SoftwareX 10 (2019) 100333

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

CameraTransform: A Python package for perspective corrections and image mapping

Richard C. Gerum [a,*], Sebastian Richter [a,b], Alexander Winterl [a,b], Christoph Mark [a], Ben Fabry [a], Céline Le Bohec [c,d], Daniel P. Zitterbart [a,b]

[a] Biophysics Group, Department of Physics, University of Erlangen-Nürnberg, Germany
[b] Applied Ocean Physics and Engineering, Woods Hole Oceanographic Institution, Woods Hole, MA, USA
[c] Centre Scientifique de Monaco, Département de Biologie Polaire, Principality of Monaco, Monaco
[d] Université de Strasbourg, CNRS, IPHC, UMR 7178, Strasbourg, France

ARTICLE INFO

Article history:
Received 5 June 2019
Received in revised form 24 September 2019
Accepted 24 September 2019

Keywords:
Perspective projection
Quantitative image analysis
Geo-referencing
Camera lens distortions

ABSTRACT

Camera images and video recordings are simple and non-invasive tools to investigate animals in their natural habitat. Quantitative evaluations, however, often require an exact reconstruction of object positions, sizes, and distances in the image. Here, we provide an open source software package to perform such calculations. Our approach allows the user to correct for perspective distortion, transform images to "bird's-eye" view projections, or transform image-coordinates to real-world coordinates and vice versa. The extrinsic camera parameters that are necessary to perform such image corrections and transformations (elevation, tilt/roll angle, and heading of the camera) are obtained from the image using contextual information such as a visible horizon, GPS coordinates of landmarks, known object sizes, or images of the same object obtained from different viewing angles. All mathematical operations are implemented in the Python package *CameraTransform*. The performance of the implementation is evaluated using computer-generated synthetic images with known camera parameters. Moreover, we test our algorithm on images of emperor penguin colonies, and demonstrate that the camera tilt and roll angles can be estimated with an error of less than one degree, and the camera elevation with an error of less than 5%. The *CameraTransform* software package simplifies camera matrix-based image transformations and the extraction of quantitative image information. An extensive documentation and usage examples in an ecological context are provided at http://cameratransform.readthedocs.io.

Code metadata

Current code version	v1.1
Permanent link to code/repository used for this code version	https://github.com/rgerum/cameratransform
Legal Code License	MIT
Code versioning system used	git
Software code languages, tools, and services used	Python, Matplotlib, Pylustrator, SciPy, Numpy, Pandas
Compilation requirements, operating environments & dependencies	Python v3.4.x and higher
If available Link to developer documentation/manual	http://cameratransform.readthedocs.org/
Support email for questions	richard.gerum@fau.de

1. Introduction

Camera traps, time-lapse recordings, or video recordings are widely used tools in ecology research [1,2], for example for estimating abundance [3], or for behavioral studies [4,5]. However, such images inherently contain perspective distortions that need

* Corresponding author.
E-mail address: richard.gerum@fau.de (R.C. Gerum).

https://doi.org/10.1016/j.softx.2019.100333

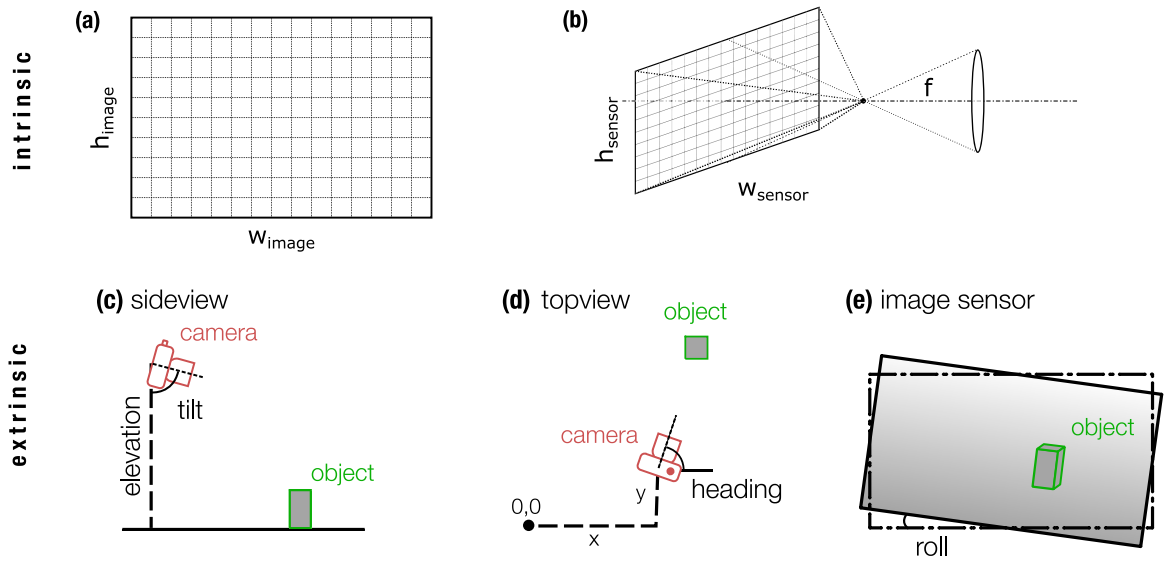

Fig. 1. Camera parameters. Intrinsic parameters: (a) the dimensions of the image in pixels $w_{\text{image}} \times h_{\text{image}}$, (b) the size of the sensor in mm ($w_{\text{sensor}} \times h_{\text{sensor}}$) and the focal length f in mm. Extrinsic parameters: (c) side view: the *elevation* specifies the height of the camera above a reference altitude, e.g. above ground, the *tilt* specifies the angle between the vertical direction and the viewing direction (sensor normal). (d) top view: the *offset* (x, y) specifies the x, y coordinates of the camera relative to a reference position ($x = 0, y = 0$), and the *heading* specifies the angle between the x-direction and the viewing direction (sensor normal). (e) Image sensor: the *roll* specifies the angle between the lower sensor edge and the horizontal direction.

to be accounted for when accurate positions and distances need to be measured. To correct for perspective distortions and to map image points to real-world positions, it is paramount to know the intrinsic and extrinsic camera parameters [6,7]. Intrinsic parameters are the sensor and lens properties. Extrinsic parameters are the geographic camera position relative to landmarks in the scene, the camera elevation, tilt/roll angle, and heading. Some of the extrinsic parameters, however, are often difficult or impractical to measure in the field at the time of the recording. Here we present methods to reconstruct unknown extrinsic camera parameters from features in the image. The mathematical procedure behind this reconstruction is based on linear algebra and is implemented in the Python package *CameraTransform*. In addition to reconstructing extrinsic camera parameters, *CameraTransform* provides a toolbox to transform point coordinates in the image to geographic coordinates. In the following, we explain the mathematical details, estimate the reconstruction uncertainties, and describe practical applications.

