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, Bundle Adjustment, Reprojection-error, Optimization, 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 (**??**)) of the enclosed environment are heavily used. These HD maps (**??**) 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 (**??**) 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 (**??**) are accessible via the two GitHub repositories: `https://github.com/Brucknem/GuidedResearch` and `https://github.com/Brucknem/OpenDRIVE`.

2. Related Work

There are many ITS projects emerging recently [13, 4, 5, 1]. They propose novel approaches to the detection, tracking and traffic prediction problems that arise with the goal to provide additional environment information to human drivers and autonomous vehicles. Erdelean *et al*. [8] give a detailed overview over the existing projects up to the year 2019.

In the *Test Area Autonomous Driving Baden-Württemberg* [11] (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 a 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. 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 [14] and associating pixels within the multi-view setup is an inherently hard problem.

Müller *et al*. [20] present an approach based on a cooperative intelligent vehicle. The vehicle moves through the scene and passes cooperative awareness messages containing positional information 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 [14].

An overview over the general structure of Bundle Adjustment problems as used in subsection 3.2 is given by Triggs *et al*. in [24].

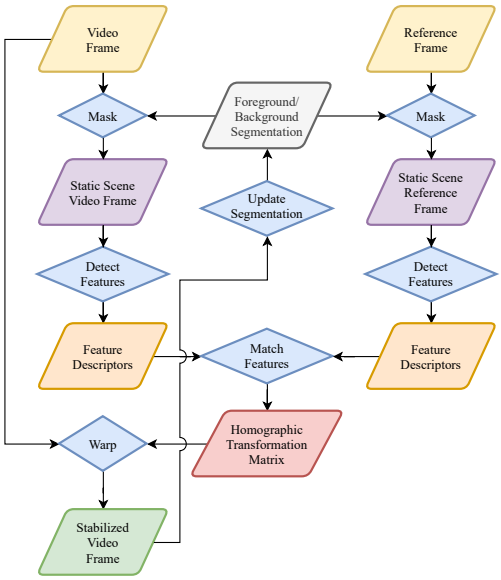

Figure 1. The proposed dynamic stabilization pipeline. The input is the video and a stable reference frame. The frames are segmented into the foreground and background to extract the static scene. On the static scene visual features are detected and the feature descriptors are matched. We use the homographic transformation minimizing the reprojection-error based on the matching to warp the video frame. The stabilized frame is used to update the background segmentation mask and the pipeline repeats.

3. Approach

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

3.1. Dynamic Stabilization

The camera sensors used in the project are mounted onto gantry bridges spanning the highway. Environmental influences, *e.g.* wind or vibrations from passing vehicles, bring the bridges into a swinging state that spreads onto the cameras. This unwanted real world movement propagates to the video feeds output by the camera and introduces jittery motion in image space. We assume the cameras to be mostly static, thus we only face disturbances within a small range around the resting position. Nonetheless, the small disturbances get amplified by the huge distances covered by the cameras and introduce substantial error.

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

Extract the Static Background We stabilize the input frame by minimizing the reprojection-error between the video frame and the reference frame, thus aligning the images. This alignment is based on the matching of features between the frames, whereas we do not align the moving vehicles, but only the static non-moving background, *e.g.* the road, poles, guardrails and bridges. We assume that the background does not move in the real world, thus aligning it during image warping ensures that the scene stays static and only the vehicles real movement is kept.

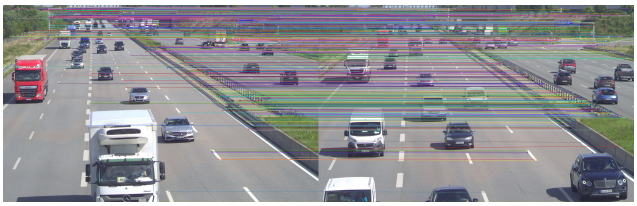

Figure 2. Left: The input frame. Right: The stable reference frame. The colorful lines display the matching between image features. The ends of the lines are the location of the features.

Hence, we mask the input 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*. [25, 26, 7].

Feature detection We search the image for pixel locations that are prominent and depict specific patterns that are unique and can easily be compared. The detected features describe the location based on different metrics, *e.g*. the local image gradient, oriented histograms or haar-like features [23]. Kumar *et al*. [15] describe a multitude of variations of feature detectors and descriptors.

We implemented the SURF [6] and ORB [22] feature detectors and descriptors and the Fast [12] feature detector with FREAK [2] feature descriptors. The algorithmic implementation is taken from the OpenCV library [7].

Feature matching We compare the detected features from the input frame with the features from the stable reference frame. A match is reported if the feature descriptors of two compared features surpass the Lowe's ratio test [17] regarding some feature dependent metric [15]. These feature matches establish a spatial relationship in pixel space between the two frames. Figure 2 displays an exemplary feature matching.

We estimate a homographic transformation H that maps the homogeneous pixel locations $(x_i, y_i, 1)^T$ of the input frame to the matched pixel locations $(x_i', y_i', 1)^T$ in the stable reference frame. We minimize the reprojection-error between the pixels so that for each match i it holds

$$z_i * (x_i', y_i', 1)^T \sim H * (x_i, y_i, 1)^T \tag{1}$$

where z_i is the homogeneous component used in the perspective division. We use a RANSAC [10] based estimation procedure to robustify the minimization against outliers.

The implementation is included in the OpenCV library [7].

Image alignment We use the found homographic transformation to warp the whole input frame, thus aligning the background of the frames. As the keyframe is stable and does not change over time the current video frame is now also stabilized. The alignment minimizes the motion of the static background scene and leaves only the real expected movement of the vehicles.

Foreground/Background segmentation upate We use the stabilized frame to update the segmentation of the foreground and background. We assume the motion between frames to be relatively small, thus we use the segmentation of the stabilized frame for the next frame. 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.

3.2. Static Calibration

ITS are inherently dependent on the calibration of the different sensors. The system has to know the poses of the different sensors relative to some reference coordinate system to accurately measure the position of vehicles within the single sensor ranges and at the overlapping boundaries.

We propose a calibration procedure based on a Bundle Adjustment (BA) problem formulation. We map visual landmarks in the video feed to their partially known world positions from high definition road maps (HD maps). We recover the pose by jointly optimizing for the camera intrinsic and extrinsic parameters as well as the real world positions of the landmarks.

Retrieve Objects from High Definition Maps In our project we use HD maps in the OpenDRIVE standard format. In this work we focus on the permanent delineator (PD) objects that are easily visible in the video feeds.

We extract the world position of the PDs using the mathematical operations defined in the OpenDRIVE standard. This gives us the the base origin point $o = (x, y, z)^T$ of the PDs in the Universal Transverse Mercator (UTM) projection [16, 21]. This point o is the world position of the lower end of the PD where it touches the ground or another object. Additionally, we retrieve a directional heading axis $d = (x, y, z)^T$ and the height h of the PD.

1D Approximation of Objects In the original BA setting the optimization is done jointly over multiple cameras and observations, and the arising stereo vision problem is solved jointly for the 3D positions of the objects and the camera parameters. For this system of equations to be solvable it requires multiple cameras from different viewing angles and large overlapping fields of view between the cameras.

In our project we have neither of requirements and calibrate each camera separately to the HD map. We relax the BA problem by the 1D approximation of the PDs

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

where S is the set of points along its central axis between its base at $\lambda = 0$ and its top at $\lambda = h$. This approximation allows for a joint optimization of their world positions and the camera intrinsic and extrinsic parameters.

subsubsection 4.3.2 derives a minimal number of points for the resulting system of equations to be solvable.

Mapping Objects to Pixels We solve the BA problem by minimizing the reprojection-error over the PDs. We thus require a set C of correspondences that map world points s_c of the PDs to their respective pixel p_c. Figure 3 displays an exemplary mapping from a HD map to the video feed.

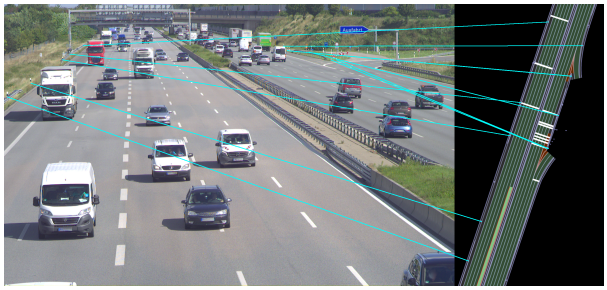

Figure 3. Left: The current camera frame. Right: A part of the HD map. Light blue lines: An exemplary mapping $s_c \mapsto p_c$ from objects (right) to their corresponding pixels (left).

Figure 4. Top: Points of PDs mapped to pixel locations (green) and points without known corresponding pixels (red) rendered by a poorly calibrated camera model. The mapping from the projected points to their expected pixels is drawn in light blue. Bottom: The same points after the calibration procedure. The rendered positions of the mapped points align with their respective pixels. The drawn mapping disappears as the distances approach 0.

