Solar Energy 300 (2025) 113803

Contents lists available at ScienceDirect

Solar Energy

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

A simulation environment for UAV-based real-time condition monitoring of solar tower power plants

Alexander Schnerring [a,c,*], Rafal Broda [a,c], Adrian Winter [a,1], Michael Nieslony [a], Julian J. Krauth [a], Marc Röger [a], Sonja Kallio [a], Robert Pitz-Paal [b,c]

[a] German Aerospace Center (DLR), Institute of Solar Research, Calle Doctor Carracido 44, 04005, Almería, Spain
[b] German Aerospace Center (DLR), Institute of Solar Research, Linder Höhe, 51147, Cologne, Germany
[c] RWTH Aachen University, Chair of Solar Technology, Linder Höhe, 51147, Cologne, Germany

ARTICLE INFO

Keywords:
Unmanned aerial vehicle
Simulation
Condition monitoring
Calibration
Concentrated solar power
Heliostat

ABSTRACT

The development and testing of unmanned aerial vehicle (UAV)-based condition monitoring systems is time consuming, costly and poses safety risks. While numerous examples show that simulation environments are well suited to support the development process, existing environments fall short of simulating quantities specific to the condition monitoring of solar tower power plants. To bridge this gap, we present a simulation environment that provides quantities necessary to investigate such systems in simulation, prior to their application in real solar tower power plants. The presented environment models the state of the solar field and computes observations of the field and reflections of a point light source, as seen from a virtual camera. In addition, it allows for the navigation of a simulated UAV in the virtual solar field in response to realistic UAV control signals. The simulated concentrator corner points were found to match the concentrator corner points determined by a bundle adjustment measurement up to an RMSE = 23.7 mm before and RMSE = 4.6 mm after accounting for translational, rotational and scale errors. The simulated reflections of a point light source were found to match the measured reflections up to an RMSE of 2.25 mrad in X-direction and 2.09 mrad in Y-direction in the concentrator coordinate system. After eliminating errors in the camera position estimate, concentrator orientations and mirror surface slope errors, the remaining RMSE is 0.35 mrad in X-direction and 0.22 mrad in Y-direction. We conclude that the proposed simulation environment is a valuable tool for the development of UAV-based condition monitoring systems of solar tower power plants.

1. Introduction

The efficiency of concentrating solar power (CSP) tower plants is, among other factors, negatively affected by optical errors in the solar field such as heliostat tracking errors, canting errors and slope errors. A common approach to detect such errors is to use computer vision algorithms on images recorded with a camera mounted on an unmanned aerial vehicle (UAV) [1–4]. Currently, flight routes for these UAVs are planned prior to their flight, while measurement data is evaluated after the flight. The measurement data quality is often only recognized to be insufficient when the UAV has already landed. Performing the image data analysis in real-time, i.e. while the UAV is still flying, opens up new possibilities for automated monitoring, as the UAV flight route can be planned dynamically, i.e. based on real-time data analysis results. However, the development of UAV-based systems comes with several challenges: Field tests are time-intensive, dependent on unplannable or unpredictable environmental conditions and always pose the risk of material damage. These problems can be mitigated using a simulation environment, in which UAV-based systems can be developed and tested safely, at low cost, and efficiently prior to their application in the real world.

UAV-based CSP condition monitoring systems are typically comprised of several components, such as a perception module and a geometric computation pipeline [1,4]. While various UAV simulators have been proposed [5], none of them offers the simulation of quantities required for the development of CSP condition monitoring systems, such as characteristic features in the solar field or reflections as seen

* Corresponding author at: German Aerospace Center (DLR), Institute of Solar Research, Calle Doctor Carracido 44, 04005, Almería, Spain.
E-mail address: alexander.schnerring@dlr.de (A. Schnerring).
[1] Now with Hamburg University of Applied Sciences, Department Aeronautical Engineering, Stiftstraße 69, 20999, Hamburg, Germany.

https://doi.org/10.1016/j.solener.2025.113803
Received 31 January 2025; Received in revised form 30 April 2025; Accepted 15 July 2025
Available online 18 August 2025

Acronyms

CCS	concentrator coordinate system
CSP	concentrating solar power
DLR	German Aerospace Center
EOR	exterior orientation
FCS	facet coordinate system
FOMS	fitted orientations, measured surfaces
GCS	global coordinate system
GPS	global positioning system
GSD	ground sampling distance
ICS	image coordinate system
IOR	interior orientation
LED	light-emitting diode
MAE	mean absolute error
OCS	observer coordinate system
RMSE	root mean squared error
ROIS	raw orientations, ideal surfaces
SITL	software-in-the-loop
SSE	steady state srror
STJ	Solar Tower Jülich
UAV	unmanned aerial vehicle

from a UAV camera. Conversely, simulation tools in the CSP community [6,7] focus on modeling optical properties of the solar field but do not provide the quantities required for the development of UAV-based systems. This work aims to bridge this gap by presenting and validating a novel simulation environment, designed to facilitate the development and testing of UAV-based CSP condition monitoring systems. It is important to note that the purpose of this work is to present the simulation infrastructure itself, rather than a specific system. Throughout this paper, the term *System Under Development* will be used in an abstract sense to refer generically to any system being developed and tested using the proposed simulation environment.

The presented simulation environment extends AirSim [8], an open-source UAV simulator that allows for real-time flight dynamics simulation in response to standard flight control commands. Built on Unreal Engine [9], AirSim enables realistic rendering of virtual environments, allowing a *System Under Development's* perception module to be integrated into the simulation loop. The developed extensions add a solar field model and an image feature computation model, enabling the simulation of characteristic image features, as seen by a UAV camera. The modular structure allows for replacing perception outputs with the computed image feature data and systematically controlling noise levels during development and testing.