2. Software

The complete software *CameraTransform*, released under the MIT license, is implemented in Python 3.6 [8], an interpreted programming language optimized for scientific purposes. For maximum efficiency, several open-source libraries are used. For numerical operations, such as matrix operations, we use the Numpy library [9]. Statistical distributions are implemented using the SciPy package [10]. The data are visualized using the Matplotlib [11] and Pylustrator [12] libraries, and are stored using the Pandas library [13].

3. Camera matrix

All information required for mapping real-world points (x, y, z coordinates in meters) to image points are stored in the camera matrix. The camera matrix is expressed in projective coordinates, and is split into two parts — the intrinsic matrix and the extrinsic matrix that correspond to the intrinsic and extrinsic camera parameters, respectively [14].

3.1. Intrinsic parameters

The intrinsic matrix entries contain information about the focal length f of the camera in mm, the effective sensor dimensions ($w_{\text{sensor}} \times h_{\text{sensor}}$) in mm, and the image dimensions ($w_{\text{image}} \times h_{\text{image}}$) in pixels (see Fig. 1a,b). Specifically, the intrinsic matrix entries are the effective focal length f_{pix} and the center of the sensor ($w_{\text{image}}/2$, $h_{\text{image}}/2$)

$$C_{\text{intr.}} = \begin{pmatrix} f_{\text{pix}} & 0 & w_{\text{image}}/2 & 0 \\ 0 & f_{\text{pix}} & h_{\text{image}}/2 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

$$f_{\text{pix}} = f \cdot w_{\text{image}}/w_{\text{sensor}} \tag{2}$$

The diagonal elements account for the re-scaling from pixels in the image to a position in mm on the sensor. The off-diagonal elements account for the offset between image and sensor coordinates, whereby the origin of the image is at the top left corner, and the origin of the sensor coordinates is at the center.

Eq. (1) only applies for the rectilinear projection of a pinhole camera (camera obscura). The *CameraTransform* package also supports cylindrical or equirectangular projections, which cannot be expressed in matrix notation. These are commonly used for panoramic images (see Supplementary Information). The package also supports corrections for radial lens distortions (see Supplementary Information).

3.2. Extrinsic parameters

The extrinsic matrix consists of the offset (x, y, z) of the camera relative to an arbitrary fixed real-world reference point (0,0,0). Customarily, ($x = 0$, $y = 0$) is the position of the camera, and the z-coordinate of the reference point is the ground level so that z is the elevation of the camera. The x, y plane of the coordinate system is usually the horizontal plane. Furthermore, the extrinsic matrix contains the tilt angle α_{tilt}, the heading angle α_{heading}, and the roll angle α_{roll} of the camera (see Fig. 1c–e and Supplementary Information).

The extrinsic parameters are used to rotate and translate the intrinsic camera matrix, with the aim to map or project 3-D

points from real-world coordinates to 2-D image coordinates. The backprojection from a 2-D image point to 3-D real-world coordinates, however, is inherently underdetermined due to the lack of depth information in the image, and therefore requires one additional constraint, e.g. the z-coordinate of that real-world point or its distance to another real-world point. Supplementary Information explains several strategies to perform the rectilinear, cylindrical, or equirectangular backprojection.

4. Fitting of the extrinsic camera parameters

While the intrinsic camera parameters describing camera and lens properties are usually well known, this is often not the case for the extrinsic parameters that define the orientation of the camera. Below, we start with the simplest case where the heading and position of the camera is irrelevant and thus can be set to arbitrary values (e.g. 0). This leaves only three free parameters *elevation*, *tilt* and *roll*, unless the camera was properly horizontally aligned, in which case *roll* is approximately zero. The more complicated case where knowledge of camera heading and position is important, e.g. for multi-camera setups, or if the image needs to be geographically mapped, is described further below.

The *CameraTransform* package provides several fitting routines that allow the user to infer the extrinsic parameters from characteristic features in the image.

4.1. Fitting by object height

If the true height of objects in the image is known, for example for a group of animals, or more generally if distances between points seen in the image are known, the camera parameters can be fitted. This works especially well for the tilt angle as it most sensitively affects the apparent object height (see Supplementary Information, Fig. B.5b). To evaluate the fit parameter uncertainties, we use Metropolis Monte-Carlo sampling [16,17]. The input for this sampling process is a list of base (foot) and top (head) positions of the objects. Optionally, also the position of the horizon, landmarks, or reference objects such as rulers or survey poles can be provided to improve the algorithm (for details see below). The algorithm projects the foot positions from the image to real-world coordinates, using the constraint $z = 0$, then projects the head positions from the image to real-world coordinates, using as a constraint the x position of the projected foot points. The distance between these pairs of points is the estimated height of the object. This height is assigned a probability according to a known height distribution (e.g. the user does not need to know the exact height but can instead provide an estimate for the mean and standard deviation, or any non-Gaussian distributions of the expected object heights).

The Metropolis algorithm starts with arbitrarily chosen parameter values (for elevation, tilt, and roll) and evaluates the probability p_0 assigned to these parameter values. Optionally, the user can provide starting values, but usually the algorithm converges well from a random initialization. Subsequently, small random numbers are added to the parameter values, and the corresponding probability p_1 is re-evaluated. If the probability increases, the new parameter sample is saved, if the probability decreases, the new sample is only saved with a probability of p_1/p_0, otherwise discarded. After many such iterations (typically 10 000 in our case, which takes roughly 0.5 min on a standard desktop PC), the saved set of parameter samples represents the distribution of the fitted parameters (elevation, tilt, and roll). From these parameter samples, one can finally compute the mean value, denoting the best-fit parameter values, and the standard deviation or credible intervals, denoting the uncertainty of the parameters.

Optionally, if a horizon is visible in the image, *CameraTransform* uses the horizon line as an additional constraint to determine the extrinsic camera parameters. Based on the elevation, tilt and roll, the astronomical horizon line of a perfectly spherical earth is projected onto the image, and its distance to the user-provided horizon is minimized.

To evaluate this method, an artificial image (Fig. 2a) is created using the *CameraTransform* package. A set of rectangles with a width of 30 cm and a height of 0.75 m are randomly placed at distances ranging from 50 m to 150 m, and subsequently projected to the image plane using the following camera parameters: focal length 14 mm, sensor size 17.3 × 9.7 mm with 4608 × 2592 px, camera height 16.1 m, tilt angle 85.3°, and roll angle 0.3°. Using the software *ClickPoints* [18], we mark the base and top positions of these rectangles in the image (Fig. 2c) and provide them as input for the sampling routine. We then investigate how the estimated distribution of elevation, tilt angle, and roll angle vary with the number of provided objects. The analysis is performed with and without a horizon, for 5, 10, 25, and 50 randomly chosen objects. The results show that by including a larger number n of objects, the uncertainty of the parameter estimate (as indicated by the width of the distribution) decreases roughly as $n^{-0.5}$ (Fig. 2). Moreover, we find that both the camera elevation and the tilt angle can be fitted with considerably less uncertainty if a horizon is provided (Fig. 2g,h,i), compared to parameter estimates without horizon (Fig. 2d,e,f).

Furthermore, the uncertainty of the parameter estimates depends on the position of the objects in the image (see Supplementary Information). Objects that are evenly distributed throughout the image provide better estimates compared to objects that are clustered in the front or in the back of the image.

