A robust state estimation for an electric race car

Student Research Project / T3100

for the study program
Computer Science

at the
Baden-Wuerttemberg Cooperative State University Stuttgart

by
Dominik Stiller

Submission Date	June 8, 2020
Thesis Supervisor	B.Sc. Marco Busch
University Supervisor	Prof. Dr. Zoltán Ádam Zomotor
Matriculation Number, Course	4369179, TINF17A

Declaration of Authorship

I hereby declare that the thesis submitted with the title *A robust state estimation for an electric race car* is my own unaided work. All direct or indirect sources used are acknowledged as references.

Neither this nor a similar work has been presented to an examination committee or published.

Sindelfingen	June 8, 2020	
Place	Date	Dominik Stiller

Confidentiality Clause

This thesis contains confidential data of *DHBW Engineering Stuttgart e.V.* This work may only be made available to the university supervisor. Any publication and duplication of this thesis–even in part–is prohibited.

An inspection of this work by third parties requires the expressed permission of the author and *DHBW Engineering Stuttgart e.V.*

Abstract

Real-time computer vision applications with deep learning-based inference require hardware-specific optimization to meet stringent performance requirements. Frameworks have been developed to generate the optimal low-level implementation for a certain target device based on a high-level input model using machine learning in a process called autotuning. However, current implementations suffer from inherent resource utilization inefficiency and bad scalability which prohibits large-scale use.

In this paper, we develop a load-aware scheduler which enables large-scale autotuning. The scheduler controls multiple, parallel autotuning jobs on shared resources such as CPUs and GPUs by interleaving computations, which minimizes resource idle time and job interference. The scheduler is a key component in our proposed Autotuning as a Service reference architecture to democratize autotuning. Our evaluation shows good results for the resulting inference performance and resource efficiency.

Contents

Acronyms

4WD four-wheel drive

CAN controller area network

CG center of gravity

DV driverless vehicle

ECU electronic control unit

EKF extended Kalman filter

EV electric vehicle

GNSS global navigation satellite system

GPS Global Positioning System

IMU inertial measurement unit

TC traction control

TV torque vectoring

UKF unscented Kalman filter

UT unscented transform

VDC vehicle dynamics control

List of Figures

List of Tables

List of Source Codes

List of Symbols

α	Angular accleration	$\mathrm{rad\,s^{-2}}$
β	Vehicle sideslip angle	rad
$\mathbb{0}$	Zero vector	1
ω	Angular velocity	$\mathrm{rad\,s^{-1}}$
ω_{motor}	Motor speed	$\mathrm{rad\,s^{-1}}$
ω_{wheel}	Wheel speed	$\mathrm{rad\,s^{-1}}$
ϕ	Angular displacement around x-axis/roll angle	rad
ψ	Angular displacement around z-axis/yaw angle	rad
τ	Threshold for outlier detection methods	$*$
θ	Angular displacement around y-axis/pitch angle	rad
φ	Angular orientation	rad
a	Linear accleration	$\mathrm{m\,s^{-2}}$
i_{gear}	Gear ratio from motor to wheels	1
p	Linear displacement/position in earth-fixed coordinates	m
R	Corner radius	m
r	Linear displacement/position in vehicle coordinates	m
r_{dyn}	Dynamic tire radius	m
v	Linear velocity	$\mathrm{m\,s^{-1}}$

x	x-axis component of linear position in earth-fixed coordinates	m
y	y-axis component of linear position in earth-fixed coordinates	m
z	z-axis component of linear position in earth-fixed coordinates	m

1 Introduction

Race cars have fascinated people, built quickly after first car instead of comfort, max performance more examples of tradeoffs

1.1 Problem

more than xx years since first car while mindset is same, now there are electric race cars new possibilities because of four-wheel drive (4WD) and computing power individual torque on each wheel computer assist driver in getting max performance, execution of driver commands with optimal use of vehicle potential TC, TV, battery management components need estimate of vehicle state task of state estimation which delivers good estimate even in face of sensor failures and unpredictable environment

1.2 Scope

this thesis describes design of a robust, accurate, flexible state estimation for a formula student race car maybe a research question? this state estimation fuses available sensors and detects outliers first background, then design and implementation then evaluation with measurement data

project for DHBW engineering team but works in other cars as well

2 Background

The state estimation of a vehicle sits at the junction of vehicle dynamics and control theory. Both knowledge of the dynamics of a race car and the algorithms to model their physics in equations and software is required to design a successful solution. Therefore, we explore the fundamentals of vehicle dynamics, estimation algorithms and sensor outlier detection in this chapter.

Throughout this thesis, the conventions of ISO 8855 [1] will be used, which assumes a right-handed coordinate system. The vehicle coordinate system uses an upward z-axis with a forward x-axis and a leftward y-axis, while the earth-fixed coordinate system uses an upward z-axis with an eastward x-axis and a northward y-axis. The vehicle'scenter of gravity (CG) is used as origin/reference point of the vehicle coordinate system. In case of mixed coordinate systems, left-superscript will be used to denote the reference frame (e.g., ${}^{V}x$ for vehicle coordinates, ${}^{E}x$ for earth-fixed coordinates).

2.1 Rigid Body Kinematics

The fundamental laws of mechanics apply to race cars as they do to any other body. These laws relate, among others, the body's linear and angular position and its time derivatives, resulting in translational and rotational changes. Their three-dimensional vector definitions are shown in equations 1 and 2. To simplify the equations, a rigid body is assumed. This means, that deformations which occur in the vehicle during dynamic maneuvers are negligibly small or vanish. Therefore, points on the body maintain the same distance relative to each other at all times. Furthermore, a two-dimensional motion in the road plane can be assumed in many cases because the effects vertical dynamics are negligible. The simplified equations for that case will also be shown.

$$p = \begin{bmatrix} x, y, z \end{bmatrix}^T \tag{1a}$$
$$v = \begin{bmatrix} v_x, v_y, v_z \end{bmatrix}^T \tag{1b}$$
$$a = \begin{bmatrix} a_x, a_y, a_z \end{bmatrix}^T \tag{1c}$$
$$\beta = \arctan\left(\frac{v_x}{v_y}\right) \approx \frac{v_y}{v_x} \tag{1d}$$

$$\varphi = \begin{bmatrix} \phi, \theta, \psi \end{bmatrix}^T \tag{2a}$$
$$\omega = \begin{bmatrix} \dot{\phi}, \dot{\theta}, \dot{\psi} \end{bmatrix}^T \tag{2b}$$
$$\alpha = \begin{bmatrix} \ddot{\phi}, \ddot{\theta}, \ddot{\psi} \end{bmatrix}^T \tag{2c}$$

The linear displacement p of a body describes its position relative to the origin of its reference frame. It comprises a longitudinal component along the x-axis, a lateral component along the y-axis and a vertical component along the z-axis. Its first time derivative $v = \frac{dr}{dt}$ and second time derivative $a = \frac{d^2r}{dt^2}$ are the body's linear velocity and acceleration, respectively. The scalar magnitude $\|v\|$ of the velocity is called speed. In vehicles, the arctangent of the ratio of lateral and longitudinal velocities, often approximated as just the ratio, is furthermore denoted the *vehicle sideslip angle* β.

The angular orientation φ of a body describes its rotation in the reference frame. It can be described by the roll angle ϕ around the x-axis, the pitch angle θ around the y-axis and the yaw angle, or heading, ψ around the z-axis. The angular velocity ω and acceleration α describe the element-wise time derivatives of these angles. Talking about p and φ only makes sense in earth-fixed coordinates.

2.1.1 Transformation of Linear Velocities

All points on a rigid body experience the same angular velocity, i.e. $\omega^A = \omega^B$ for any two points A, B on that body at all times. However, these two points will generally not have the same linear velocity vector, since it is affected by their location r relative to the CG. Only when $\omega = \mathbb{0}$, the linear velocity is equal for all points on the body, i.e. $v^A = v^B$. If there is a non-zero angular velocity, equation 3 holds for any point P, assuming the velocity at the CG is known (the expanded form can be found in Appendix B.1).