Following this introduction, Section 2 describes the conventions and notation used throughout the paper and outlines the mathematical preliminaries. Section 3 provides an overview of the simulation system and describes the functionality of each components in detail. The validation methodology for key quantities of the simulation environment is presented in Section 4. Section 5 presents the validation results and discusses their implications for the transferability of systems developed in simulation to real-world applications. Finally, Section 6 concludes the paper and outlines directions for future research.

2. Preliminaries

2.1. Conventions and notation

The solar field is assumed to consist of N heliostats, where each heliostat consists of two tracking axes and a concentrator with M facets. The concentrator coordinate system (CCS) of a heliostat n is denoted by CCS_n and the facet coordinate system (FCS) of a facet m of concentrator n is denoted by $\mathrm{FCS}_{n,m}$. The state of the solar field is characterized by the pose of every $\mathrm{FCS}_{n,m}$. These coordinate systems are defined w.r.t. the global coordinate system (GCS), chosen to be a local east-north-up coordinate system on the northern hemisphere in accordance with the *SolarPACES Guideline for Heliostat Performance Testing* [10]. Likewise, the pose of a camera moving through the solar field is described by its own coordinate system, which changes over time as the camera navigates the environment. As the acronym CCS is already in use, the camera coordinate system is denoted by observer coordinate system (OCS). The 2D coordinate system associated to the camera sensor is denoted by image coordinate system (ICS). Fig. 1 illustrates the coordinate systems used throughout this work.

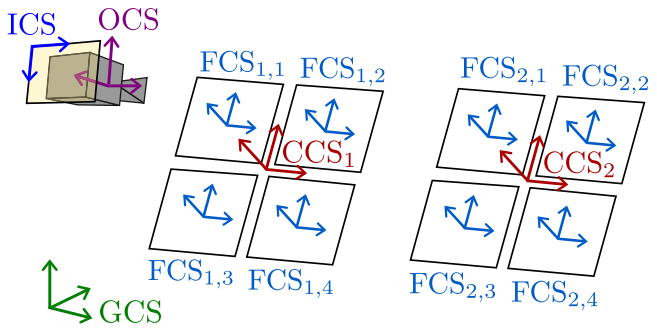


Fig. 1. Overview of the coordinate systems for the case $N = 2$ and $M = 4$. The yellow rectangle represents the camera sensor plane.

Scalar values are represented as lowercase letters with regular font, vectors as lowercase letter with bold font, and matrices in uppercase letters with bold font.

2.2. Coordinate transformations

The spatial relationship of the aforementioned coordinate systems is described within the framework of coordinate transformations. The transformation between two coordinate systems CS1 and CS2 is conveniently summarized in a 4×4 matrix of the form

$$\boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_{\mathrm{CS2}}^{\mathrm{CS1}} & \boldsymbol{t}_{\mathrm{CS2}}^{\mathrm{CS1}} \\ \boldsymbol{0} & 1 \end{pmatrix} \tag{1}$$

where $\boldsymbol{t}_{\mathrm{CS2}}^{\mathrm{CS1}}$ denotes the translation from CS1 to CS2 and $\boldsymbol{R}_{\mathrm{CS2}}^{\mathrm{CS1}}$ denotes the rotation from CS1 to CS2.

2.3. Transforming points and point clouds

Characteristic features in the solar field, e.g. facet corner points or a point light source, are represented by points and point clouds. A point $\boldsymbol{p}^{\mathrm{CS2}} = (p_{\mathrm{X}}^{\mathrm{CS2}}, p_{\mathrm{Y}}^{\mathrm{CS2}}, p_{\mathrm{Z}}^{\mathrm{CS2}})^T$ expressed w.r.t. CS2 can be expressed w.r.t. CS1 in the following way:

$$\begin{pmatrix} p^{\mathrm{CS1}} \\ 1 \end{pmatrix} = \boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}} \cdot \begin{pmatrix} p^{\mathrm{CS2}} \\ 1 \end{pmatrix} . \tag{2}$$

To simplify notation, appending the entry 1 to the multiplied vector and only considering the first three entries of the resulting vector is implicitly assumed whenever a point is multiplied by a transformation matrix, i.e.

$$\boldsymbol{p}^{\mathrm{CS1}} = \boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}} \cdot \boldsymbol{p}^{\mathrm{CS2}} . \tag{3}$$

Conversely to Eqs. (2) and (3), a point expressed w.r.t. CS1 can be expressed w.r.t. CS2 by multiplying with the inverse of $\boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}}$:

$$\boldsymbol{p}^{\mathrm{CS2}} = \left(\boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}}\right)^{-1} \cdot \boldsymbol{p}^{\mathrm{CS1}} = \boldsymbol{T}_{\mathrm{CS1}}^{\mathrm{CS2}} \cdot \boldsymbol{p}^{\mathrm{CS1}} . \tag{4}$$

A point cloud $\boldsymbol{P}$ is a collection of N points and can be expressed in form of a matrix:

$$\boldsymbol{P}^{\mathrm{CS2}} = \begin{pmatrix} p_{\mathrm{X1}}^{\mathrm{CS2}} & p_{\mathrm{Y1}}^{\mathrm{CS2}} & p_{\mathrm{Z1}}^{\mathrm{CS2}} \\ \vdots & \vdots & \vdots \\ p_{\mathrm{XN}}^{\mathrm{CS2}} & p_{\mathrm{YN}}^{\mathrm{CS2}} & p_{\mathrm{ZN}}^{\mathrm{CS2}} \end{pmatrix}^{T} . \tag{5}$$

Analogously to Eqs. (2) and (3), a point cloud can be expressed w.r.t. another coordinate system, i.e. $\boldsymbol{P}^{\mathrm{CS1}} = \boldsymbol{T}_{\mathrm{CS2}}^{\mathrm{CS1}} \cdot \boldsymbol{P}^{\mathrm{CS2}}$.

