applications. It captures total available information on object state at time k, and should be continuously updated as new sensory data arrives.

In Kalman filter tracking algorithm, this updating is done via equations (2.30) and (2.31), stated and derived in previous theorem, which we repeat here for convenience:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k \big(\boldsymbol{y}_k - \boldsymbol{H} \hat{\boldsymbol{x}}_{k|k-1} \big), \tag{2.39}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_{k|k-1}. \tag{2.40}$$

Together with connections (2.23) and (2.24) from proposition 2.4.1 on predicted state density:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F} \hat{\boldsymbol{x}}_{k-1|k-1}, \tag{2.41}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{Q}_k + \boldsymbol{F} \boldsymbol{P}_{k-1|k-1} \boldsymbol{F}^{\tau}, \tag{2.42}$$

these equations are called **Kalman filtering equations**. The first recursion concerns the mean of the multivariate normal distribution of the posterior state, and the second its covariance matrix.

Suppose that at time $k-1$, distribution of $\boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{k-1}}$ is known, meaning that $\hat{\boldsymbol{x}}_{k-1|k-1}$ and $\boldsymbol{P}_{k-1|k-1}$ are given. The object state ($\hat{\boldsymbol{x}}_{k-1|k-1}$) and its uncertainty ($\boldsymbol{P}_{k-1|k-1}$) are then propagated through time using object dynamics assumptions. This is done in (2.41) and (2.42), giving $\hat{\boldsymbol{x}}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$. These predictions are then updated in Bayesian fashion with new evidence in eqs. (2.39) and (2.40).

Observe the mechanism of the updating procedure in (2.39). Predicted state $\hat{\boldsymbol{x}}_{k|k-1}$ is corrected for the quantity proportional to the *error* between currently observed $\boldsymbol{y}_k$ and predicted observation (from measurement process assumptions of Kalman filter) $\boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}$. The coefficient of proportionality is given by Kalman gain matrix $\boldsymbol{K}_k$.

Let us revisit the redundancy of assumption (2.20). This requires inspecting the initial conditions in Kalman filtering equations. Using (2.17) instead of predicted state distribution, we can replicate verbatim the proof of the theorem 2.4.4, and conclude that posterior distribution at every step is indeed Gaussian. More specifically, we set

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{X}_1, \\ \boldsymbol{Z} &= \boldsymbol{Y}_1, \end{aligned}$$

repeat the proof and arrive at following formulas, which are analogous to (2.30), (2.31), (2.32) and (2.33) but have $\hat{\boldsymbol{x}}_1$ and $\boldsymbol{P}_1$ in place of $\hat{\boldsymbol{x}}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$:

$$\begin{aligned} \hat{\boldsymbol{x}}_{1|1} &= \hat{\boldsymbol{x}}_1 + \boldsymbol{K}_1(\boldsymbol{y}_1 - \boldsymbol{H}\hat{\boldsymbol{x}}_1), \\ \boldsymbol{P}_{1|1} &= (\boldsymbol{I} - \boldsymbol{K}_1\boldsymbol{H})\boldsymbol{P}_1, \\ \boldsymbol{K}_1 &= \boldsymbol{P}_1\boldsymbol{H}^\tau(\boldsymbol{R}_1 + \boldsymbol{H}\boldsymbol{P}_1\boldsymbol{H}^\tau)^{-1}, \\ q(\boldsymbol{y}_1) &= \phi(\boldsymbol{y}_1|\boldsymbol{H}\hat{\boldsymbol{x}}_1, \boldsymbol{R}_1 + \boldsymbol{H}\boldsymbol{P}_1\boldsymbol{H}^\tau). \end{aligned}$$

2.5 Extended Kalman filter

In this section we shortly describe another famous and widely used filter - **extended Kalman filter**. It is an elegant application of ideas from standard Kalman filtering to the case of *nonlinear* systems, prevalent in many engineering fields. Nonlinearity here is in terms of object dynamics and measurement equations - their linear forms (2.13) and (2.14) are no longer suitable. We keep other assumptions of Kalman filter intact.

We begin by reminding ourselves of the first order approximation of multivariable scalar valued functions due to Taylor's theorem. Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at point $\boldsymbol{x}_0 \in \mathbb{R}^n$, with the gradient vector $\nabla f(\boldsymbol{x}_0) \in \mathbb{R}^n$. Then for $\boldsymbol{x}$ in vicinity of $\boldsymbol{x}_0$, we approximate:

$$f(\boldsymbol{x}) \approx \hat{f}(\boldsymbol{x}) := f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^\tau(\boldsymbol{x} - \boldsymbol{x}_0).$$

For sufficiently smooth vector valued function $\boldsymbol{f} = (f_1, \ldots, f_n) : \mathbb{R}^n \to \mathbb{R}^n$ we then assemble the approximations for all the component functions f_i and thereby get:

$$\boldsymbol{f}(\boldsymbol{x}) \approx \hat{\boldsymbol{f}}(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x}_0) + \begin{pmatrix} \nabla f_1(\boldsymbol{x}_0)^\tau \\ \vdots \\ \nabla f_n(\boldsymbol{x}_0)^\tau \end{pmatrix} (\boldsymbol{x} - \boldsymbol{x}_0). \tag{2.43}$$

The matrix in the second term on the right hand side of (2.43) is of course the Jacobian matrix of $\boldsymbol{f}$ which we denote $\mathbf{J}_{\boldsymbol{f}}$:

$$\mathbf{J}_{\boldsymbol{f}}(\boldsymbol{x}) := \begin{pmatrix} \nabla f_1(\boldsymbol{x})^{\tau} \\ \vdots \\ \nabla f_n(\boldsymbol{x})^{\tau} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\boldsymbol{x}) \end{pmatrix}.$$

In extended Kalman filter we assume that object dynamics and measurement process are captured by differentiable functions $\boldsymbol{f} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ and $\boldsymbol{h} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$:

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}(\boldsymbol{x}_{k-1}), \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k). \end{aligned}$$

Gaussian noise assumptions **K0-KP** from the Kalman filter setup are replicated in the context of extended Kalman filtering, only with revised expressions . Transition density, likelihood function and posterior density are therefore given by:

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \phi(\boldsymbol{x}_k|\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{Q}_k), \tag{2.44}$$

$$q(\boldsymbol{y}_k|\boldsymbol{x}_k) = \phi(\boldsymbol{y}_k|\boldsymbol{h}(\boldsymbol{x}_k), \boldsymbol{R}_k), \tag{2.45}$$

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_1, \dots, \boldsymbol{y}_{k-1}) = \phi\big(\boldsymbol{x}_{k-1}|\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}\big). \tag{2.46}$$

Exact derivation of the posterior distribution $p(\boldsymbol{x}_k|\boldsymbol{y}_1, \dots, \boldsymbol{y}_k)$ at next time step (i.e. integrating in eq. (2.9) given the relations (2.44), (2.45) and (2.46)) is not trivial regardless of the normality assumptions. Therefore we resort to linearization procedure from (2.43). Specifically, we approximate $\boldsymbol{f}(\boldsymbol{x}_{k-1})$ by the first order terms in Taylor series expansion of $\boldsymbol{f}(\boldsymbol{x}_{k-1})$ at $\hat{\boldsymbol{x}}_{k-1|k-1}$:

$$\boldsymbol{f}(\boldsymbol{x}_{k-1}) \approx \hat{\boldsymbol{f}}(\boldsymbol{x}_{k-1}) = \boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big) + \boldsymbol{F}_k\big(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1|k-1}\big), \tag{2.47}$$

where we denoted:

$$\boldsymbol{F}_k = \mathbf{J}_{\boldsymbol{f}}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big).$$

We regroup terms in (2.47) and apply this approximation in transition density (2.44):

$$p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \approx \phi\big(\boldsymbol{x}_k|\boldsymbol{F}_k\boldsymbol{x}_{k-1} + (\boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big) - \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1|k-1}), \boldsymbol{Q}_k\big). \tag{2.48}$$

In the similar fashion, we approximate function $\boldsymbol{h}(\boldsymbol{x}_k)$ with first order terms from Taylor expansion about $\hat{\boldsymbol{x}}_{k|k-1}$:

$$\boldsymbol{h}(\boldsymbol{x}_k) \approx \hat{\boldsymbol{h}}(\boldsymbol{x}_k) = \boldsymbol{h}\big(\hat{\boldsymbol{x}}_{k|k-1}\big) + \boldsymbol{H}_k\big(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}\big),$$

where we denoted:

$$\boldsymbol{H}_k = \mathbf{J}_{\boldsymbol{h}}\big(\hat{\boldsymbol{x}}_{k|k-1}\big)$$

to approximate likelihood function (2.45):

$$q(\boldsymbol{y}_k|\boldsymbol{x}_k) \approx \phi\big(\boldsymbol{y}_k|\boldsymbol{H}_k\boldsymbol{x}_k + (\boldsymbol{h}\big(\hat{\boldsymbol{x}}_{k|k-1}\big) - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1}), \boldsymbol{R}_k\big).$$

Having linearized the means of both transition distribution and likelihood function, we quickly show how results of section on Kalman filtering equations (previous section) can be rewritten for extended Kalman filter setup.