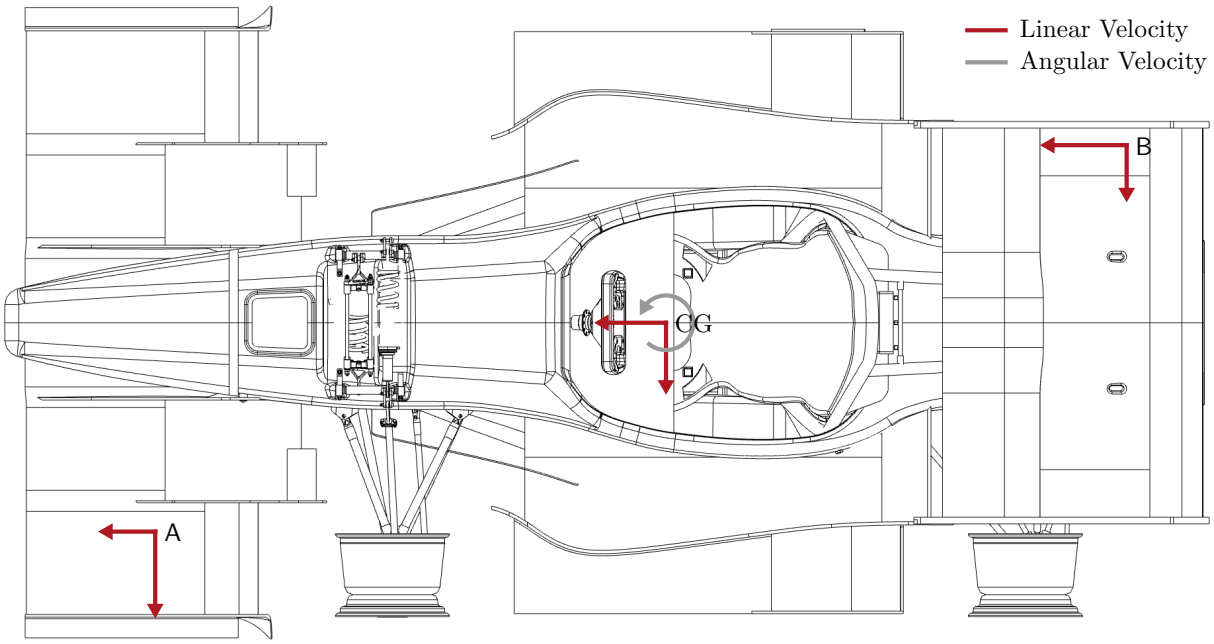


Figure 1: Experienced velocities at off-center points

$$v^P = v^{CG} + \omega \times r^P \tag{3}$$

This becomes easier to visualize when regarding the two-dimensional case, where v_z, $\dot{\phi}$ and $\dot{\theta}$ are disregarded, as shown in equation 4.

$$v^P = \begin{bmatrix} v_x^{CG} \\ v_y^{CG} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} v_x^{CG} \\ v_y^{CG} \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\psi} \cdot r_y \\ \dot{\psi} \cdot r_x \\ 0 \end{bmatrix} \tag{4}$$

Let us regard the example scenario shown in figure 1, where the vehicle has a positive yaw rate and the CG is moving forward and to the left. Point A is at the front left ($r_x > 0$, $r_y > 0$), while point B is at the rear right ($r_x < 0$, $r_y < 0$). Since r is defined relative to the CG, its position is $\mathbb{0}$. Due to the positive yaw rate, A experiences a higher v_y but lower v_x than the CG. B, on the other hand, experiences a higher v_x but lower v_y than the CG. If the yaw rate were zero, all points would experience the same linear velocity.

2.1.2 Transformation of Linear Accelerations

Like the angular velocity, the angular acceleration is the same at every point on a rigid body, i.e. $\alpha^A = \alpha^B$ for any two points A, B on that body. However, these two points will generally not experience the same linear acceleration. Only if the rotational motion components ω and α are $\mathbb{0}$, the linear acceleration is equal for all points on the body, i.e.

$a^A = a^B$. In the general case, equation 5 [2, p. 140] holds for any point P at location r (the expanded form can be found in Appendix B.2).

$$a^P = a^{CG} + \alpha \times r^P + \omega \times (\omega \times r^P) \tag{5}$$

We can identify two additional components which affect the experienced linear acceleration. The term $\alpha \times r$ is the tangential acceleration along the circular path around the center of rotation, which is $a_{tan} = \alpha \cdot r$ in scalar notation. More significant due to the squared angular velocity, however, is the introduction of the centripetal acceleration a_c in the term $\omega \times (\omega \times r)$. This term is the vector-equivalent of the acceleration resulting from the centripetal force F_c in equation 6.

$$F_c = \frac{mv^2}{r} \iff a_c = \frac{v^2}{r} = \omega^2 r \tag{6}$$

The two-dimensional form, shown in equation 7, is more intuitive than its three-dimensional counterpart. Here, a_z, $\dot{\phi}$, $\dot{\theta}$, $\ddot{\phi}$ and $\ddot{\theta}$ are assumed to be zero. The inward-direction of the centripetal effect can be seen by the negative signs of the angular velocity part.

$$a^P = \begin{bmatrix} a_x^{CG} \\ a_y^{CG} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \ddot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \left(\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \right) = \begin{bmatrix} a_x^{CG} \\ a_y^{CG} \\ 0 \end{bmatrix} + \begin{bmatrix} -\ddot{\psi} \cdot r_y \\ \ddot{\psi} \cdot r_x \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\psi}^2 \cdot r_x \\ -\dot{\psi}^2 \cdot r_y \\ 0 \end{bmatrix} \tag{7}$$

We demonstrate these effects in the example scenario shown in figure 2, which is similar to the one in the previous subsection, but with an additional positive yaw acceleration. In point A, the tangential and centripetal acceleration cancel out and even surpass the linear acceleration experienced in the CG, resulting in a negative a_x and a much lower a_y. The opposite effect occurs in point B, where both a_x and a_y are amplified.

2.1.3 Calculate Angular Acceleration from Linear Acceleration

Direct measurement of the angular acceleration is often not possible. However, individual components of α can be calculated from the difference of two known linear acceleration vectors at different points A, B with known locations r^A, r^B using equation 8. It is derived from equation 5, with $\Delta a = a^A - a^B$, $\Delta r = r^A - r^B$, and a^{CG} being eliminated by the difference. The points must not be on the rotation axis of the calculated component of α, otherwise they experience no angular acceleration. Thus, at least three non-collinear

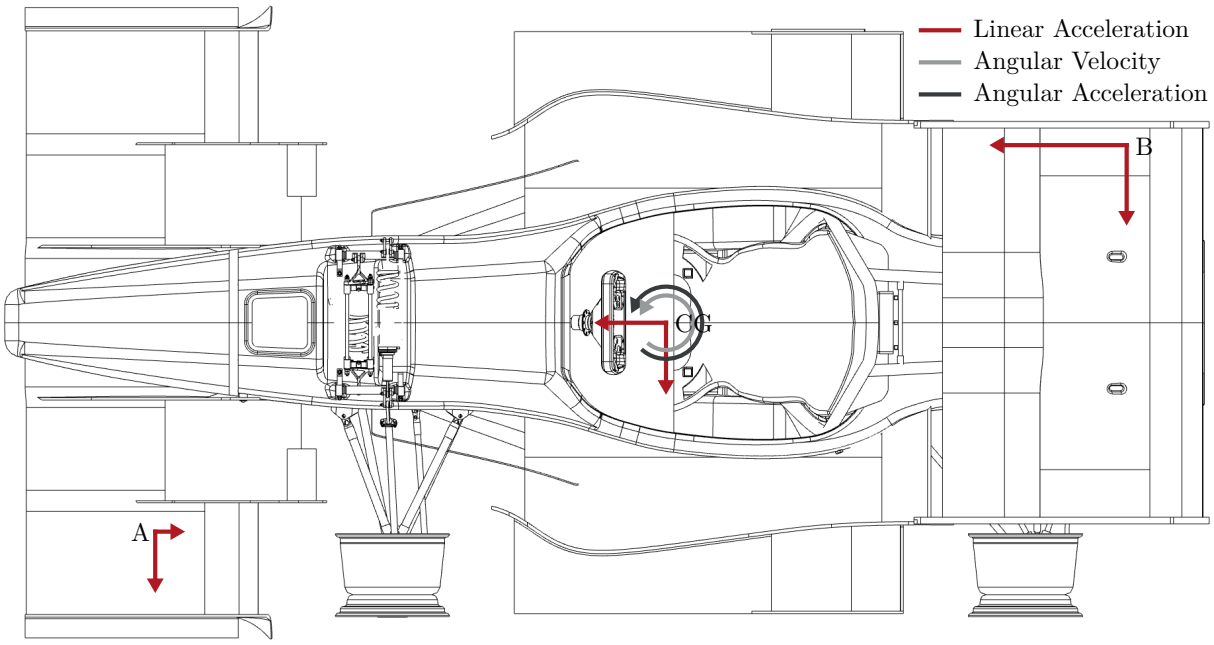


Figure 2: Experienced acceleration at off-center points

points are required to determine all components of α. While the inverse of a cross product is not uniquely determined, as can be seen by the scalar factor t, $t = 0$ works well in practice.