3. Simulation methodology

3.1. Simulation system overview

The simulation environment is designed to enable the development and testing of various *Systems Under Development*. While this paper does not aim to provide a detailed description of any specific system, Section 3.6 briefly discusses possible configurations and characteristics of such systems. Based on the operational principles of typical UAV-based CSP condition monitoring systems, the following models have been identified as necessary:

1. A model representing where the facet corner points of each heliostat are located w.r.t. to both GCS and ICS. This information is used in systems utilizing the projected facet corner points, e.g. camera pose estimation [1] or coarse calibration algorithms [2].
2. A reflection model, enabling to compute how objects reflected by the observed heliostats map to the ICS of an observing camera. This information is used in systems that estimate the solar field state from reflections visible in the observed heliostats [1,4].
3. A model to render image data, allowing to include the *Systems Under Development's* perception module in the simulation loop. Through this, characteristics specific to the perception module (such as noise processes) can be considered in simulation.

In addition to the aforementioned models, the development of real-time systems introduces the following requirements:

4. A model to compute the UAV position and velocity (and hence the camera pose) as a function of UAV control signals.
5. All of the above models should be implemented such that they can be run in real-time.

Fig. 2 illustrates the modular architecture of the simulation environment (highlighted in gray) in which the *System Under Development* (highlighted in yellow) can be developed and tested.

The *Solar Field State* (Section 3.2) serves as a virtual representation of the solar field. It is parametrized by field data such as heliostat positions, concentrator geometry and surface measurements and describes the motion of each $\mathrm{FCS}_{n,m}$ w.r.t. the GCS using a *Kinematic Model* and a *Surface Model*.

The *Ground Truth Image Feature Computation* (Section 3.4) simulates which image features a virtual camera observes as a function of the *Solar Field State* and the *Camera State* (Section 3.3). These image features can represent both facet corner points and object reflections. Since the current development of condition monitoring systems by German Aerospace Center (DLR) is based on point light reflections, the scope of this paper is limited to the simulation of point lights. However, the presented framework can easily be extended to the reflection of other objects, such as tower edges, astronomical objects, etc.

Alternatively, the *UAV & Graphic Simulation* (Section 3.5) can be used to render entire images, which can be passed to the *System Under Development's* perception module in order to extract relevant image features. In addition, the *UAV & Graphic Simulation* computes the subsequent UAV pose as a function of UAV control signals, which can be output by the *System Under Development's* UAV control logic.

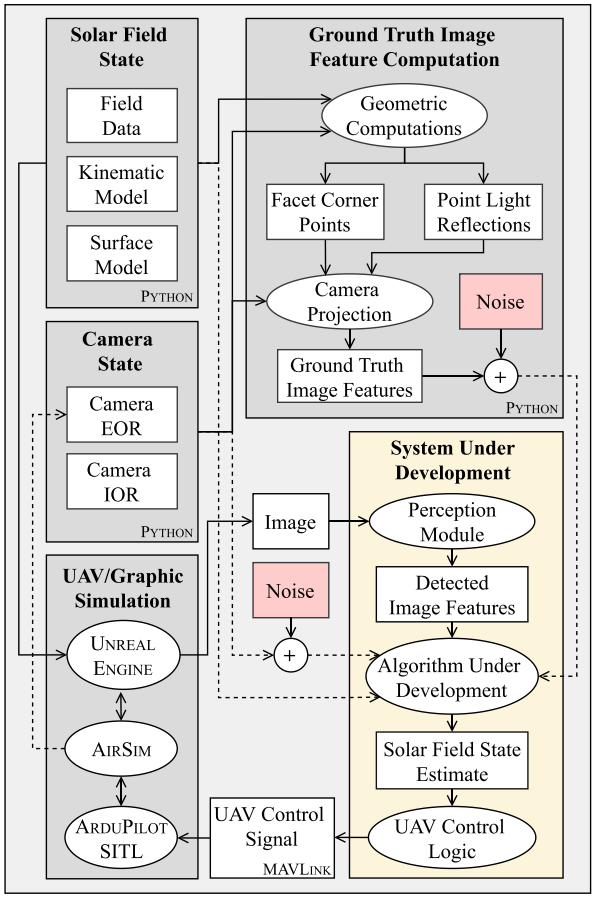


Fig. 2. The presented simulation environment consists of four components (gray colored boxes), communicating with each other and the *System Under Development* (yellow colored box). Optional paths are shown as dashed lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2. Solar field state

As mentioned in Section 3.1, relevant quantities for solar tower power plant condition monitoring systems include facet corner points and point light reflections. Prior to their projection onto the ICS, these points first have to be expressed w.r.t. the GCS. This computation is supported by the *Solar Field State*, a geometric model composed of the *Kinematic Model* and the *Surface Model*. The *Solar Field State* describes the spatial relation between GCS, CCS_n and $\mathrm{FCS}_{n,m}$. This information, combined with the concentrator and facet geometry and the *Camera State*, is used to compute the projection of these points onto the ICS.

