Supplemental Material for: Why Having 10,000 Parameters in Your Camera Model is Better Than Twelve

Thomas Schöps[1] Viktor Larsson[1] Marc Pollefeys[1,2] Torsten Sattler[3]
[1]Department of Computer Science, ETH Zürich [3]Chalmers University of Technology
[2]Microsoft Mixed Reality & AI Zurich Lab

In this supplemental material, we present additional information that did not fit into the paper for space reasons. In Sec. 1, we present the feature detection process that we use to find the approximate locations of star center features in images of our calibration pattern. In Sec. 2, we present the initialization process for camera calibration. In Sec. 3, we discuss additional details of our bundle adjustment. In Sec. 4, we present larger images for the camera model validation experiment. In Sec. 5, we present more details of the results of the Structure-from-Motion experiment that tests the impact of different camera models in an application context.

1. Feature detection

First, we find all AprilTags in the image using the AprilTag library [4]. The four corners of each detected AprilTag provide correspondences between the known calibration pattern and the image. We use these to compute a homography that approximately maps points on the pattern into the image. This will only be perfectly accurate for pinhole cameras and planar patterns. However, in general it will be locally appropriate next to the correspondences that were used to define the homography. With this, we roughly estimate the positions of all star features that are directly adjacent to an AprilTag. Each feature location is then refined and validated with the refinement process detailed below. After refinement, the final feature positions provide additional correspondences between the pattern and the image. For each newly detected and refined feature location, we compute a new local homography from the correspondences that are closest to it in order to predict its remaining adjacent feature locations that have not been detected yet. We then repeat the process of feature refinement, local homography estimation, and growing, until no additional feature can be detected.

The initial feature locations predicted by the above procedure can be relatively inaccurate. Thus, we first apply a well-converging matching process based on the known pattern appearance to refine and validate the feature predictions, before we apply an accurate refinement step with a smaller convergence region afterwards.

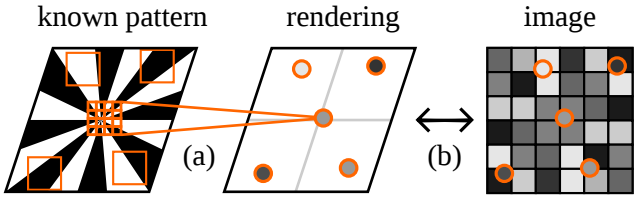


Figure 1. Sketch of matching-based feature refinement. (a) Based on an estimate for a local homography between the pattern and the image, the known pattern is rendered with supersampling (subpixels illustrated for center sample only for clarity). The result are rendered grayscale samples shown in the center. (b) The samples are (rigidly) moved within the image to find the best matching feature position, while accounting for affine brightness differences.

Matching-based refinement and validation. The goal of matching-based refinement is to improve an initial feature position estimate and reject wrong estimates. As input, the process described above yields an initial rough estimate of the feature position and a homography that locally approximates the mapping between the known calibration pattern and the image. The matching process uses the homography to render a synthetic image of the pattern in a small window around the feature position. Then it locally optimizes the alignment between this rendering and the actual image. This is illustrated in Fig. 1.

In detail, we define a local window for refinement, for which we mostly use 21×21 pixels. The optimum size depends on many factors such as the blur introduced by out-of-focus imaging, internal image processing in the camera, and clarity of the calibration pattern. It is thus a parameter that should be tuned to the specific situation. Within this window, we sample as many random points as there are pixels in the window. We assume that the window is centered on the feature location, and given the local homography estimate, we determine the intensity that would be expected for a perfect pattern observation at each sample location.

We use 16x supersampling to more accurately predict the intensities.

Next, we match this rendering with the actual image observation by optimizing for a 2-dimensional translational shift $\mathbf{x}$ of all samples in the image. In addition, since the black and white parts of the pattern will rarely be perfectly black and white under real imaging conditions, we optimize for an affine brightness transform to bring the sample intensities and image intensities into correspondence. In total, we thus optimize for four parameters with the following cost function:

$$C_{\text{match}}(\mathbf{x}, f, b) = \sum_{i}^{n} \left(f \cdot p_i(\mathbf{x}) + b - q_i\right)^2 \quad , \qquad (1)$$

where f and b are the affine factor and bias, $p_i(\mathbf{x})$ is the bilinearly interpolated image intensity at the sample position i with the current translation shift $\mathbf{x}$, and q_i is the rendered intensity of sample i. Given the initial translation offset $\mathbf{x} = \mathbf{0}$, we can initialize f and b directly by minimizing C_{match}. Setting both partial derivatives to zero eventually yields (dropping i from the notation for brevity):

$$f = \frac{\sum(qp) - \frac{1}{n}\sum p \sum q}{\sum(pp) - \frac{1}{n}(\sum p)^2} \quad , \quad b = \frac{\sum q - f \sum p}{n} \quad . \qquad (2)$$