This mapping is currently done by human interaction and not fully automated. We implemented an annotation tool to mark pixels that outputs a list of pixels that can easily be mapped to the list of objects.

Calibration Procedure We model our cameras using the pinhole camera model. The projection from points $s_c = o_c + \lambda_c * h_c$ of the PDs world positions to pixels p_c is formulated as

$$p_c = \pi\left(R * T * (o_c + \lambda_c * h_c)\right) \tag{3}$$

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

$$z * \pi(x) = \begin{pmatrix} f_x, & 0, & c_x, & 0 \\ 0, & f_y, & c_y, & 0 \\ 0, & s, & 1, & 0 \end{pmatrix} * x \tag{4}$$

where x is an 3D input vector in homogeneous coordinates, f_x, f_y are the focal lengths in pixels, c_x, c_y is the principle point and z is the homogeneous component used in the perspective division.

The optimal values for π, R, T are found if for all correspondences the distance between the expected pixel $\hat{p}_c$ and the projected pixel p_c is minimal and it holds for all c

$$0 = \hat{p}_c - \pi\left(R * T * (o_c + \lambda_c * h_c)\right) = \hat{p}_c - p_c \tag{5}$$

This places constraints on the values π, R, T can take and enables us to recover the camera pose, the intrinsic parameters and the 1D approximate positions relative to the λs only from the correspondences.
Figure 4 displays rendering the PDs before and after calibration.

Reprojection-Error Formulation We solve the BA problem by minimizing a modified version of the least-squares reprojection-error E formulated as

$$\begin{aligned} E(P,S,\pi,T,R,W) = &\sum_{c \in C} \rho(\|w_c * [\hat{p}_c - p_c]\|_2^2) \\ &+ \sum_{c \in C} \alpha * \rho(\|(1 - w_c)\|_2^2) \\ &+ \sum_{c \in C} \beta * \rho(\|\Delta(\lambda_c, 0, h_c)\|_2^2) \\ &+ \sum_{\pi_i \in \pi} \gamma * \rho(\|\Delta(\pi_i, \pi_i * 0.9, \pi_i * 1.1)\|_2^2) \\ &+ \delta * \rho(\|\Delta(R_x, 60, 110)\|_2^2) \\ &+ \delta * \rho(\|\Delta(R_y, -10, 10)\|_2^2 \end{aligned} \tag{6}$$

where P is the set of mapped pixels $\hat{p}_c$ in the image and S is the set of mapped corresponding points s_c from the PDs. The additional weights $w_c \in W$ are associated to the correspondences to downweight outliers and are enforced to stay near 1. We use a robust loss function ρ to further decrease the influence of outliers.

We guide the algorithm to feasible solutions by penalizing λ_c values that are negative or exceed the PDs height h_c using the distance

$$\Delta(x, l, u) = \begin{cases} x - u, & \text{if } x > u \\ x - l, & \text{if } x < l \\ 0, & \text{else} \end{cases} \tag{7}$$

from the interval $[l, u]$.

We rely on a rough initialization for the values $\pi_i \in \pi$ and allow the optimizer to adjust the values within a ± 0.1 interval around the initialization.

Finally, we constrain the X axis rotation R_x (Pitch) to be within the interval $[60, 110]$ degree and the Y axis rotation R_y (Roll) to be within the interval $[-10, 10]$ degree.

The factors $\alpha, \beta, \gamma, \delta$ are used to explicitly scale the remaining losses of the terms, thus giving them different influence on the optimizer. We saw empirically that good values for the factors are $\alpha = 1 * 10^{100}, \beta = 2, \gamma = 10, \delta = 50$.

Initialization In contrast to most BA problems our approach drops the need for good initialization. The regularization of the λ values gives the optimizer enough flexibility to optimize over the infinite space of possible values, but enforces the values of the λ to lie within the interval of $\lambda \in [0, h]$.

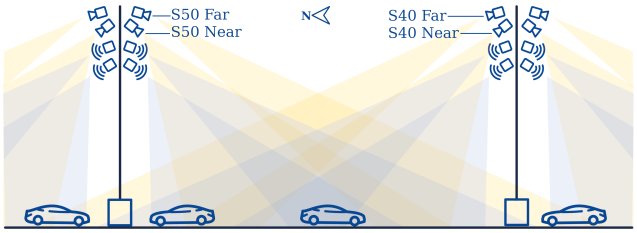

Figure 5. The schematic camera setup along the highway A9. The cameras S40 Far and S40 Near are facing north, the cameras S50 Far and S50 Near are facing south.

The optimizer calculates its own initial guess

$$T_0 = T(\bar{s}_x, \bar{s}_y, \bar{s}_z + \bar{d}) \tag{8}$$

for the camera translation based on the mean

$$\bar{s} = \frac{1}{|C|} \sum_{c \in C} o_c \tag{9}$$

of the base origin points o_c of the PDs and some distance $\bar{d}$ from the mean.

The optimizer initializes the rotation

$$R_0 = R(x, y, z) \tag{10}$$

based on x, y, z Euler angle values drawn from the uniform distribution $U(-35, 35)$, where $R(0,0,0)$ is the camera facing in negative Z direction onto the mean $\bar{s}$. This initialization ensures that all points lie in front of the camera and can be projected onto the image plane.

It is sufficient to initialize $\lambda_c = 0$ for all correspondences.

4. Evaluation

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

4.1. 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 5.

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 pipeline described in subsection 3.1.

4.2. Dynamic Stabilization

We evaluate and compare three pipeline instances based on the SURF [6, 7] feature detector (SURF), ORB [22, 7] feature detector (ORB) and FAST [12, 7] feature detector with FREAK [2, 7] feature descriptors (FAST) and present two measures to evaluate the dynamic stabilization pipeline described in subsection 3.1.

4.2.1 Optical Flow

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

We use the dense optical flow estimation algorithm proposed by Farnebäck [9, 7] 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 6 displays one frame of optical flow calculated pre and after stabilization. It qualitatively shows that the static background scene is dark in the stabilized frames which indicates a low optical flow and thus nearly no movement. We see a car driving, indicated by the yellow patch, that remains as expected using SURF and ORB, but gets filtered out by FAST. This problem is described in section 4.2.1.

Figure 7 displays the mean pixel displacement (MPD) per frame calculated as the mean of the lengths of the vectors in the optical flow field and the damping as the difference of the mean displacements over the original and stabilized frame. It shows that the original frames have a MPD of up to 5 pixels per frame. This implies that on average every pixel on the dynamic scene and static background had moved by 5 pixels per frame. The displacement is damped by all of the stabilizers by a mean of up to 4.5 pixels, whereas the remaining displacement is due to the actually moving vehicles in the dynamic foreground.

Figure 8 displays the percentage of frames that are more stable than the original frame. We define a frame to be more stable if its MPD is lower than the MPD of the original frame. It shows that for SURF and ORB at least 88.9% ranging up to 99.7% of frames are more stable. For the FAST detector it shows that except for the S40 Far camera only between 67.2% to 79.4% of frames are more stable, whereas a value of 50% does not directly indicate that the other 50% of frames are worse as pointed out in section 4.2.1.

We included plots for the MPD and the damping for the other three cameras in the Appendix (section 7).

Problem of Optical Flow as a Measure Optical flow cannot distinguish between dynamically moving objects and static scene. This is especially a problem when a jitter of the camera moves the pixels in the opposite direction as the vehicles path is pointing in image space. This jitter blurs the movement of the dynamic objects into the background and thus removes some of the real movement. The dynamic stabilization then reintroduces the real movement of the vehicle, thus showing 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 hints at the overall stabilization capabilities of the presented pipeline, but cannot give a direct explanation for single worse frames.

4.2.2 Track features and calculate path smoothness

We randomly took three sample vehicles per camera and tracked their pixel locations over the sequence. As the vehicles move through the image the bounding box of the object is found using

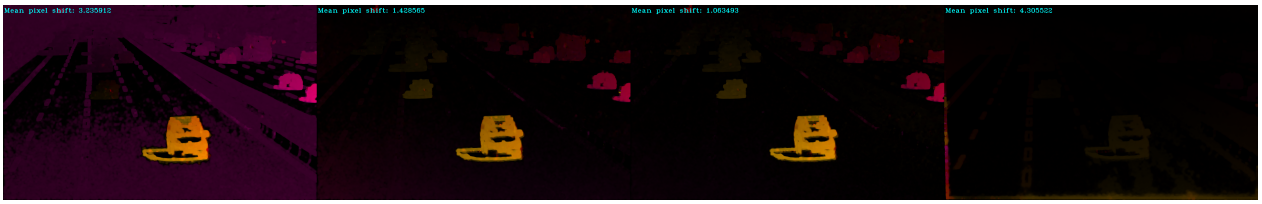

Figure 6. From left to right: Original frame, stabilized using SURF, ORB and FAST. The dense optical flow displays the pixel displacement, the lighter the color the further the displacement. The angle of displacement is color coded according to the HSV color circle. In the original frame the violet background color indicates a jittery camera movement. The car in the lower half is driving in the opposite direction as the camera jitters.