Consider proposition 2.4.1 which derives the predicted state distribution $p(\boldsymbol{x}_k|\boldsymbol{y}_1, \dots, \boldsymbol{y}_{k-1})$. Define random vectors $\boldsymbol{W}$ and $\boldsymbol{Z}$ as in the proof of the proposition:

$$\begin{aligned} \boldsymbol{W} &= \boldsymbol{X}_{k-1}|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}}, \\ \boldsymbol{Z} &= \boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}}. \end{aligned}$$

We follow the remainder of the proof as well, applying approximated transition density (2.48) and (2.46) to get:

$$\begin{aligned} \boldsymbol{W} &\sim N\big(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{P}_{k-1|k-1}\big), \\ \boldsymbol{Z}|_{\boldsymbol{W}=\boldsymbol{x}_{k-1}} &\sim N\big(\boldsymbol{F}_k\boldsymbol{x}_{k-1} + (\boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big) - \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1|k-1}), \boldsymbol{Q}_k\big). \end{aligned} \tag{2.49}$$

The proof of the proposition finishes by invoking theorem 1.2.2. This is the case here as well. Term $\boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big) - \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1|k-1}$ is independent of $\boldsymbol{x}_{k-1}$, therefore mean in (2.49) is linear function and theorem applies with conclusions:

$$\begin{aligned} \boldsymbol{X}_k|_{\boldsymbol{y}_1,\dots,\boldsymbol{y}_{k-1}} &\sim N\big(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}\big), \\ \boldsymbol{P}_{k|k-1} &= \boldsymbol{Q}_k + \boldsymbol{F}_k\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_k^{\intercal}, \\ \hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big) - \boldsymbol{F}_k\hat{\boldsymbol{x}}_{k-1|k-1} = \\ &= \boldsymbol{f}\big(\hat{\boldsymbol{x}}_{k-1|k-1}\big). \end{aligned}$$

Similarly, one can replicate the procedure from theorem 2.4.4 and arrive at the rest of extended Kalman filtering equations:

$$\begin{aligned} \boldsymbol{S}_k &:= \boldsymbol{R}_k + \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\intercal}, \\ q(\boldsymbol{y}_k|\boldsymbol{y}_1, \dots, \boldsymbol{y}_{k-1}) &= \phi(\boldsymbol{y}_k|\boldsymbol{h}\big(\hat{\boldsymbol{x}}_{k|k-1}\big), \boldsymbol{S}_k), \\ \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\intercal}\boldsymbol{S}_k^{-1}\big(\boldsymbol{y}_k - \hat{\boldsymbol{x}}_{k|k-1}\big), \\ \boldsymbol{P}_{k|k} &= \boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\intercal}\boldsymbol{S}_k^{-1}\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}. \end{aligned}$$

While there are many similarities and analogies between Kalman and extended Kalman filters, notice that posterior covariance matrix $\boldsymbol{P}_{k|k}$ is dependent on measurements in extended version. Indeed, Jacobian $\boldsymbol{H}_k$ is evaluated at object state estimate $\hat{\boldsymbol{x}}_{k|k-1}$, which in turn depends on observations $\boldsymbol{y}_1, \dots, \boldsymbol{y}_{k-1}$.

2.6 Unscented Kalman filter

In section 2.4 we derived a closed form solution to the recursion (2.9), under general assumptions **C0-CM** from section (2.2) and Kalman filter assumptions **K0-KP**. In the previous section we dropped assumption that underlying dynamical system is linear, and approximated the object dynamics function $\boldsymbol{f}$ and measurement function $\boldsymbol{h}$ with linear terms from their respective Taylor expansions, to get the approximate solution to (2.9).

In this section, we continue this program by presenting the **unscented Kalman filter** - another approximate solution of (2.9). It has similar computational load as extended Kalman filter, but covers broader class of object dynamics and measurement functions and usually achieves better performance. In fact, it is in general on par with the *second-order* extended Kalman filter, without requiring calculation of Jacobian or Hessian ([6], [13]).

Unscented Kalman filter is based on the **unscented transformation** - numerical algorithm for approximating moments of non-linearly transformed random variable or vector. Let $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ be a non-linear function and $\boldsymbol{X} : \Omega \to \mathbb{R}^n$ random vector on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We are interested in calculating the mean and covariance of the random vector $\boldsymbol{f}(\boldsymbol{X})$:

$$\begin{aligned}
\mathbb{E}(\boldsymbol{f}(\boldsymbol{X})) &= \int_{\mathbb{R}^n} p(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})d\boldsymbol{x}, \\
\operatorname{cov}(\boldsymbol{f}(\boldsymbol{X})) &= \int_{\mathbb{R}^n} p(\boldsymbol{x})(\boldsymbol{f}(\boldsymbol{x}) - \mathbb{E}(\boldsymbol{f}(\boldsymbol{X})))(\boldsymbol{f}(\boldsymbol{x}) - \mathbb{E}(\boldsymbol{f}(\boldsymbol{X})))^{\tau} d\boldsymbol{x}.
\end{aligned}$$

For that purpose, we compute the weighted sum of the function values on the certain collection of points from the range of the given random vector. We call this collection of points *sigma points* and denote $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n \in \mathbb{R}^n$. The corresponding weights are $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n$. We impose the following condition on the sigma points and weights:

$$\frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i \boldsymbol{\xi}_i}{n} = \mathbb{E}(\boldsymbol{X}), \tag{2.50}$$

$$\frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i (\boldsymbol{\xi}_i - \mathbb{E}(\boldsymbol{X}))(\boldsymbol{\xi}_i - \mathbb{E}(\boldsymbol{X}))^{\tau}}{n} = \operatorname{cov}(\boldsymbol{X}). \tag{2.51}$$

In short - we want the sample mean and covariance of sigma points with respect to "probability distribution" given by the weights to correspond to the mean and covariance of the random vector $\boldsymbol{X}$. We then approximate the moments of the transformed random vector $\boldsymbol{f}(\boldsymbol{X})$ by the following sums:

$$\widehat{\mathbb{E}(\boldsymbol{f}(\boldsymbol{X}))} = \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i \boldsymbol{f}(\boldsymbol{\xi}_i)}{n}, \tag{2.52}$$

$$\widehat{\operatorname{cov}(\boldsymbol{f}(\boldsymbol{X}))} = \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i \Big(\boldsymbol{f}(\boldsymbol{\xi}_i) - \widehat{\mathbb{E}(\boldsymbol{f}(\boldsymbol{X}))}\Big)\Big(\boldsymbol{f}(\boldsymbol{\xi}_i) - \widehat{\mathbb{E}(\boldsymbol{f}(\boldsymbol{X}))}\Big)^{\tau}}{n}, \tag{2.53}$$

$$\widehat{\operatorname{cov}(\boldsymbol{X}, \boldsymbol{f}(\boldsymbol{X}))} = \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i \Big(\boldsymbol{\xi}_i - \mathbb{E}(\boldsymbol{X})\Big)\Big(\boldsymbol{f}(\boldsymbol{\xi}_i) - \widehat{\mathbb{E}(\boldsymbol{f}(\boldsymbol{X}))}\Big)^{\tau}}{n}. \tag{2.54}$$

Determining weights and sigma points that satisfy (2.50) and (2.51) is a problem in itself, skipped here for brevity. See [6] for an example of the construction.