3.2.1. Kinematic model

The kinematic model $\boldsymbol{K}_n$ describes the motion of CCS_n as a function of two tracking angles, θ_n and τ_n, i.e.

$$\boldsymbol{K}_n = \boldsymbol{K}(\theta_n, \tau_n | \boldsymbol{o}_n, \boldsymbol{k}_n) = \boldsymbol{T}_{\mathrm{CCS}_n}^{\mathrm{GCS}} , \tag{6}$$

where $\boldsymbol{o}_n$ denotes the heliostat origin w.r.t. the GCS and the set of parameters $\boldsymbol{k}_n$ account for the heliostat geometry and potential mounting errors. In this work, $\boldsymbol{k}_n$ is obtained by fitting the kinematic model to a number of calibration points, collected using the camera-target method [11]. θ_n and τ_n are computed from a motor movement model, which takes motor positions provided by the field operator as an input and outputs the respective tracking angles [6].

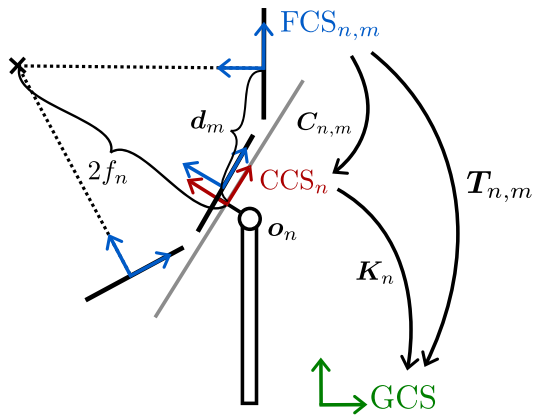


Fig. 3. The *Solar Field State* is described by a collection of coordinate transformations, obtained by chaining the *Kinematic Model* and the *Surface Model*.

3.2.2. Surface model

The surface of a concentrator consists of M facets, where each facet is canted, i.e. slightly adjusted w.r.t. the CCS$_n$ in such a way that incoming solar irradiation is concentrated at the focal length f_n. In this work, two cases for the *Surface Model* are considered: The first case (referred to as the ideal surface case) assumes each facet to be a perfectly flat mirror surface. The transformation $\boldsymbol{C}_{n,m}$ from facet coordinate system FCS$_{n,m}$ to CCS$_n$ is described by

$$\boldsymbol{C}_{n,m} = \boldsymbol{C}(f_n, \boldsymbol{d}_m) = \boldsymbol{T}^{\mathrm{CCS}_n}_{\mathrm{FCS}_{n,m}} , \tag{7}$$

where f_n denotes the concentrator focal length and $\boldsymbol{d}_m$ is the vector pointing from the origin of CCS$_n$ to the origin of FCS$_{n,m}$. The overall transformation $\boldsymbol{T}_{n,m}$ from facet m of heliostat n to the GCS is obtained by chaining the transformations:

$$\boldsymbol{T}_{n,m} = \boldsymbol{K}_n \cdot \boldsymbol{C}_{n,m} = \boldsymbol{T}^{\mathrm{GCS}}_{\mathrm{CCS}_n} \cdot \boldsymbol{T}^{\mathrm{CCS}_n}_{\mathrm{FCS}_{n,m}} = \boldsymbol{T}^{\mathrm{GCS}}_{\mathrm{FCS}_{n,m}} . \tag{8}$$

A simplified 2D illustration of a heliostat with two facets is shown in Fig. 3.

In the second case (referred to as the measured surface case), each facet is further subdivided into a collection of smaller tiles, where each tile is assumed to be a perfectly flat mirror surface. The orientation of each tile is adjusted w.r.t. the ideal surface according to a slope deviation map, e.g. obtained by the QDEC-H system [12]. More precisely, each tile is assigned the orientation of the facet it belongs to and corrected by the mean of all measured slope deviations inside this tile, effectively discretizing the slope deviation map. This way, both the ideal and the measured cases can be treated computationally identically, since they both describe a concentrator as a collection of flat surfaces. A subdivision of an exemplary concentrator surface into a grid of tiles of size 32×32 is shown in Appendix B.

The *Solar Field State* is implemented in Python, especially building on the packages NumPy [13] and SciPy [14] to achieve efficient vectorized computations for the entire solar field. The collection of facet coordinate transforms $\boldsymbol{T}_{n,m}$ is stored in a Python class. This class is then passed to the *Ground Truth Image Feature Computation*. Additionally, all $\boldsymbol{T}_{n,m}$ are passed to Unreal Engine in form of a JSON file, from which the virtual heliostat field is constructed.

3.3. Camera state

The *Camera State* is composed of the exterior orientation (EOR) and the interior orientation (IOR). The EOR describes the position and orientation of the simulated camera w.r.t. the GCS, i.e. EOR = $\boldsymbol{T}^{\mathrm{GCS}}_{\mathrm{OCS}}$, and describes how points defined w.r.t. the GCS are mapped to the OCS. The IOR describes how points in the OCS are mapped to points on the ICS. The overall projection of points expressed w.r.t. the GCS onto the ICS is denoted by

$$\boldsymbol{p}^{\mathrm{ICS}} = \mathrm{projection}(\boldsymbol{p}^{\mathrm{GCS}} | \mathrm{IOR}, \mathrm{EOR}) . \tag{9}$$

The *Camera State* is stored in a Python class and passed to the *Ground Truth Image Feature Computation*. The projection is implemented using the package opencv-python, a Python wrapper for the open-source computer vision software OpenCV [15]. A detailed description of the camera model is provided in Appendix A.

