Automated Camera Stabilization and Calibration for Intelligent Transportation Systems

Marcel Bruckner
Technische Universität München
Boltzmannstraße 15, 85748 Garching
marcel.bruckner@tum.de

Abstract

In the emerging field of Intelligent Transportation Systems one main challenge is the fusion of different sensor types. To fuse the measurements correctly each sensor needs to be free from noise and calibrated accurately.

In this work we focus on two problems resulting from environmental influences on RGB cameras within the Providentia++ project.

First we propose an online vision-based framework to remove jitter from shaky camera streams to dynamically stabilize the video feed. We show that our approach based on visual features and an image space homographic transformation gives good stabilization results regarding the optical flow in the image. By exemplary tracking objects and measuring their travelled pixel distance we show a substantial decrease of the jittery image motion.

Second we propose an online reprojection-error based algorithm to statically calibrate RBG-only cameras w.r.t. a high definition map and to mitigate drift of the intrinsic and extrinsic camera parameters over time. By relaxing the minimization problem using a 1D-approximation of road signs we achieve high accuracy in the calibration of the cameras. We show the minimal number of needed correspondences between the video and the map, the structure they have to exceed and give a lower bound on the remaining calibration error.

Keywords— Intelligent Transportation Systems, Computer Vision, Video stabilization, Feature detection, Feature matching, Camera calibration, Reprojection-error, OpenDRIVE

1. Introduction

Within the PROVIDENTIA project, a section of the highway A9 between Munich and Nuremberg was converted to a testing site for autonomous driving. As part of this, a large sensor network system has been set up along the highway to allow monitoring and steering of traffic as well as to improve the coordination between autonomous and traditional cars. The primary task of the intelligent system is to create a digital traffic twin that accurately represents the physical road situation in real-time. Based on this digital twin, the smart infrastructure can provide a far-reaching and comprehensive view to the drivers and autonomous cars in order to improve their situational awareness within the current traffic environment. (Taken from the description)

A key challenge of ITS lies in the reliable and accurate calibration of the different sensors. The calibration is especially challenging when the sensor is subject to real-life disturbances like vibration of its mounting pole caused by wind or displacements due to temperature expansion. These disturbances introduce noise to the measurements. We focus on the implications of this noise for the vision system built upon the cameras.

The focus of the project is not to accurately view and measure the vehicles in the sensor ranges, but rather to build upon these measurements. The backend tracking the vehicles are sensible noise in the measurements despite the strong efforts of to extend the robustness using extended filtering techniques. These filters already mitigate a broad range of noise but nonetheless cannot remove all.

The projects main focus is to provide accurate inter-vehicle information and to predict the future traffic development. The digital twin modeling the state of the traffic and environment drifts over time and the calibration between the nodes and of the nodes gets less accurate.

To tackle real-life disturbances and to remove noise from the system we provide two vision based frameworks. The problems arising from disturbances can be roughly grouped into problems concerning the Dynamic Stabilization of the video feed and the Static Calibration of the camera pose.

Dynamic stabilization The dynamic shaky motions of the camera due to wind and vibrations from passing vehicles are stabilized using a digital image stabilization approach. The DIS approach is based on visual image features that are matched between the current and a keyframe. Using the feature matching the homographic transformation between the frames is computed and the current frame is warped to minimize the pixel distances between the static background scene. This enables us to mitigate the real-world motions of the camera in the image space.

Static calibration Within the project high definition maps (HD maps (2.3)) of the enclosed environment are heavily used. These HD maps (2.3) offer the real-world positions of the highway lanes, the gantry bridges, objects like poles and permanent delineators and traffic signals like speed limits or exit markers.

Due to the environmental influences on the mounting constructions as well as the gantry bridges (Explain these earlier) and the natural wear of the materials the cameras positional and rotational parameters are changing over time.

Using the spatial information of the HD maps (2.3) and a mapping to pixels in the video frame the reprojection error is minimized to find the optimal camera pose for the observations.

The code for the dynamic stabilization, static calibration and object position retrieval from the HD maps (2.3) are accessible via the two GitHub repositories: `https://github.com/Brucknem/GuidedResearch` and `https://github.com/Brucknem/OpenDRIVE`.

2. Terms and Definitions

This section provides explanations of extensively used terms and definitions.

2.1. Digital Twin

2.2. Dynamic foreground and static scene

We group parts of the world seen by the cameras into the two groups of dynamic foreground and static scene. Dynamic foreground describes all pixels that represent objects that are meant to be moving, *e.g.* vehicles on the roads. On the other hand the static scene are all pixels that are not dynamic foreground, *e.g.* the road, guardrails or bridges.

2.3. High definition road maps (HD maps (2.3))

Several HD maps are used to generate the Digital Twin (2.1). The HD maps adhere to the OpenDRIVE standard V1.4. They contain spatial and relational information between the world, roads, the road coordinate frames and objects within these frames.

3. Related Work

There are many ITS projects emerging *e.g.* Koster *et al.* at the *Testfeld für Autonomes und vernetztes Fahren in Niedersachsen* [11]. Also cooperative perception using infrastructure sensors is an emerging field [4].

None of the teams - to the best of our knowledge - provide in depth public information about their camera stabilization and calibration approaches.

In the *Test Area Autonomous Driving Baden-Württemberg* [9] (TAADBW) project multiple optical camera sensors are attached to large poles with overlapping fields of view. They calibrate their cameras relative to a high-precision map and assume the calibration and the intrinsic camera parameters to be static during runtime. The team relies on an manual a-priori selection of visual landmarks and perform calibration at system startup. The team of the TAADBW estimate the extrinsic calibration with exact world position from the map by minimizing the squared distances between the visible landmarks and the respective projected objects.