Subsequently, we optimize all four parameters with the Levenberg-Marquardt method. The factor parameter f is not constrained in this optimization and may become negative, indicating that we have likely found a part of the image that looks more like the inverse of a feature than the feature itself. While this may appear like a deficiency, since pushing the parameter to be positive might have nudged the translation $\mathbf{x}$ to go towards the proper feature instead, we can actually use it to our advantage, as we can use the condition of f to be positive as a means to reject outliers. This allows us to obtain virtually outlier-free detections.

Note that for performance reasons, in this part of the refinement we do not optimize for the full homography that is used at the start to render the pattern prediction. This changes in the following symmetry-based refinement step, which is described in Sec. 3.2 in the paper.

2. Calibration initialization

In this section, we describe how we obtain an initial camera calibration that is later refined with bundle adjustment, as described in the paper.

For each image, we first interpolate the feature detections over the whole pattern area for initialization purposes, as in [2]. This is important to get feature detections at equal pixels in different images, which is required for the relative pose initialization approach [2] that is used later. Since we know that the calibration pattern is approximately planar, we can use a homography to approximately map between the pattern and the image (neglecting lens distortion). Each square of four adjacent feature positions is used to define a homography, which we use for mapping within this square. This allows to obtain dense approximate pattern coordinates for all image pixels at which the pattern is visible. These approximately interpolated matches are only used for initialization, not for the subsequent refinement.

We then randomly sample up to 500 image triples from the set of all available images. We score each triple based on the number of pixels that have a dense feature observation in each of the three images. The image triple with the highest number of such pixels is used for initialization.

Since all tested cameras were near-central, we always assume a central camera during initialization (and switch to the non-central model later if requested). We thus use the linear pose solver for central cameras and planar calibration targets from [2]. For each image pixel which has an (interpolated) feature observation in each of the three images chosen above, the corresponding known 3D point on the calibration pattern is inserted into a 3D point cloud for each image. The initialization approach [2] is based on the fact that for a given observation, the corresponding points in the three point clouds must lie on the same line in 3D space. It solves for an estimate of the relative pose of the three initial images, and the position of the camera's optical center. This allows to project the pattern into image space for each pixel with a matched pattern coordinate, which initializes the observation direction for these pixels.

Next, we extend the calibration by localizing additional images using the calibration obtained so far with standard techniques [1]. Each localized image can be used to project the calibration pattern into image space, as above, and extend the calibrated image region. For pixels that already have an estimate of their observation direction, we use the average of all directions to increase robustness. If more than one calibration pattern is used, we can determine the relative pose between the targets from images in which multiple targets are visible. We localize as many images as possible with the above scheme.

As a result, we obtain a per-pixel camera model which stores an observation direction for each initialized pixel. We then fit the final camera model to this initialization by first setting the direction of each grid point in the final model to the direction of its corresponding pixel in the per-pixel model. If the pixel does not have a direction estimate, we guess it based on the directions of its neighbors. Finally, using a non-linear optimization process with the Levenberg-Marquardt method, we optimize the model parameters such that the resulting observation directions for each pixel match the per-pixel initialization as closely as possible.