To demonstrate the fitting procedure, we analyze an image (Fig. 3a) from a wide-angle camera overseeing an Emperor penguin (*Aptenodytes forsteri*) colony at Pointe Géologie, Antarctica (micrObs system, see [15]). The camera was positioned on an island with an elevation above sea ice level of 19 m as measured by differential GPS. We estimate the extrinsic camera parameters by providing the feet and head positions of 50 animals, assuming an average height of 0.75 m with an uncertainty that is left as a free parameter. Furthermore, we assume that the z-position of all animals is exactly equal as they are standing on the frozen sea ice, which we assume to be flat. Fig. 3b shows the projected top view based on the extracted extrinsic camera parameters. The estimated extrinsic parameters are: elevation 18.9 ± 0.7 m, tilt 84.6 ± 0.42°, and roll −0.2 ± 0.41°. The size variation of the penguins, which was left as a free parameter for the metropolis algorithm to sample, was estimated to be 0.059 m. The obtained parameters can now be used e.g. to estimate the area of huddles in the image. For this purpose, the user paints the region occupied by the heads of huddling penguins (pink line in Fig. 3a), and the circumference of this region is transformed to a top-view projection (pink line in Fig. 3d,f) and moved down by a distance of 0.75 m (one penguin height) to indicate the huddle area (green line in Fig. 3d,f).

4.2. Fitting by geo-referencing

For small tilt angles, e.g. images taken from a helicopter (Fig. 4a), the size of the objects in the image does not vary sufficiently over the range of y-positions, and hence fitting by object height fails. If in addition there is no visible horizon, such images require a different method. If an accurate map or a high-resolution satellite image of the area of interest is available, point correspondences between the image and the map can be used instead to estimate the camera parameters using an image registration approach.

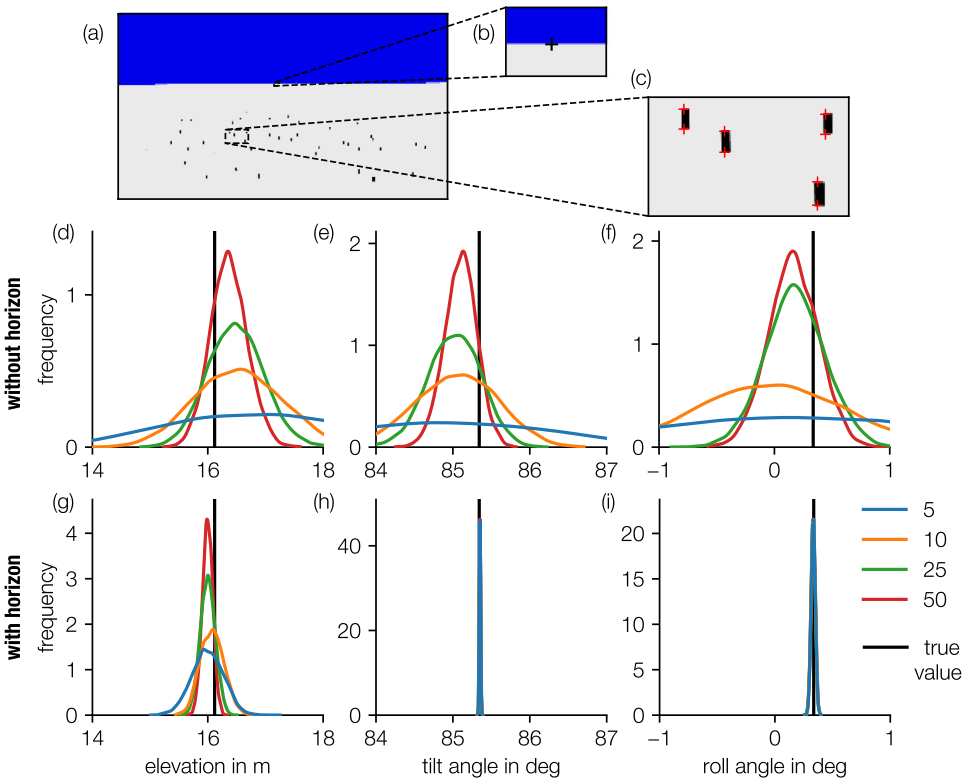

Fig. 2. Influence of number of objects to determine camera elevation, tilt angle and roll angle. (a) Artificial image with randomly placed objects and horizon. (b) 3 points on the horizon and (c) foot and head points of 50 objects are manually selected. Metropolis-sampled uncertainty of the obtained parameters (elevation, tilt, and roll) for a fit without using horizon points (d)–(f) and with horizon points (g)–(i), for different number of randomly selected (without replacing) objects (N = 5, blue line; N = 10, orange line; N = 25, green line; N = 50 red line; true value, black line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the example shown in Fig. 4, photographs of a King penguin (*Aptenodytes patagonicus*) colony at the Baie du Marin (Possession Island, Crozet Archipelago) have been taken from a helicopter flying approximately 300 m above ground. We choose eight distinct points that are recognizable in both the camera image and the satellite image provided by Google Earth (Fig. 4a,c). To calculate the cost function for image registration, we project the image points (blue points in Fig. 4a) onto the satellite image (blue points in Fig. 4c) and calculate the distance to the target points (red points in Fig. 4b,c). The fit routine of *CameraTransform* then computes the *height* and *tilt* of the camera as well as the camera's x,y-position and heading. The example in Fig. 4 with an rms error of 0.76 m between transformed image and target points demonstrates that the fit routine matches all except one point, which is the branch point of a river that has shifted from the time the satellite image was taken (Fig. 4c).

5. Stereo images

If object sizes are unknown, a stereo camera setup can be used to determine the size of multiple objects that are recorded from two different positions and angles, provided that corresponding points in both images can be unambiguously marked, and that either the distance between the two cameras or the absolute size of a single object in the image is known.

CameraTransform estimates the relative orientation of the cameras by minimizing the back-projected distance of a marked point in one image to the epipolar line of the corresponding point in the other image (the epipolar line is the line in image B that corresponds to a single point in image A), and vice versa (see Supplementary Fig. C7). We found that a minimization of the back-projection error in pixels is preferable to a more direct minimization of the distance between the corresponding rays of the points in the world space, due to scaling issues.

In contrast to existing stereo fitting methods (e.g. OpenCV), our method is based on the Monte Carlo approach and provides complete distributions of the estimated parameters to assess the uncertainty of the estimates. Furthermore, our method directly provides the relative orientation of the two cameras, whereas the commonly employed Eight-Point Algorithm [19] only yields the fundamental matrix, from which the relative orientation cannot be unambiguously extracted.