$$\Delta a = \alpha \times \Delta r + \omega \times (\omega \times \Delta r) \implies \alpha = \frac{\Delta r \times (\Delta a - \omega \times (\omega \times \Delta r))}{\|\Delta r\|^2} + t \cdot \Delta r, t \in \mathbb{R} \quad (8)$$

In two dimensions, this becomes easier to understand. Calculation of the yaw acceleration from the longitudinal and lateral accelerations of two points is shown in equation 9. We can see that the accelerations and distances from different axes are being correlated. The equation even shows how positioning both points on the z-axis would result in a zero division, which shows that points off the rotation axis are required. When A and B are directly in front of and behind the CG on the x-axis so $\Delta r_y = 0$, the equation simplifies to $\ddot{\psi} = \frac{\Delta a_y}{\Delta r_x}$. For practical applications, increasing the distance between the two points minimizes the effects of measurement uncertainty in the result.

$$\alpha = \frac{\begin{bmatrix} \Delta r_x \\ \Delta r_y \\ 0 \end{bmatrix} \times \left(\begin{bmatrix} \Delta a_x \\ \Delta a_y \\ 0 \end{bmatrix} - \begin{bmatrix} -\dot{\psi}^2 \cdot r_x \\ -\dot{\psi}^2 \cdot r_y \\ 0 \end{bmatrix} \right)}{\left\| \begin{bmatrix} \Delta r_x \\ \Delta r_y \\ 0 \end{bmatrix} \right\|^2} \implies \ddot{\psi} = \frac{\Delta r_x \Delta a_y - \Delta r_y \Delta a_x}{\Delta r_x^2 + \Delta r_y^2} \quad (9)$$

2.1.4 Transformation of Velocity Between Coordinate Systems

Most sensor measurements are done in the vehicle coordinate system. Others, such as position measurements from a global navigation satellite system (GNSS) (e.g., Global Positioning System (GPS), Galileo, GLONASS), will be made in earth-fixed coordinates, however. To relate these, we need to be able to transform the vehicle velocity from the vehicle coordinate system to the earth-fixed coordinate system, which has an inertial (i.e. non-accelerating) reference frame. Equation 10 shows, how this can be achieved by a simple rotation using the heading ψ. A two-dimensional treatment is sufficient, since GNSS altitude information are not relevant here. The transformation extracts the east-/northward components of the local velocities and adds them.

$$\begin{bmatrix} {}^{E}v_x \\ {}^{E}v_y \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} {}^{V}v_x \\ {}^{V}v_y \end{bmatrix} \tag{10}$$

2.1.5 Cornering Acceleration

The motion of a rigid body in the earth-fixed reference frame can be understood as a composition of a rotation and a translation, in our case with the CG as center of rotation. In vehicles, this rotation results from cornering. The acceleration as experienced in the CG is the result of the direct linear acceleration and the centripetal force resulting from the rotation, as shown in equation 11 [3, p. 146]. Other than in equation 5, the radius R is not the distance to the CG but the corner/curve radius. When driving straight, the radius approaches infinity and the centripetal force vanishes. Since R is usually not known, we can get rid of it using $R = v_\perp/\dot{\psi}$ with the tangential velocity $v_\perp$.

$$a_x = \dot{v}_x - \dot{\psi}^2 R = \dot{v}_x - \dot{\psi} v_y \tag{11a}$$

$$a_y = \dot{v}_y + \dot{\psi}^2 R = \dot{v}_y + \dot{\psi} v_x \tag{11b}$$

2.1.6 Velocity Estimation from Motor Speeds

While there is a plethora of methods for measuring vehicle velocity, such as optical cross-correlation, radar, 5^{th} wheel and GNSS, most of them are too expensive for widespread use. Therefore, velocity estimation in many automotive applications is based mainly on motor or wheel speed measurements, which are readily available, especially in electric cars. Equation 12 shows arguably the simplest method to estimate the longitudinal velocity,

involving only the dynamic tire radius r_{dyn}, the wheel speed ω_{wheel}, the motor speed ω_{motor} and the motor-to-wheel ratio i_{gear}.

$$v_x = \omega_{wheel} r_{dyn} = \frac{\omega_{motor}}{i_{gear}} r_{dyn} \tag{12}$$

This method can only be used for a crude approximation of the longitudinal velocity component. It falls short when the wheel slip, i.e. the relative motion of tire and road surface, increases or in the most severe case even locks up. While the previously mentioned methods are infeasible in most cases, they can be used as ground truth to evaluate estimation methods.

A simple improvement is averaging over all four wheels. More advanced methods are presented in [4]. For transient maneuvers with high slip, wheel speed data can be augmented with acceleration measurements. The authors present three approaches. The first estimator uses equation 12 until a high-slip situation occurs, at which point acceleration measurements are integrated over time with the last wheel-based velocity as initial condition. The second estimator reduces noise through a Kalman filter which averages both measurements, weighted by a constant obtained with fuzzy logic. Finally, the third estimator reduces adverse effects of acceleration bias and an incorrect dynamic roll radius through regression.

2.2 Estimation Algorithms

Obtaining the true state of some physical system is next to impossible. Any sensor measurement contains noise and other errors, which distract from the underlying process. Therefore, the challenge is discerning errors from the true value. This is known as *filtering problem*, where we want to obtain the best estimate $\hat{x}_k$ of the system state x_k at time step k based on past observations $z_i, i \leq k$ [5, p. 67]. Usually, this involves the fusion of multiple sensors and physical models. In this section we explore common estimation methods to solve the filtering problem problem. We will regard their discrete-time instead of their continuous-time versions, since digital observations are based on sampling with a finite rate.

2.2.1 Extended Kalman Filter

The Kalman filter [6] is the most widely used estimation algorithm [7, p. 401]. First developed for trajectory estimation in the Apollo space program, it now finds application

in most vehicles, aircraft, spacecraft, but even in finance and econometrics. The Kalman filter assumes a linear system, which is often not the case, or only true in a small region of the full range. Therefore, a non-linear version based on the same ideas has been developed with the extended Kalman filter (EKF), which we will regard in this section.

The idea of the EKF is simple yet powerful: we observe the physical system through noisy measurements but predict the state with a mathematical model of the system. At every time step, we fuse the observations with the predictions based on our confidence in either, basically resulting in a weighted average. For example, a noisy measurement is rather unreliable, while models can never capture all nuances and uncertainties. The fusion yields a good estimate of the true state of the system, better than only measurements or predictions could alone.

The model describes the *state* x_k of the system at time k. For example, a train on a track can be described by its position and velocity which are related, so the state is two-dimensional. How the state evolves over time according to a process is shown in the *process equation* 13. The process function $f(x_{k-1}, u_k)$ propagates the previous state x_{k-1} and is controlled by external inputs u_k, e.g., the train engine accelerating the train, and disturbed process noise w_k, which represents unmodeled behavior, e.g., air resistance or friction. The process noise is assumed to be Gaussian with zero mean and a diagonal covariance matrix Q_k, i.e no noise terms are correlated.

$$x_k = f(x_{k-1}, u_k) + w_k \tag{13}$$

While the state is inherent to the system, it can not be directly observed. Rather, we have *observations* z_k. The elements of z_k and x_k must not necessarily coincide, i.e. the observation space and state space can differ; for example, we might have multiple observations of the train's velocity but none of the position. The measurement is described by the *measurement equation* 14. The measurement function $h(x_k)$ gives the observation vector which is disturbed by measurement noise, e.g., resulting from the sensor itself or transmission errors. The measurement noise is assumed to be Gaussian with zero mean and a diagonal covariance matrix R_k. We use the measurement equation to be able to fuse real measurements with pseudo-measurements from our model.

$$z_k = h(x_k) + v_k \tag{14}$$

The process and measurements equations acknowledge, that the state can only be modeled and observed with some degree of uncertainty. Therefore, in addition to the state estimate $\hat{x}_k$, we want to keep track of its covariance P_k, which is zero mean as well. Together with

linearizations, this assumption allows the EKF to be much more computationally efficient than more generally applicable algorithms like the particle filter.

The EKF algorithm [8, p. 16 ff.] is recursive, with each iteration comprising a prediction step and a correction step, i.e. the previous state is first propagated to the current time step using the model and then corrected with the measurements from the real system. In the following description of the algorithm, we distinguish between the predicted state estimate $\hat{x}_k^-$ and covariance P_k^-, and the corrected state estimate $\hat{x}_k^+$ and covariance P_k^+.

Prediction At the beginning of every iteration, the previous corrected state is propagated using the process function to predict the current state (equation 15). Note that the prediction only depends on the directly previous state, not any other states in the history, revealing that the EKF assumes an underlying Markov process.