Due to the size of the local window in feature refinement

Label	Used resolution	Field-of-view (FOV) (approximate)	Type	Description
D435-C	1920×1080	$70^\circ \times 42^\circ$	RGB	Color camera of an Intel RealSense D435
D435-I	1280×800	$90^\circ \times 64^\circ$	Mono	Infrared camera of an Intel RealSense D435
SC-C	640×480	$71^\circ \times 56^\circ$	RGB	Color camera of an Occipital Structure Core (color version)
SC-I	1216×928	$57^\circ \times 45^\circ$	Mono	Infrared camera of an Occipital Structure Core (color version)
Tango	640×480	$131^\circ \times 99^\circ$	Mono	Fisheye camera of a Google Tango Development Kit Tablet
FPGA	748×468	$111^\circ \times 66^\circ$	Mono	Camera attached to an FPGA
GoPro	3000×2250	$123^\circ \times 95^\circ$	RGB	GoPro Hero4 Silver action camera
HTC One M9	3840×2688	$64^\circ \times 47^\circ$	RGB	Main (back) camera of an HTC One M9 smartphone

Table 1. Specifications of the cameras used in the evaluation: The resolution at which we used them, and the approximate field-of-view, which is measured horizontally and vertically at the center of the image. SC-I provides images that are pre-undistorted by the camera.

(*cf*. the section on feature extraction in the paper), features cannot be detected close to the image borders (since the window would leave the image). We thus restrict the fitted model to the axis-aligned bounding rectangle of the feature observations.

3. Bundle adjustment details

Gauge freedom. As mentioned in the paper, in our setting there are more dimensions of Gauge freedom than for typical Bundle Adjustment problems. Note that we do not use any scaling information for the pattern(s) during bundle adjustment, but scale its result once as a post-processing step based on the known physical pattern size. For the central camera model, the Gauge freedom dimensions are thus: 3 for global translation, 3 for global rotation, 1 for global scaling, and 3 for rotating all camera poses in one direction while rotating all calibrated observation directions in the opposite direction. For the non-central camera model, the Gauge freedom dimensions are those listed for the central model and additionally 3 for moving all camera poses in one direction while moving all calibrated lines in the opposite direction. Furthermore, if the calibrated lines are (nearly) parallel, there can be more directions, since then the cost will be invariant to changes of the 3D line origin points within the lines.

Calibration data bias. For parametric models, whose parameters affect the whole image area, different densities in detected features may introduce a bias, since the camera model will be optimized to fit better to areas where there are more feature detections than to areas where there are less. Note that this is a reason for image corners typically being modeled badly with these models, since there typically are very few observations in the corners compared to the rest of the image, and the corners are at the end of the range of radius values that are relevant for radial distortion.

For our generic models, while there is some dependence among different image areas due to interpolation within the grid, they are mostly independent. Thus, this kind of calibration data bias should not be a concern for our models. However, unless using regularization, all parts of the image need to contain feature detections to be well-constrained (which is again most difficult for the image corners).

4. Camera model validation

In Fig. 2, we present larger images for the camera model validation experiment in the paper (Fig. 7 in the paper), such that more details are visible when zooming in a digital version of the PDF. Furthermore, results for additional cameras are included in the figure. The specifications of all cameras in the figure are given in Tab. 1. Note that the "Tango" camera has a fisheye lens and shows hardly any image information in the corners. Thus, there are no feature detections in the image corners, which causes large Voronoi cells to be there in Fig. 2.

5. Example Application: Camera Pose Estimation

In Fig. 3, we present example images of the sparse 3D reconstructions that we evaluated in the paper to determine how much the results of bundle adjustment in Structure-from-Motion are affected by the choice of camera model. The videos used for these reconstructions were recorded with the SC-C and D435-I cameras by walking in a mostly straight line on a forest road, looking either sideways or forward. For D435-I, we use 10 videos with 100 frames each, whereas for SC-C, we use 7 videos with 264 frames on average.