6. Impact

We present the Python package *CameraTransform* for estimating extrinsic camera parameters based on image features, satellite images, and images obtained from multiple viewing angles. Moreover, *CameraTransform* allows users to geo-reference images or to correct images for perspective distortions (e.g. to obtain top-view or "bird's-eye" view projections). The package has been previously applied for studying Emperor and King penguin colonies [5, 15,20], but is generally applicable for other quantitative image analysis where the extrinsic camera parameters were not or could not be measured in the field at the time of the recording. The package is published under the GPLv3 open source license to allow for continuous use and application in science. The documentation is hosted on http://cameratransform.readthedocs.io and contains further details on how to install the package. The documentation also provides numerous usage examples.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

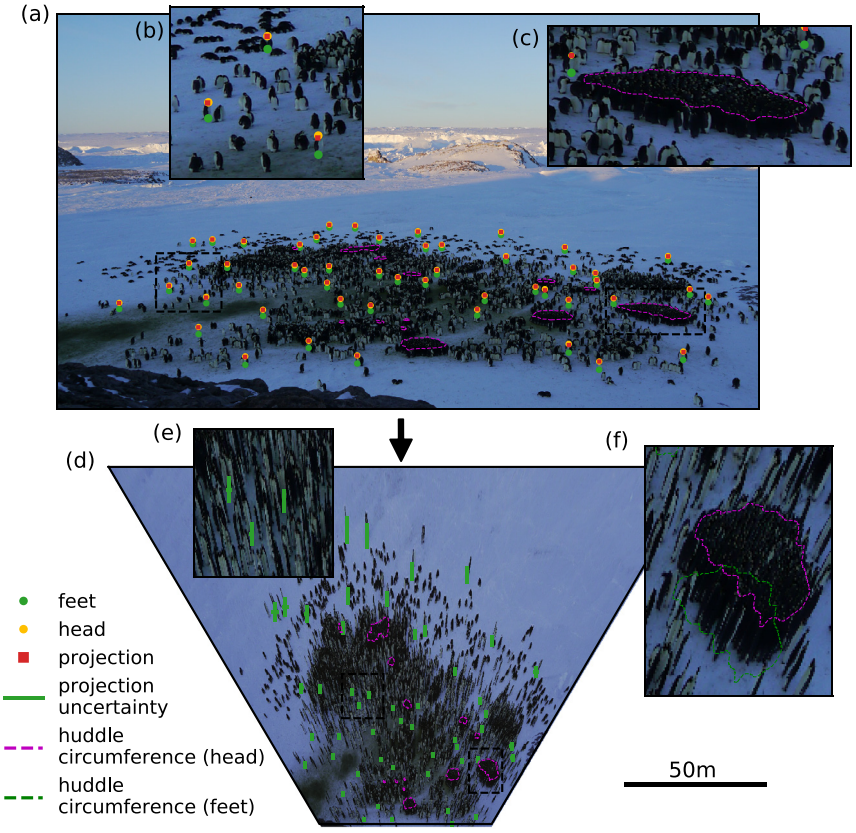

Fig. 3. Fit of camera parameters using image objects. (a) Image taken with the micrObs system [15] of a penguin colony. The feet (green) and head (yellow) positions of 50 penguins were manually marked (shown in inset (b)) and the circumference of a huddle was marked (purple, shown in inset (c)). The head and feet positions are used to estimate the camera perspective (estimated head positions are shown as red squares) for projecting the image to a top view (d). Inset (e) shows the uncertainty of the projected penguin positions and inset (f) shows the projected huddle circumference.

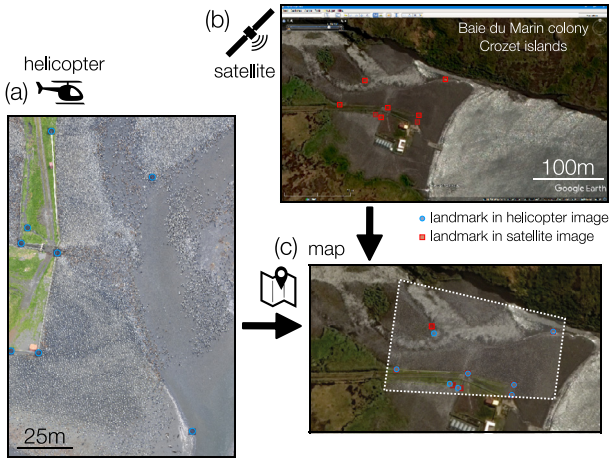

Fig. 4. Fit of camera parameters by image registration. (a) Camera image from a helicopter flight at the Baie du Marin colony at the Crozet islands (Dec 09, 2014) [20]. (b) Satellite image provided by Google Earth. (c) Image fitted over points in the image (blue) with points in the map image (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CRediT authorship contribution statement

Richard C. Gerum: Conceptualization, Formal analysis, Methodology, Software, Visualisation, Vailidation, Writing - original draft, Writing - review & editing. **Sebastian Richter:** Investigation, Software, Data curation, Validation, Writing - review & editing. **Alexander Winterl:** Investigation, Software, Data curation, Validation. **Christoph Mark:** Software, Writing - review & editing. **Ben Fabry:** Project administration, Supervision, Writing - original draft, Writing - review & editing, Funding acquistion. **Céline Le Bohec:** Investigation, Writing - review & editing, Funding acquistion. **Daniel P. Zitterbart:** Investigation, Supervision, Writing - original draft, Writing - review & editing, Funding acquistion.

Acknowledgments

This work was supported by the National Institutes of Health, USA (HL120839), the Institut Polaire Français Paul-Emile Victor, France (IPEV, Program no. 137 to CLB), the RTPI NUTRESS (Réseau Thématique Pluridisciplinaire International “NUTrition et RESistance aux Stress environnementaux” - CSM/CNRS/Unistra), Monaco, and the Deutsche Forschungsgemeinschaft (DFG), Germany grant FA336/5-1 and ZI1525/3-1 in the framework of the priority program “Antarctic research with comparative investigations in Arctic ice areas”.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2019.100333.

References

[1] Gregory T, Carrasco Rueda F, Deichmann J, Kolowski J, Alonso A. Arboreal camera trapping: taking a proven method to new heights. Methods Ecol Evol 2014;5(5):443–51. http://dx.doi.org/10.1111/2041-210X.12177, URL http://doi.wiley.com/10.1111/2041-210X.12177.

[2] Cutler TL, Swann DE. Using remote photography in wildlife ecology: A review. Wildl Soc Bull 1999;27(3):571–81, URL http://www.jstor.org/stable/3784076.

[3] Lynch TP, Alderman R, Hobday AJ. A high-resolution panorama camera system for monitoring colony-wide seabird nesting behaviour. In: Schielzeth H, editor. Methods Ecol Evol 2015;6(5):491–9. http://dx.doi.org/10.1111/2041-210X.12339, URL http://doi.wiley.com/10.1111/2041-210X.12339.

[4] Zitterbart DP, Wienecke B, Butler JP, Fabry B. Coordinated movements prevent jamming in an emperor penguin huddle. PLoS One 2011;6(6). e20260. http://dx.doi.org/10.1371/journal.pone.0020260, URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3106014{&}tool=pmcentrez{&}rendertype=abstract.

[5] Richter S, Gerum RC, Schneider W, Fabry B, Le Bohec C, Zitterbart DP. A remote-controlled observatory for behavioural and ecological research: A case study on emperor penguins. Methods Ecol Evol 2018;9(5). http://dx.doi.org/10.1111/2041-210X.12971, URL http://doi.wiley.com/10.1111/2041-210X.12971.