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+, u_k) \tag{15}$$

The previous covariance also needs to be propagated, since the estimate uncertainty might change with every step (equation 16). The Jacobian matrix F_k of the process function, which can be though of as a vector derivative, needs to be evaluated for this step (equation 17). Additionally, the process noise needs to be accounted for model mismatches, since there are always aspects of the real-world which cannot be modeled. Therefore, leveraging the additive property of Gaussian noise, the process covariance Q_k is added to the propagated covariance.

$$P_k^- = F_k P_{k-1} F_k^T + Q_k \tag{16}$$

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}_{k-1}^+} \tag{17}$$

The measurement function then generates pseudo-measurements from the estimated state (equation 18).

$$\hat{z}_k = h(\hat{x}_k^-) \tag{18}$$

Correction Once the prediction has been made, it can be corrected with observations obtained from the real system by taking a weighted average of both. The weighting is determined by the *Kalman gain* $K_k \in [0, 1]$ (equation 19), which also maps back from the observation space into the state space using the Jacobian matrix H_k of the measurement function (equation 20). The Kalman gain is defined as the ratio of the state-innovation covariance P_{xy} to the innovation covariance P_{yy}, and so it captures the uncertainty in the measurements and the model prediction. When the

measurement covariance R_k is small relative to the model covariance, the Kalman gain is high and so the measurement is weighted more. On the other hand, when the measurement noise is high, the Kalman gain approaches zero and the model prediction is weighted more.

$$K_k = \underbrace{P_k^- H_k^T}_{P_{xy}} \underbrace{(H_k P_k^- H_k^T + R_k)}_{P_{yy}}^{-1} \tag{19}$$

$$H_k = \left.\frac{\partial h}{\partial x}\right|_{x=\hat{x}_k^-} \tag{20}$$

In the best case, the model and the measurements coincide. However, this is usually not the case, so the real measurements z_k and the pseudo-measurements $\hat{z}_k$ differ, resulting in an *innovation* y_k, also called *residual*. The absolute innovation is higher when there is a larger mismatch between the model and the real system. The Kalman gain now determines, how much of the residual is used to correct the prediction. The posterior state estimate $\hat{x}_k^+$ is the final, best estimate for the iteration k and will be used in the next time step (equation 21).

$$\hat{x}_k^+ = \hat{x}_k^- + K(\underbrace{z_k - \hat{z}_k}_{y_k}) \tag{21}$$

Additionally, the covariance matrix needs to be updated to account for the correction (equation 22, I being the identity matrix). The covariance is reduced based on the Kalman gain, signifying how the certainty in our estimate has increased by the correction.

$$P_k^+ = (I - KH_k)P_k^- \tag{22}$$

2.2.2 Unscented Kalman Filter

While the EKF has been successfully applied to problems for a long time, it suffers a number of limitations [7, p. 402]:

- The error propagation in equation 16 is only reliably if the linearization in form of the Jacobian matrix F_k is sufficiently accurate. This might not be the case in highly non-linear systems or under the wrong initial estimate. In the worst case, the estimate might even diverge.

- Linearization can only be applied to differentiable functions, i.e. processes for which a Jacobian exists. Discontinuities and singularities might exist in some cases, making differentiation impossible.
- The derivation of Jacobian matrices can be very difficult and error-prone for complex systems, both the equations themselves and the conversion to code.

Therefore, alternatives to the EKF have been developed. A promising algorithm is the unscented Kalman filter (UKF) [9].

At its core, the UKF's improvement relies on the unscented transform (UT). This transformation is a method to approximate the properties of a statistical distribution undergoing a non-linear transformation. A small set of points ($2n + 1$ for an n-dimensional random variable) is sampled from the distribution using a specific, deterministic algorithm (not randomly as in Monte Carlo algorithms). These points contain second-order information about the distribution and can be transformed to capture properties such as mean and variance of the resulting distribution. Therefore, the mean is calculated to a higher order of accuracy than in an EKF, whereas the covariance is calculated to the same order of accuracy. Through the UT, linearization of the Jacobian is rendered unnecessary, which aids the filter's numerical stability in non-linear systems and enables application in discontinuous systems. Furthermore, no manual derivation of the Jacobian is necessary. Still, the efficiency and accuracy is equal or better than the EKF.

The UKF leverages the UT to adjust the estimate and covariance in both the prediction and correction step. While the general structure of the Kalman filter including prediction and correction remains, the sigma points are now used in every step.

The UKF has been shown to outperform the EKF in terms of accuracy in a large number of applications. While it too struggles with second-order moments, only the second-order EKF [10] can match its accuracy, but is more computationally complex [11]. Therefore, the EKF's popularity over the objectively better UKF can probably attributed to its long existence.

2.3 Outlier Detection

The quality of the estimate produces by the estimation algorithm is limited by the quality of its input data, especially if the algorithm is not robust. In practice, sensor data can contain anomalous sensor samples, also called *outliers*. While these samples might actually reflect reality, they usually stem from errors during measurement or transmission. In this section, the term *outlier* will refer to bad samples that should be discarded, not surprising

but correct samples which can occur in random processes. The challenge is to discern these two while minimizing the number of false positives (leads to discarding good measurements) and false negatives (degrades estimate quality). This improves estimation quality and can be crucial in safety-critical applications. This section presents common outlier types and methods to detect them.

2.3.1 Outlier Types

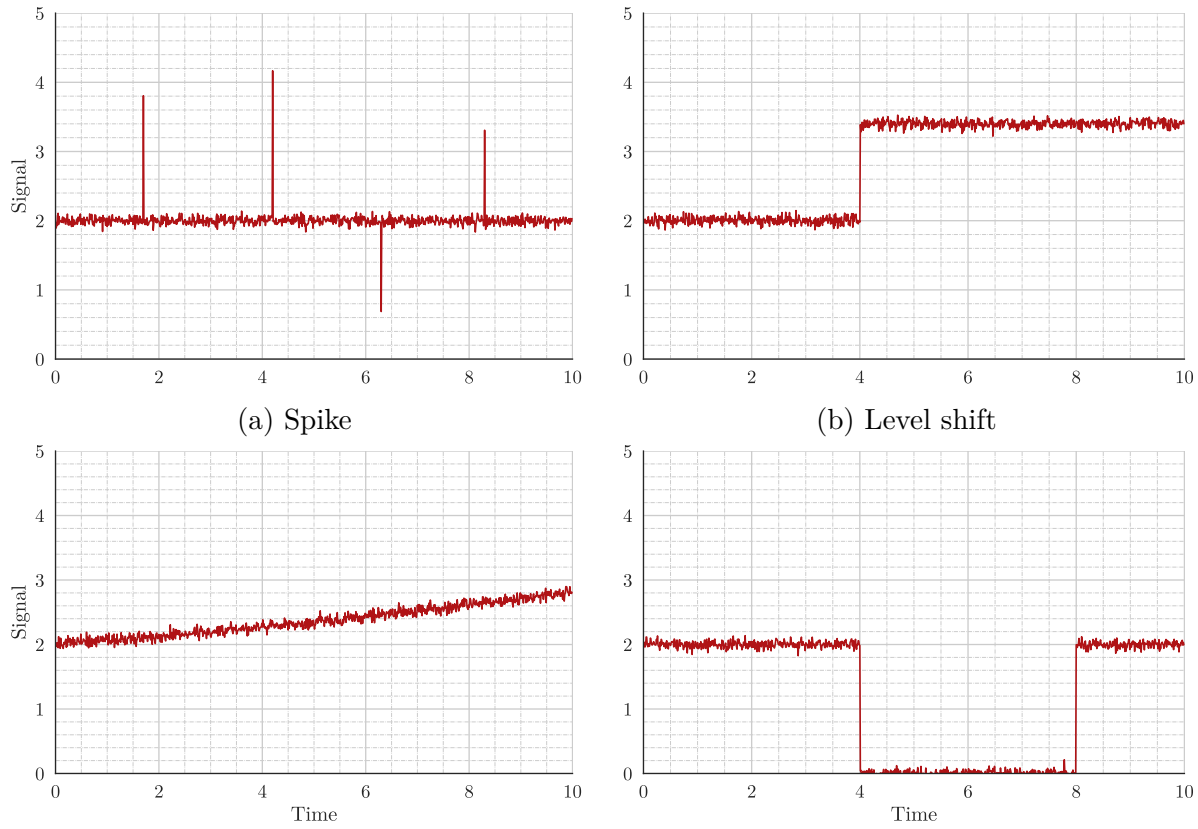
Outliers can be the result of a variety of issues. For example, heavy vibration and shocks might cause acceleration spikes and an optical cross-correlation sensor might see a featureless surface, so the recognized velocity drops to zero. In cabled transmission, a physical connection might be defect, while in radio transmission, the signal might drop out temporarily or there might be electromagnetic interference. Possible causes can vary a lot between environments, but a highly dynamic environment with many mechanical and electric parts such as an electric race car is likely to produce outliers at some point.