Assumptions of unscented Kalman filter are similar as those in standard Kalman filter - big difference is that object dynamics and measurement functions are non-linear. Object state and measurement processes are given by:

$$\boldsymbol{X}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{v}_k, \tag{2.55}$$

$$\boldsymbol{Y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \tag{2.56}$$

where

$$\boldsymbol{v}_k \sim N(\mathbf{0}, \boldsymbol{Q}_k), \tag{2.57}$$

$$\boldsymbol{w}_k \sim N(\mathbf{0}, \boldsymbol{R}_k). \tag{2.58}$$

We further suppose that at time k, joint distribution of latest object state and latest observation, conditional on the past data, is multivariate normal:

$$f\big(\boldsymbol{x}_k, \boldsymbol{y}_k | \mathbf{y}^{k-1}\big) = \phi\left(\begin{pmatrix} \boldsymbol{x}_k \\ \boldsymbol{y}_k \end{pmatrix} \middle| \begin{pmatrix} \hat{\boldsymbol{x}}_{k|k-1} \\ \hat{\boldsymbol{y}}_{k|k-1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{P}_{k|k-1} & \boldsymbol{C}_k \\ \boldsymbol{C}_k^{\tau} & \boldsymbol{S}_k \end{pmatrix}\right). \tag{2.59}$$

Here it was implicitly assumed:

$$\begin{aligned} \boldsymbol{X}_k|_{\mathbf{y}^{k-1}} &\sim N\big(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}\big) \\ \boldsymbol{Y}_k|_{\mathbf{y}^{k-1}} &\sim N\big(\hat{\boldsymbol{y}}_{k|k-1}, \boldsymbol{S}_k\big) \\ \boldsymbol{C}_k &:= \operatorname{cov}\big(\boldsymbol{X}_k|_{\mathbf{y}^{k-1}}, \boldsymbol{Y}_k|_{\mathbf{y}^{k-1}}\big) \end{aligned}$$

The goal is to determine posterior state distribution at time k, $p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)$. But observe that this is just a conditional distribution with respect to the joint distribution

$f\big(\boldsymbol{x}_k, \boldsymbol{y}_k | \mathbf{y}^{k-1}\big)$, conditioned on $\boldsymbol{Y}_k = \boldsymbol{y}_k$. Hence retrieving the distribution $p\big(\boldsymbol{x}_k|\mathbf{y}^k\big)$ amounts to estimating all the parameters in (2.59), and then using the theorem 1.1.1.

Focusing on the predicted state distribution for the moment, observe that (2.55) and (2.57) give:

$$\begin{aligned}\hat{\boldsymbol{x}}_{k|k-1} &= \mathbb{E}\big(\boldsymbol{X}_k|\mathbf{y}^{k-1}\big) = \\ &= \mathbb{E}\big(\boldsymbol{f}(\boldsymbol{X}_{k-1}) + \boldsymbol{v}_k|\mathbf{y}^{k-1}\big) = \\ &= \mathbb{E}\big(\boldsymbol{f}(\boldsymbol{X}_{k-1})|\mathbf{y}^{k-1}\big),\end{aligned}$$

and similarly:

$$\begin{aligned}\boldsymbol{P}_{k|k-1} &= \operatorname{cov}\big(\boldsymbol{X}_k|\mathbf{y}^{k-1}\big) = \\ &= \operatorname{cov}\big(\boldsymbol{f}(\boldsymbol{X}_{k-1}) + \boldsymbol{v}_k|\mathbf{y}^{k-1}\big) = \\ &= \operatorname{cov}\big(\boldsymbol{f}(\boldsymbol{X}_{k-1})|\mathbf{y}^{k-1}\big) + \boldsymbol{Q}_k.\end{aligned}$$

Mean and covariance of the distribution $p\big(\boldsymbol{f}(\boldsymbol{x}_{k-1})|\mathbf{y}^{k-1}\big)$ are now to be estimated using unscented transformation. Let us denote sigma points and weights at time $k-1$ with $\boldsymbol{\xi}_1^{(k-1)}, \ldots, \boldsymbol{\xi}_n^{(k-1)}$ and $\boldsymbol{\psi}_1^{(k-1)}, \ldots, \boldsymbol{\psi}_n^{(k-1)}$ respectively. These are of course chosen so that (2.50) and (2.51) are satisfied for the posterior object state at time $k-1$, $\boldsymbol{X}_k|_{\mathbf{y}^{k-1}}$. We propagate sigma points one time step via function $\boldsymbol{f}$:

$$\boldsymbol{\xi}_i^{(k)} = \boldsymbol{f}\Big(\boldsymbol{\xi}_i^{(k-1)}\Big)$$

and facilitate estimates for $\hat{\boldsymbol{x}}_{k|k-1}$ and $\boldsymbol{P}_{k|k-1}$ using (2.52) and (2.53) as follows:

$$\hat{\boldsymbol{x}}_{k|k-1} = \frac{\sum_{i=1}^n \boldsymbol{\psi}_i^{(k-1)} \boldsymbol{\xi}_i^{(k)}}{n},$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{Q}_k + \frac{\sum_{i=1}^n \boldsymbol{\psi}_i^{(k-1)} \Big(\boldsymbol{\xi}_i^{(k)} - \hat{\boldsymbol{x}}_{k|k-1}\Big)\Big(\boldsymbol{\xi}_i^{(k)} - \hat{\boldsymbol{x}}_{k|k-1}\Big)^\tau}{n}.$$

Parameters $\boldsymbol{S}_k$ and $\hat{\boldsymbol{y}}_{k|k-1}$ in (2.59) are estimated in the similar fashion, using already establishd unscented transformation for the posterior distribution at time $k-1$. This time however, sigma points are propagated via function $\boldsymbol{h}$:

$$\boldsymbol{\eta}_i^{(k)} = \boldsymbol{h}\Big(\boldsymbol{\xi}_i^{(k)}\Big).$$

We get:

$$\hat{\boldsymbol{y}}_{k|k-1} = \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i^{(k-1)} \boldsymbol{\eta}_i^{(k)}}{n},$$

$$\boldsymbol{S}_k = \boldsymbol{R}_k + \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i^{(k-1)} \Big(\boldsymbol{\eta}_i^{(k)} - \hat{\boldsymbol{y}}_{k|k-1}\Big) \Big(\boldsymbol{\eta}_i^{(k)} - \hat{\boldsymbol{y}}_{k|k-1}\Big)^{\tau}}{n}.$$

To estimate final parameter, cross-covariance $\boldsymbol{C}_k$, employ (2.54):

$$\boldsymbol{C}_k = \frac{\sum_{i=1}^{n} \boldsymbol{\psi}_i^{(k-1)} \Big(\boldsymbol{\eta}_i^{(k)} - \hat{\boldsymbol{y}}_{k|k-1}\Big) \Big(\boldsymbol{\xi}_i^{(k)} - \hat{\boldsymbol{x}}_{k|k-1}\Big)^{\tau}}{n}.$$

Joint distribution (2.59) is now conditioned on $\boldsymbol{Y}_k = \boldsymbol{y}_k$ using theorem 1.1.1 directly. First determine the relevant partitioned inverted covariance matrix elements using lemma 1.2.1:

$$\begin{aligned} \boldsymbol{\Lambda}_{\mathbf{11}}^{-1} &= \boldsymbol{P}_{k|k-1} - \boldsymbol{C}_k \boldsymbol{S}_k \boldsymbol{C}_k^{\tau}, \\ \boldsymbol{\Lambda}_{\mathbf{12}} &= -\big(\boldsymbol{P}_{k|k-1} - \boldsymbol{C}_k \boldsymbol{S}_k \boldsymbol{C}_k^{\tau}\big)^{-1} \boldsymbol{C}_k \boldsymbol{S}_k^{-1}. \end{aligned}$$

Then apply statement (1.6) to calculate the mean:

$$\begin{aligned} \hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \big(\boldsymbol{P}_{k|k-1} - \boldsymbol{C}_k \boldsymbol{S}_k \boldsymbol{C}_k^{\tau}\big)\big(\boldsymbol{P}_{k|k-1} - \boldsymbol{C}_k \boldsymbol{S}_k \boldsymbol{C}_k^{\tau}\big)^{-1} \boldsymbol{C}_k \boldsymbol{S}_k^{-1}\big(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1}\big) = \\ &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{C}_k \boldsymbol{S}_k^{-1}\big(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_{k|k-1}\big) \end{aligned}$$

and covariance

$$\boldsymbol{P}_{k|k} = \boldsymbol{\Lambda}_{\mathbf{11}}^{-1} = \boldsymbol{P}_{k|k-1} - \boldsymbol{C}_k \boldsymbol{S}_k \boldsymbol{C}_k^{\tau}$$

of the posterior state distribution at time k.

Chapter 3

Monocular 3D object tracking

3.1 Introduction

The interest in the problem of accurate vehicle detection and tracking in the context of autonomous driving has been on the rise in recent years. The most successful approaches in commercial applications rely on fusion of variety of non-trivial and expensive sensors such as radar, lidar, sonar, GPS, odometry and inertial measurement units. Specialized control systems are used to analyse sensory output and deduce navigation paths and obstacles.

One of the drivers of progress in the area was the emergence of public benchmarks that supply datasets of real-world traffic pictures and scenes, with accurate $3D$ annotations and lidar sensor outputs, such as a pioneer KITTI ([11], used for testing in this work), and in recent times Waymo Open Dataset ([27]), NuScenes ([5]) and Argoverse ([7]). The objective on these benchmarks is to infer type and exact $3D$ information (position, size and orientation) for all objects in the given image, using various sensor data (such as lidar point cloud) and images themselves. In the case of tracking benchmarks, the goal is to identify and give tracks for objects across a given video sequence. Example of an image in KITTI detection benchmark is shown in figure 3.1.