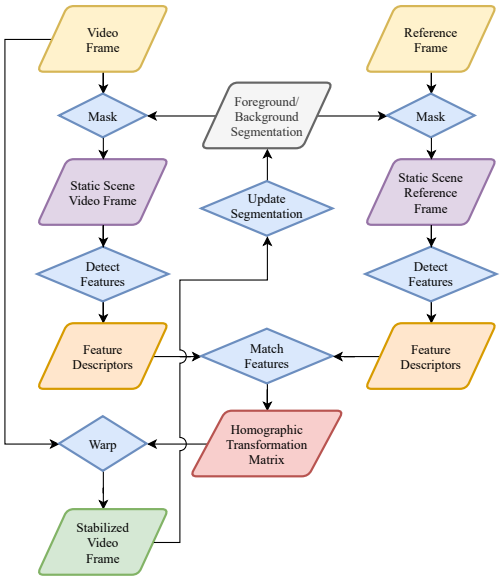

Figure 1

The large overlaps in the fields of view in their setup allow a global optimization strategy. This is not feasible in our project due to the small overlaps in the fields of view between the cameras [12] and associating pixels within the multi-view setup is an inherently hard problem.

Müller *et al.* [18] present an approach based on a cooperative intelligent vehicle. The vehicle moves through the scene and passes cooperative awareness messages from the vehicle to the infrastructure. This removes the need for overlapping fields of view completely and enables a fully automated and sensor-independent registration. The team calibrates a multitude of different sensor types to the world frame and recovers their extrinsic parameters.

Calibration between cameras and radar sensors has been solved previously by Schöller *et al.* within the *Providentia++* project [12].

4. Approach

In this paper we propose two algorithms to solve distinct problems that arise from the real-world disturbances that act upon the Intelligent Transportation System.

4.1. Dynamic stabilization

The gantry bridges to which the cameras are mounted are prone to environmental influences. These influences include, but not exclusively, wind and vibrations from passing vehicles. These external influences bring the cameras into a swinging state which introduces jittery motion in the video feeds.

Although the displacements of the camera only span a small range around its resting position, the influences in on the images amplify by the huge distances the cameras overview.

To mitigate the noise added by the jittery motion we propose the pipeline displayed in Figure 1.

Removing dynamic foreground scene We only want to find features on the non-moving static scene (2.2) objects, *e.g.* the road, poles, guardrails and bridges. As these features are expected to not move between frames, aligning these during image warping ensures that the scene stays static. Moving vehicles are thus excluded from the detection and do not contribute in the algorithm.

Hence, we mask the current video frame and the stable reference frame using a background segmentation based on the *Improved Adaptive Gaussian Mixture Model for Background Subtraction* proposed by Zoran Zivkovic *et al.* [21, 22, 6].

Additionally it speeds up the algorithm.

Maybe append more about warmup for MOG2.

Feature detection We are looking for specific patterns or specific features which are unique, can be easily tracked and can be easily compared. (Taken from OpenCV)

Hence, on the static scene (2.2) of the video frame and reference frame we perform feature detection to retrieve the descriptors and pixel locations of prominent image parts.

There exists many different implementations of feature detectors and descriptors as described by Kumar *et al.* [13]. We mainly focus on the implementations of SIFT [15], SURF [5] and ORB [20] feature detectors and descriptors, Fast [10] feature detector with FREAK [2] feature descriptors and Star [1] feature detector with BRIEF [7] feature descriptors coming from the OpenCV library [6].

Feature matching Feature matching compares the features of one frame to the features of the other frame. A match is reported if the feature descriptors of two compared features surpass a specific dynamic threshold [15] regarding some feature dependent metric [13]. These feature matches establish a spatial relationship in pixel space between the two frames.

Based on this spatial pixel relationship we can now compute the homographic transformation between the two frames. The transformation describes a perspective transformation H between $(x'_i, y'_i, 1)^T$ and $(x_i, y_i, 1)^T$ so that

$$s_i * (x'_i, y'_i, 1)^T \sim H * (x_i, y_i, 1)^T \tag{1}$$

and minimizes the back-projection error.

The implementation is included in the OpenCV library [6].

Image alignment With the given homographic transformation the current video frame is now warped so that the matches align in pixel space. As a consequence the static scene is now aligned perfectly between the two frames. As the keyframe is stable and does not change over time the current video frame is now also stabilized as the motion of the static scene is minimized.

Foreground/Background segmentation upate As a last step the foreground/background segmentation is updated using the new stabilized frame. This ensures that for the next frame the segmentation distinguishes the static scene and dynamic foreground.

This reduces the search space for the feature detectors and prohibits matches between static scene and dynamic foreground objects. Hence, the algorithm is robustified and speed up.

4.2. Static calibration

Intelligent Transportation System are inherently dependent on the calibration of the different sensors. To track and predict traffic the system has to know the poses of the different sensors relative to some reference coordinate system. This enables the ITS to accurately measure the position of vehicles within the single sensor ranges and at the overlapping boundaries.

Previous experiments have shown that a calibration process based on an IMU is not feasible in our case. Instead we focus on a calibration procedure based on visual landmarks in the video feed. The landmarks are mapped to their partially known world positions from high definition road maps.

Retrive objects from the HD maps (2.3) In this work we focus on the permanent delineator objects that are easily visible in the video feeds.

The world position of the objects can be retrieved using the mathematical operations defined in the OpenDRIVE standard.