Outliers can be classified in four broad categories [12, p. 19], [13, p. 165 ff.], shown in figure 3:

- Spike/intermittent noise: a transient deviation from previous samples
- Level shift: a transient deviation of the mean value
- Drift/spurious trends: a slow variation of the mean value over time
- Null/signal dropout: not receiving input or zero

Drift often occurs as the result of integrating a biased signal, e.g., caused by low frequency noise or temperature changes. The true mean does not change, but the accumulating error results in an observed trending mean. While there are other cases of poor data quality such as clipping and excessive noise, they rather stem from calibration problems and hardware errors, so they are not regarded further.

Outliers can be temporary or persistent. While spikes are inherently temporary, outliers of the other three categories might or might not return to normal. For example, a GPS signal can drop out because the sensor has no satellite connectivity for a couple of seconds but then recovers (as in figure 3d), while another sensor can be permanently damaged and will not recover (as in figure 3b). Persistent but not transient outliers are hard to detect, and especially spurious trends cannot easily be distinguished from real trends by their nature. This motivates the need for two separate methods, an approach that is also employed by [12]:



(a) Spike

(b) Level shift

(c) Drift

(d) Null

Figure 3: Outlier types

- A simple but strict method for transient outliers
- A complex but lenient method for persistent outliers

With this dual approach, false negatives for easy-to-detect transient outliers can be minimized while false positives for hard-to-discern persistent outliers can be avoided. Note that the choice of detection thresholds influences the strictness and therefore sensitivity to a high degree.

2.3.2 Rudimentary Methods

Arguably the simplest method is the range check, shown in equation 23. A plausible interval $[\tau_{min}, \tau_{max}]$ of values is defined in advance, and if the measurement z is outside of that range, it is marked as outlier. For example, a speed of $200\,\mathrm{m\,s^{-1}}$ is highly unlikely for a race car, and also high negative speeds are implausible.

$$\tau_{min} < z < \tau_{max} \tag{23}$$

Another rudimentary method for detecting spikes is the difference of consecutive samples z_t, z_{t-1}, shown in equation 24. When the difference is higher than the maximum plausible change rate τ, the current measurement is marked as outlier.

$$|z_t - z_{t-1}| < \tau \tag{24}$$

Performing a sanity check is a good first approach to detect gross outliers. However, the thresholds should be set rather high to avoid false positives.

2.3.3 Statistical Methods

Methods based on the statistical properties of a signal can provide more granular checks than the previous rudimentary methods. One of the simplest robust statistics is the median, i.e. the center value when sorting a list of values, which is used in the method shown in equation 25 [14, p. 142]. An expected value $\tilde{z}_t$ is calculated using both the median of the last k samples and the median of the differences of the last $k+1$ samples. If the difference between the expected and the actual value exceeds the threshold, the current sample is marked as outlier. The window size k influences the robustness but also the detection delay, and should therefore be chosen carefully. Note that, once the outlier persists for more than k samples, following measurements will be regarded as valid.

$$|z_t - \underbrace{median(z_{t-k}, \ldots, z_{t-1}) + k \cdot median(z_{t-k} - z_{t-k-1}, \ldots, z_{t-1} - z_{t-2})}_{\tilde{z}_t}| < \tau \tag{25}$$

If an EKF is used, a chi-squared test can be used to detect anomalies based on the residuals, as shown in equation 26 [15, p. 4292], [16, p. 2050 f.]. Since the residual y is Gaussian distributed with the innovation covariance matrix P_{yy} (for nomenclature, refer to section 2.2.1), the normalized sum of squares of the residual values should be distributed according to a chi-squared distribution χ^2 with as many degrees of freedom as there are measurements. An outlier is detected when the chi-squared test at a significance level of τ failed.

$$y = z - h(\hat{x}^-) \tag{26a}$$

$$P_{yy} = HP^-H^T + R \tag{26b}$$

$$r^T P_{yy}^{-1} r < \chi^2(\tau) \tag{26c}$$

Drift detection becomes easier when multiple sensors measure the same variable and can be compared. A simple variance-based approach based on this idea is shown in equation

27 [12, p. 20]. For each time step, the summed variance of n sensors is calculated, based on the mean μ_z of their measurements at that instant. Once the threshold $\tau \in (0, 1)$ is exceeded, the measurement with the highest contribution to the summed variance is marked as outlier.

$$\sum_{i=1}^{n} (z_i - \mu_z)^2 < \tau \tag{27}$$

The last approach we will regard in this chapter can be used for spike and drift detection, but requires $n > 2$ sensors for a variable like the previous method. Multiple instances of the same EKF, collectively called a bank of n EKFs, are executed in parallel. However, in the i-th filter, the measurement of the i-th sensor is disabled. In the event that an outlier in the i-th measurement occurs, all filters except the one that does not incorporate the faulty measurement will show a very high residual.
Equation 28 [17, p. 3] shows how the sum of squared residuals[1] $NSSR^i$, normalized by the diagonal measurement covariance matrix R^i, is calculated from the residual y_i of the i-th filter. The NSSR combines all elements of the residual vector into a single metric, enabling easy comparison with a threshold τ_i. This threshold can be chosen empirically, or based on a desired statistical significance of the resulting χ^2 distribution [18, p. 3]. Note that a more sophisticated approach to fault isolation, i.e. locating the sensor producing outliers, is required for the EKF bank to work with multiple simultaneous outliers.

$$y^i = z^i - h^i(\hat{x}^-) \tag{28a}$$

$$\underbrace{(y^i)^T (R^i)^{-1} y^i}_{NSSR^i} < \tau_i \tag{28b}$$

[1] While the original paper talks about a *weighted* sum of squared residuals, we omitted the weight coefficient in favor of an adjustable threshold.

3 Design

In this chapter, we will look at the design of the state estimation and the vehicle platform it will be deployed to. The state estimation is built from the ground up, so there is a lot of flexibility regarding the design choices. The design is deliberately described in terms of generic components, enabling adoption on other platforms and software environments.

3.1 Vehicle Platform

The state estimation described in this thesis will be deployed to two electric 4WD race cars:

- a non-autonomous race car which is newly designed and manufactured
- an autonomous race car which is already existing but repurposed

These vehicles will in the following be referred to as EV and DV, respectively. Note that the DV is electric as well but has additional components required for autonomous driving.

3.1.1 Sensor Setup

The vehicles are equipped with a plethora of sensors, which can be used for state estimation. While some sensors like the ones for brake pressure are not relevant, most provide useful information. The relevant sensors are listed in table 1, with the last two columns denoting in which vehicle they are available. The sensors' locations in the vehicle shown in figure 4 In the ideal case, both vehicles would have the all sensors, since working with a homogeneous set of measurements is simpler, but financial and weight considerations mean that only the sensors necessary for the vehicle's use case are available.

Sensor type	Measured variables	EV	DV
Inertial measurement unit with gyrometer	a, ω	×	×
Optical cross-correlation velocity sensor	v, β	×	
Motor speed sensor	ω_{motor}	×	×
Global navigation satellite system	$p, \|v\|$	×	×
Global navigation satellite system	ψ		×