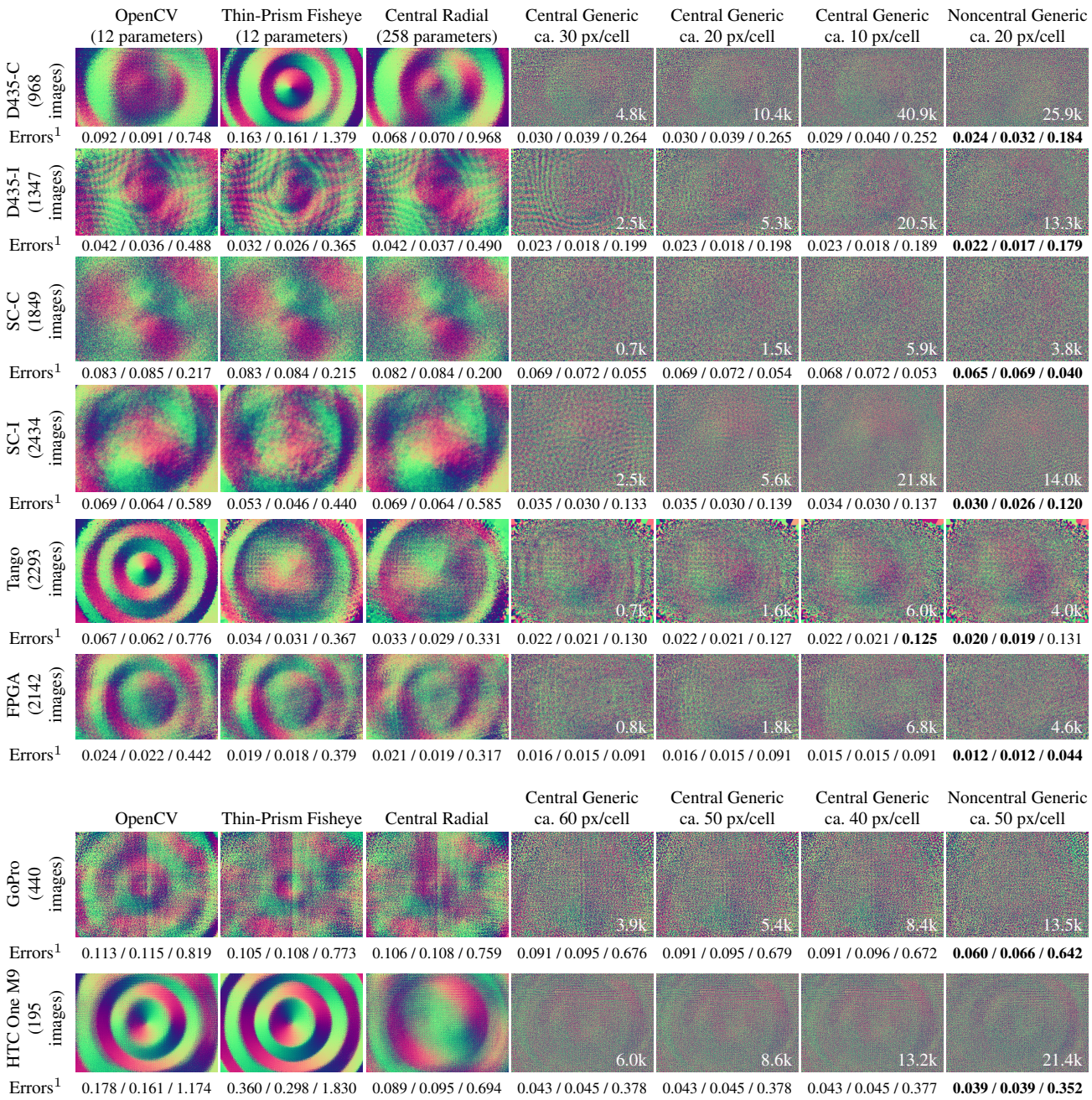
References

[1] Laurent Kneip and Paul Furgale. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *ICRA*, 2014. 2

[2] Srikumar Ramalingam and Peter Sturm. A unifying model for camera calibration. *PAMI*, 39(7):1309–1319, 2016. 2

[3] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *PAMI*, 13(4):376–380, 1991. 5

[4] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. In *IROS*, October 2016. 1



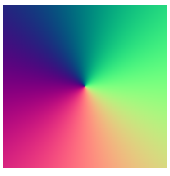


Figure 2. Directions (encoded as colors, see legend on the left) of all reprojection errors for calibrating the camera defined by the row with the model defined by the column. Each pixel shows the direction of the closest reprojection error (*i.e.*, the images are Voronoi diagrams) from all used images. Ideally, the result is free from any systematic pattern. Patterns indicate biased results arising from not being able to model the true camera. Parameter counts for generic models are given in the images. [1]Median training error [px] / test error [px] / biasedness.