In case that the *System Under Development* controls the simulated UAV, the EOR can be sampled from the *UAV & Graphic Simulation* through the Python API for AirSim. If no feedback-based control of the UAV is required, the *UAV & Graphic Simulation* can also be excluded from the simulation. In this case, the *Camera State* can be fed with a collection of predefined camera poses. For example, a recorded flight can be reproduced in simulation by obtaining the camera poses from the recorded data, either from image data using computer vision techniques or from the UAV sensor data, e.g. recorded using a real-time kinematics and an inertial measurement unit.

3.4. Ground truth image feature computation

The *Solar Field State* provides the coordinate transformation collection $\boldsymbol{T}_{n,m}$, describing the position and orientation of every facet coordinate system FCS$_{n,m}$ w.r.t. the GCS. This information is used in the *Ground Truth Image Feature Computation* component together with the *Camera State* to compute the projection of both facet corner points and a point reflected at the facet mirror plane onto the ICS.

3.4.1. Facet corners

The facet geometry is represented by the point cloud $\boldsymbol{P}^{\mathrm{FCS}_{n,m}}_{\mathrm{Corner}}$, containing the corner points of each facet expressed w.r.t. FCS$_{n,m}$. The facet geometry is assumed to be identical for every facet in the field, i.e. $\boldsymbol{P}^{\mathrm{FCS}_{n,m}}_{\mathrm{Corner}} = \boldsymbol{P}^{\mathrm{FCS}}_{\mathrm{Corner}}$. The facet corners are expressed w.r.t. the GCS by applying $\boldsymbol{T}_{n,m}$ and then projected onto the ICS using the previously described *Camera State*:

$$\boldsymbol{P}^{\mathrm{ICS}}_{\mathrm{Corner}} = \mathrm{projection}(\boldsymbol{T}_{n,m} \cdot \boldsymbol{P}^{\mathrm{FCS}}_{\mathrm{Corner}} | \mathrm{IOR}, \mathrm{EOR}) . \tag{10}$$

Several filters are applied to the collection of all facets to ensure that only those facet corner points are projected that lie in front of the camera and within the camera field of view.

3.4.2. Point light reflections

Assuming a point light at position $\boldsymbol{p}^{\mathrm{GCS}}_{\mathrm{Source}}$ and a flat mirror surface defined by the XY-plane of FCS$_{n,m}$, the position of the reflection as seen from a camera can be computed by projecting the mirrored point onto the ICS:

$$\begin{aligned}
\boldsymbol{p}^{\mathrm{FCS}_{n,m}}_{\mathrm{Source}} &= \boldsymbol{T}^{-1}_{n,m} \cdot \boldsymbol{p}^{\mathrm{GCS}}_{\mathrm{Source}} \\
\boldsymbol{p}^{\mathrm{FCS}_{n,m}}_{\mathrm{Source,Mirrored}} &= (p^{\mathrm{FCS}_{n,m}}_{\mathrm{Source,X}},\ p^{\mathrm{FCS}_{n,m}}_{\mathrm{Source,Y}},\ -p^{\mathrm{FCS}_{n,m}}_{\mathrm{Source,Z}})^T \\
\boldsymbol{p}^{\mathrm{GCS}}_{\mathrm{Source,Mirrored}} &= \boldsymbol{T}_{n,m} \cdot \boldsymbol{p}^{\mathrm{FCS}_{n,m}}_{\mathrm{Source,Mirrored}} \\
\boldsymbol{p}^{\mathrm{ICS}}_{\mathrm{Reflection}} &= \mathrm{projection}(\boldsymbol{p}^{\mathrm{GCS}}_{\mathrm{Source,Mirrored}} | \mathrm{IOR}, \mathrm{EOR})
\end{aligned} \tag{11}$$

This work describes a *HelioPoint* measurement setup [4], where the light source is mounted next to the camera on the UAV. For simplicity, the point light position is assumed to lie in the camera focal point, i.e. $\boldsymbol{p}^{\mathrm{GCS}}_{\mathrm{Source}} = \boldsymbol{t}^{\mathrm{GCS}}_{\mathrm{OCS}}$. However, any point light position can be assumed for the computation in Eq. (11). The reflection $\boldsymbol{p}^{\mathrm{ICS}}_{\mathrm{Reflection}}$ for facet (n, m) will only be visible in the image if its projection lies inside the polygon defined by the projected facet corner points $\boldsymbol{P}^{\mathrm{ICS}}_{\mathrm{Corner}}$. This can be checked using a simple point-in-polygon test [16].

Fig. 4 illustrates the facet corner computation and the reflection computation. While the reflection computation is described based on

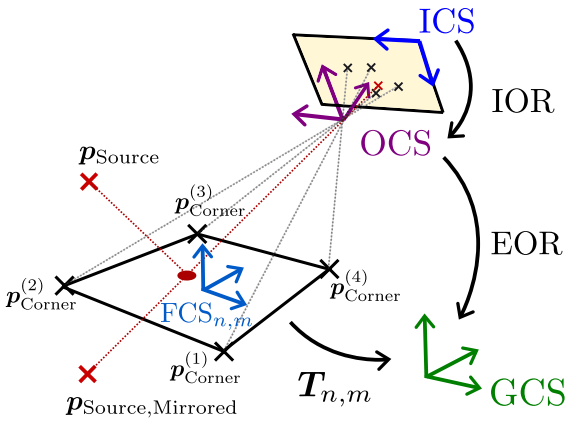


Fig. 4. The facet corner points and the reflection of p_{Source} as seen from the virtual camera are obtained by means of a camera projection of the facet corner points $P^{\text{GCS}}_{\text{Corner}}$ and the point $p^{\text{GCS}}_{\text{Source, Mirrored}}$.

the ideal surface case, the same computation applies to the measured surface case by treating each discretization tile as a facet.