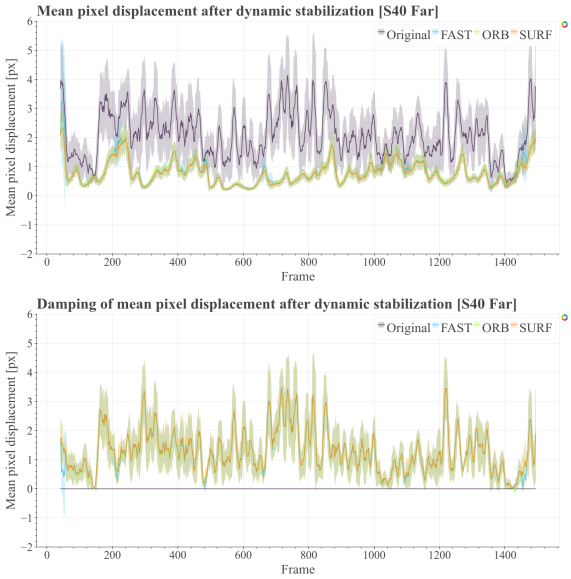

Figure 7. Top: Comparison of the three implemented dynamic stabilizers (SURF, ORB, FAST) and the original not stabilized video feed using optical flow as measure (lower is better). The graphs display the mean pixel shift at each frame. Bottom: The damping capabilities of the same three stabilizers (higher 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.

the *Discriminative Correlation Filter With Channel and Spatial Reliability* proposed by Lukezic *et al*. [18, 7]. The center point of the bounding box is then written to disk and used as the predicted pixel location p of the vehicle.

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 to also stabilize the bounding box. This gives us comparable results for the pixel locations.

We calculate the travelled distance of the pixel locations of the objects as they move through the image. This distance is a direct measure for the jittery motion in the image as it introduces large jumps of the tracked positions, thus directly increasing the travelled distance.

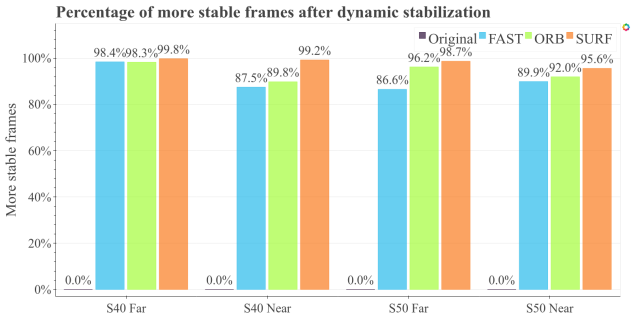

Figure 8. Comparison of the percentages of more stable frames after dynamic stabilization per camera and stabilizer. A frame is classified as more stable if the MPD is lower than for the original video feed.

The travelled distance is formulated as the arc length

$$arc(vehicle) = \sum_{n=1}^{|frames|} \|p_i - p_{i-1}\|_2 \tag{11}$$

based on the finite differences between consecutive tracked positions. To get comparable results a normalization

$$narc_{stabilizer}(vehicle) = \frac{arc_{stabilizer}(vehicle)}{arc_{original}(vehicle)} \tag{12}$$

is performed by the arc length of the original video frame.

Figure 9 displays the vehicles tracked in the video sequence taken from the camera S40 Far. There is one red truck, a black pickup and a white car taken as random samples. The tracking is performed als long as the vehicles are visible. It shows that the travelled distances of the pixels after dynamic stabilization remain at between 43% and 68.4% compared to the original one. This implies that much of the additional motion introduced by the environmental influences are removed and the remaining path displays the real projected path of the vehicle.

We included plots for the tracked pixel locations and the comparison of the arc lengths for the other three cameras in the Appendix (section 7).

4.2.3 Speed Comparison

Table 1 compares the average time needed to stabilize a frame per stabilizer instance. It shows that, although the name, the FAST stabilizer is the slowest, followed by SURF and beaten by ORB.

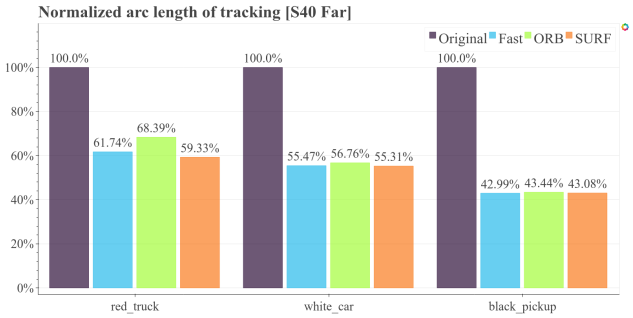

Figure 9. Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The removed jitter is directly proportional to the decrease in the normalized arc lengths, as the pixel only follows the real vehicles movement.

Stabilizer	Time $[ms]$	σ $[ms]$
FAST	18.649	2.102
ORB	14.816	1.466
SURF	16.058	1.443

Table 1. Comparison of the average milliseconds needed to stabilize the frame and the respective standard deviation in milliseconds per stabilizer instance.

Nonetheless, all stabilizers run at more than 50 frames per second and are thus all realtime capable.

4.3. Static calibration

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

4.3.1 Ensure no Systematic Error

We solve the BA problem proposed in subsection 3.2 by minimizing the reprojection-error as formulated in Equation 6. The algorithm converges to a pair of optimal translation T and rotation R values for the camera pose. These T,R values best describe the camera pose relative to the mapped objects as stated in Equation 3.

Using a maps provider we assured that the resulting values are within reasonable ranges and that there are no systematic errors in the optimization. Figure 10 displays the positions of the four cameras and their respective looking directions. It shows that except for the S50 Far the expected position is within the expectable range around the actual position. All cameras are looking along the highway as expected.

4.3.2 Number of Correspondences Needed for Convergence

The correspondences build up a system of linear equations as described in Equation 3. This system of equations is solvable if the number of constraints on the parameters is equal to or exceeds the degrees of freedom in the system, thus being over constrained. This is the case if for the number $|C|$ of correspondences it holds that

$$\begin{aligned} 2*|C| &\geq 5+3+3+1*|C| \\ |C| &\geq 11 \end{aligned} \tag{13}$$

as each of the expected pixels $\hat{p}_c$ gives us two constraints and we optimize for the 5 intrinsic parameters, 3 extrinsic translation parameters, 3 extrinsic rotation parameters and one λ_c parameter per correspondence. It shows that 11 points are enough to recover the pose, though more points improve the robustness of the algorithm.

4.3.3 Structure of Correspondences

We empirically see that recovering the camera pose performs best when there are at least two correspondences per object. These correspondences need to be the top and bottom most visible pixel of the object. If there exists only 1 or a low number of dense packed pixels the algorithm cannot precisely recover the camera pose as it is free to move the correspondences along the center line of the object. The algorithm therefore cannot distinguish solutions where it places the camera low, thus projecting a high point of an object to a low pixel correspondence, and solutions where it places the camera high and lowers the world position of the correspondence along the center line.

4.3.4 Expectable Error Bounds

Due to measurement uncertainty in the camera sensors we expect some remaining error after pose estimation. We derive a lower bound on this error starting from the optimized focal length f_px in pixels, the sensor width w_{mm} in millimeters and w_{px} in pixels. We use the focal length in millimeters given by

$$f_{mm} = f_{px} * \frac{w_{mm}}{w_{px}} \tag{14}$$

to calculate the field of view (FOV) in radians by

$$FOV_x = 2 * \arctan\left(\frac{f_{mm}}{2 * w_{mm}}\right) \tag{15}$$

Equivalent the FOV_y with the height of the sensor h_{mm}. The angle spanned by each pixel is calculated by

$$\alpha_{px} = \frac{w_{px}}{FOV_x} \tag{16}$$

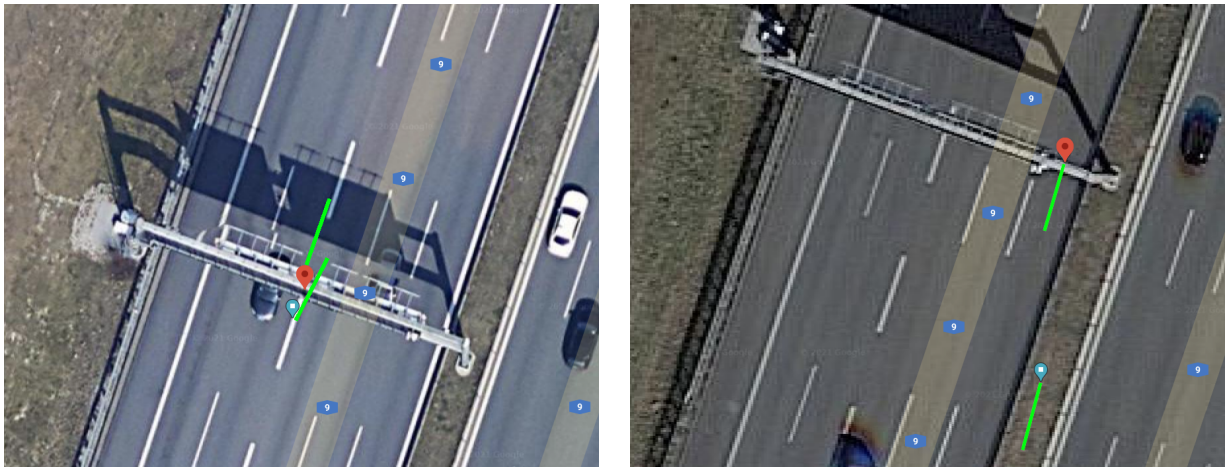

Figure 10. Left: The positions of the cameras S40 Near (red) and S40 Far (blue) and their respective looking directions (green). Right: The positions of the cameras S50 Near (red) and S50 Far (blue) and their respective looking directions (green). The rotations of the cameras are in a reasonable range so that the cameras look along the highway as expected. All cameras, except the S50 Far camera, are within reasonable translational bounds around their real world location as subsubsection 4.3.4 shows.