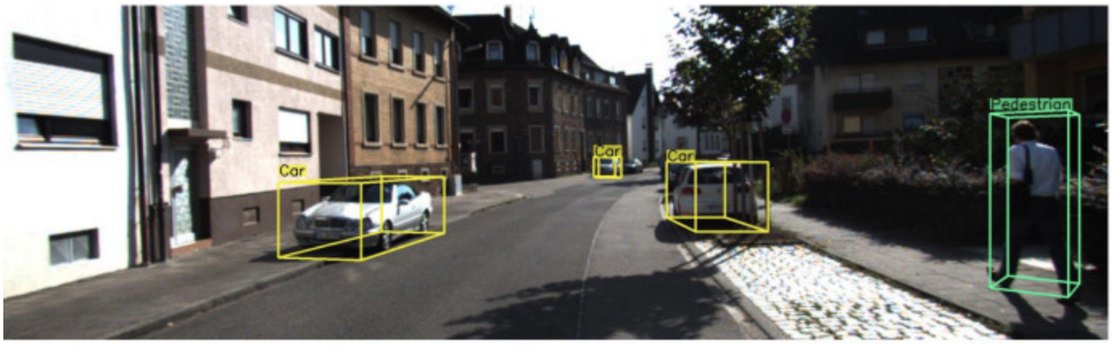

Figure 3.1: Sample image from KITTI detection benchmark. Bounding boxes are due to a recent completely *monocular* method [4].

Lidar device produces precise depth information in terms of 3D point clouds, and since the seminal work of [20] and its variants showed how to directly manipulate point cloud by way of deep neural networks, works based on these principles in both detection and tracking area followed. They became the most performant in aforementioned public benchmarks; interested reader can follow [25] for an example.

On the other hand, drawbacks of lidar such as high cost and sensitivity to weather are reason for the recent popularity of **monocular methods**, which are pursued in this work. Monocular methods try to infer the object position in $3D$ world from *single image*. This task is inherently ill-posed, due to the overall lack of reliable depth information. While the gap in performance of monocular and lidar-based methods is still substantial, the trend is positive. The basic tool in these methods are again deep neural networks, only this time applied directly to the RGB image produced by the camera, instead of the expensive lidar point cloud.

In this work, object detection method is borrowed from a recent paper [9] by Ding M. *et al.* (2019) in its entirety, and Kalman filter was used as a main component of the tracking algorithm. Therefore, solution to the problem of monocular $3D$ vehicle tracking presented here is naturally divided in 2 self-contained parts - detection and tracking. Their respective implementations are consequently modular, in a sense that a different detector can be used in conjunction with the tracker, and different tracker can be built upon the detector. Finally, their descriptions are going to be separate as well - detection framework is discussed in section 3.4 and the Kalman filter setup in section 3.5. But we begin by introducing KITTI Vision Benchmark in section 3.2, and important technical details in section 3.3.

3.2 KITTI Vision Benchmark

The multiple object tracking (MOT) system described in this work was tested on the publicly available KITTI benchmark.

This benchmark was developed in 2012 as a joint project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago, with a mission to provide accurate and challenging benchmark for various visual recognition endeavours such as:

- $2D$ and $3D$ object detection
- Multiple object tracking
- Visual odometry / SLAM, Scene flow, Depth prediction and others

The choice of systems of coordinates in this work is in accordance with the specifications of KITTI MOT and $3D$ detection benchmarks.

The reference point (origin) is the camera position, i.e. the origin is moving with the ego-vehicle. The actual coordinate system used is depicted in figure 3.2.

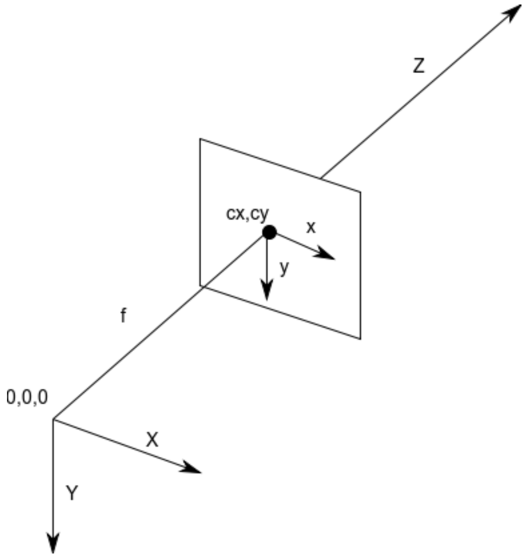

Figure 3.2: Camera and image coordinates.

The origin is at position $(0,0,0)^\tau \in \mathbb{R}^3$, y axis points vertically downward and z axis represents depth (from the camera standpoint). The separate coordinate system is that of the image plane, also depicted above. Its origin is at the top left corner of the image, with directions of the axes as indicated in the figure.

3.3 Bounding box representation

In this discussion, we employ the *aircraft principal axes* terminology, where x, y and z axis from figure 3.2 are called pitch or transverse axis, yaw or normal axis and roll or longitudinal axis respectively.

The corresponding Euler angles, i.e. rotations around the axes are denoted with ϕ (*pitch*), θ (*yaw*) and ψ (*roll*). Positive direction of rotation is determined by right-hand rule.

The location and orientation of detected objects in $3D$ world are described by way of their **bounding boxes**, which are rectangular cuboids enclosing the object.

There are various ways to specify such a bounding box. We choose the following set of parameters, guided by the KITTI labeling conventions:

- $\mathbf{p} = (x_0, y_0, z_0)^\tau \in \mathbb{R}^3$ is the location of the *bottom center* of the cuboid in camera coordinate system,
- $\theta \in [-\pi, \pi]$ - yaw, heading angle on the ground plane,
- w, h and l - width, height and length of the cuboid.

In KITTI MOT benchmark video sequences, as well as in $3D$ detection benchmark images, roll and pitch are assumed to be zero, i.e. $\psi = \phi = 0$. Therefore they don't feature in bounding box parametrization.

Furthermore, we introduce the observation angle $\alpha \in [-\pi, \pi]$, the relative rotation of the bounding box with respect to the line connecting the camera (ego-vehicle) with the detected object. Although this angle is not important in specification of the bounding box, it is crucial in recovering the bounding box from the image patch containing the observed object.

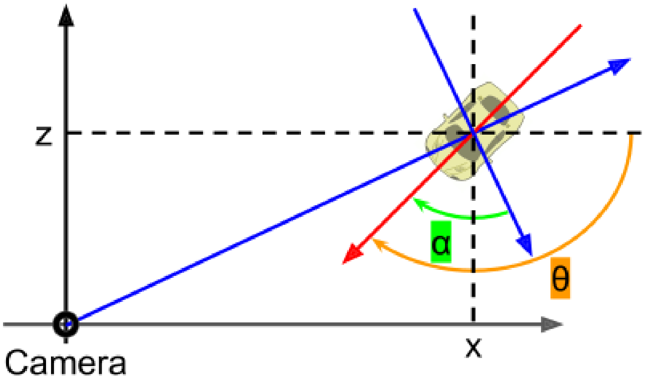

Figure 3.3: Global (θ) and local (α) orientation of the object from the bird-eye view. Image taken from [17].

The definition of α and relationship with heading angle θ, and their positive orientations, are much clearer from the bird-eye view in figure 3.3. We can see that the difference between θ and α, for the case in the figure where both $\alpha > 0$ and $\theta > 0$, is in fact the ray angle (angle between positive z-axis and the ray which connects the camera and the object center):

$$\theta = \alpha + \arctan\left(\frac{x}{z}\right).$$

3.4 Detection

As mentioned earlier, monocular methods in $3D$ object detection are experiencing rapid development. This development is primarily driven by advancements in deep learning applications to computer vision domain, which has been nothing short of magical throughout the last decade.

The aim in monocular $3D$ detection is to provide the $3D$ bounding boxes (introduced in section 3.3) for all the relevant objects captured in the single image of the scenery. The starting point in majority of the approaches is to use an accurate $2D$ detector, i.e. to obtain $2D$ bounding box (a rectangle) for the relevant objects, thereby for the moment ignoring depth and orientation estimation.