This gives us the the base origin point $o = (x, y, z)^T$ of the object in the transverse mercator projection [19]. The base origin point is the world position of the lower end of the object where it ends in the ground or another object.

Additionally we retrieve a directional heading axis $d = (x, y, z)^T$ and the height h of the object.

These values enable us to approximate the real-world objects by sampling points $s \in S$ in world position along the center line of the object, where

$$S = \{o + \lambda * d : \quad \lambda \in [0, h]\} \tag{2}$$

Mapping objects to pixels To calibrate the camera we need to establish a mapping

$$s_c \mapsto p_c \Leftrightarrow o_c, d_c, \lambda_c \mapsto p_c \tag{3}$$

from pixels $p_c = (u, v)^T \in P$ from the video stream to the corresponding sampled points s_c from the object they belong to.

This leaves us with a set C of correspondences $\{p_c, s_c\}$.

This mapping is currently done by human interaction and not fully automated. To minimize the work an aiding system to mark pixels was implemented that outputs a list of pixels that can easily be mapped to the list of objects.

Relaxation of problem by line approximation Approximating the objects by lines removes the need for exactly known 3D correspondences as usually needed in calibration problems. Nonetheless it does not require to jointly estimate the full world position of the objects and camera pose jointly as in Bundle-Adjustment problems. The assumptions we made are:

- Objects are symmetric around their directional heading axis.
- Projected pixels of the objects are also symmetric around the projected directional axis.

We then take the mean over each row of pixels as the approximate pixel positions of the center line.

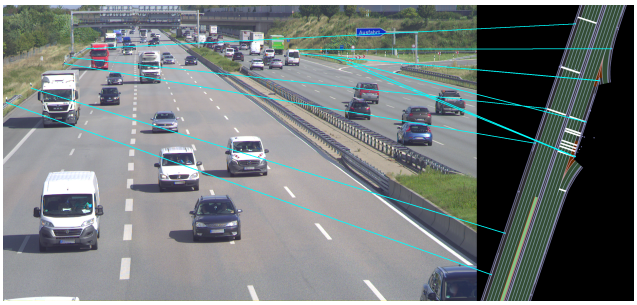

Figure 2: Left: The current camera frame. Right: A part of the HD maps (2.3). Cyan: The mapping $s_c \mapsto p_c$ from objects (right) to their corresponding pixels (left).

Calibration procedure Our camera is modelled using the pinhole camera model. The pinhole projection from samples of the world objects to pixels is formulated as

$$p_c = \pi\,(R * T * (o_c + \lambda_c * h_c)) \tag{4}$$

$$z * \pi(x) = \begin{bmatrix} f_x, & 0, & c_x, & 0 \\ 0, & f_y, & c_y, & 0 \\ 0, & s, & 1, & 0 \end{bmatrix} * x \tag{5}$$

where R is the cameras world rotation in Euler angles, T is the cameras world translation and π is the camera projection to image space based on the camera intrinsic parameters.

The optimal value for π, R, T is found, if it holds for all correspondences:

$$0 = p_c - \pi\,(R * T * (o_c + \lambda_c * h_c)) = p_c - \hat{p}_c \tag{6}$$

This places constraints on the values π, R, T can take and enables us to recover the camera pose and intrinsics only from the correspondences.

We estimate the camera pose by minimizing a modified version of the least-squares reprojection error

$$\min_{T,R,\Lambda,W} E(P,S,\pi,T,R,\Lambda,W) \tag{7}$$

formulated as

$$\begin{aligned} E(P,S,\pi,T,R,\Lambda,W) = & \sum_{c\in C} \rho(\|w_c * [p_c - \hat{p}_c]\|^2) \\ & + \sum_{c\in C} \alpha * \rho(\|(1-w_c)\|^2) \\ & + \sum_{c\in C} \beta * \rho(\|\Delta(\lambda_c, 0, h_c)\|^2) \\ & + \sum_{\pi_i\in\pi} \gamma * \rho(\|\Delta(\pi_i, \pi_i * 0.9, \pi_i * 1.1)\|^2) \\ & + \delta * \rho(\|\Delta(R_x, 60, 110)\|^2) \\ & + \delta * \rho(\|\Delta(R_y, -10, 10)\|^2 \end{aligned} \tag{8}$$

where P is the set of mapped pixels in the image, S is the set of mapped corresponding sampled points from the objects and Λ is the set of λ values associated with the sampled points. This formulation allows the optimization over the line approximations of the objects and jointly optimizes for the camera parameters T, R and the $\lambda \in \Lambda$ parameters of the line objects.

The calculation of the exact position of $s_c \sim \lambda$ allows the optimizer to search the whole space of real numbers for λ. Nonetheless we penalize values for λ that exceed the physical height of the object by

$$\Delta(x,l,u) = \begin{cases} x-u, & \text{if } x > u \\ x-l, & \text{if } x < l \\ 0, & \text{else} \end{cases} \tag{9}$$

This regularization enables a robust estimation procedure that can flexibly adjust to the missing exact world positions.

Initialization In contrast to most pose estimation problems our approach drops the need for good initialization. By regularization of the λ values enough flexibility is given to optimize over an infinite space of values, but enforces the solution of the λ to lie within the interval of $\lambda \in [0,h]$.

$$\bar{s} = \frac{1}{|C|} \sum_{c\in C} o_c \tag{10}$$

$$T_0 = \begin{pmatrix} 1,0,0, & \bar{s}_x \\ 0,1,0, & \bar{s}_y \\ 0,0,1, & \bar{s}_z + 1000 \\ 0,0,0, & 1 \end{pmatrix} \tag{11}$$

$$R_0 = \mathbb{I}^{4\times 4} \tag{12}$$

It is sufficient to initialize with $\lambda_c = 0$ for all correspondences. The camera rotation is defined to be zero with the camera facing in negative world z axis. By placing the camera at some distance over the mean of the known object positions with zero rotation the optimization always converges to the desired minimum.