Camera	f_{px}	$FOV_x[^\circ]$	$\alpha_{px}[rad]$	$d[m]$	$u[cm]$
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 2. The focal lengths f_{px}, fields of view $FOV_x[^\circ]$ and spanned angle per pixel $\alpha_{px}[rad]$ compared to the measurement uncertainty $u[cm]$ at some distance $d[m]$ from the cameras. It shows a linear increase of the uncertainty proportional to the distance of the object from the camera.

Using the pythagorean formula we can then calculate the uncertainty

$$u = \tan(\alpha_{px}) * d \tag{17}$$

of the camera as the spanned meters per pixel relative to the distance d from the camera.

Table 2 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 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.

The highest possible achievable precision for the translational parameters of the camera is bound by the uncertainty in the measurements, as a misplacement of the camera can only be detected if it is higher than the sensor uncertainty. Nonetheless, in practice the imprecision will sum up and the translation parameters will drift inevitable. This implies that objects in close distance to the camera are more reliable when estimating the translation.

The rotational parameters can be estimated more precisely, as small rotational deviations introduce large deviations in the objects projected pixels. The expectable precision for the rotation is thus bound by the angle a_{px} spanned per pixel, as a deviation by one a_{px} would project the objects onto the next pixel. This implies that the precision of the rotation is at least in the range of $1.12 * 10^{-4}$ to $3.52 * 10^{-4}$ radians.

4.3.5 Estimations of the Algorithmic Error

We solve the BA problem proposed in subsection 3.2 by minimizing the reprojection-error as formulated in Equation 6. The optimization jointly optimizes for the 5 intrinsic parameters, 3 extrinsic translation parameters, 3 extrinsic rotation parameters and one λ_c parameter per correspondence. Convergence is reached when the gradient of the optimized parameters is zero and the found solution is a minimum. The resulting high-dimensional problem contains a multitude of local minima, whereas each represents a configuration for the camera pose that well explains the dependency (Equation 3) between pixels, world objects and the camera.

Figure 11 displays the resulting parameters for the camera S40 Far. We plot each of the parameters against the scale of the remaining loss of the correspondences and the λ_c. The translation parameters are in meters and relative to the UTM projection [16]. The rotation parameters are in degrees of Euler angles.

The plots show that the standard deviation σ_T of the translations does not exceed $6 * 10^{-8}m = 60nm$, σ_R of the rotation is at most $2 * 10^{-9}deg$ and σ_π of the focal length is at most $1 * 10^{-6}$ pixels. This implies that compared to the uncertainty introduced by the measurements (subsubsection 4.3.4) the algorithmic error can be neglected.

We evaluate the algorithmic error for the remaining cameras in the Appendix (section 7). The results are the same as expected.

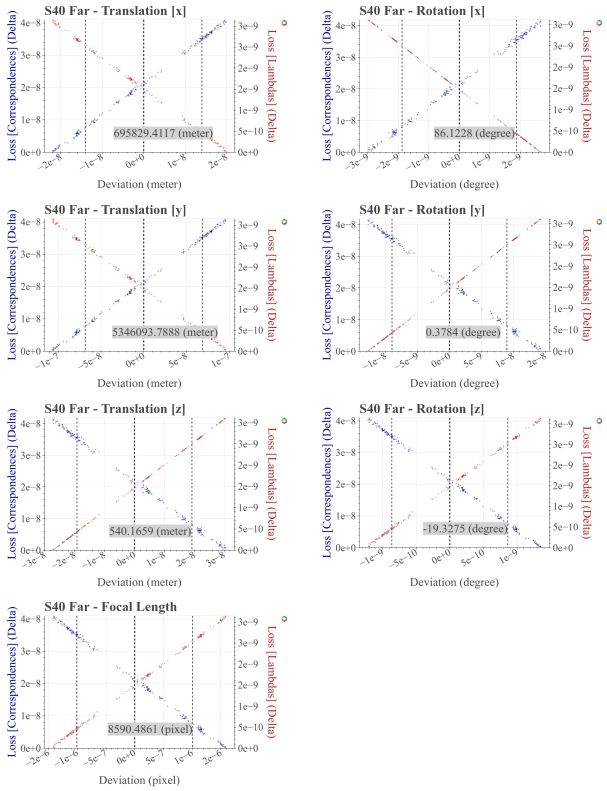

Figure 11. Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. Bottom: The resulting focal length plotted against the remaining losses.

5. Future Work

The project leaves us with the opportunity to continue the research in multiple directions.

5.1. Test on different weather/lighting conditions

We tested and evaluated the implementations on recordings with good weather and lighting conditions, thus a next step is to test the implementations in bad weather and lighting conditions, *e.g.* by night, rain and snow. From out current perspective the feature based dynamic stabilization approach will not suffer in performance as the detected features only depend on features in the image space. Only by night and if the static background is occluded the stabilization pipeline will fail.

We implemented the solver for the BA problem to include human interaction when mapping from PDs to pixels. The mapping will be harder in bad weather and lighting conditions based on the worse visibility of the landmarks. We propose an automatic mapping scheme in subsubsection 5.3.3. This scheme will be affected by changing weather and lighting conditions as the detection of new landmarks is also based on the visibility of landmarks.

5.2. Dynamic Stabilization

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

5.2.1 Warp Field Stabilization based on Optical Flow

We use the optical flow to measure the performance of the stabilizers as described in subsubsection 4.2.1. The optical flow is a 2D vector field where each vector is a displacement vector showing the movement of pixels between frames caused by movement of the objects or cameras. The image can be stabilized using the inverse vector field that also minimizes the reprojection error between frames.

5.2.2 Deep Learning based approaches

The ongoing success of deep learning based approaches in computer vision, especially with convolutional neural networks (CNN), a self-learning stabilization procedure might be developed. The CNN expects the current input and reference frame and outputs the homographic transform or the warped frame. This might speed up the pipeline and inherently adds a measure for the uncertainty of the results by modelling the probability of a homographic transformation. This approach can be used to fuse the feature detection, matching and warping steps into one joint step that is learned by the CNN from labeled data.

5.3. Static Calibration

For the static calibration there are several improvements possible.

5.3.1 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.

5.3.2 Automatic detection of more landmarks after initial calibration

Currently the detection and mapping of landmarks is done by hand using a watershed algorithm [19, 7] 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.

5.3.3 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 (**??**). 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 (**??**) 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.

5.3.4 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.

5.3.5 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. 5.3.2 and Sec. 5.3.3 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.

6. 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. 2

[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, 5, 12

[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] E. Arnold, M. Dianati, R. de Temple, and S. Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–13, 2020. 2

[6] 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, 12

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 3, 5, 6, 9, 12

[8] Isabela Erdelean, Abdelmename Hedhli, Martin Lamb, Niklas Strand, and Ewa Zofka. Catalogue of connected and automated driving test sites: Deliverable no2. 1. 2019. 2

[9] 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

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[11] 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

[12] Morteza Ghahremani, Yonghuai Liu, and Bernard Tiddeman. Ffd: Fast feature detector. *IEEE Transactions on Image Processing*, 30:1153–1168, 2021. 3, 5, 12

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

[14] 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

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

[16] Richard B Langley. The utm grid system. *GPS world*, 9(2):46–50, 1998. 3, 8

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

[18] 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. 6

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

[20] 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

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

[22] 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, 5, 12

[23] David G Stork, Richard O Duda, Peter E Hart, and D Stork. Pattern classification. *A Wiley-Interscience Publication*, 2001. 3

[24] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. 2

[25] 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

[26] 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

7. Appendix

To declutter the report we moved many of the plots in this section. Please see the next pages for the plots.