The seminal work in this regard is [22], Faster R-CNN (**R**egions with **C**onvolutional **N**eural **N**etworks), a culmination of the R-CNN family of papers that changed the landscape of object detection in computer vision. These papers cover the gradual transition from traditional methods (template matching techniques using HOG and

SIFT descriptors, SVM classifiers, selective search etc.) to completely neural network based, accurate and blazingly fast end-to-end frameworks popular today. These networks usually consist of:

- Region proposal algorithm to generate proposals of bounding boxes for possible objects in the image. Prior to the final paper a selective search procedure was used, but in Faster R-CNN this is a neural network as well, called RPN (**R**egion **P**roposal **N**etwork).

- Feature generation network. This is usually a convolutional neural network that generates features for objects.

- Classification layer which assigns classes to objects.

- Regression layer which refines the coordinates of the bounding boxes.

Many monocular $3D$ detection methods use Faster R-CNN (or variation) in some way or the other in their pipelines. They differ in their approach to solve the fundamental problem of inferring depth (distance from the camera) of the object from single view of the scene, i.e. arrive at $3D$ bounding box for the object, instead of the rectangle obtained from Faster R-CNN framework (and its variations). We outline two popular approaches:

- Distance estimation through 2D/3D constraints

- Direct generation of $3D$ proposal

In the first approach, consistency between $2D$ and $3D$ box is imposed in order to derive the missing information. One observes that although the inference problem of depth and orientation from single view is not well posed, the vehicles are rigid bodies and as such can be constrained to particular geometric setups of known shape and size. The pioneering work in this direction is so called Deep3DBox, [18]. In this work, the local orientation (observation angle α from figure 3.3) and size of the object (w, h and l of the bounding box) are directly regressed from the corresponding image patch. The authors then leverage the assumption that $3D$ bounding box projected onto the image fits the $2D$ box tightly to obtain the missing center location. This amounts to all 4 sides of the $2D$ bounding box having at least one vertex of the $3D$ bounding box projected onto them (visualized in figure 3.4). The resulting over-constrained optimization problem is tractable, and multiple solutions were proposed over the recent years, improving on mentioned [18].

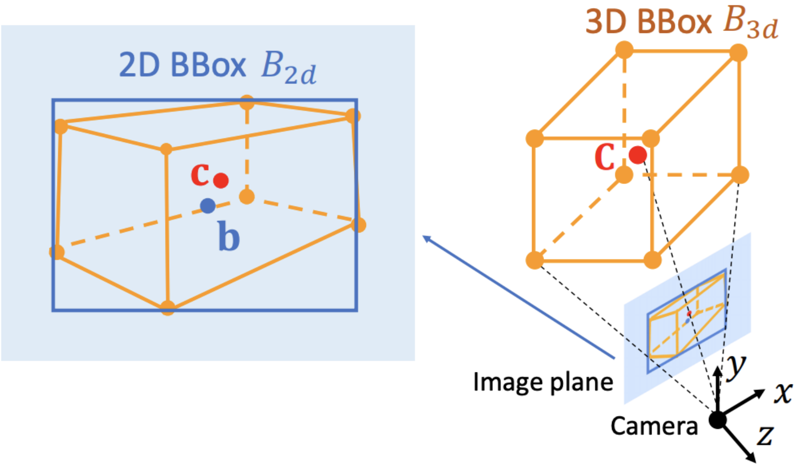

Figure 3.4: Tight $2D/3D$ constraint. Image taken from [21].

A more direct approach traces its roots to the influential work of Uber Advanced Technologies Group, [8]. Their method relies on densely (exhaustively) generating $3D$ proposals for object bounding boxes using various priors (e.g. cars are constrained to the ground-plane). Each proposal is then scored according to hand-crafted features, and final result is obtained by the algorithm of *non-maximum suppression* (NMS) - a simple technique of filtering proposals (of which there are possibly many) based on a given criterion (in this case the score obtained from various features). It is worth noting that non-maximum suppression is also employed in filtering the proposals in the Faster R-CNN framework introduced earlier. We explain this procedure immediately.

Let us denote with $\mathcal{P} = \{1, \dots, N\}$ the input of the algorithm, i.e. set of N proposal boxes. The set of their individual scores is in turn denoted $\mathcal{S} = \{s_1, \dots, s_N\}$. Let $\delta > 0$ be a positive real which we call *overlap threshold*. The output of the algorithm will be a set of filtered proposals $\mathcal{F} \subseteq \mathcal{P}$, initially empty. Define a function $0 \leq \mathrm{IOU}(i, j) \leq 1$ which takes two box proposals i and j and calculates their *intersection over union* (IOU), which is literally the quotient of the area of intersection of two boxes over the area of their union. This function serves as a similarity measure of two proposal boxes. Repeat as follows, until $\mathcal{P} = \varnothing$:

1. Select the proposal i with highest score s_i, $i = \arg\max \mathcal{S}$, remove it from $\mathcal{P}$ and add it to output $\mathcal{F}$.

2. For every $j \in \mathcal{P}$, remove j from $\mathcal{P}$ if $\mathrm{IOU}(i, j) > \delta$.

The monocular detection method used in this work is D^{4}LCN from [9], recently published at 2020 Conference on Computer Vision and Pattern Recognition. It is currently the best performing purely monocular method on the KITTI $3D$ detection benchmark.

This method is in line with the direct proposal generation approach, though it differs greatly in its implementation details. We limit ourselves to a short summary - full technical description of the method is beyond the scope of this text.

The key component in understanding depth in a given scene (and scene in general) is a *depth map*, which encodes depth of various points of the image. The authors choose fully monocular state of the art depth map generator (again a specialized CNN system) from [10] called DORN. The example of a depth map is shown in figure 3.5.

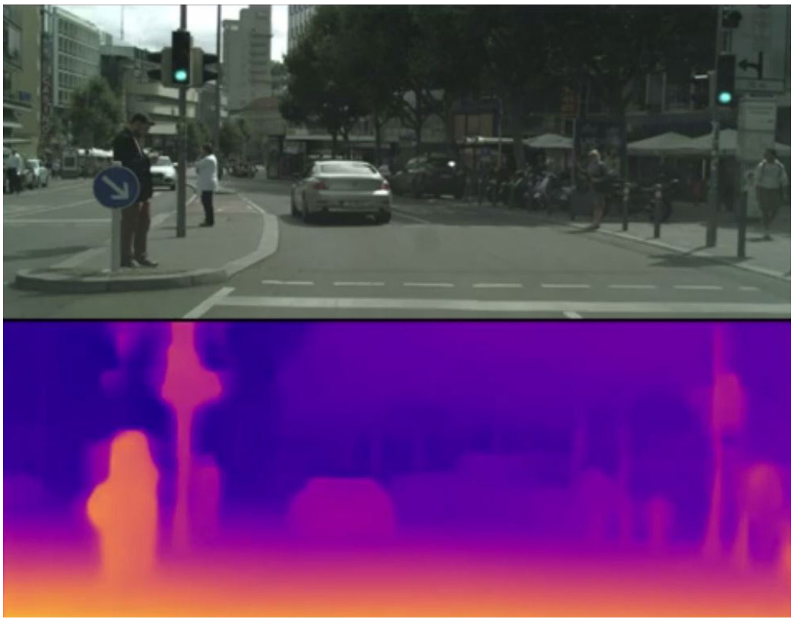

Figure 3.5: Example of a depth map

Instead of using hand-crafted features to score proposals, D^{4}LCN employs a novel CNN pipeline to serve as feature extraction network. Rather than learning global convolutional kernels and applying them to all images, kernels in this work are specific to each pixel, taking into account pixel's location and depth (as retrieved from the depth map).

It is worth mentioning that availability of depth maps gives rise to another type of monocular methods, called *pseudo-lidar*. Depth maps are transformed into a point cloud similar to one produced by lidar devices. Then, state of the art lidar-based CNN systems are applied. The problem with this approach is heavy reliance on accuracy of depth maps. Although D^{4}LCN (the method we employ) uses depth maps as well, authors claim to have somewhat reduced this dependence.

3.5 Tracking

In this work, Kalman filter algorithm is used as a base of the tracking pipeline. More complicated filtering algorithms are used in recent works, such as Poisson multi-Bernoulli mixture filter in [23] and unscented Kalman filter in [19], but more complexity does not necessarily lead to a significant increase in performance. Apart from filtering, *data association* is important part of any tracking system - one has to match predicted trajectories with new detection results at each frame.