5. Evaluation

We assure the correctness and quantify the visual improvements resulting from the algorithms by an empirical study on the video streams of the highway cameras.

5.1. Study objects

We use video recordings from the four cameras mounted to the two gantry bridges internally named S40 and S50. The schematic camera setup is displayed in Figure 4.

The dataset consists of four recordings, each with of a length of 1495 frames over $\sim$ 60 seconds at 25 frames per second.

The recordings are taken on a day with strong winds to ensure high jitter in the video feed to optimally test the Dynamic Stabilization algorithm described in subsection 4.1.

5.2. Dynamic stabilization

To evaluate the Dynamic Stabilization algorithm described in subsection 4.1 we have come up with two metrics to quantify the environmental influences.

Figure 3: Left: Sampled points of objects that are mapped to pixel locations (green) and sampled points without known corresponding pixels (red) rendered by a poorly calibrated camera model. The mapping from the points to their expected pixels is drawn in cyan. Right: The same sampled points after the calibration procedure. The rendered positions of the sampled points align with the pixels of the objects they are mapped to and the drawn mapping disappears as the distance approaches 0.

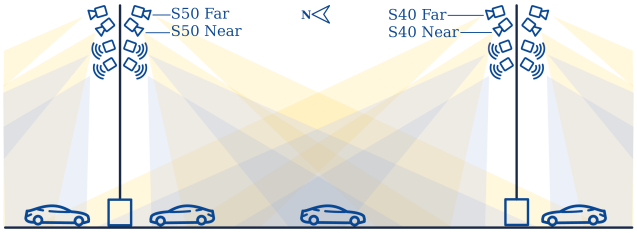

Figure 4: The schematic camera setup along the highway A9. The cameras S40 Far and S40 Near are facing north, the cameras S50 Far and S50 Near are facing south.

5.2.1 Optical Flow

The Optical Flow describes the apparent motion of image objects between two consecutive frames. It is a 2D vector field where each vector is a displacement vector showing the movement of points between the frames caused by movement of the objects or cameras.

We use the dense Optical Flow estimation algorithm proposed by Farnebäck [8, 6] to measure the displacement of each pixel between the frames. We calculate the mean displacement over the whole image to get the overall displacement. Figure 5 shows one frame of Optical Flow calculated.

We see that the dynamically stabilized video feed exhibit a substantially lower mean displacement between frames as the displacement from camera jitter is removed. This leaves us with only the expected displacement resulting from moving objects, *e.g.* vehicles on the highway or shaking trees.

Figure 6 displays the mean pixel displacement per frame. It shows that the displacement with especially the SURF [5, 6] feature detector is lowered greatly. The damping approximates the removed jitter and also displays a huge lowering in mean displacement.

Figure 7 further quantifies the stabilization of the frames.

Problem of Optical Flow as a metric Unfortunately, Optical Flow exhibits one major problem as a metric: It cannot distinguish between dynamically moving objects and static scene. This especially introduces a problem when a jitter of the camera moves the pixels in the same direction as the vehicles path is pointing in image space. By this jitter the movement of the camera blurs the movement of the dynamic objects into the background and thus removes some of the real movement. This shows some frames after stabilization to be worse than before. This should be taken with caution as the Optical Flow cannot detect the relative motion and thus is not a definite measure for the jitter. Nonetheless it gives a good hint at the overall stabilization capabilities and can become a measure together with the following measure of the path length of a pixel on a dynamic object.

5.2.2 Track features and calculate path smoothness

To overcome the problems of the Optical Flow as a measure of stability we randomly took three sample vehicles per camera and tracked their pixel locations over time. As the vehicles move the bounding box of the object is found using the Discriminative Correlation Filter With Channel and Spatial Reliability proposed by Lukezic *et al.* [16, 6]. The middle point of the bounding box is then written to disk and used as the predicted pixel location of the object.

We only track in the original frame to remove inaccuracies in the tracking and use the homographic transformation matrix from Equation 1 that stabilizes the frame. This gives us comparable results for the pixel locations.

To calculate the arc length we use the sum of distances between the middle point pixel locations over frames. The middle point is formulated as

$$p = \begin{pmatrix} x + 0.5 * w \\ y + 0.5 * h \end{pmatrix} \tag{13}$$

with x being the x-coordinate of the top left corner of the bounding box, y being the y-coordinate of the top left corner of the bounding box, w being the width of the bounding box and h being the height of the bounding box.

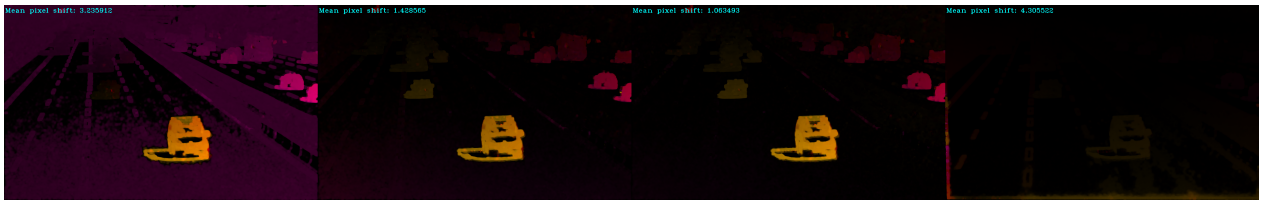

Figure 5: The dense Optical Flow displaying the pixel displacement. The lighter the color the further the displacement. The angle of displacement is color coded according to the HSV color circle. From left to right: Original frame, stabilized using SURF [5, 6] feature detector, ORB [20, 6] feature detector and FAST [10, 6] feature detector with FREAK [2, 6] feature descriptors. In the original frame the violet background color indicates a jittery camera movement. The car in the lower half is driving in the opposite direction as the camera jitters.