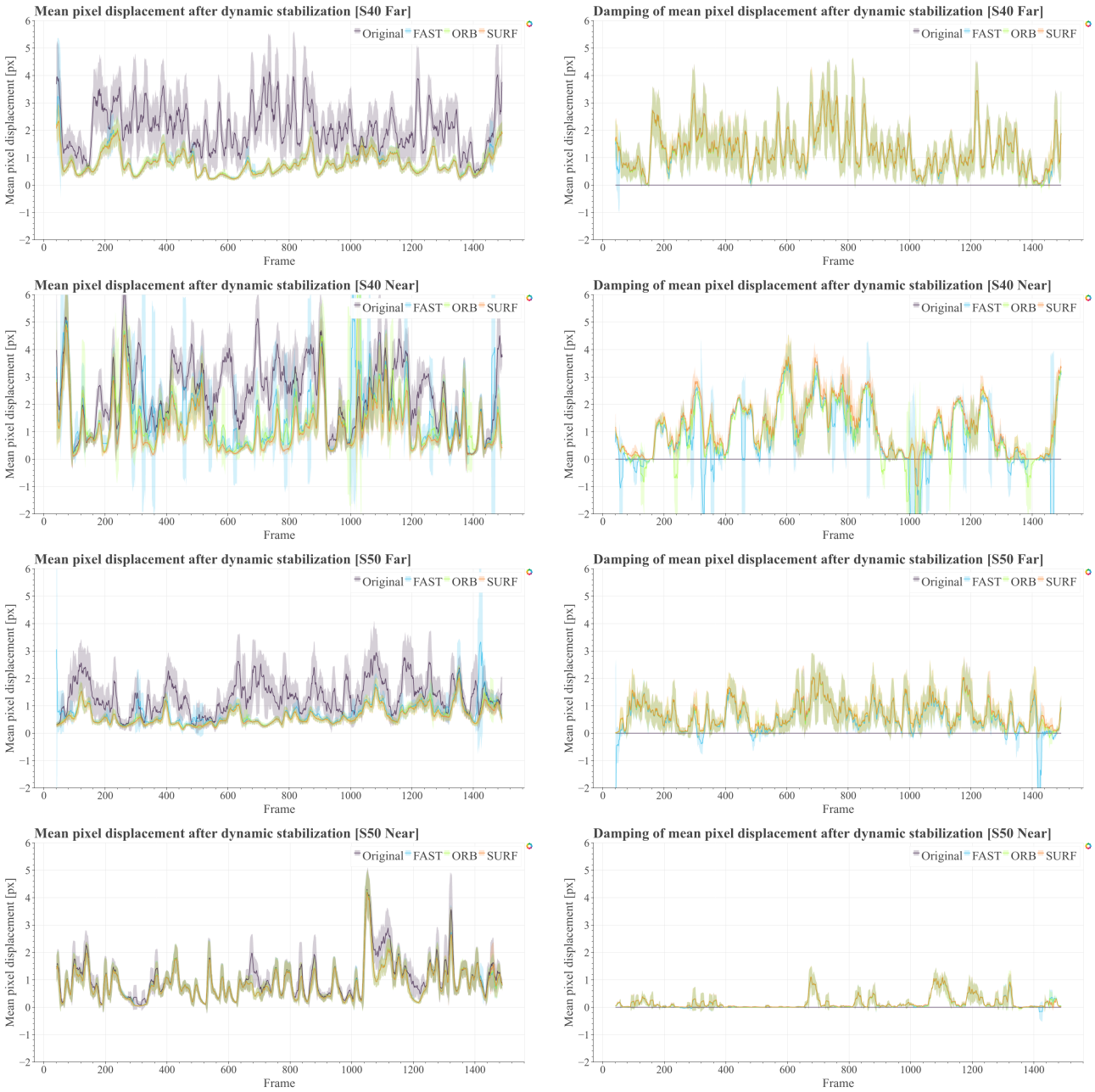

Figure 12. Left: Comparison of the three implemented dynamic stabilizers and the original not stabilized video feed using Optical Flow as metric (lower is better). The stabilizers are based on the FAST [12, 7] feature detector with FREAK [2, 7] feature descriptors, SURF [6, 7] feature detector and ORB [22, 7] 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.

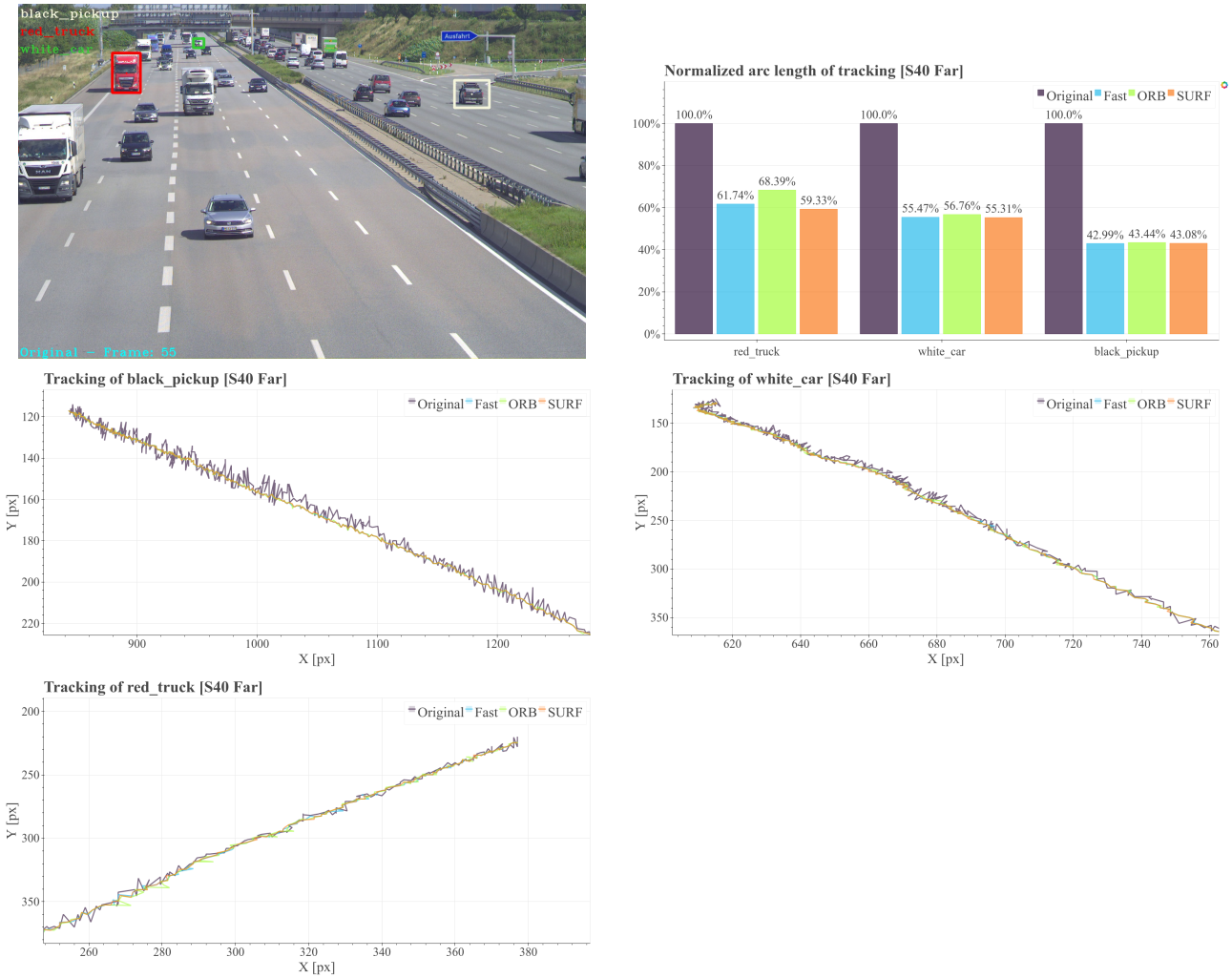

Figure 13. Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The length is normalized by the original arc length, hence the 1.0 factor for the original video. As the jitter is removed, the pixels 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.