[6] Ito M. Robot vision modelling—camera modelling and camera calibration. Adv Robot 1990;5(3):321–35. http://dx.doi.org/10.1163/156855391X00232, URL http://www.tandfonline.com/doi/abs/10.1163/156855391X00232.

[7] Pollefeys M, Koch R, Van Gool L. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. Int J Comput Vis 1999;32(1):7–25. http://dx.doi.org/10.1023/A:1008109111715, URL http://link.springer.com/10.1023/A:1008109111715.

[8] Van Rossum G, Drake Jr FL. Python tutorial. The Netherlands: Centrum voor Wiskunde en Informatica Amsterdam; 1995.

[9] Van Der Walt S, Colbert SC, Varoquaux G. The NumPy array: A structure for efficient numerical computation. Comput. Sci. Eng. 2011;13(2):22–30. http://dx.doi.org/10.1109/MCSE.2011.37, arXiv:1102.1523, URL http://ieeexplore.ieee.org/document/5725236/.

[10] Oliphant TE. Python for scientific computing. Comput Sci Eng 2007;9(3):10–20. http://dx.doi.org/10.1109/MCSE.2007.58, URL https://ieeexplore.ieee.org/abstract/document/4160250/.

[11] Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng 2007;9(3):90–5. http://dx.doi.org/10.1109/MCSE.2007.55.

[12] Gerum R. Pylustrator: code generation for reproducible figures for publication. arXiv e-prints 2019. arXiv:1910.00279.

[13] McKinney W. Data Structures for Statistical Computing in Python, in: Proc. 9th python sci. conf. (SCIPY 2010), 2010, p. 51–6, http://dx.doi.org/10.1016/S0168-0102(02)00204-3, URL https://pdfs.semanticscholar.org/f6da/c1c52d3b07c993fe52513b8964f86e8fe381.pdf.

[14] Hartley R, Zisserman A. Multiple view geometry in computer vision. Cambridge: Cambridge University Press; 2003, p. 657.

[15] Richter S, Gerum RC, Winterl A, Houstin A, Seifert MA, Peschel J, Fabry B, Le Bohec C, Zitterbart DP. Phase transitions in huddling emperor penguins. J Phys D Appl Phys 2018;51(21):214002. http://dx.doi.org/10.1088/1361-6463/aabb8e, URL http://stacks.iop.org/0022-3727/51/i=21/a=214002?key=crossref.b52459c985c989270a3ae1d74f2e8477http://iopscience.iop.org/article/10.1088/1361-6463/aabb8e.

[16] Hastings W. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 1970;51(1):97–109, URL https://academic.oup.com/biomet/article-abstract/57/1/97/284580.

[17] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. J Chem Phys 1953;21(6):1087–92. http://dx.doi.org/10.1063/1.1699114, arXiv:5744249209, URL http://aip.scitation.org/doi/10.1063/1.1699114.

[18] Gerum RC, Richter S, Fabry B, Zitterbart DP. ClickPoints: an expandable toolbox for scientific image annotation and analysis. Methods Ecol Evol 2016;8(6):750–6. http://dx.doi.org/10.1111/2041-210X.12702.

[19] Longuet-Higgins HC. A computer algorithm for reconstructing a scene from two projections. Nature 1981;293(5828):133–5. http://dx.doi.org/10.1038/293133a0, URL http://www.nature.com/articles/293133a0.

[20] Gerum RC, Richter S, Fabry B, Le Bohec C, Bonadonna F, Nesterova A, Zitterbart DP. Structural organisation and dynamics in king penguin colonies. J Phys D Appl Phys 2018;51(16):164004. http://dx.doi.org/10.1088/1361-6463/AAB46B, URL http://iopscience.iop.org/article/10.1088/1361-6463/aab46b.

CameraTransform: a Python Package for Perspective Corrections and Image Mapping **Supplementary Material**

Richard C. Gerum[a], Sebastian Richter[a,b], Alexander Winterl[a,b], Christoph Mark[a], Ben Fabry[a], Céline Le Bohec[c,d], Daniel P. Zitterbart[a,b]

[a] *Biophysics Group, Department of Physics, University of Erlangen-Nürnberg, Germany*
[b] *Applied Ocean Physics and Engineering, Woods Hole Oceanographic Institution, Woods Hole, MA, USA*
[c] *Centre Scientifique de Monaco, Département de Biologie Polaire, Monaco, Principality of Monaco*
[d] *Université de Strasbourg, CNRS, IPHC, UMR 7178, Strasbourg, France*

Keywords: Perspective projection, quantitative image analysis, geo-referencing, camera lens distortions

---

Appendix A. Camera Matrix

All information about the mapping of real-world points to image points are stored in a camera matrix. The camera matrix is expressed in projective coordinates, and can be split into two parts - the intrinsic matrix and the extrinsic matrix [1]. The intrinsic matrix depends on the camera sensor and lens, the extrinsic matrix depends on the camera's position and orientation.

Appendix A.1. Projective coordinates

Projective coordinates, also known as homogeneous coordinates, are used to represent projective transformations as matrix multiplications [2] whereby the vector representation of a point is extended by an additional entry. This entry defaults to 1, and all scalar multiples of a vector are considered equal:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \hat{=} \begin{pmatrix} s \cdot x \\ s \cdot y \\ s \end{pmatrix} \tag{A.1}$$

For example, the point (5,7) can be represented by the tuple of projective coordinates (5,7,1) or (10,14,2) and so on. The scalar s need not be an

integer. Projective coordinates allow us to write the camera projection $\vec{y}$ as:

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \tag{A.2}$$

where $\vec{x}$ specifies the point in the 3D world, which is multiplied with the camera matrix C to obtain the point in the camera image $\vec{y}$.

Appendix A.2. Intrinsic parameters

The intrinsic parameters are given by Equation (1) & (2) in the main text.

Appendix A.3. Extrinsic parameters

To compute the extrinsic camera matrix, we first need the three rotation matrices and the translation matrix:

$$R_{\text{tilt}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_{\text{tilt}}) & \sin(\alpha_{\text{tilt}}) \\ 0 & -\sin(\alpha_{\text{tilt}}) & \cos(\alpha_{\text{tilt}}) \end{pmatrix} \tag{A.3}$$

$$R_{\text{roll}} = \begin{pmatrix} \cos(\alpha_{\text{roll}}) & \sin(\alpha_{\text{roll}}) & 0 \\ -\sin(\alpha_{\text{roll}}) & \cos(\alpha_{\text{roll}}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

$$R_{\text{heading}} = \begin{pmatrix} \cos(\alpha_{\text{heading}}) & \sin(\alpha_{\text{heading}}) & 0 \\ -\sin(\alpha_{\text{heading}}) & \cos(\alpha_{\text{heading}}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.5}$$

$$t = \begin{pmatrix} x \\ y \\ -\text{height} \end{pmatrix} \tag{A.6}$$

$$\tag{A.7}$$

The extrinsic camera matrix then consists of the 3x3 rotation matrix R and the 3x1 translation matrix t side by side, as a 4x4 matrix in projective coordinates.