Our choice of object state (in the context of filtering algorithm) is pretty standard. Constant velocity model is assumed, and the object state at time k is set to be

$$\boldsymbol{x}_k = [x_k, y_k, z_k, \theta_k, l_k, w_k, h_k, \dot{x}_k, \dot{y}_k, \dot{z}_k]^\tau \in \mathbb{R}^{10}.$$

Object here (what we are trying to track) is in fact a bounding box. Its state is described with vector $\boldsymbol{x}_k$, where x, y, z is the position of the bottom center of the box, l, w, h are dimensions of the box and θ is the heading angle. Note that these definitions are in line with the bounding box representation in section 3.3. Finally, $\dot{x}_k$, $\dot{y}_k$, $\dot{z}_k$ are velocities of the bounding box in respective directions. From the constant velocity model, we derive object dynamics equations (regarded deterministically, i.e. not including the noise terms):

$$\begin{aligned} x_k &= x_{k-1} + T\dot{x}_k, \\ y_k &= y_{k-1} + T\dot{y}_k, \\ z_k &= z_{k-1} + T\dot{z}_k, \end{aligned}$$

where $T = 0.1s$ is time interval between frames, specific of course to the KITTI benchmark setup. Taking into account that in the case of constant velocity heading angle and velocities do not change, we are ready to specify the object dynamics equation of the Kalman filter:

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1},$$

where transition matrix $\boldsymbol{F} \in \mathbb{R}^{10\times10}$ is given by:

$$\boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Detection module described in section 3.4 returns bounding boxes (specified in section 3.3) at every frame. We denote the measurement vector with $\boldsymbol{y}_k$ for $k \in \mathbb{N}$ to be exactly this detection output:

$$\boldsymbol{y}_k = [x_k, y_k, z_k, \theta_k, l_k, w_k, h_k]^\tau \in \mathbb{R}^7.$$

Furthermore, since measurement components correspond without any transformation to state components, we have the measurement equation:

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k,$$

with the measurement matrix $\boldsymbol{H} \in \mathbb{R}^{7\times10}$ given by

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

This completes the specification of our Kalman filter setup.

Chapter 4

Implementation and results

4.1 Detection module

This work was implemented in Python, and the implementation will be made publicly available at [1]. Due to lack of appropriate hardware (strong GPU specifications are required for the task) we use Google Colaboratory environment for running CNN models (GPU: 1xTesla K80 , compute capability 3.7, having 2496 CUDA cores and 12GB GDDR5 VRAM).

In detection module, we use the models provided by the authors of D^4LCN([9]) (Pytorch as deep learning framework) and DORN([10]) (Caffe framework), pretrained on the KITTI $3D$ detection and depth prediction benchmarks, respectively.

KITTI single image depth prediction benchmark consists of close to a hundred thousand training images, providing ample training dataset. In contrast, KITTI $3D$ detection benchmark training set contains merely 7481 images, due to arduous task of correct data labeling. In case reader wonders if the size of this set is sufficient to adequately train complex CNN systems used in D^4LCN, such as backbone network ResNet-50, the answer lies in the concept of *transfer learning*, which pervades the modern deep learning methodology. Deep neural networks (e.g. ResNet-50) are trained on huge datasets such as COCO by Microsoft with hundreds of thousands and ImageNet with millions of very diverse images. Then *final layers* of these pretrained models are additionally trained on the target dataset. The objective is to have initial layers of networks learning generic features such as edges, shapes, textures etc. (*transferable* across multiple domains), while the final layers are specialized for task at hand.

These pretrained models showcased decent performance on the images from KITTI multiple object tracking benchmark. The main drawback with respect to lidar based detectors is high number of *false negatives* - while the provided detections turned out to be solid input to our tracking pipeline, more distant or occluded vehicles often remained undetected. The example of a vehicle occluded by a cyclist and not picked up by D^4LCN detector is displayed in figure 4.1.

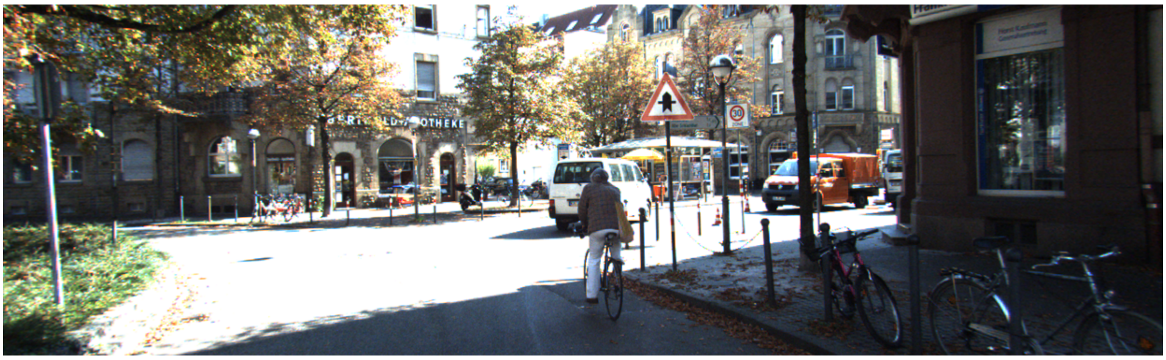

Figure 4.1: Example of a missed partially occluded target. White van is not detected, as well as the orange vehicle which is probably too far away.

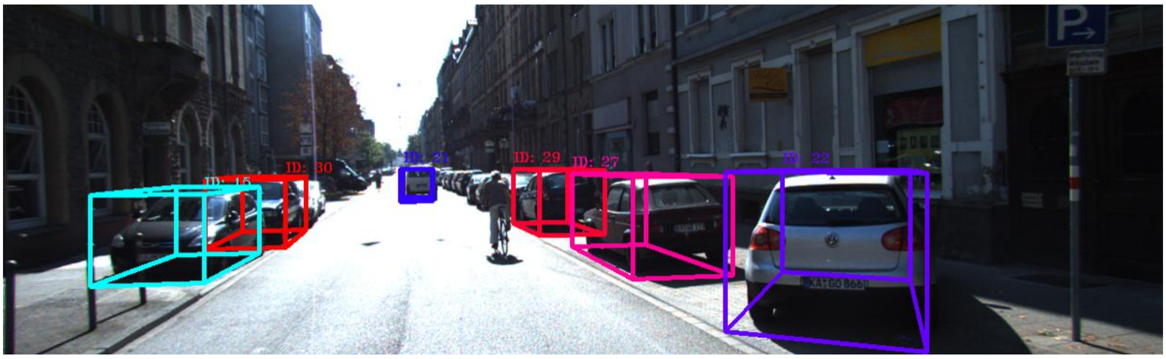

Figure 4.2: Distant future from figure 4.1 (around 100 frames). White van is now far away, but it is not occluded and we track it successfully.

4.2 Tracking module

Python library FilterPy was used to implement the Kalman filtering logic, outlined in the previous chapter. While this is the main component of the tracking module, in realistic tracking scenarios more than one object is encountered in each frame, hence this has to be accounted for.

Indeed, suppose that at time (frame) $k-1$, m_{k-1} tracks are available. We denote them with $\boldsymbol{S}_{k-1}$, therefore:

$$\boldsymbol{S}_{k-1} = \{\boldsymbol{x}_{k-1}^1, \ldots, \boldsymbol{x}_{k-1}^{m_{k-1}}\}.$$

Denote with $\boldsymbol{S}_k^p$ the predicted object tracks (states) at frame k. These are obtained by propagating $\boldsymbol{S}_{k-1}$ from $k-1$ to k using object dynamics equation, which amounts to multiplying by matrix $\boldsymbol{F}$.

On the other hand, denote with $\boldsymbol{D}_k$ a set of n_k bounding boxes that are *detected* at frame k:

$$\boldsymbol{D}_k = \{\boldsymbol{y}_k^1, \ldots, \boldsymbol{y}_k^{n_k}\}.$$

In order to utilize this novel information, we update the object states (trajectories) $\boldsymbol{S}_k^p$ via Kalman filtering equations derived in chapter 2. Important practical consideration here is that of the data association problem. i.e. properly matching detections with trajectories.

While $\boldsymbol{S}_k^p$ still has m_{k-1} tracks, this number is generally different than number of detected objects in the current frame, n_k. Vehicles enter and exit camera field of view frequently. Spatial distance between individual objects may be small, and changes from one frame to another abrupt. Apart from matching detections from the current frame with the existing trajectories, identity of the vehicles throughout the duration of a sequence should be preserved.

A convenient way to interpret this problem is as a bipartite graph matching. First, we introduce the weighted bipartite graph on sets of predicted trajectories $\boldsymbol{S}_k^p$ (predicted bounding boxes at frame k) and detection results $\boldsymbol{D}_k$ (observed bounding boxes at frame k), where the weight between trajectory i and detection j is their intersection over union $\mathrm{IOU}(i, j)$. A matching in a graph is a subset of its edges, where no two selected edges share a vertex. We then solve the problem of *maximum weight matching*, i.e. finding the matching where the sum of the weights is the greatest.