Table 1: Sensor setups for EV and DV

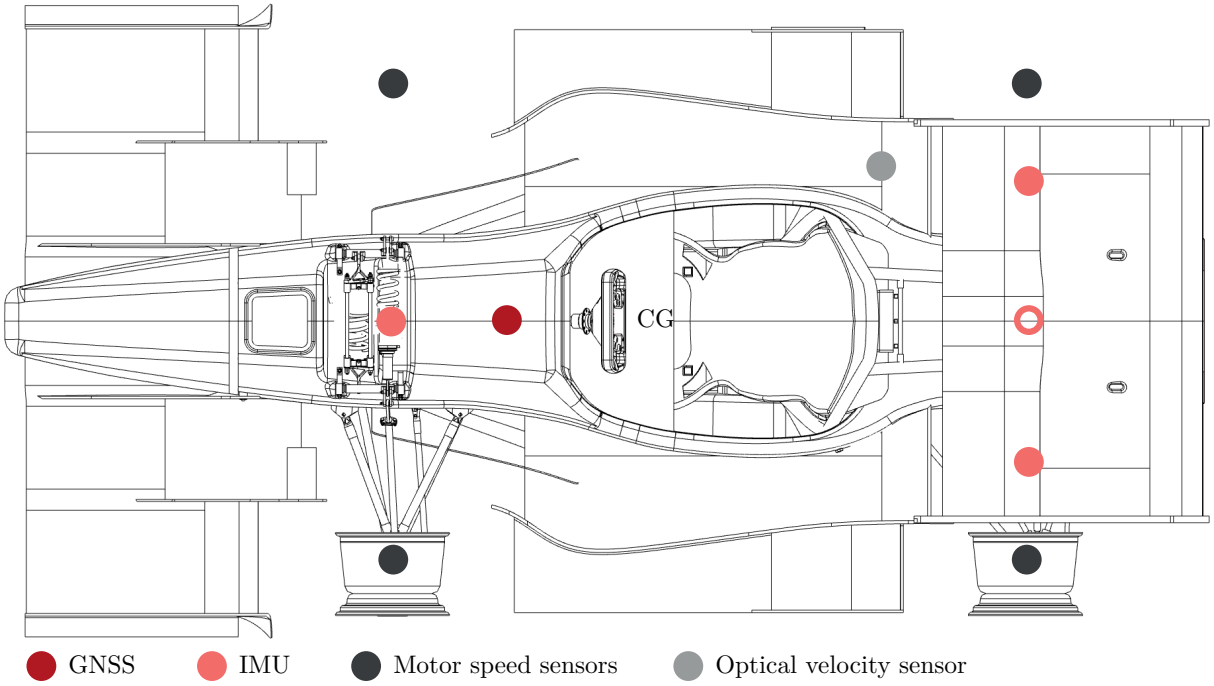


Figure 4: Locations of sensors in vehicle

IMU The EV features three inertial measurement units (IMUs) with gyrometers: one directly in front of the CG, and two situated on the rear left and right. This enables redundancy, since accelerations and rotations in the two-dimensional plane only require two IMUs. In the DV, the front one remains while only a single rear one, denoted by the circle with the missing center, is located directly behind the CG. While these sensors react very fast, the signals are rather noisy [19, p. 19 ff.].

Optical velocity sensor The optical cross-correlation velocity sensor, only available in the EV, provides longitudinal and lateral speed measurements, and therefore also measurements of the vehicle sideslip angle. It enables slip-free velocity measurements by correlating photosensor information of the surface over time, which uniquely determines the speed and direction of movement [20]. The fact that it is not influenced by slip, as velocity measurements from wheel speeds are, makes it a very valuable addition to the sensor setup. However, it is noisy and prone to temporary spikes and even failures on feature-less surfaces (see dark red line in figure 5 at 5.2 s).

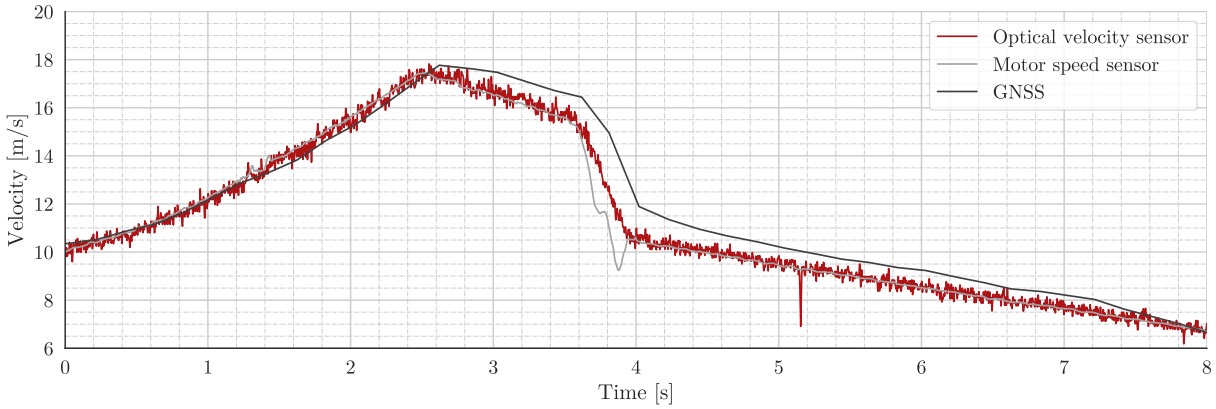


Figure 5: Comparison of velocity measurements from different sensors

Motor speed sensor The rotary encoders in each of the four motors give individual rotation speed measurements for the four wheels. They are available in both vehicles and are assumed to be reliable, since the vehicle will not drive when they fail. However, when using them to calculate the vehicle velocity, deviations due to slip occur in highly dynamic situations (see light grey line in figure 5 at 3.8 s)

GNSS A GNSS receiver is mounted in the front of both vehicles. Next to position measurements, they provide another source of speed information. The position accuracy is high in the range of a few centimeters. However, they are slow to react, with over 100 ms of delay in some situations [19, p. 27] (see dark grey line in figure 5). The receiver mounted in the DV additionally provides heading information, made possible by the relative position of two separated antennas mounted in the front and rear. This enables transformation of the speed into a velocity vector.

To summarize, the key differences between the EV and DV are as follows:

- Three IMUs in EV but two IMUs in DV
- No optical cross-correlation velocity sensor in DV
- No heading information in EV

The lack of the optical velocity sensor in the DV is inconvenient but not fatal, since the state estimation can compensate it. For track and obstacle recognition, the autonomous DV is furthermore equipped with stereo cameras and lidar sensors. These could be used for optical flow computations to gain more position, attitude and velocity information, but for this design, we rely solely on the sensors described in the previous paragraphs. The estimated state is, however, used to correct lidar measurements.

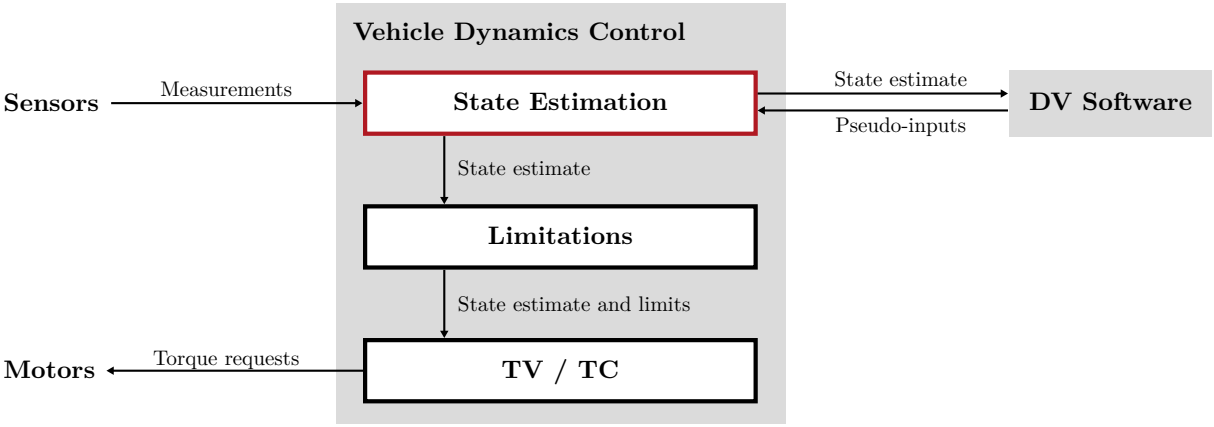


Figure 6: Integration of state estimation in software/hardware system

3.1.2 Computation System

All sensor and control signals converge at the central electronic control unit (ECU). The software running on that ECU is called vehicle dynamics control (VDC) and has several important tasks:

- Receive driver input from steering wheel and pedals
- Send torque requests to each of the four motors
- Limit speed and torque depending on situation
- Increase vehicle agility through torque vectoring (TV)
- Optimize use of tire potential through traction control (TC)

Ultimately, the VDC's task is to help the driver to exploit the maximum physical potential of the vehicle through its TV and TC, collectively called performance components.

The state estimation is located between the sensor inputs and the aforementioned performance components, and is therefore part of the VDC as well. The integration with input and output signals is shown in figure 6. While the VDC is executed at a fixed rate, sensor inputs may arrive via a controller area network (CAN) bus system at different rates. For example, the VDC runs at 1000 Hz, i.e. it is scheduled to be executed every 1 ms, but optical velocity sensor measurements may arrive at 250 Hz and GNSS measurements even slower at 5 Hz. This may be due to a measurement overhead, such as satellite communication in case of the GNSS, which prohibits higher rates, but it may also be a conscious choice to reduce bus load and thus congestion. The state estimation must be able to fuse these measurements at different rates while providing continuous estimates at the highest rate. The time since the arrival of the last measurement is provided to aid this task.