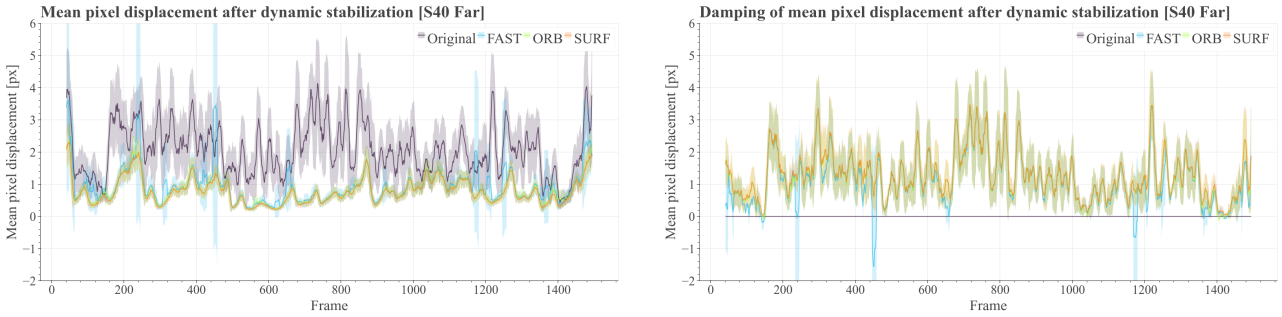

Figure 6: Left: Comparison of the three implemented dynamic stabilizers and the original not stabilized video feed using Optical Flow as metric (lower is better). The stabilizers are based on the FAST [10, 6] feature detector with FREAK [2, 6] feature descriptors, SURF [5, 6] feature detector and ORB [20, 6] feature detector. The graphs display the mean pixel shift at each frame. Right: The damping capabilities of the same three stabilizers (higher is better). The graphs approximate the removed jitter in the mean pixel shift between the original video and the stabilizer at each frame.
For visualization the values are filtered using the rolling mean over 12 frames. The light areas display the standard deviation within the window.

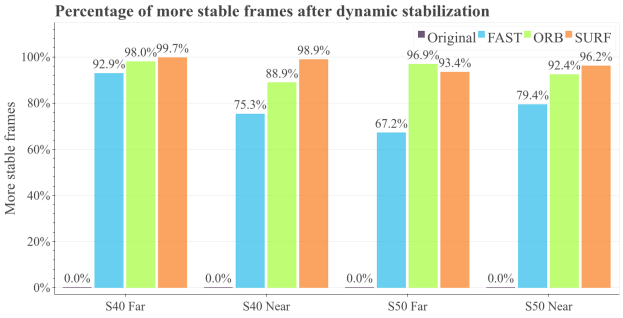

Figure 7: Comparison of the percentages of more stable frames after dynamic stabilization per camera and stabilizer. A frame is classified as more stable if the mean pixel displacement is lower as in the original video feed. The comparison shows that the stabilizers all improve the video stability, but the SURF [5, 6] feature detector based stabilizer outperforms the others.

The arc length is described by

$$arc(vehicle) = \sum_{n=1}^{|frames|} \|p_i - p_{i-1}\|_2 \tag{14}$$

To get comparable results a normalization is performed by the arc length of the original video frame.

$$narc_{stabilizer}(vehicle) = \frac{arc_{stabilizer}(vehicle)}{arc_{original}(vehicle)} \tag{15}$$

Figure 8 displays the vehicles tracked in the video sequence taken from the camera S40 Near. There is one red truck, a black pickup and a white car taken as a random sample. The tracking is performed als long as they are visible.

As our dynamic stabilization algorithms remove environmental influences from the frames the pixels only move on the projected path of their vehicles. Figure 8 displays that the arc lengths after stabilization remain at only a relative length of 0.5 of the original length. This clearly indicates that with the same vehicle movement the tracked position is much more accurate.

The **??**, **??**, **??** and **??** in **??** display the tracked pixel locations of three cars per camera.

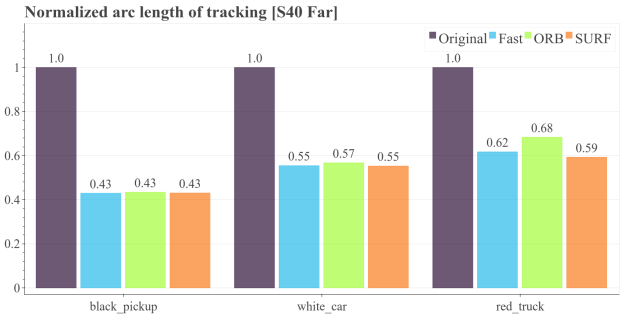

Figure 8: Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The length is normalized by the original arc length, hence the 1.0 factor for the original video. As the jitter is removed, the pixels movement is lowered significantly as it does only move with the vehicle, not the camera. This can be seen with around half of the path length remaining after stabilization for all stabilizers.

5.2.3 Speed comparison

5.3. Static calibration

In the following we evaluate the implemented static calibration algorithm and assess the algorithmic and systematic errors.

