A real-time 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	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 real-time 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

CG center of gravity

EKF extended Kalman filter

UT unscented transform

GNSS global navigation satellite system

GPS Global Positioning System

UKF unscented Kalman filter

VDC vehicle dynamics control

List of Figures

List of Tables

List of Source Codes

List of Symbols

α	Angular accleration	$\mathrm{rad\,s^{-2}}$
$\mathbb{0}$	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
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

target device

1.1 Problem

more than xx years since first car while mindset is same, now there are electric race cars new possibilities because of 4wd and computing power individual torque on each wheel computer assist driver in getting max performance 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 failure 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.

$$p = \begin{bmatrix} x, y, z \end{bmatrix}^T \tag{1a}$$
$$v = \begin{bmatrix} v_x, v_y, v_z \end{bmatrix}^T \tag{1b}$$
$$a = \begin{bmatrix} a_x, a_y, a_z \end{bmatrix}^T \tag{1c}$$

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

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 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).

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

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.

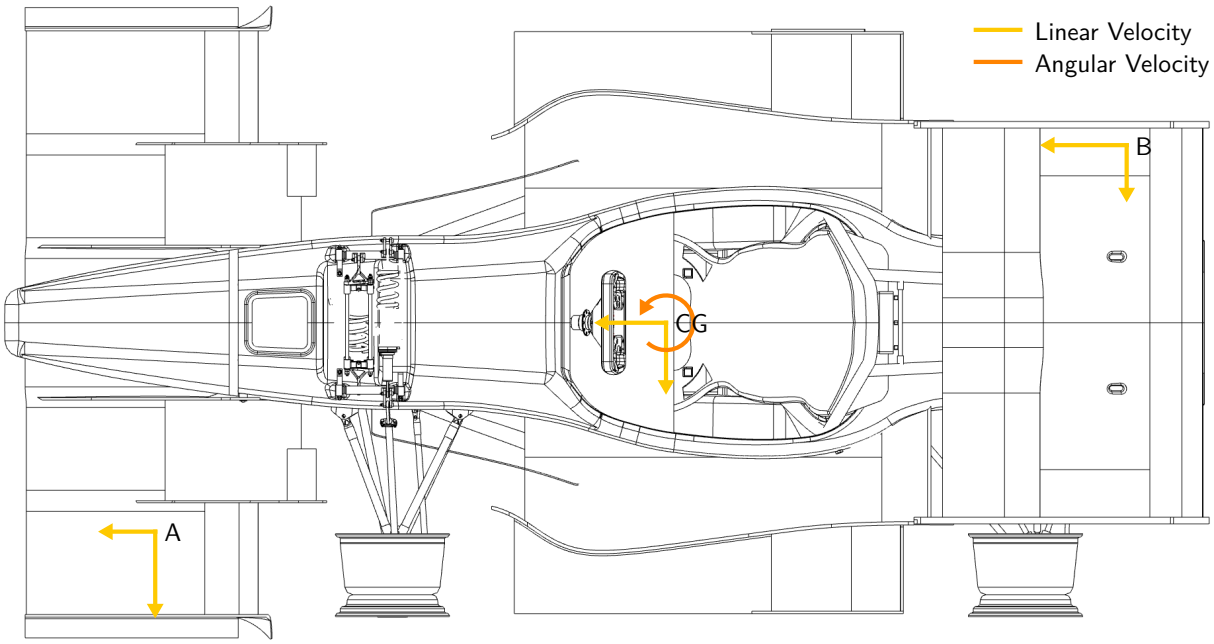

Figure 1: Experienced velocities at off-center points

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

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 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.

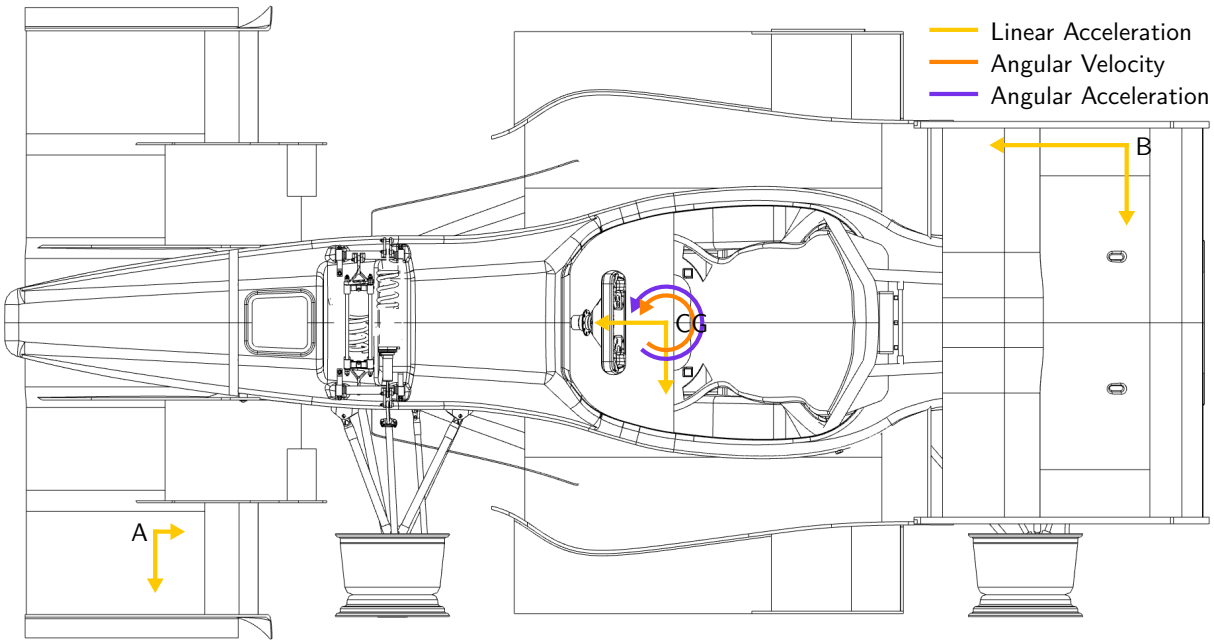

Figure 2: Experienced acceleration at off-center points

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

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 Components of Vehicle 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. 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 Failure Detection