Maximum weight matching is a known variation on the *assignment problem* (unbalanced bipartite graph, maximum weight instead of minimum cost), which is a well studied example in linear programming. Polynomial solutions exist, such as the *Hungarian algorithm*, which we employ in this work.

In case of missing detections in a given frame, existing track is propagated no more than 2 frames into the future, in hope for the next observation.

Finally, we address the problem of initializing the track. In order to decrease the number of false positives, i.e. tracking non-existent vehicles, we consider the requirement of more than one successive observation for a given vehicle before confirming a track. Number of observations required (duration of the confirmation phase) is denoted with C in further text.

4.3 KITTI tracking benchmark

KITTI multiple object tracking benchmark contains 21 video sequences in a training dataset, with the total of around 8000 frames and 30000 annotated objects. The testing dataset is bigger with 28 sequences, but no labels are provided for them. While the supported classes of objects are car, cyclist and pedestrian, we limit ourselves on the car class in this work.

Track i (i.e. a bounding box) provided by the tracking algorithm is considered a match, or a *true positive* if the $\mathrm{IOU}(i, j)$ with respect to the appropriate ground truth track j is greater than 0.5. Total number of true positives in all frames of all sequences is denoted TP. It is important to note that KITTI tracking benchmark projects bounding boxes and takes intersection over union in the image plane, as opposed to considering 3D overlap, whereas our algorithm provides 3D information.

Any unmatched tracks (according to the definition given above) are called *false positives*, while undetected ground truth tracks are called *false negatives*. Respective total occurrences of those are denoted FP and FN. Amongst true positives, *identity switch* happens when track is lost and then recovered but with changed identity (e.g. vehicle was occluded for few frames and propagation into the future was terminated due to lack of observations), or in case of collision with another track identities are not adequately preserved. We denote total number of identity switches with IDS.

Then we can define the metrics used in the benchmark:

- MOTA (Multiple Object Tracking Accuracy) aims to capture the overall performance of the tracking algorithm taking into account multiple sources of errors - false positives, false negatives and identity switches:

$$\mathrm{MOTA} = 1 - \frac{\mathrm{FN} + \mathrm{FP} + \mathrm{IDS}}{\mathrm{GT}}, \tag{4.1}$$

  where GT denotes the total number of ground truth objects.

- MOTP (Multiple object tracking precision) measures the average precision of the matched true positives. Let M be the set of all ordered pairs of (t, g), where t is true positive track, and g is its ground truth counterpart. Then $|M| = TP$, and we define:
$$\mathrm{MOTP} = \frac{\sum_{(t,g)\in M} \mathrm{IOU}(t, g)}{\mathrm{TP}}.$$
Since t is true positive, $0.5 \leq \mathrm{IOU}(t, g) \leq 1$ and hence $0.5 \leq \mathrm{MOTP} \leq 1$.
- Ground truth trajectory is classified as either *mostly tracked* (successfully tracked for at least 80% of its life span), *mostly lost* (tracked for under 20% of its lifespan) and *partially tracked* (any other track). Proportions of these with respect to the total number of ground truth tracjectories are denoted MT, ML and PT. Identity switches are not taken into account in these metrics.

Metrics introduced here can be evaluated directly on the training set of KITTI tracking benchmark, due to existence of labels. To measure performance on the test set, scientific research projects (not students who are writing master's degree thesis) are allowed to upload their results on the KITTI server, with outcomes displayed in public benchmark.

To compare monocular detection with lidar-based version, we adopt the detection results on KITTI tracking dataset from recent state of the art lidar-based detector [26]. We then use both detection methods in conjunction with our tracking module to obtain the following results on the training dataset:

Detection module	**C**	**MOTA**	**MOTP**	**MT**	**PT**	**ML**	**ID switches**
Monocular	3	0.633486	0.871845	0.3	0.54	0.16	7
Monocular	2	0.657250	0.868709	0.34	0.53	0.13	10
Monocular	1	0.673120	0.863316	0.45	0.45	0.1	243
Lidar-based	3	0.797175	0.868170	0.74	0.22	0.04	13

Table 4.1: Results on KITTI tracking benchmark training dataset

Detection module	**C**	**FP**	**TP**	**FN**	**Recall**	**Precision**
Monocular	3	904	17023	**7911**	**0.68**	0.95
Monocular	2	1003	17794	**7237**	**0.71**	0.95
Monocular	1	1271	18838	**6354**	**0.75**	0.94
Lidar-based	3	1667	25653	**3212**	**0.89**	0.94

Table 4.2: Subset of confusion matrix for different detection modules

While not state of the art (which is approximately 90%), MOTA values above 60% (for the length of confirmation phase $C = 1$ we have MOTA = 67.3%) indicate solid overall performance, and MOTP value of $\approx 87\%$ suggests that results on true positives are non trivial, i.e. IOU overlap is on average significantly bigger than 50%. Number of identity switches is very low when duration of confirmation phase is more than 1 frame.

Furthermore, we conclude that tracking module is sensible, due to admirable performance in conjunction with lidar-based detection method.

In order to highlight the main disadvantage of monocular detection performance in comparison with lidar-based method, we show the important elements of confusion matrix in table 4.2. The recall value, i.e. proportion of retrieved ground truth objects, is much bigger for the lidar-based method. It missed 3212 targets (FN), whereas monocular algorithms missed twice as many. Taking into account that both methods achieve high precision (high TP to TP+FP ratio), we conclude that missing targets is main relative drawback of the monocular method.

Best performing monocular setup according to MOTA and MT metrics is the one with $C = 1$ - every observation, if unmatched with existing track, is used to initialize a track. While this approach alleviates the problem of false negatives, we notice that it has a high number of identity switches. Although MOTA metric somewhat downplays the phenomenon of an identity switch (there are very few *vehicle trajectories* in comparison with *vehicle instances*, thus false positives/negatives should not be counted on an equal footing with identity switches in eq. (4.1)), in practice it is unquestionably an inconvenient event. It obstructs fusion of the information inferred from images with data from other sensors (e.g. radar). Furthermore, it impedes the overall performance because every new track initialization generally requires multiple frames.

In the end, we show some qualitative results as well in figure 4.3. The images are spaced about 5 frames one from the other. While tracking in the example is mostly sound, we can confirm aforementioned areas of improvement, such as losing the black car on the edge of the field of view, and what seems to be an identity switch in later frames.