5.3.1 Ensure no Systematic Error

The proposed pose estimation algorithm from Equation 7 converges to a pair of optimal translation T and rotation R values for the camera pose. These T, R values approximate the projection from the world objects to the pixels as described in Equation 4.

Using a maps provider we assured that the resulting values are within reasonable ranges and that there are no systematic errors in the optimization. The results are displayed in Figure 9

5.3.2 Points needed for convergence

The correspondences build up a system of linear equations. This system of equations is solvable if there exist a more or equal number of constraints on the parameters than there are degrees of freedoms in the system:

$$p_c = \pi\left(R * T * (o_c + \lambda_c * h_c)\right), \forall c \in C \tag{16}$$

This is the case if:

$$\begin{aligned} 2 * |C| &\geq 5 + 3 + 3 + 1 * |C| \\ |C| &\geq 11 \end{aligned} \tag{17}$$

As each of the pixel per correspondence gives us two constraints and we optimize over the 5 intrinsic parameters (Equation 5), the 3 extrinsic translation, 3 extrinsic rotation parameters and one λ parameter per correspondence. We see that 11 points are enough to recover the pose.

5.3.3 Structure of points

The best result are shown when there are at least two correspondences per object. These correspondences should be the top and bottom most visible pixel of the object. If there exists only 1 or a low number of near pixels, the algorithm cannot precisely recover the camera pose, as it is free to move the correspondence along the center line of the object. The algorithm therefore cannot distinguish the solutions where the camera is placed low, thus projecting a high point of an object to a low pixel correspondence, or if it should place the camera higher and lower the correspondences world position along the center line.

We thus conclude that the best solution is recovered the more the pixels fill the whole object, and the more spread to the top and bottom of the object the correspondences are.

5.3.4 Expectable Error Bounds

Due to measurement uncertainty in the camera sensors we expect some remaining error after pose estimation. To conclude a lower bound on this error we start from the optimized focal length f_px in pixels and the sensor width w_{mm} in millimeters and w_{px} in pixels.

The focal length in millimeters is given by

$$f_{mm} = f_{px} * \frac{w_{mm}}{w_{px}} \tag{18}$$

We then calculate the field of view (FOV) in radians by

$$FOV_x = 2 * \arctan\left(\frac{f_{mm}}{2 * w_{mm}}\right) \tag{19}$$

Equivalent the FOV_y with the height of the sensor h_{mm}.

This brings us to a formulation for the angle spanned by each pixel

$$\alpha_{px} = \frac{w_{px}}{FOV_x} \tag{20}$$

Using the pythagorean formula we can then calculate the uncertainty u in meters of the camera as the spanned meters per pixel relative to the distance d from the camera

$$u = \tan(\alpha_{px}) * d \tag{21}$$

Table 1 displays the uncertainty for our cameras. It shows that the far cameras cannot distinguish between points that are $\sim 2.2cm$ at the nearest visible distance from the camera ranging up to $\sim 7.54cm$ at the farthest distance. The near cameras cannot

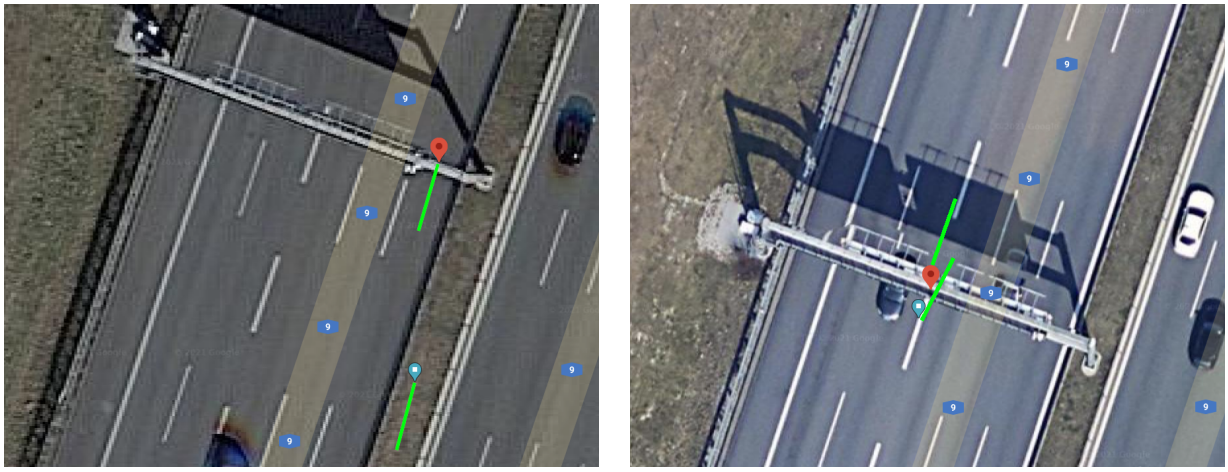

Figure 9: Left: The positions of the cameras S40 Near (red) and S40 Far (green) and their respective looking directions (yellow). Right: The positions of the cameras S50 Near (red) and S50 Far (green) and their respective looking directions (yellow). It displays that the rotation of the cameras is in a reasonable range so that the cameras look along the highway as expected. Also the cameras are within reasonable translational bounds around their real world location as subsubsection 5.3.4 shows.