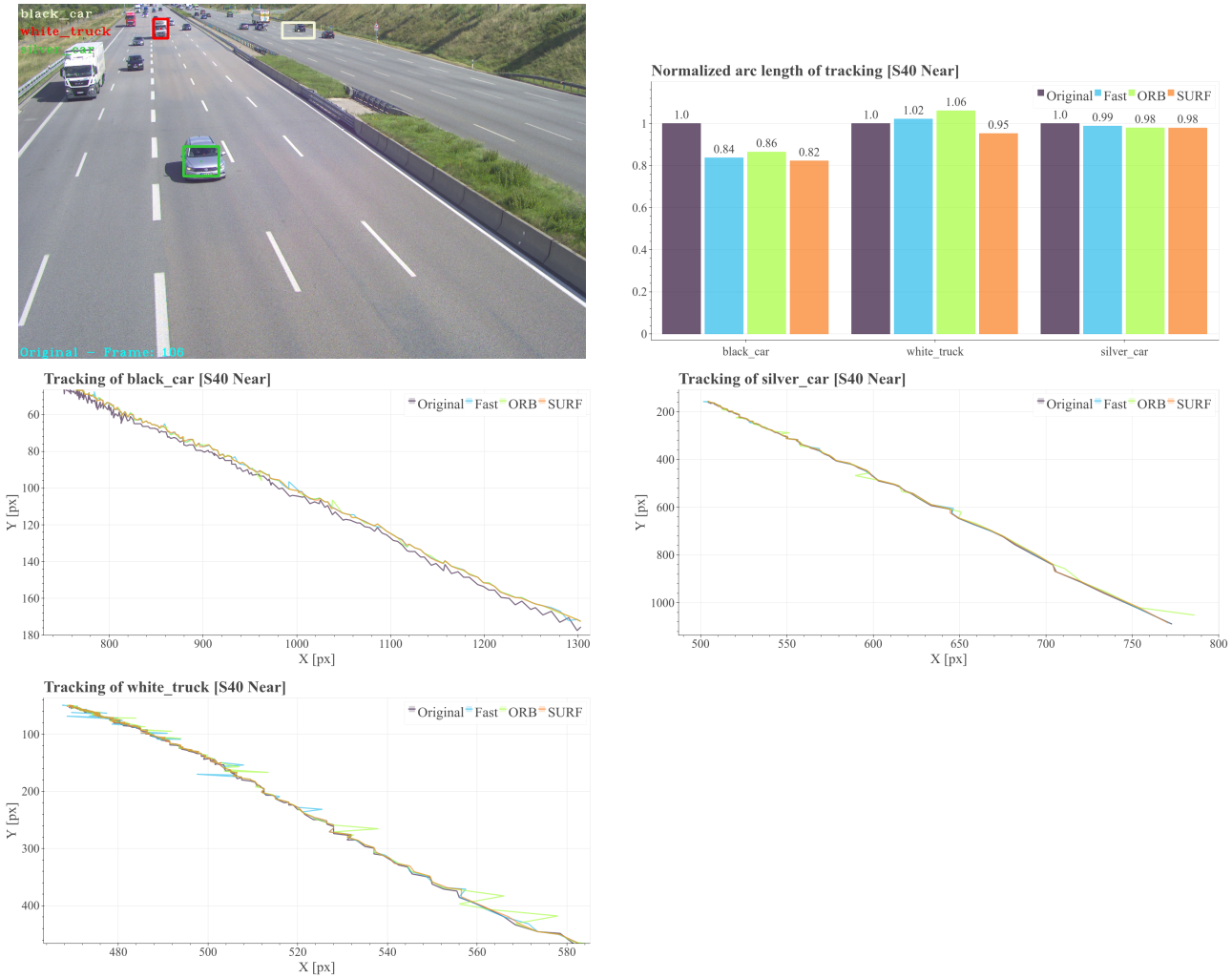

Figure 14. Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The length is normalized by the original arc length, hence the 1.0 factor for the original video. As the jitter is removed, the pixels 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.

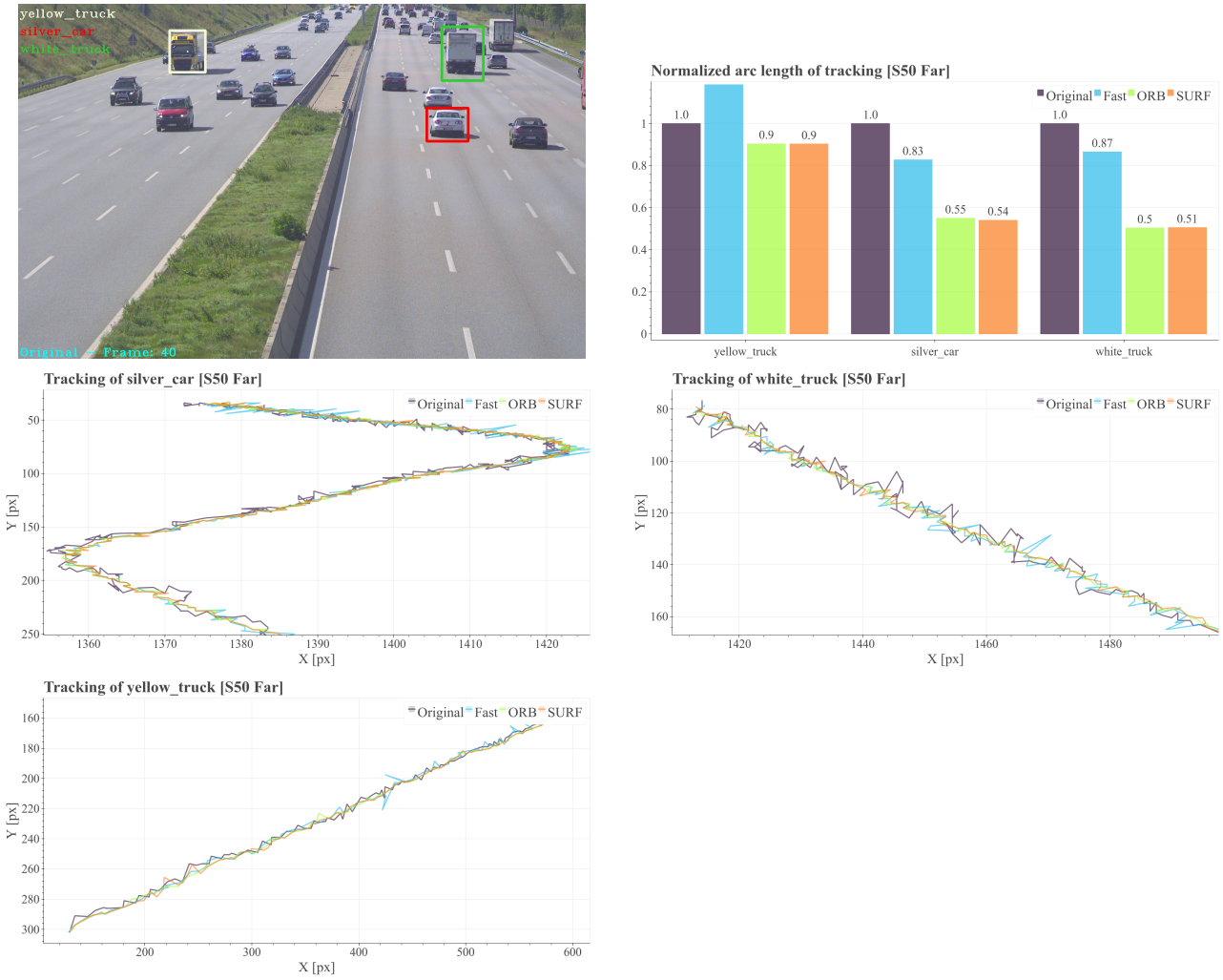

Figure 15. Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The length is normalized by the original arc length, hence the 1.0 factor for the original video. As the jitter is removed, the pixels 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.