$$R = R_{\text{roll}} \cdot R_{\text{tilt}} \cdot R_{\text{heading}} \tag{A.8}$$

$$T = R_{\text{tilt}} \cdot R_{\text{heading}} \cdot t \tag{A.9}$$

$$C_{\text{extr.}} = \left(\begin{array}{c|c} R & T \\ \hline 0 & 1 \end{array}\right) \tag{A.10}$$

The final camera matrix C is the product of the intrinsic and the extrinsic camera matrix.

$$C = C_{\text{intr.}} \cdot C_{\text{extr.}} \tag{A.11}$$

Appendix A.4. Projecting from the World to the Camera image

To map a real-world point to a pixel of the acquired image, we first write the real-world point $\vec{p}_{\text{world}}(x_1, x_2, x_3)$ in projective coordinates:

$$\tilde{p}_{\text{world}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \tag{A.12}$$

The image point $\tilde{p}_{\text{im}}$ can then be computed according to:

$$\tilde{p}_{\text{im}} = C \cdot \tilde{p}_{\text{world}} \tag{A.13}$$

Finally, the point $\tilde{p}_{\text{im}}$ is converted from projective coordinates (which has 3 entries) to "conventional" coordinates $\vec{p}_{\text{im}}$ (with two entries) by division with the additional scaling factor s (which is the 3rd entry of $\tilde{p}_{\text{im}}$):

$$\vec{p}_{\text{im}} = \begin{pmatrix} \tilde{p}_{\text{im}_1} / \tilde{p}_{\text{im}_3} \\ \tilde{p}_{\text{im}_2} / \tilde{p}_{\text{im}_3} \end{pmatrix} \tag{A.14}$$

where the subscript denotes the entry of the vector $\tilde{p}_{\text{im}}$.

Appendix A.5. Projecting from the camera image to real-world coordinates

While projecting from the 3D real-world to the 2D image is a straight forward matrix multiplication, projecting from the image back to the real-world is more difficult. As the information of the 3rd dimension is lost during the transformation from the real-world to the image, there exists no unique back-transformation. An additional constraint is needed to transform a point back to the 3D world, e.g. one of the 3D coordinates must be fixed. For example, if the real-world point $\vec{p}_{\text{world}}$ has a known x_3 coordinate (e.g. the height above ground is known), and the image coordinates y_1 and y_2 are given, the back-transformation can be performed as follows:

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{pmatrix} \cdot \begin{pmatrix} s \cdot x_1 \\ s \cdot x_2 \\ s \cdot x_3 \\ s \end{pmatrix} \tag{A.15}$$

$$= \begin{pmatrix} c_{11} & c_{12} & c_{13} \cdot x_3 & c_{14} \\ c_{21} & c_{22} & c_{23} \cdot x_3 & c_{24} \\ c_{31} & c_{32} & c_{33} \cdot x_3 & c_{34} \end{pmatrix} \cdot \begin{pmatrix} s \cdot x_1 \\ s \cdot x_2 \\ s \\ s \end{pmatrix} \tag{A.16}$$

$$= \begin{pmatrix} c_{11} & c_{12} & c_{13} \cdot x_3 + c_{14} \\ c_{21} & c_{22} & c_{23} \cdot x_3 + c_{24} \\ c_{31} & c_{32} & c_{33} \cdot x_3 + c_{34} \end{pmatrix} \cdot \begin{pmatrix} s \cdot x_1 \\ s \cdot x_2 \\ s \end{pmatrix} \tag{A.17}$$

$$= \tilde{C} \begin{pmatrix} s \cdot x_1 \\ s \cdot x_2 \\ s \end{pmatrix} \tag{A.18}$$

$$\tilde{C}^{-1} \cdot \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} s \cdot x_1 \\ s \cdot x_2 \\ s \end{pmatrix} \tag{A.19}$$

This means that the information about the fixed 3D coordinate has to be incorporated in the camera matrix. The inverse of the resulting matrix, when multiplied with the image point in projective coordinates, gives the unknown x_1 and x_2 entries of the real-world 3D point. After re-scaling the vector entries (division by s), the known x_3 value is added to the vector to retrieve the real-world coordinates of the 3D point $\vec{p}_{\text{world}}$.

For transformations that cannot be expressed as a matrix, the ray of the given pixel is obtained, then transformed with the extrinsic matrix, and finally the intersection of this ray with e.g. a coordinate plane is calculated.

Appendix A.5.1. Rectilinear Projection

This rectilinear projection is the standard "pin-hole" camera model, which is the most common projection for single images. The 3D point (x, y, z) is projected on a plane $(x_{\text{im}}, y_{\text{im}})$. The transformations here are given for a camera pointing in z direction. For an arbitrary orientation of the camera, the 3D point has to be first transformed to account for the camera's position and orientation.

$$x_{\text{im}} = f_x \cdot \frac{x}{z} + x_{\text{offset}} \tag{A.20}$$

$$y_{\text{im}} = f_y \cdot \frac{y}{z} + y_{\text{offset}} \tag{A.21}$$

Where f_x and f_y are the focal lengths in pixel for the x and y direction (for an ideal camera with square pixels, f_x and f_y are equal). The offsets account for the origin (0,0) of the image usually being the top left corner and not the center of the image.

Because information is lost when projecting from 3D to 2D, there is no unique backtransformation. For every 2D point, however, a unique line in 3D space can be specified on which the 3D point lies. This so called "ray" is defined as follows:

$$\vec{r} = \begin{pmatrix} (x_{\text{im}} - x_{\text{offset}})/f_x \\ (y_{\text{im}} - y_{\text{offset}})/f_y \\ 1 \end{pmatrix} \tag{A.22}$$

Appendix A.5.2. Cylindrical Projection

The cylindrical projection is a common projection used for panoranic images. The 3D point is projected on the 2D surface of a cylinder. This projection is often used for wide panoramic images, as it can cover the full 360°range in the x-direction. The poles, however, cannot be represented in this projection.

$$x_{\text{im}} = f_x \cdot \arctan\left(\frac{x}{z}\right) + x_{\text{offset}} \tag{A.23}$$

$$y_{\text{im}} = f_y \cdot \frac{y}{\sqrt{x^2 + z^2}} + y_{\text{offset}} \tag{A.24}$$

The image rays are defined as follows:

$$\vec{r} = \begin{pmatrix} \sin\left(\frac{x_{\text{im}} - x_{\text{offset}}}{f_x}\right) \\ \frac{y_{\text{im}} - y_{\text{offset}}}{f_y} \\ \cos\left(\frac{x_{\text{im}} - x_{\text{offset}}}{f_x}\right) \end{pmatrix} \tag{A.25}$$

Appendix A.5.3. Equirectangular Projection

The equirectangular projection is a common projection used for panoranic images. The 3D point is projected on the 2D surface of a sphere. The projection can cover the full range of angles in both x and y direction.