causes: missing connection, no feature correlation for sfii, electromagnetic interference

outlier types: persistent, transient [12, p. 170 ff.] outlier, drift, null [13, p. 19 f.] transient is easier to detect separate mechanisms

statistical methods

transient: range and

EKF bank

3 Design

state estimation to be designed will be deployed in two cars

non-autonomous EV, newly built -> sensors can be chosen

partly autonomous DV, reuse old car with camera/lidar -> fixed set of sensors

3.1 Vehicle Characteristics

car is equipped with a plethora of sensors, but only some are important to us

e.g., sensors for brake pressure or accelerator pedal actuation not relevant

top down view of car with sensor positions

table: sensor name, sensor type, measured variables, columns EV and dv with X

refer to characterization in TC paper

sfii is key sensor for velocity estimation because it is slip free as opposed to wheels and fast as opposed to GPS

(compare velocity with only gps, sfii and whee)

ES910 as ECU, can be programmed in Matlab/Simulink, C, ASCET-MD [14, p. 17]

runs vehicle dynamics control (VDC) with TC, TV, motor request...

The VDC is a tool that helps the driver to exploit the maximum physical potential of the vehicle.

state estimation will be part of VDC

In DV vehicle, DV software runs on a separate computer that needs part of state as well

inputs from sensor via CAN at different frequencies to avoid congestion

it is unclear whether sfii will be in DV

3.2 Requirements

goal: provide robust, accurate estimate of vehicle state for VDC and DV software

flexible: support arbitrary sensor setups within full sensor setup

robust: detect and handle sensor failures

state variables to estimate

estimate these variables in CoG, this describes motion of vehicle since it is assumed to be a rigid body

real-time: needs to be computable in 1 ms frequency

complete in all areas, so no developments in near future are necessary

design principle: use simple methods if they work just as well (occam's razor)

only if the results are not adequate, try more complicated approaches

easier to understand and troubleshoot

make less assumptions and thus generalize better

because maintainers change every year

on a higher level, clear architecture to facilitate maintenance and extension

We assume that vertical dynamics are negligible and only motion in the two-dimensional plane of the track are relevant, simplifying equations.

3.3 Architecture

in the literature, there is no concrete solution for our problem

however, we can combine common components into our own solution

At high level: state estimation = preprocessing + outlier detection + state estimation

To fulfill requirements of robustness and accuracy: outlier detection and sensor fusion

For practical reasons: preprocessing

preprocessing and outlier detection for each sensor in parallel

all IMUs have same treatment

out bus creation calculates vehicle side slip angle from vx and vy

For flexibility: outlier detection and sensor setup detection using same mechanism

in both cases, sensor cannot or should not be used in sensor fusion

does not matter if not connected, not sending, invalid values

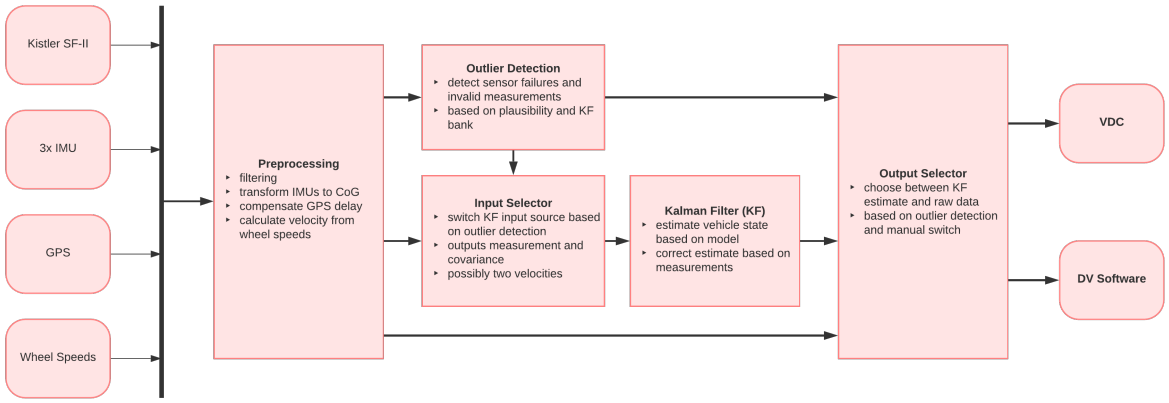

Figure 3: High-level architecture

Show all inputs and outputs

3.4 Preprocessing

convert to SI units and deg to rad so formulas can be applied

show necessary transformations for each measurement

IMU fusion

very important because used extensively in EKF and other parts of VDC like TC and TV (especially dpsi)

fast response

contains outlier detection because available sensors need to be known before fusion in preprocessing

GPS WGS84 coordinates transformed to track with first known point as origin

sfii transformed to CG

3.5 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

Plausibility for velocities rather conservative

debouncing to increase robustness

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

3.6 EKF

outlier detection sensor state is used to create measurement mask

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 [15, p. 156]

prediction using differential equation

initialization

disable measurements using separate mask

input cov as process noise

4 Implementation

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 [15, 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

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

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

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] 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.

[13] 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.

[14] 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.

[15] 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.

Glossary

target device

the device that inference will be performed on; usually an accelerator located at the edge

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

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