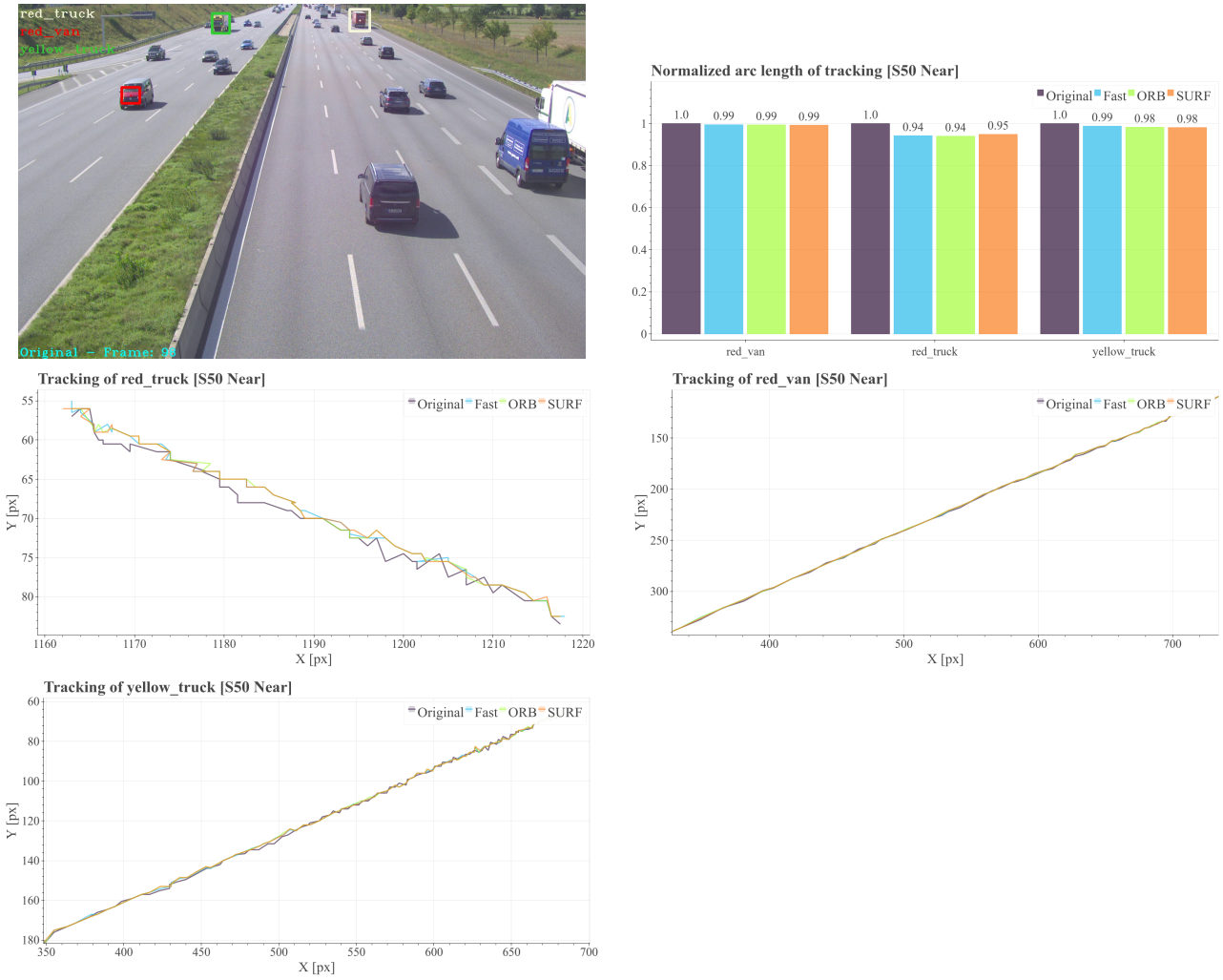

Figure 16. Left: The exemplary vehicles tracked through the video sequence of the camera S40 Near. Right: The corresponding normalized arc lengths of the pixel path. The length is normalized by the original arc length, hence the 1.0 factor for the original video. As the jitter is removed, the pixels 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.

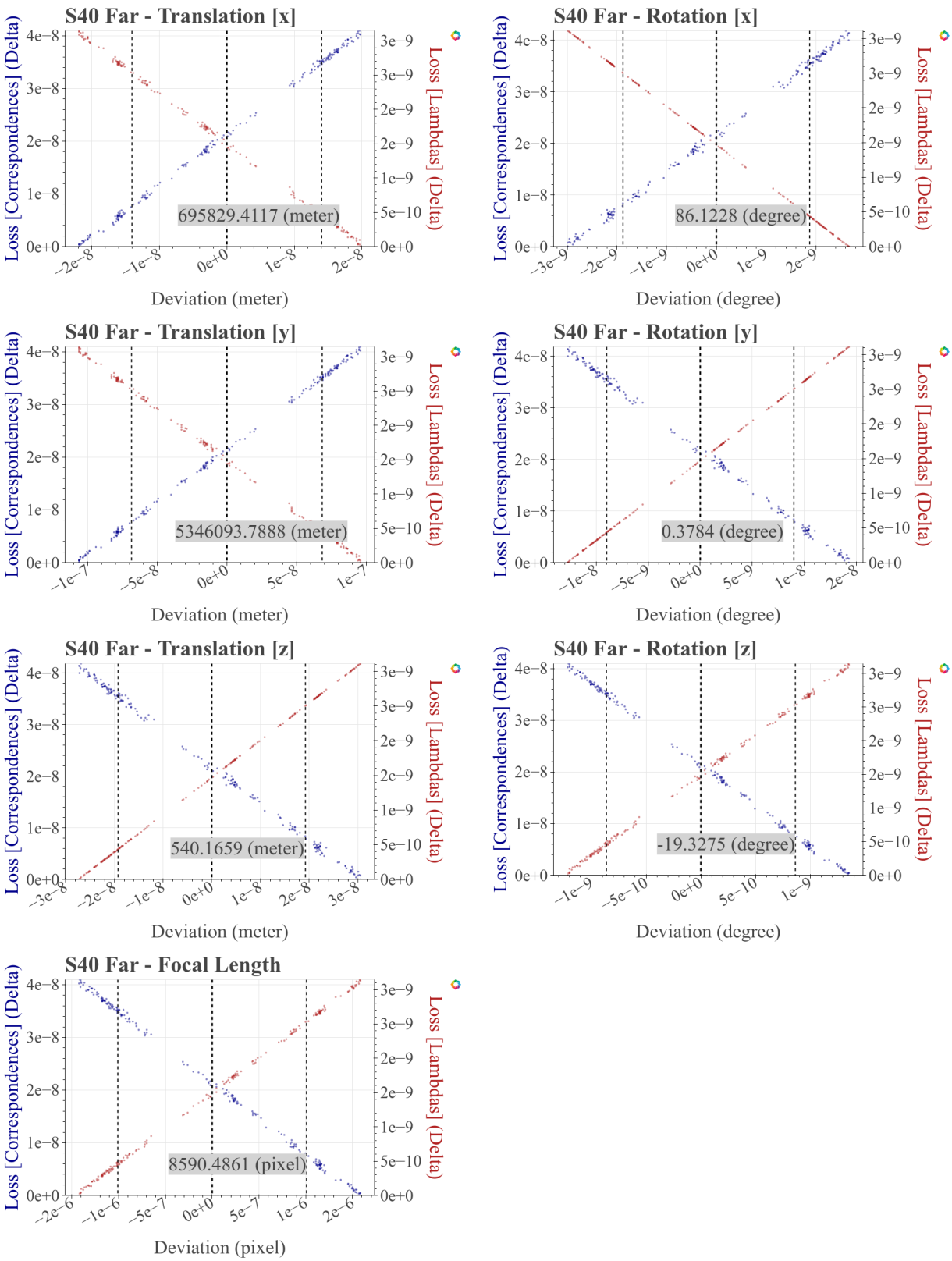

Figure 17. Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. Bottom: The resulting focal length plotted against the remaining losses.

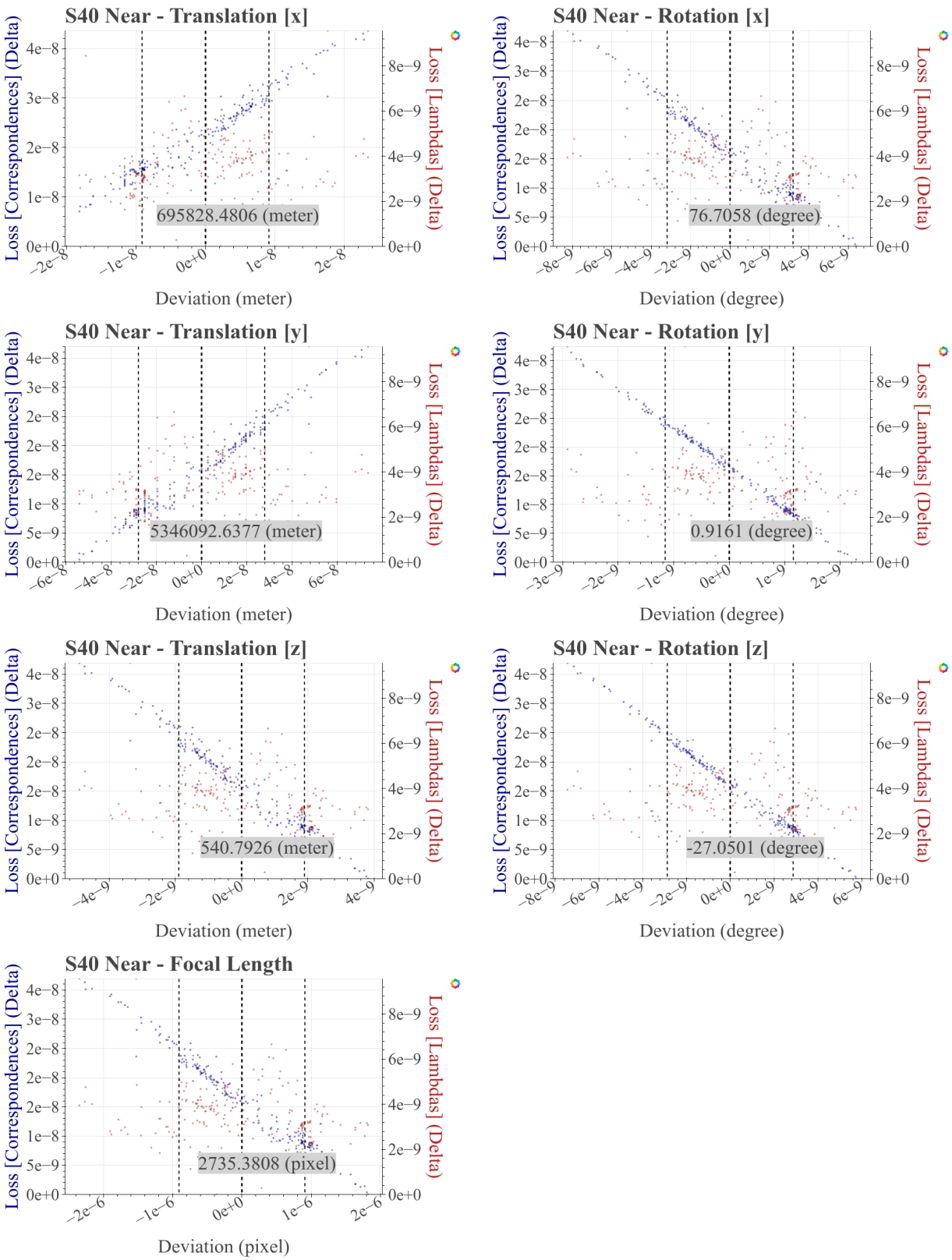

Figure 18. Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. Bottom: The resulting focal length plotted against the remaining losses.