Variable	Symbol	Unit	Coordinate system
Position	x, y	m	earth-fixed
Heading	ψ	rad	earth-fixed
Linear velocity	v_x, v_y	$\mathrm{m\,s^{-1}}$	vehicle
Linear acceleration	a_x, a_y	$\mathrm{m\,s^{-2}}$	vehicle
Yaw velocity	$\dot{\psi}$	$\mathrm{rad\,s^{-1}}$	–
Yaw acceleration	$\ddot{\psi}$	$\mathrm{rad\,s^{-2}}$	–

Table 2: State variables to be estimated

Note that the software to control autonomous driving in the DV runs on a separate, more powerful computer, which receives the state estimate from the ECU. The ECU then receives pseudo-driver inputs back from the motion controller and sends torque requests to the motors. However, the interface with the DV software is beyond the scope of this thesis.

3.2 Requirements

The goal of the state estimation is to provide a robust and accurate estimate of the vehicle state. This estimate will be used by the performance components of the VDC and the DV software, which therefore dictate the required state variables, which are listed in table 2. All quantities refer to the CG, which is regarded as center of the vehicle. However, all angular variables are valid for every point on the vehicle because it is assumed to be a rigid body. The position is defined relative to a reference point, but can easily be translated. Note that the vehicle state is only defined in the plane of two dimensions, since no three-dimensional state is required by subsequent components and vertical dynamics are negligible.

Given its goal and environment, several requirements must be fulfilled by our state estimation design:

1. Accurate estimation of vehicle state: the best possible estimate of the vehicle state given the current measurements and physical knowledge is required for maximal performance

2. Support for flexible sensors: both the EV and DV sensor setups must be supported, in the best case the state estimation should handle an arbitrary setup

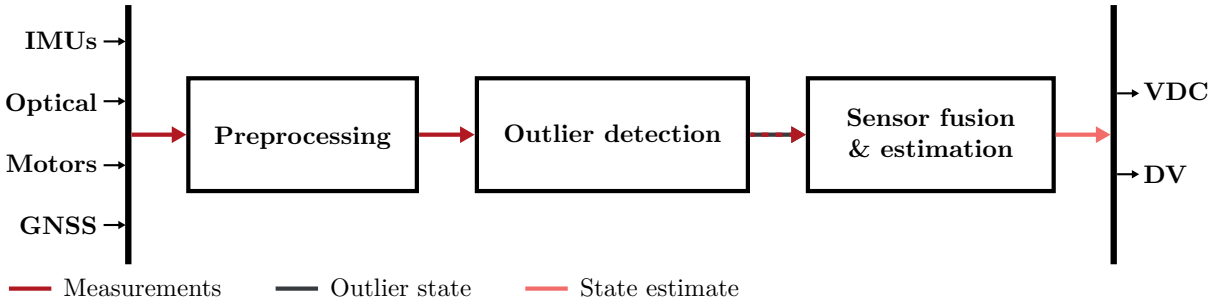


Figure 7: Architecture of state estimation

3. Robustness towards outliers and failures: outliers and sensor failures may occur at any time and should not have an adverse effect on the estimation, which requires detection and appropriate handling

4. Support for variable sampling rates: measurements arriving at different rates must be supported, as mentioned in the previous section

Additionally, our philosophy for the design and architecture is a preference for simplicity, according to Occam's razor ("the simplest solution is most likely the right one"). This is especially true if two methods work equally well but one is simpler. Only if simple methods do not yield the desired result, try more complex and complicated approaches. This philosophy has numerous benefit. A simple solution is one that makes less assumptions, and therefore generalizes better. It is also easier to understand, reason about and troubleshoot. This is of special significance for the application of the state estimation in a Formula Student team, where members and component maintainers change yearly and cannot dive deep into every topic. We apply this philosophy to small components but also to the high-level architecture, which facilitates maintenance and extension.

3.3 Architecture

Following our preference for simplicity, we propose a three-stage architecture, shown in figure 7. Measurements arriving from different sensors are first preprocessed, which, for example, includes unit conversions and coordinate transformations. In the next steps, outliers are detected. The outlier state informs the fusion of the measurements into an accurate state estimate (requirement 1), which is then used by the subsequent VDC's performance components and DV software.

A key feature is the unified mechanism for outlier and sensor setup detection (requirements 2 and 3). Foundation is the realization that ultimately it does not matter if a measurement is unavailable due to an invalid signal or a missing sensor, therefore both cases can be

treated the same. This allows a focused effort on a single component while reducing complexity.

The following sections describe each component in detail.

3.3.1 Preprocessing

Goal of the preprocessing stage is to harmonize all measurements. Mostly, this is only a simple conversion to SI units, e.g., velocities from $\mathrm{km\,h^{-1}}$ to $\mathrm{m\,s^{-1}}$, or angles from deg to rad. Doing these conversions as early as possible simplifies subsequent computations and reduces error potential due to wrong units. For other signals, more preprocessing is necessary.

IMU The most sophisticated preprocessing is required for the IMUs. The measurements are first rotated in three dimensions to correct any misalignment with the vehicle axes. Then, we need a generic IMU fusion algorithm which can take an arbitrary number of sensors with known positions and fuse them for a better estimate of the linear accelerations, yaw rate and yaw acceleration. An accurate estimate of these variables is important, because they are extensively used in the sensor fusion and performance components. This helps with reducing noise, but also enables calculation of the angular acceleration, which cannot be measured directly with our sensors. We always calculate the linear acceleration and angular velocity in three dimensions, and, in case of more than two IMUs, the angular acceleration as well. We provide two approaches for IMU fusion with the same interface, enabling drop-in replacement. Since the available IMUs need to be known at this point, performing the outlier detection here instead of the outlier detection state is necessary.

The first approach fuses measurements by averaging, and is therefore called *mean-based IMU fusion*. First, the angular velocity measurements from the gyrometers are averaged. Then, the angular accelerations are derived using equation 8 from all combinations of two IMUs and averaged. For example, in the EV, the combinations 1–2, 2–3 and 1–3 are regarded. In case only two IMUs are available, they are assumed to have a zero z-position and the angular acceleration is calculated using the simplified equation 9. In case only a single IMU is available, we assume $\alpha = \mathbb{0}$. Finally, the linear accelerations are transformed to the CG using equation 5 and averaged as well, which eliminates the effect of tangential and centripetal acceleration.

The second *maximum-likelihood-based IMU fusion* approach is more sophisticated and is presented in [21], with an extension proposed in [22]. The idea is to first find a maximum-likelihood estimate for ω given the linear acceleration and angular

acceleration measurements of all sensors. This is done by solving a least-squares problem using the Gauss-Newton algorithm, with a fixed number of 10 iterations instead of checking for convergence. The maximum-likelihood estimate for a and α is then as simple as plugging the previously found estimate into a linear equation.

show necessary transformations for each measurement

GPS WGS84 coordinates transformed to track with first known point as reference point maybe later mechanism to set reference point while only speed instead of velocity is known, use as information for vx because vy is much smaller (β usually smaller than 0.1 rad) and still good enough for outlier detection

sfii transformed to CG

3.3.2 Outlier Detection

Show diagram of AND and OR For all EKF inputs Plausibility check for all For velocity, use wheels, GPS and sfii in EKF bank, different covs than in normal EKF Plausibility for velocities rather conservative observability and controllability goal: support one sensor failure, more is unlikely

debouncing to increase robustness allow manual three-way override to enable/disable sensors and override outlier detection errors

3.3.3 EKF

outlier detection sensor state and dt signal is used to create measurement mask (requirement 4) Show input selection and state equations, jacobians euler forward discretization GPS is not included because it is not beneficial due to its slow response kinematic model from [23, p. 156] simple without tire model: occams razor and empiric by wisch reduces effects of noise no heading measurements in ev, especially lack of initial heading is bad normalize angles

prediction using differential equation initialization disable measurements using separate mask input cov as process noise

4 Implementation

ES910 as ECU, can be programmed in Matlab/Simulink, C, ASCET-MD [24, p. 17]

Since whole VDC is in Simulink because supported by ES910, state estimation as well

Describe Simulink features: subsystems, busses, blocks

Simulink acts as glue, great for modelling signal flow

Matlab code for key algorithm implementations, more readable and unit testable, clear architecture requirement

State estimation is separate model to allow for independent development

Architecture components as subsystems

Reused components referenced model

Strict busses and datatypes, units in simulink to avoid logical errors that are hard to find at compile-time

Integration in VDC

only model-in-the-loop testing with measurement data because complete setup was not available during development for HIL

Build process and toolchain incl intecrio and inca for live telematics, monitoring and parametrization

requires application for covariances during testing

5 Evaluation

Fundamental: no ground truth, therefore residuals are a good indicator

non-zero-mean residual means model mismatch [23, p. 158]

Results

bad position in EV without heading measurements, because velocity cannot be used properly to estimate position

compare raw measurements with velocity estimate

show results for each state variable

effect of different debouncing times and thresholds

compare max change rate and mean based

max likelihood IMU fusion not better for omega and a, but less noise for alpha when more than 2 imus -> better EKF results

use mean based fusion because occams razor

show all wssrs for ekf bank

maybe compare with other outlier detection approaches

Discussion, impact of results on esleek

prepare for DV

6 Conclusion

Summary

future work

try unscented KF because race cars are highly non-linear

wheel speed stuff

integrate better with DV and use lidar/cam information

possibly reimplement in DV software in future

Bibliography

[1] ISO, *Road vehicles — Vehicle dynamics and road-holding ability — Vocabulary*, 2011.