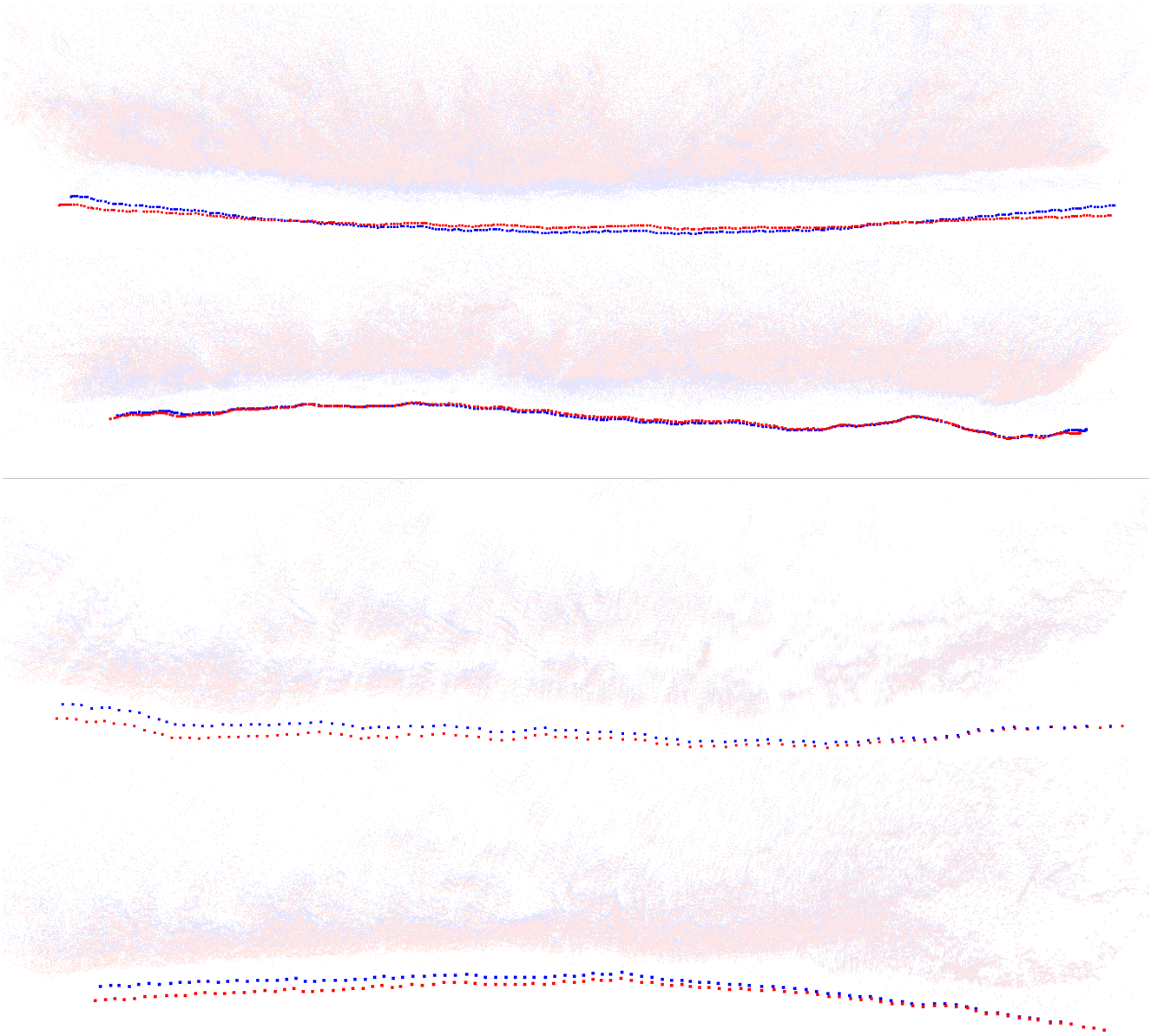


Figure 3. Example reconstructions of forest scenes used for the Structure-from-Motion experiment in the paper, bundle-adjusted with a noncentral-generic and a Thin-Prism-Fisheye calibration. For the noncentral-generic model, camera poses are shown in blue and reconstructed points in light blue. For the Thin-Prism-Fisheye model, camera poses are shown in red and reconstructed points in light red. The two images on the top show reconstructions of videos by the SC-C camera. These reconstruction pairs are aligned by matching the camera positions with the Umeyama method [3] to visualize the difference in the shape of the trajectories. The two images on the bottom show reconstructions of a video by the D435-I camera. Since the differences in shape are smaller for this camera, these reconstructions are aligned at the first camera pose of the video (on the right side) to visualize the accumulated difference when starting from the same pose.