Camera	f_{px}	$FOV_x[^\circ]$	α_{px}	$d[m]$	$u[cm]$
S40 Far	8591	12.753	$1.16e^{-4}$	200	2.32
S40 Far	8591	12.753	$1.16e^{-4}$	650	7.54
S40 Near	2735	38.678	$3.52e^{-4}$	25	0.88
S40 Near	2735	38.678	$3.52e^{-4}$	450	15.82
S50 Far	8868	12.357	$1.12e^{-4}$	200	2.22
S50 Far	8868	12.357	$1.12e^{-4}$	650	7.30
S50 Near	2747	38.527	$3.50e^{-4}$	25	0.88
S50 Near	2747	38.527	$3.50e^{-4}$	450	15.76

Table 1

distinguish between points that are $\sim 0.88cm$ at the nearest visible distance from the camera ranging up to $\sim 15.82cm$ at the farthest distance.

This gives a lower bound for the uncertainty in the translational parameters of the camera. The camera translation cannot be more precise as the lowest uncertainty in the measurements. Nonetheless, in practice the error will sum up and the translation parameters will drift inevitable.

This concludes that objects near to the camera are more reliable to detect and to use for estimation.

5.3.5 Expectable Deviations among Estimations

The proposed pose estimation algorithm is based on the minimization of the reprojection-error Equation 4. As with all optimization problems convergence is reached when the values that are optimized don't change anymore.

The optimization jointly optimizes for the 6 camera parameters, 3 for the translation, 3 for the Euler angle rotation and one λ parameter per correspondence. The resulting high-dimensional problem exceeds multiple minima, whereas each represents a configuration for the camera pose that well explains the dependency between pixels, world objects and the camera.

As stated previously the loss landscape does exhibit a multitude of local minima, thus the optimization procedure converges to different sets of parameters.

Figure 10 displays the resulting parameters for the camera S40 Far. We plot the loss of the correspondence residuals and the λ residual blocks against each of the parameters. The translation parameters are in meters and relative to the transverse mercator projection [14]. The rotation parameters are in degree of Euler angles.

The plots show that the standard deviation σ of the translations does not exceed $2 * 10^{-4} m = 0.2mm$. For the rotation σ is at most $4 * 10^{-4} deg$.

We have shown the expectable error for the parameters resulting from the sensor and map inaccuracies in subsubsection 5.3.4. We conclude that in relation to these the algorithmic error can be neglected.

In the appendix **??** we additionally evaluate the other cameras.

6. Future Work

The project leaves us with the opportunities to research in a broad range of directions.

6.1. Test on different weather/lighting conditions

The implementations are currently only tested on good weather and lighting conditions. A next step is to test the implementations in bad weather and lighting conditions, *e.g.* by night, rain and snow. As far as we can tell by now the feature based dynamic stabilization approach will suffer in performance as the detected features will be compromised. The static calibration nonetheless

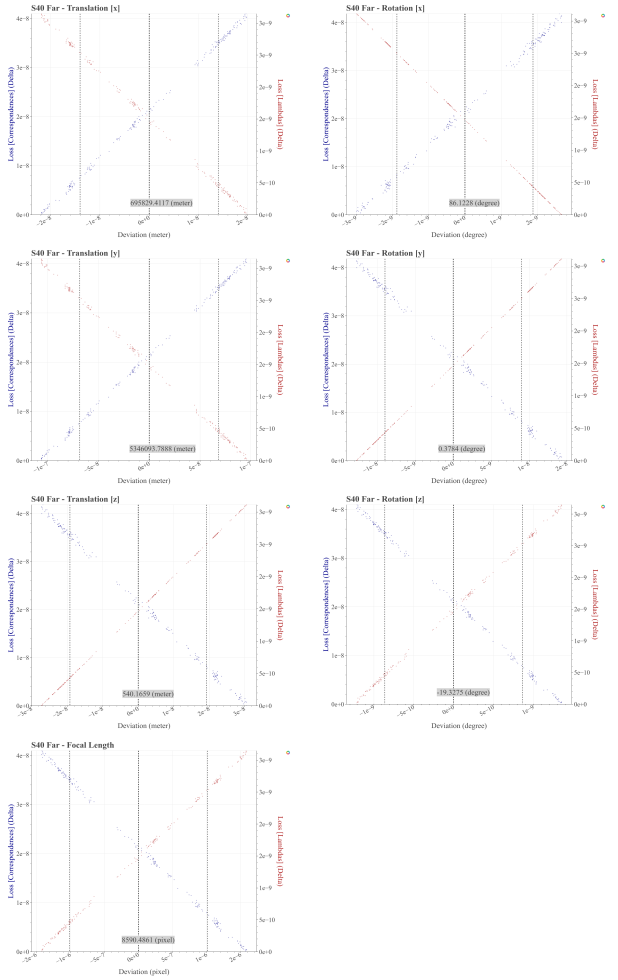

Figure 10: Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. The mean of the distributions is displayed as thick dashed line. The smaller dashed lines display the standard deviation σ. The σ of the translations does not exceed $2 * 10^{-4}m = 0.2mm$. For the rotation σ is at most $4 * 10^{-4}deg$.

wont be impacted as we have the human-in-the-loop approach for correspondence mapping. When we switch to an automatic landmark detection algorithm the weather conditions will impact the detection and thus the calibration.

6.2. Dynamic stabilization

In the subfield of the dynamic stabilization we have until now implemented the visual feature based approach.

6.2.1 Dropping stabilization if not enough correspondences found

We showed in subsection 5.2 that the number of better frames is not at 100%. This is partially due to the shortcomings of the optical flow used as a metric as described in section 5.2.1. Another problem arises when the feature matching does not give enough matches. Finding the homographic transformation is not well defined in this case anymore and the found solution can make the image stability worse.