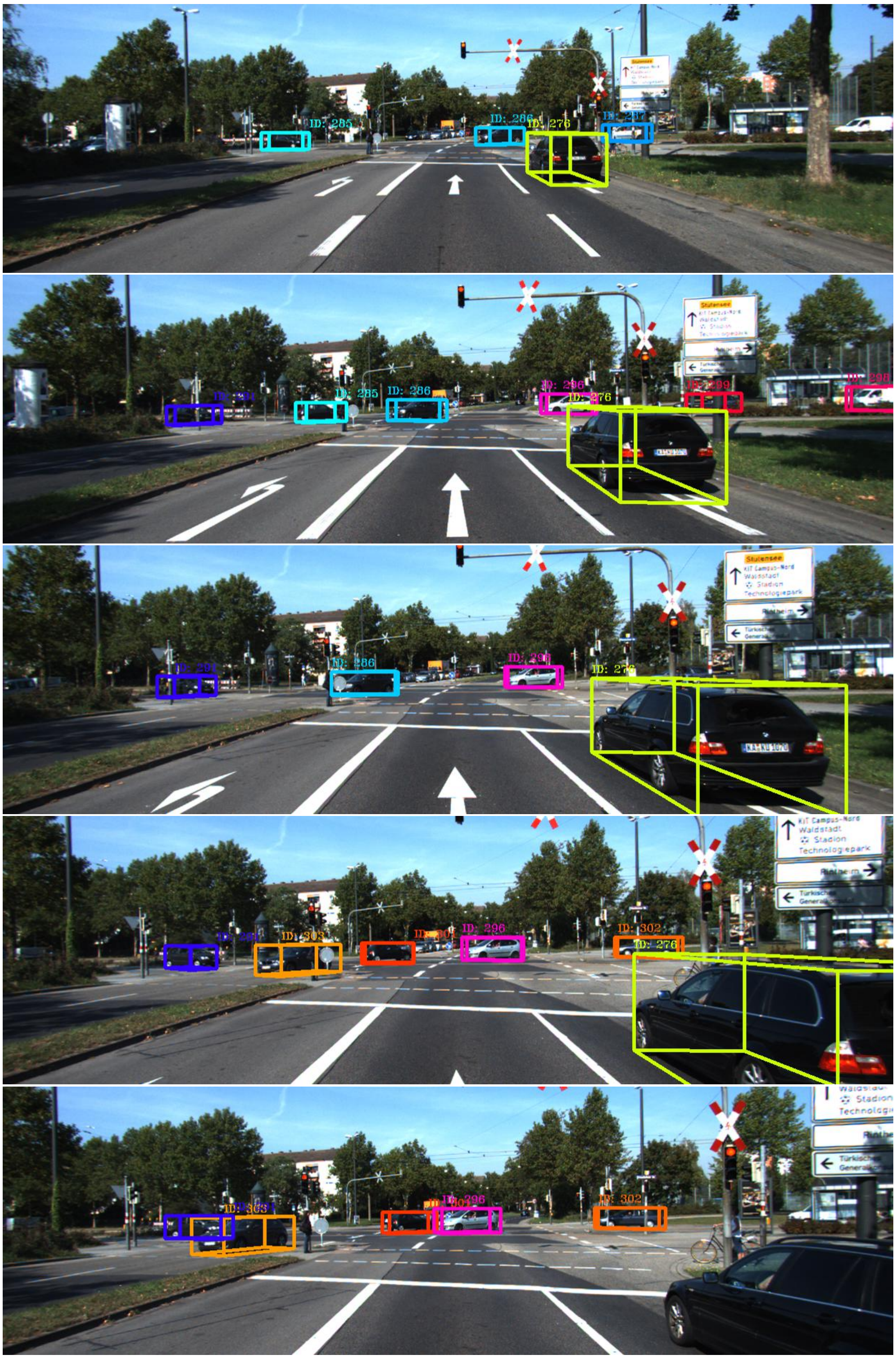

Figure 4.3: Output of the tracking algorithm. Time flows top to bottom.

Bibliography

[1] `https://github.com/Florijan-Iljazovic/MOT-Master-Thesis`.

[2] D. Bernstein, *Matrix mathematics: Theory, facts, and formulas*, 2009.

[3] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, 2006.

[4] G. Brazil and X. Liu, *M3d-rpn: Monocular 3d region proposal network for object detection*, The IEEE International Conference on Computer Vision (ICCV), October 2019.

[5] H. Caesar, V. Bankiti, A. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, *nuscenes: A multimodal dataset for autonomous driving*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020), 11618–11628.

[6] S. Challa, M. Morelande, D. Musicki, and R. Evans, *Fundamentals of object tracking*, 2011.

[7] M. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. T. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, *Argoverse: 3d tracking and forecasting with rich maps*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019), 8740–8749.

[8] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, *Monocular 3d object detection for autonomous driving*, CVPR, 2016.

[9] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo, *Learning depth-guided convolutions for monocular 3d object detection*, The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2020.

[10] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, *Deep Ordinal Regression Network for Monocular Depth Estimation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[11] A. Geiger, P. Lenz, and R. Urtasun, *Are we ready for autonomous driving? the kitti vision benchmark suite*, 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012), 3354–3361.

[12] A. Graves, A. Mohamed, and G. E. Hinton, *Speech recognition with deep recurrent neural networks*, 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (2013), 6645–6649.

[13] A. Jazwinski, *Stochastic processes and filtering theory*, 1970.

[14] R. E. Kalman, *A new approach to linear filtering and prediction problems*, Transactions of the ASME–Journal of Basic Engineering **82** (1960), no. Series D, 35–45.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, CACM, 2017.

[16] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object recognition with gradient-based learning*, Shape, Contour and Grouping in Computer Vision, 1999.

[17] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, *Gs3d: An efficient 3d object detection framework for autonomous driving*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019), 1019–1028.

[18] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, *3d bounding box estimation using deep learning and geometry*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[19] A. Patil, S. Malla, H. Gang, and Y. Chen, *The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes*, 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 9552–9557.

[20] C.R. Qi, H. Su, K. Mo, and L.J. Guibas, *Pointnet: Deep learning on point sets for 3d classification and segmentation*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[21] Z. Qin, J. Wang, and Y. Lu, *Monogrnet: A geometric reasoning network for monocular 3d object localization*, AAAI, 2019.

[22] S. Ren, K. He, R. Girshick, and J. Sun, *Faster R-CNN: Towards real-time object detection with region proposal networks*, Neural Information Processing Systems (NIPS), 2015.

[23] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, *Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering*, 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 433–440.

[24] A. Senior, R. Evans, J. Jumper, J. R. M. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, *Improved protein structure prediction using potentials from deep learning*, Nature **577** (2020), 706–710.

[25] S. Shi, C. Guo, Li.J, Z. Wang, J. Shi, X. Wang, and H. Li, *Pv-rcnn: Point-voxel feature set abstraction for 3d object detection*, The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[26] S. Shi, X. Wang, and H. Li, *Pointrcnn: 3d object proposal generation and detection from point cloud*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019), 770–779.

[27] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, *Scalability in perception for autonomous driving: Waymo open dataset*, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020), 2443–2451.

[28] T. Young, D. Hazarika, S. Poria, and E. Cambria, *Recent trends in deep learning based natural language processing [review article]*, IEEE Computational Intelligence Magazine **13** (2018), 55–75.

[29] S. Zhang, L. Yao, Aixin Sun, Y. Tay, and A. Sun, *Deep learning based recommender system: A survey and new perspectives*, 2018.

Sažetak

Neuralne mreže i duboko učenje su moćan i brzo rastući alat prilikom analize slika prometa, čemu pomažu brojni javno dostupni resursi za treniranje i testiranje neuronskih mreža na velikim skupovima podataka kao što je KITTI. To je moguće iskoristiti i povratne podatke iz CNN sistema za raspoznavanje objekata (kamera je montirana na vozilu) poslati u odgovarajući algoritam za praćenje u realnom vremenu. Ovaj rad koristi recentnu metodu ([9]) dobivanja detekcija vozila prisutnih na slici i Kalmanov filter kao algoritam praćenja. Testiranje se provodi na KITTI benchmarku.

U prvom poglavlju navodimo neka bitna svojstva višedimenzionalne normalne distribucije, s naglaskom na Bayesov teorem za normalnu razdiobu, koji se koristi u izvodu Kalmanovog filtra.

U drugom poglavlju, nakon kratkog uvoda u područje algoritama za praćenje i odgovarajuću terminologiju, koristimo bayesovski pristup kako bismo dobili općenito rekurzivno rješenje za praćenje jednog objekta. Zatim diskutiramo i detaljno izvodimo Kalmanov filter, a kraće komentiramo i njegove bitne ekstenzije, prošireni Kalmanov filter i *unscented* Kalmanov filter.

U trećem poglavlju dajemo kratki osvrt na moderne metode raspoznavanja vozila i njihovo praćenje, te prezentiramo našu metodu.

U četvrtom poglavlju navodimo neke implementacijske detalje i iznosimo rezultate na KITTI podacima. Glavni problem sustava u odnosu na ekvivalent baziran na lidaru je manji broj deketiranih objekata. Usprkos tome, rezultati su solidni.

Summary

Neural networks and deep learning are a powerful tool for detection of vehicles in camera images, growing steadily due to advance in ideas and large-scale public benchmarks such as KITTI. In principle, this can be exploited by using output of appropriate deep CNN system and feeding it as measurements input to the object tracking algorithm. This work combines state of the art monocular detection system ([9]) with the Kalman filter algorithm into multiple object tracking pipeline. The performance is tested on public KITTI benchmark video sequences.

In the first chapter we review some important properties of multivariate normal distribution, with the stress on the Bayes' theorem for Gaussian densities, which is empoyed in the derivation of the Kalman filter in the subsequent chapter.

In chapter 2, after giving some introduction to object tracking problem and terminology, we first consider more general tracking setup employing Bayesian probabilistic framework. Then we give detailed derivation of Kalman filter, and overview of its notable variations, extended Kalman filter and unscented Kalman filter.

In chapter 3 we shortly review state of the art object detection and tracking systems, and present our method.

In chapter 4 we remark some implementation details and report results of our method on the KITTI tracking benchmark. While it achieves solid performance, detector has too many false negatives (missed target) to achieve state of the art results.

Životopis

Rodio sam se u Zagrebu, 24. rujna 1995., gdje sam pohađao Osnovnu školu Tituša Brezovačkog i zagrebačku V. gimnaziju. Predstavljao sam Hrvatsku na *International physics olympiad* (IPhO) 2014. i sudjelovao na državnim natjecanjima iz fizike, logike i hrvatskog jezika.

Obrazovanje sam nastavio na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu, gdje sam 2014. godine upisao preddiplomski studij *Matematika*, te potom 2017. godine diplomski studij *Matematička statistika.* Tijekom studija bio sam demonstrator iz mnogih kolegija. Po završetku oba studija nagrađen sam za izniman uspjeh od Vijeća Matematičkog odsjeka.