$$x_{\text{im}} = f_x \cdot \arctan\left(\frac{x}{z}\right) + x_{\text{offset}} \tag{A.26}$$

$$y_{\text{im}} = f_y \cdot \arctan\left(\frac{y}{\sqrt{x^2 + z^2}}\right) + y_{\text{offset}} \tag{A.27}$$

The image rays are defined as follows:

$$\vec{r} = \begin{pmatrix} \sin\left(\frac{x_{\text{im}} - x_{\text{offset}}}{f_x}\right) \\ \tan\left(\frac{y_{\text{im}} - y_{\text{offset}}}{f_y}\right) \\ \cos\left(\frac{x_{\text{im}} - x_{\text{offset}}}{f_x}\right) \end{pmatrix} \tag{A.28}$$

Appendix B. Sensitivity Analysis

Appendix B.1. Extrinsic Camera parameters

To evaluate the sensitivity of the perspective projection with respect to uncertainties in the camera parameters, we computationally place objects of 1 m height in world coordinates at different distances from the camera (50 – 300 m) and project them to the camera image. The positions in the camera image are then projected back to real-world coordinates using a different parameter set where we vary the camera elevation and tilt angle. We use a *Panasonic DMC-G5* camera with a focal length of 14 mm and a sensor size of 17.3×9.7 mm with 4608×2592 px. The camera is placed at an elevation of 20 m with a tilt angle of 80°. For the back projection, the elevation and tilt are varied by ±10% (Fig. B.1) and for each parameter configuration the apparent object height is calculated. Since we know the true object height, the reconstructed object height indicates the error that is introduced by the uncertainties in the extrinsic camera parameters. We find that the apparent object height is only weakly dependent on variations in camera elevation regardless of the distance between object and camera (Fig. B.1b). By contrast, the apparent object height is sensitive to variations in the camera's tilt angle, especially for objects with a larger distance to the camera (Fig. B.1c).

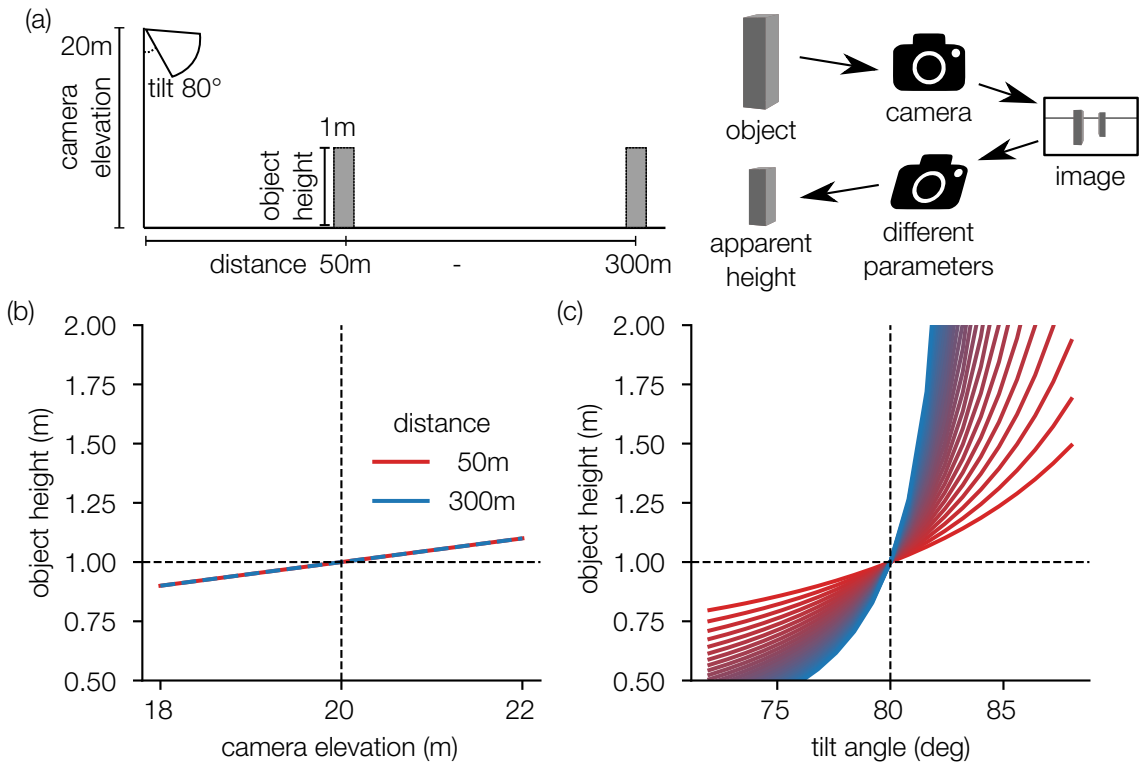

Figure B.1: **Influence of elevation and tilt angle variation of ± 10%.**
(a) Objects with a height of 1 m (dashed line, in (b),(c)) and different distances (50 m – 300 m) projected to the camera and back to the world with changed camera parameters. (b) Object height for variation of the elevation parameter (20 m ± 10%). (c) Object height for variation of the the tilt parameter (80°± 10%).

Appendix B.2. Object positions

Fitting the camera parameters from objects in the image not only depends on the number of objects used to estimate the camera parameters, but also on the position of the objects in the image. As an estimate of this dependence, we use an artificially create image with known camera parameters (focal length 3863.64 px, image 2592 x 4608 px, elevation 16.12 m, tilt 85.3°, roll 0.34°), where we place 50 objects at different distances from the camera (between 46 m and 151 m).

Using the objects in the foreground yielded better results (smaller uncertainty) for the elevation parameter than using the objects in the background of the image (see Fig. B.2). For the tilt and roll parameters, this behaviour is reversed. Using objects from both foreground and background for the fitting routine, by contrast, gave the best results for all parameters.