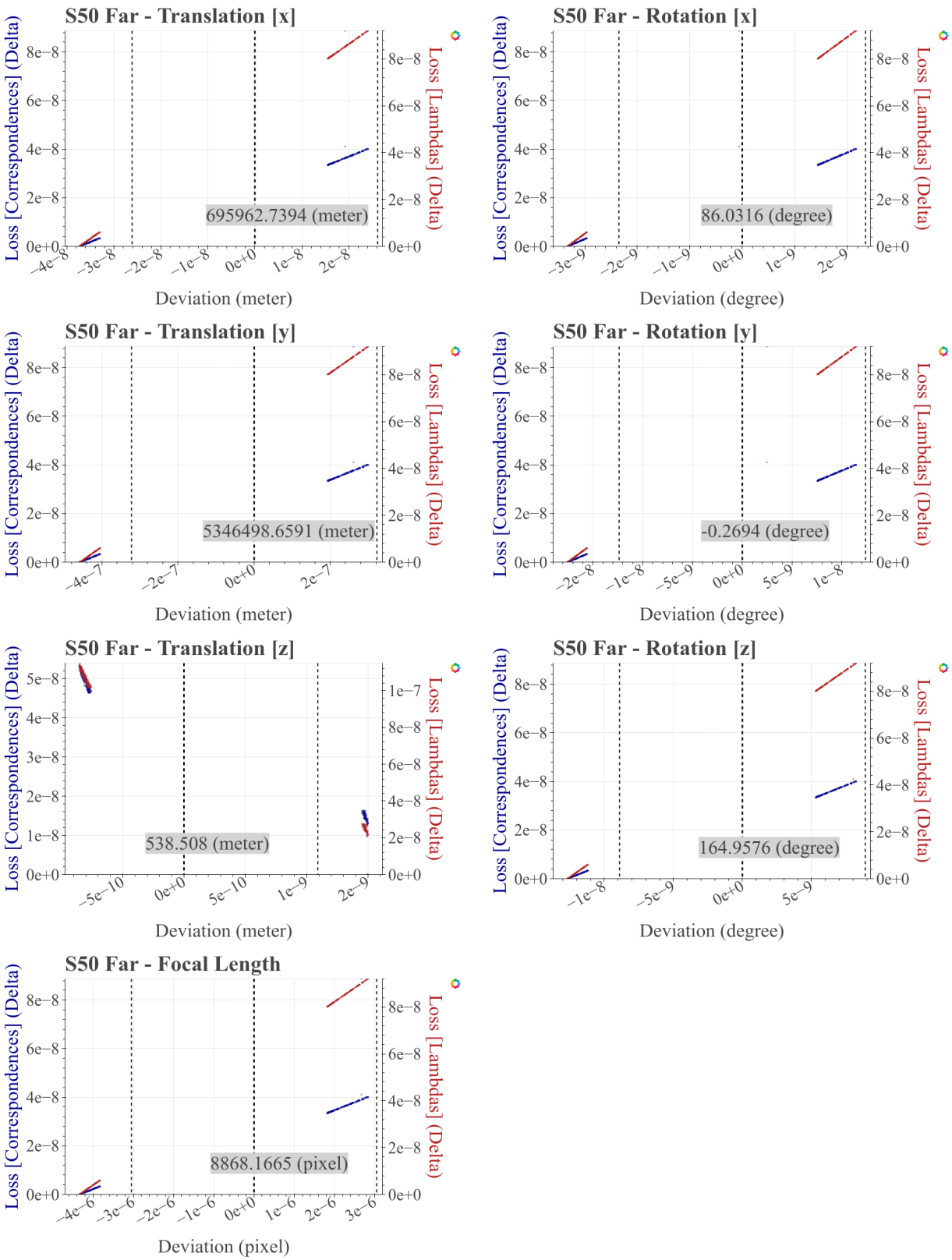

Figure 19. Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. Bottom: The resulting focal length plotted against the remaining losses.

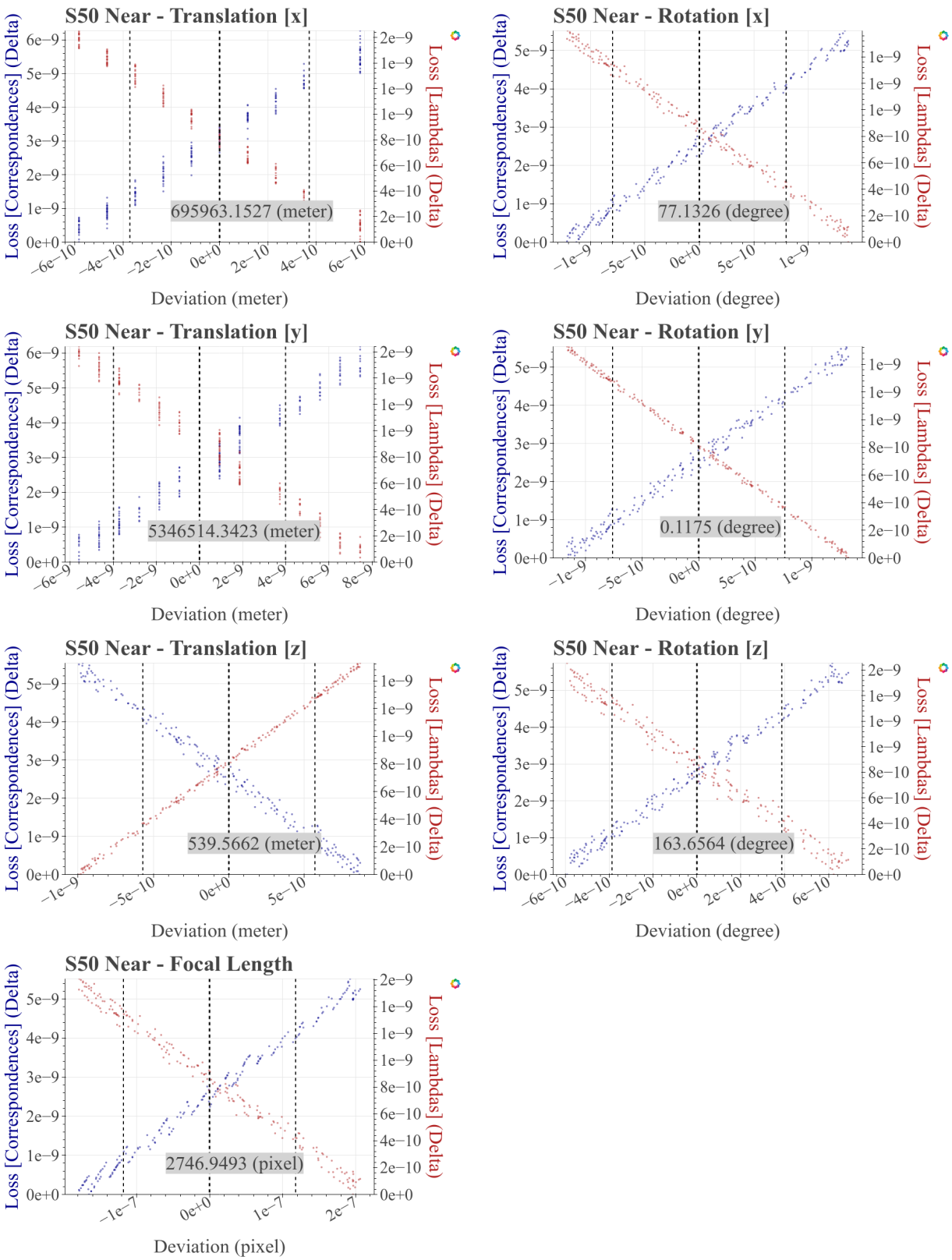

Figure 20. Left: The resulting translational parameters plotted against the remaining losses. Right: The resulting rotational parameters plotted against the remaining losses. Bottom: The resulting focal length plotted against the remaining losses.