The *Ground Truth Image Feature Computation* is implemented in Python in a vectorized form to process all facets at once, leveraging efficient computations in NumPy, SciPy and OpenCV. The projected facet corner points and point light reflections are passed to the *System Under Development* in form of NumPy arrays.

3.5. UAV & Graphic Simulation

3.5.1. AirSim

The *UAV & Graphic Simulation* is based on AirSim [8], an open-source UAV simulator developed by Microsoft. AirSim models the UAV flight dynamics using a vehicle model and a physics engine. It supports a range of virtual sensors, including the simulation of global positioning system (GPS) sensor noise. AirSim supports software-in-the-loop (SITL) integration with popular UAV firmware stacks such as ArduPilot [17], enabling the communication between simulated sensor inputs and real-world autopilot software over a User Datagram Protocol connection. Implemented primarily in C++, AirSim combines the performance required for real-time simulation with accessibility through a Python API. In the presented simulation environment, the flight control firmware ArduPilot is used.

AirSim integrates with Unreal Engine [9] to render high-fidelity 3D environments, supporting the development and testing of perception algorithms under photorealistic conditions. Strictly speaking, the presented simulation system is built on Colosseum [18], an extension of AirSim to support Unreal Engine 5. As Collosseum is largely built on AirSim, the simulator is still referred to as AirSim in this work. For more details on the functionality, refer to [8].

3.5.2. ArduPilot *SITL*

ArduPilot is an open-source autopilot firmware that supports various types of vehicles, including multirotor UAVs [17]. It provides essential functionality for autonomous flight, such as mission planning, waypoint navigation, basic fail-safe mechanisms and velocity control. ArduPilot fuses data from common onboard sensors, such as GPS, inertial measurement units, barometer and magnetometer, and supports communication over the MAVLink protocol [19]. ArduPilot is designed to operate both with physical hardware and in simulation, using the same communication interfaces. This enables the seamless transfer from an algorithm developed in simulation to a real system.

3.5.3. Unreal engine

Unreal Engine is a real-time 3D rendering engine primarily used for game development, but also adopted in simulation and robotics for its ability to generate high-fidelity, photorealistic environments. In the presented setup, the collection of coordinate transforms $T_{n,m}$ is loaded into Unreal Engine. Flat, reflective mirror surfaces of specified facet dimensions are instantiated accordingly. AirSim adds a UAV actor to the scene, with an attached camera actor to simulate onboard vision. Unreal Engine provides directional light actors in order to simulate the incoming sunlight as a function of geolocation, date and time, enabling the simulation of sun reflections. At runtime, Unreal Engine receives the UAV pose from AirSim and updates the position and orientation of the UAV actor and camera actor accordingly. The rendered images can be accessed through the AirSim Python API.

3.6. System under development

To clarify the intended use of the simulation environment, this subsection outlines characteristics and components of a possible *System Under Development*. The descriptions are intended to illustrate potential designs of a system, without presenting their specific implementation.

The *System Under Development* is assumed to be implemented in Python, though other implementations (e.g. C++) are possible if performance or integration requirements demand it. Depending on the usecase, the *System Under Development* receives the facet corner image coordinates and point light reflection image coordinates as NumPy arrays from the *Ground Truth Image Feature Computation*, or the images rendered by Unreal Engine in JPG or PNG format. In addition, the *System Under Development* may require an estimate of the camera EOR. While AirSim provides a model for GPS noise, it does not include a noise model for gimbal inaccuracies. However, the ground truth camera EOR can be directly sampled from AirSim and any noise process manually applied as needed. This enables the simulation of gimbal inaccuracies, as well as more general camera EOR uncertainty in systems that estimate the camera EOR from sources other than GPS and gimbal sensors, such as vision-based methods. As a result, the system design does not need to rely on AirSim's GPS model, supporting more flexible development scenarios. Similarly, the precision of facet corner point positions and point light reflection positions can be controlled in both GCS and ICS to simulate realistic levels of uncertainty in field data a-priori knowledge and perception inaccuracies. Based on this input, the *System Under Development* estimates the *Solar Field State* and derives UAV control actions. These actions are then encoded in MAVLink messages. If the *System Under Development* is implemented in Python, these messages can be generated using Pymavlink, a Python wrapper for the MAVLink protocol. This setup allows the system to operate consistently across both simulated and real-world environments.

Principally, any system utilizing the previously described quantities can be developed and tested within the presented simulation environment. The quantities estimated by the *System Under Development* can be fed back into the flight route planning in real-time, provided the estimated quantities are available in real-time.

One example of such a *System Under Development* could be a flexible exploration system for solar fields. In such a system, a confidence measure could be maintained in-flight, to represent the uncertainty associated with already estimated quantities. Based on this measure, new tasks could be scheduled to further explore regions of the solar field that have not been sufficiently observed. For instance, a corner-based coarse calibration system could plan its next UAV waypoint depending on the confidence of already computed concentrator orientation estimates.

Another relevant *System Under Development* could rely on algorithms requiring a certain object or reflection to be seen in the captured images during the flight. For instance, specific camera poses are required to observe the reflections of a tower edge [1], a light-emitting diode (LED) [4] or other markers reflected by the heliostats. By incorporating a feedback signal derived from the observed reflections, the system can

dynamically adjust its flight route to make these quantities more reliably observable. The real-time capability of the simulation environment enables the prototyping of such systems.