6.2.2 Warp field stabilization based on Optical Flow

Currently the optical flow is only used to measure performance but could be used as an alternative approach to the dynamic stabilization problem. The optical flow is modelled as a vector field over the image and contains the pixel movement as direction vector and distance per pixel. By inverting this vector field and thus applying the inverse of the jitter the image could be stabilized using image warping.

6.2.3 Deep Learning based approaches

With the recent emerge of deep learning based approaches in computer vision, especially with convolutional neural networks (CNN), a self-learning stabilization procedure might be developed. This might speed up the procedure and would inherently add a measure for the uncertainty of the results when finding the homographic transform. Additionally the feature detection, matching and warping steps could be fused into one step and optimized jointly.

As far as we can tell this can be broken down into two approaches based on different loss functions.

Image MSE for calibration (unsupervised) Input the current frame and reference frame, output the homographic transformation. Learning is done by minimizing the mean squared error of pixel distances between the frames with the current transformation.

Logistic regression on pose (supervised) Input the current frame and reference frame, output the homographic transformation. Learning is done by logistic regression over the expected and predicted transform.

6.3. Static calibration

For the static calibration there are several improvements possible.

6.3.1 IMU based sensor fusion

Previous by-hand calibrations have shown that an inertia measurement unit (IMU) can be used to initially find a camera pose, but the lack of performance and huge time needed makes this not feasible. This is just empirical knowledge, not backed up by research. This should be looked into further.

6.3.2 Automatic detection of more landmarks after initial calibration

Currently the detection and mapping of landmarks is done by hand using a watershed algorithm [17, 6] based external tool. This tool relies on a by hand marking of landmarks as the markers used to

find the segmentation. By applying template, color or gradient matching approaches the detection of the markers might be automated. This automated detection would speed up the mapping procedure.

6.3.3 Automatic mapping of pixels to objects

To establish the mapping of the correspondences a human has to look up the ids of the landmarks in the HD maps (2.3). After an initial calibration an image region based approach might be used to automate this mapping. This positions of the known objects from the HD maps (2.3) can be projected into the camera image with the current camera pose. Starting from the resulting pixel locations one could search in a defined enclosing region to search for pixels that clearly correspond to the objects. This automatic detection of more landmarks could further improve the robustness an can be used to mitigate drift in the cameras.

6.3.4 Machine learning based pose estimation

A machine learning based approach for the bundle adjustment problem we faced and the resulting camera pose estimation problem can be formulated. As Aravkin *et al*. [3] have shown a Student's-t distribution based approach can be used to estimate and robustify the procedure against outliers.

6.3.5 New HD map

The newer OpenDRIVE standards also provide the possibility to include lane markings. These lane markings are easily detectable and can then be used for the calibration procedure in conjunction with the object landmarks. This would greatly simplify the automatic detection and mapping procedures as described in Sec. 6.3.2 and Sec. 6.3.3 as they are spatially more extend and thus easier to detect. Additionally with the color information, as the lane markings are always white (Germany) and yellow (USA) the detection is simplified a lot.

7. Conclusion

References

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008. 3

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–517, 2012. 3, 6

[3] Aleksandr Aravkin, Michael Styer, Zachary Moratto, Ara Nefian, and Michael Broxton. Student's t robust bundle adjustment algorithm. *Proceedings / ICIP ... International Conference on Image Processing*, 11 2011. 10

[4] Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 2

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 3, 5, 6

[6] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3, 5, 6, 9

[7] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, page 778–792, Berlin, Heidelberg, 2010. Springer-Verlag. 3

[8] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 5

[9] Tobias Fleck, Karam Daaboul, Michael Weber, Philip Schörner, Marek Wehmer, Jens Doll, Stefan Orf, Nico Sußmann, Christian Hubschneider, Marc René Zofka, et al. Towards large scale urban traffic reference data: Smart infrastructure in the test area autonomous driving baden-württemberg. In *International Conference on Intelligent Autonomous Systems*, pages 964–982. Springer, 2018. 2

[10] Morteza Ghahremani, Yonghuai Liu, and Bernard Tiddeman. Ffd: Fast feature detector. *IEEE Transactions on Image Processing*, 30:1153–1168, 2021. 3, 6

[11] Frank Köster and Mathias Höhne. Testfeld für autonomes und vernetztes fahren in niedersachsen. 2017. 2

[12] Annkathrin Krämmer, Christoph Schöller, Dhiraj Gulati, Venkatnarayanan Lakshminarasimhan, Franz Kurz, Dominik Rosenbaum, Claus Lenz, and Alois Knoll. Providentia - a large-scale sensor system for the assistance of autonomous vehicles and its evaluation, 2020. 2

[13] Rekhil M Kumar and K Sreekumar. A survey on image feature descriptors. *Int J Comput Sci Inf Technol*, 5:7668–7673, 2014. 3

[14] Johann Heinrich Lambert. *Anmerkungen und zusätze zur entwerfung der land-und himmelscharten*. Number 54. W. Engelmann, 1894. 8

[15] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 3

[16] Alan Lukezic, Tomas Vojir, Luka Cehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

[17] Fernand Meyer. Color image segmentation. In *1992 international conference on image processing and its applications*, pages 303–306. IET, 1992. 9

[18] J. Müller, M. Herrmann, J. Strohbeck, V. Belagiannis, and M. Buchholz. Laci: Low-effort automatic calibration of infrastructure sensors. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3928–3933, 2019. 2

[19] PROJ contributors. *PROJ coordinate transformation software library*. Open Source Geospatial Foundation, 2021. 3

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011. 3, 6

[21] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*, ICPR '04, page 28–31, USA, 2004. IEEE Computer Society. 3

[22] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recogn. Lett.*, 27(7):773–780, May 2006. 3