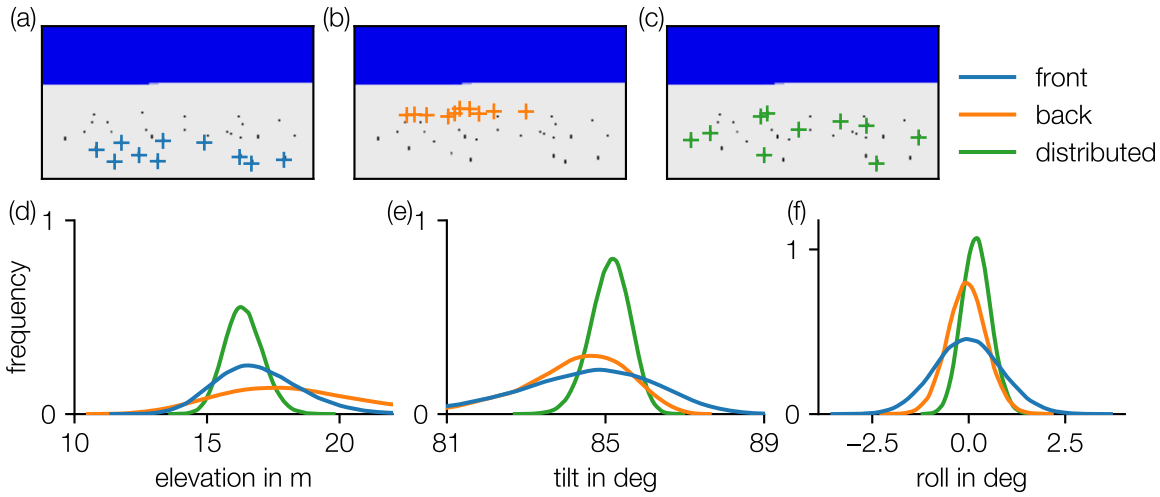

Figure B.2: **Influence of the position of the provided objects on the parameter uncertainty.**
On an artificial image (known camera parameters) with 50 objects where the foot and head positions are manually marked. For every analysis, 10 objects are used. The 10 nearest objects (blue, shown in (a)), the 10 farthest objects (orange, shown in (b)) and 10 randomly distributed objects (green, shown in (c)). (d)-(f) The uncertainty of the camera parameters (elevation, tilt, and roll) for the different conditions.

Appendix C. Fitting from Stereo Image

To test the reconstruction of the camera parameters from stereo images, two sample images (Fig. C.3g,h) of a table with various objects of known size are taken with a camera that is laterally moved between the images. A total of 16 point correspondences in the two images are then marked, which serve as the input for the metropolis sampling algorithm. During sampling, the baseline (distance between the two cameras) remains fixed to unity and is later re-scaled to fit the known sizes of the objects in the images, resulting in a fitted baseline of 87 cm. This re-scaling is essential as point correspondences alone cannot provide information on scaling. For the sampling, a total of 5 parameters (camera A: heading, roll; camera B: tilt, heading, roll) are sampled, while the other parameters (camera A: position (x=0, y=0, z=0), heading = 90°; camera B: position (x=1, y=0, z=0)) are fixed to define a unique reference frame.

After sampling, distances and object sizes in the image are measured by assigning corresponding points in the stereo images. The calculated sizes and distances are then verified with a ruler and are found to be accurate to within ± 1 mm.

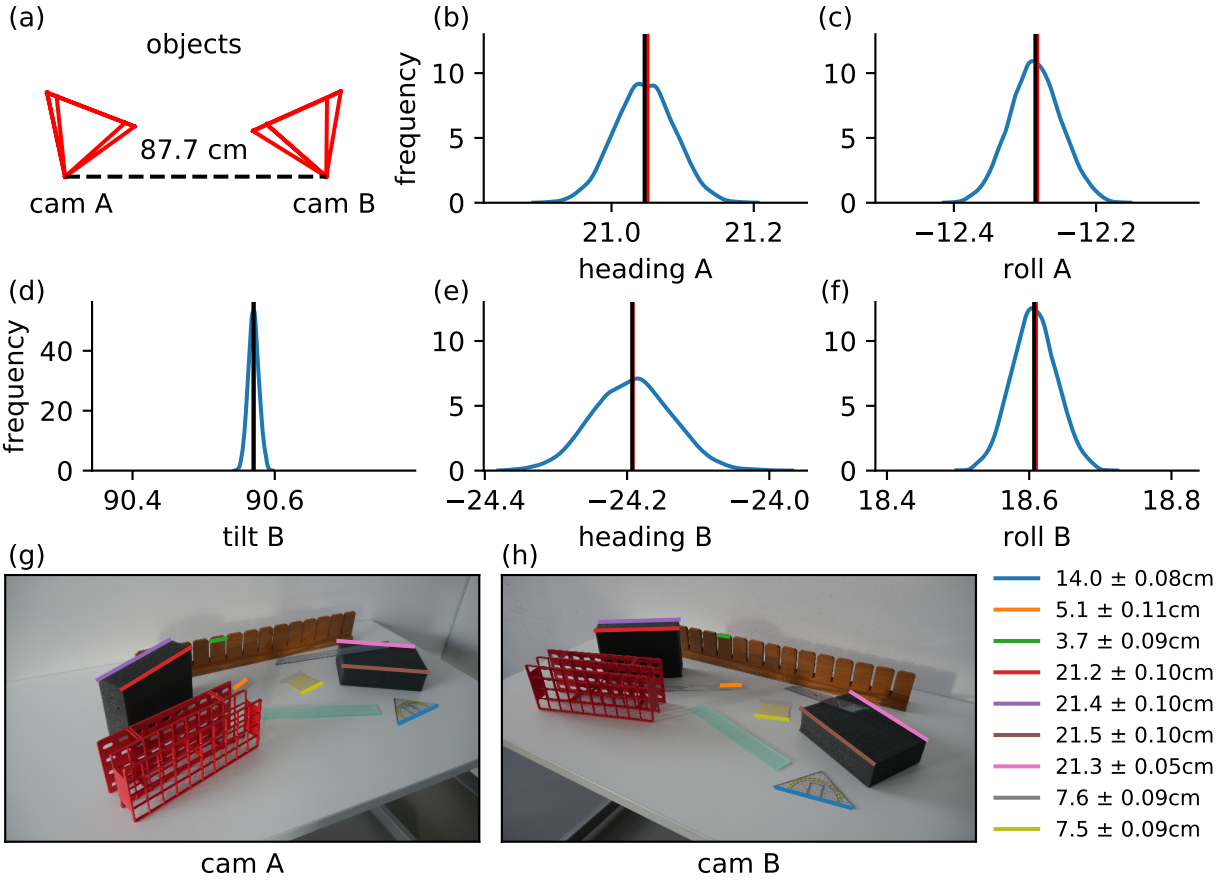

Figure C.3: **Fit of camera parameters by point-correspondences in a stereo setup.**
(a) Setup of the two cameras. (b,c,d,e,f) The fitted distributions of the parameter uncertainties for camera A (b,c) and camera B (d,e,f). (g) Image of the left camera and image of the right camera (h). Colored lines show distances that have been measured using point correspondences.

Appendix D. Lens distortions

The accuracy of the image transformation package can be further improved by considering image distortions caused by imperfect camera lenses. *CameraTransform* is currently able to deal with radial lens distortions but not skew and tangential distortions that are usually less severe. *CameraTransform* directly removes lens distortions when projecting from image coordinates to world coordinates without the need to first compute an undistorted image, which would introduce rounding and pixelation errors. Conversely, it is also possible to apply lens distortions when back-projecting from world coordinates to image coordinates.

CameraTransform implements two lens distortion models, the commonly used Brown model and the ABC model. The Brown model moves image points along the line formed by the image point and the optical axis point radially according to a polynomial function of the radial distance with even polynomial powers. This distortion model is also used e.g. by OpenCV or AgiSoft PhotoScan. The ABC model uses a 4th order polynomial for the radial shift and is used e.g. in PTGui.

References

[1] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge University Press, Cambridge, 2003 (2003).

[2] A. F. Möbius, Der barycentrische Calcul, ein Hülfsmittel zur analytischen Behandlung der Geometrie, Barth, Leipzig, 1827 (1827).