In this way, the simulation environment can serve as a platform for the development and testing of autonomous systems, capable of actively managing uncertainty and incorporating feedback to improve the measurement data.

4. Validation methodology

This section presents the methodology for validating the simulation environment through the evaluation of key quantities relevant to the *System Under Development*. Based on the requirements introduced in Section 3.1, the following simulated quantities are compared with optical data obtained from a measurement campaign conducted at the Solar Tower Jülich (STJ):

1. The 3D positions of facet corner points are compared in the GCS (see Section 4.3). Only the four outer concentrator corner points are selected, as these points can be detected using DLR's state-of-the-art AI-based perception module and are most relevant for the corner-based heliostat coarse-calibration systems currently developed by DLR.
2. The image coordinates of point light reflections are compared in the ICS (see Section 4.4). The measured reflection image coordinates are obtained with a *HelioPoint* measurement setup [4], where the reflected object is an LED mounted next to the camera. This measurement setup was selected since the reflections of LEDs are most relevant for reflection-based heliostat fine-calibration systems currently developed by DLR.

The optical measurements focus on the validation of the *Solar Field State* and the *Ground Truth Image Feature Computation*. For the *UAV & Graphics Simulation*, the consistency of the concentrator corner image coordinates (computed by the *Ground Truth Image Feature Computation* component) with the images rendered by Unreal Engine is assessed (see Section 4.5). If the computed image coordinates align with the rendered images, a perception module could, in principle, be integrated into the simulation loop and produce outputs equivalent to the ground truth.

In UAV-based condition monitoring systems that incorporate feedback for dynamic flight route planning, accurate position and velocity control are particularly important. To ensure a realistic simulation of the UAV flight dynamics, the responses of both the simulated and a real UAVs to identical MAVLink input commands are compared (see Section 4.6). As the authors in the original AirSim paper [8] show the validity of position control, this work focuses on the validation of responses to velocity control commands.

4.1. Measurement camera

The measurement images for the validation of both concentrator corner points and point light reflections were recorded using a DJI Zenmuse P1 camera, mounted on a DJI Matrice 300 UAV platform. This camera is equipped with a sensor with a resolution of $8192\,\mathrm{px} \times 5460\,\mathrm{px}$ and a physical sensor size of $35.9\,\mathrm{mm} \times 24\,\mathrm{mm}$, resulting in a pixel size of approximately $4.4\,\mu\mathrm{m}$. The camera was operated with a fixed focal length of $35\,\mathrm{mm}$, with the focus set to infinity to ensure consistent sharpness across the entire scene and different images. The exposure mode was set to manual to ensure consistent brightness and sharpness across varying lighting conditions. The images were captured using the camera's global mechanical shutter, ensuring geometric accuracy. The camera IOR was obtained through a photogrammetric reconstruction using the commercial software Aicon 3D Studio. The calibration was performed using image data recorded using a dedicated flight pattern over a collection of Aicon optical markers. From this calibration dataset, Aicon 3D Studio jointly estimates the 3D positions of the markers, the camera EORs, and the camera IOR.

4.2. Error metrics

This subsection introduces the error metrics used for the validation of two quantities, namely the concentrator corner points and the point light reflections. For a given quantity, suppose that N pairs of simulated points $\boldsymbol{x}_n^{\mathrm{Sim}}$ and measured points $\boldsymbol{x}_n^{\mathrm{Meas}}$ are collected, where $n = 1, \dots, N$. The error metric for this quantity is then computed over the collection of errors $\boldsymbol{e}_n = \boldsymbol{x}_n^{\mathrm{Meas}} - \boldsymbol{x}_n^{\mathrm{Sim}}$. For the validation of concentrator corner points, the errors $\boldsymbol{e}_n$ are three-dimensional vectors, expressed w.r.t. the GCS. For the validation of point light reflections, the errors $\boldsymbol{e}_n$ are two-dimensional vectors, referenced to quantities derived from measurements in the ICS.

The estimated mean $\bar{\boldsymbol{\mu}}$ and the estimated standard deviation $\bar{\sigma}$ are defined as

$$\bar{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{e}_n\,, \quad \bar{\sigma} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{e}_n - \bar{\boldsymbol{\mu}})^2}\,. \tag{12}$$

The root mean squared error (RMSE) is given as

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|\boldsymbol{e}_n|^2} \tag{13}$$

and the mean absolute error (MAE) is given as

$$\mathrm{MAE} = \frac{1}{N}\sum_{n=1}^{N}|\boldsymbol{e}_n|\,, \tag{14}$$

where $|\boldsymbol{e}_n|$ denotes the Euclidean norm of the error vector.

When the RMSE is to be expressed for a specific component of the quantity, it is denoted with a subscript corresponding to that component, e.g. $\mathrm{RMSE}_{\mathrm{X}}$ for the X-component. In Section 5, both RMSE and MAE are presented: As the RMSE penalizes large errors more strongly than the MAE, differences between the two metrics may be explained by the presence of outliers or a non-symmetric error distribution [20].