[2] D. Gross, W. Hauger, J. Schröder, W. A. Wall, and S. Govindjee, *Engineering Mechanics 3*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, ISBN: 978-3-642-53711-0. DOI: 10.1007/978-3-642-53712-7.

[3] W. F. Milliken and D. L. Milliken, *Race Car Vehicle Dynamics*. Great Britain: Society of Automotive Engineers Inc, 1996.

[4] C. K. Song, M. Uchanski, and J. K. Hedrick, "Vehicle Speed Estimation Using Accelerometer and Wheel Speed Measurements," in *SAE Technical Paper Series*, ser. SAE Technical Paper Series, SAE International400 Commonwealth Drive, Warrendale, PA, United States, 2002. DOI: 10.4271/2002-01-2229.

[5] S. K. Mitter, "Filtering and stochastic control: a historical perspective," *IEEE Control Systems*, vol. 16, no. 3, pp. 67–76, 1996, ISSN: 1066-033X. DOI: 10.1109/37.506400.

[6] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960, ISSN: 0021-9223. DOI: 10.1115/1.3662552.

[7] S. J. Julier and J. K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004, ISSN: 0018-9219. DOI: 10.1109/JPROC.2003.823141. [Online]. Available: https://www.cs.ubc.ca/~murphyk/Papers/Julier_Uhlmann_mar04.pdf.

[8] S. S. Haykin, Ed., *Kalman filtering and neural networks*, ser. Adaptive and learning systems for signal processing, communications, and control. New York: Wiley, 2001, ISBN: 9780471221548.

[9] Simon J. Julier and Jeffrey K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Ivan Kadar, Ed., vol. 3068, SPIE, 1997, pp. 182–193. DOI: 10.1117/12.280797.

[10] M. Roth and F. Gustafsson, "An efficient implementation of the second order extended Kalman filter," in *14th International Conference on Information Fusion*, 2011, pp. 1–6.

[11] F. Gustafsson and G. Hendeby, "Some Relations Between Extended and Unscented Kalman Filters," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2012, ISSN: 1053-587X. DOI: 10.1109/TSP.2011.2172431.

[12] J. Kabzan, M. d. I. La Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart, *AMZ Driverless: The Full Autonomous Racing System*, 2019. [Online]. Available: https://arxiv.org/pdf/1905.05150.pdf.

[13] H. Himmelblau, J. H. Wise, A. G. Piersol, and M. R. Grundvig, *Handbook for Dynamic Data Acquisition and Analysis - IES Recommended Practices 012.1*. 1994. [Online]. Available: https://trs.jpl.nasa.gov/bitstream/handle/2014/33780/94-0509.pdf.

[14] S. Basu and M. Meckesheimer, "Automatic outlier detection for time series: an application to sensor data," *Knowledge and Information Systems*, vol. 11, no. 2, pp. 137–154, 2007, ISSN: 0219-1377. DOI: 10.1007/s10115-006-0026-6.

[15] K. Hausman, S. Weiss, R. Brockers, L. Matthies, and G. S. Sukhatme, "Self-calibrating multi-sensor fusion with probabilistic measurement validation for seamless sensor switching on a UAV," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4289–4296, ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487626.

[16] M. I. Valls, H. F. Hendrikx, V. J. Reijgwart, F. V. Meier, I. Sa, R. Dube, A. Gawel, M. Burki, and R. Siegwart, "Design of an Autonomous Racecar: Perception, State Estimation and System Integration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2048–2055, ISBN: 978-1-5386-3081-5. DOI: 10.1109/ICRA.2018.8462829.

[17] T. Kobayashi and D. L. Simon, Eds., *Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics*, vol. Volume 1: Turbo Expo 2003, Turbo Expo: Power for Land, Sea, and Air, 2003. DOI: 10.1115/GT2003-38550.

[18] W. Xue, Y.-q. Guo, and X.-d. Zhang, "A Bank of Kalman Filters and a Robust Kalman Filter Applied in Fault Diagnosis of Aircraft Engine Sensor/Actuator," in *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, IEEE, 2007, p. 10, ISBN: 0-7695-2882-1. DOI: 10.1109/ICICIC.2007.3.

[19] C. Biel, "Konzeptionierung und automatisierte Parametrierung der Antriebsregelung eines Formula Student Rennwagens," 2019.

[20] A. Bellof, "Method and device for determining the direction and speed of an object," DE4313497 (A1), 1994. [Online]. Available: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&CC=DE&NR=4313497A1&KC=A1.

[21] I. Skog, J.-O. Nilsson, P. Handel, and A. Nehorai, "Inertial Sensor Arrays, Maximum Likelihood, and Cramér–Rao Bound," *IEEE Transactions on Signal Processing*, vol. 64, no. 16, pp. 4218–4227, 2016, ISSN: 1053-587X. DOI: 10.1109/TSP.2016.2560136.

[22] J. Wahlstrom, I. Skog, and P. Handel, "Inertial Sensor Array Processing with Motion Models," in *2018 21st International Conference on Information Fusion (FUSION)*, I. C. o. I. Fusion, Ed., Piscataway, NJ: IEEE, 2018, pp. 788–793, ISBN: 978-0-9964527-6-2. DOI: 10.23919/ICIF.2018.8455269. [Online]. Available: https://www.researchgate.net/profile/Johan_Wahlstroem3/publication/327483205_Inertial_Sensor_Array_Processing_with_Motion_Models/links/5cd575a092851c4eab924a23/Inertial-Sensor-Array-Processing-with-Motion-Models.pdf.

[23] Alexander Wischnewski, Tim Stahl, Johannes Betz, and Boris Lohmann, "Vehicle Dynamics State Estimation and Localization for High Performance Race Cars," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 154–161, 2019, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.08.064. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896319303957.

[24] ETAS GmbH Stuttgart, "ES910.3-A Prototyping and Interface Module User's Guide," 2018. [Online]. Available: https://www.etas.com/download-center-files/products_ES900/ES910.3-A_UG_R09_EN.pdf.

Glossary

performance components

the collective term for TV and TC, the components of the VDC which maximize the vehicle's performance

A Bus Definitions

Bus definitions

B Expanded Rigid Body Equations

B.1 Transformation of Linear Velocity

$$\begin{aligned}\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} &= \begin{bmatrix} v_x^{CG} \\ v_y^{CG} \\ v_z^{CG} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \\ &= \begin{bmatrix} v_x^{CG} \\ v_y^{CG} \\ v_z^{CG} \end{bmatrix} + \begin{bmatrix} \dot{\theta} r_z - \dot{\psi} r_y \\ \dot{\psi} r_x - \dot{\phi} r_z \\ \dot{\phi} r_y - \dot{\theta} r_x \end{bmatrix}\end{aligned}$$

B.2 Transformation of Linear Acceleration

$$\begin{aligned}\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} &= \begin{bmatrix} a_x^{CG} \\ a_y^{CG} \\ a_z^{CG} \end{bmatrix} + \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \left(\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \right) \\ &= \begin{bmatrix} a_x^{CG} \\ a_y^{CG} \\ a_z^{CG} \end{bmatrix} + \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} \dot{\theta} r_z - \dot{\psi} r_y \\ \dot{\psi} r_x - \dot{\phi} r_z \\ \dot{\phi} r_y - \dot{\theta} r_x \end{bmatrix} \\ &= \begin{bmatrix} a_x^{CG} \\ a_y^{CG} \\ a_z^{CG} \end{bmatrix} + \begin{bmatrix} \ddot{\theta} r_z - \ddot{\psi} r_y \\ \ddot{\psi} r_x - \ddot{\phi} r_z \\ \ddot{\phi} r_y - \ddot{\theta} r_x \end{bmatrix} + \begin{bmatrix} \dot{\theta}(\dot{\phi} r_y - \dot{\theta} r_x) - \dot{\psi}(\dot{\psi} r_x - \dot{\phi} r_z) \\ \dot{\psi}(\dot{\theta} r_z - \dot{\psi} r_y) - \dot{\phi}(\dot{\phi} r_y - \dot{\theta} r_x) \\ \dot{\phi}(\dot{\psi} r_x - \dot{\phi} r_z) - \dot{\theta}(\dot{\theta} r_z - \dot{\psi} r_y) \end{bmatrix}\end{aligned}$$