4.3. Concentrator corner point validation methodology

4.3.1. Point cloud measurement via photogrammetry

To obtain the 3D positions of the concentrator corner points, a series of images is recorded using the camera described in Section 4.1 during a measurement flight over the solar field. The 2D image coordinates of the concentrator corners are detected in every image by an AI-based perception module [21]. These coordinates are then processed by Aicon 3D Studio, which performs a photogrammetric reconstruction by jointly estimating both the camera EORs and the 3D corner point cloud, assuming the camera IOR determined in the camera calibration process (see Section 4.1). The simulated concentrators are selected to match the subset of concentrators for which all four corner points were successfully reconstructed in the Aicon 3D Studio evaluation. They are oriented using the motor positions provided by the field operator at the image recording time as input to the *Kinematic Model*. Each facet is placed according to the ideal *Surface Model* and the simulated points are obtained by selecting those facet corner points that make up the four outer corners of the concentrator. The measured point cloud is subsequently aligned with the simulated point cloud using a Helmert transform, whose parameters are obtained through the Kabsch algorithm [22]. This alignment step ensures that both point clouds are expressed in the same coordinate frame, namely the GCS in which the field data is defined. After the alignment step, the simulated point cloud is compared with the measured point cloud by means of an error model, described in Section 4.3.3. Fig. 5 illustrates the workflow for generating both the simulated and measured point clouds.

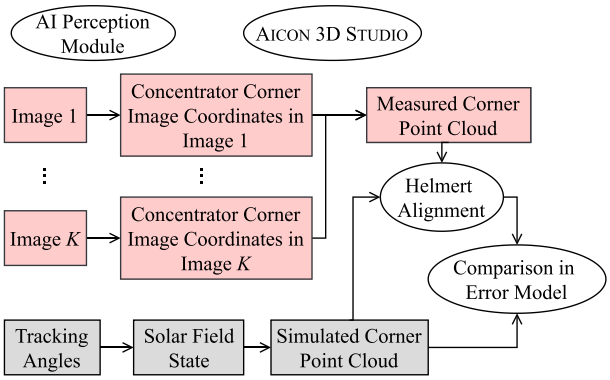


Fig. 5. Overview of the workflow used to generate and align both measured (top) and simulated (bottom) concentrator corner point clouds prior to their comparison. K denotes the total number of images used to generate the measured point cloud. All measured quantities are depicted as red rectangles, while simulated quantities are depicted as gray rectangles. Processing steps are represented by white ellipses.

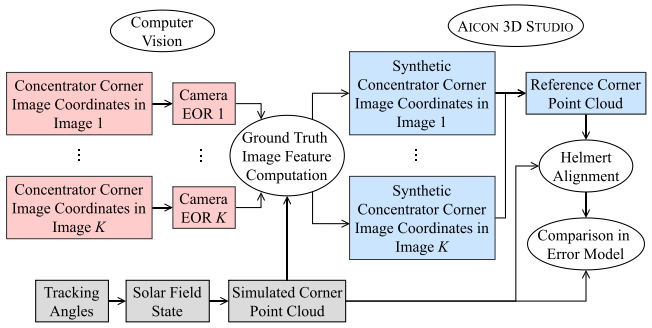


Fig. 6. Overview of the workflow used to generate and align both reference (top) and simulated (bottom) concentrator corner point clouds prior to their comparison. K denotes the total number of images used to generate the measured point cloud. All measured quantities are depicted as red rectangles and all reference quantities are depicted as blue rectangles, while simulated quantities are depicted as gray rectangles. Processing steps are represented by white ellipses.

4.3.2. Reference measurement

Since the corner point positions are estimated using AICON 3D STUDIO, the resulting measured corner point cloud is subject to noise. This uncertainty arises from factors such as noise in the perception module and suboptimal camera perspectives during the measurement flight, both of which affect the accuracy of the generated corner point cloud. To estimate this uncertainty, synthetic concentrator corner image coordinates are computed from the simulated corner point cloud using the *Ground Truth Image Feature Computation*. The required synthetic camera EORs are determined from the real concentrator corner image coordinates detected in the actual measurement images. More precisely, the camera EOR is optimized such that the synthetic image coordinates closely resemble the real image coordinates for each image. The synthetic image coordinates are passed to AICON 3D STUDIO, generating a second point cloud, referred to as the reference corner point cloud. This point cloud is then aligned with and compared to the simulated corner point cloud, analogously to the workflow outlined in Section 4.3.1. Fig. 6 illustrates the workflow used to generate the reference corner point cloud. This procedure isolates the error contribution introduced by AICON 3D STUDIO and provides an estimate of the uncertainty in the measured corner point cloud attributable to the measurement process.

4.3.3. Error model and stepwise error elimination

The following analysis models the deviation between simulated and measured corner points as a superposition of multiple error components. Each concentrator is associated with four error vectors, one for each corner, defined as the vector from the simulated to the corresponding measured corner point. The four error vectors of each concentrator are then explained as the superposition of translational errors, rotational errors, scale errors along the concentrator X- and Y-directions and residual errors. These components reflect the physically most plausible error sources:

1. **Translational errors** explain a uniform shift of all four simulated corner points of a concentrator w.r.t. its measured corner points. These errors may occur due to imprecisions in the field data: During the field commissioning, each heliostat position is measured using a Tachymeter, where translational measurement errors can be introduced.
2. **Rotational errors** explain a rotation of all four simulated corner points of a concentrator w.r.t. its measured corner points. These errors may occur due to several effects: Tracking errors cause the true concentrator orientation to deviate from the set orientation. Furthermore, the kinematic parameters $\boldsymbol{k}_n$ are optimized by the camera-target method w.r.t. the optical axis of each heliostat. A heliostat rotated around its optical axis produces the same calibration point in the camera-target method, since the reflected sun beam only depends on the surface normal. However, such a rotation alters the positions of the concentrator corner points, introducing a rotational discrepancy between the simulated and measured corner points. In addition, erroneous assumptions during calibration (e.g. errors in the sun position, target position and the heliostat position) may propagate into the normal vector computation during the calibration process. This may cause the kinematic parameters $\boldsymbol{k}_n$ to not represent the physical condition of the field, but rather to compensate for the aforementioned erroneous assumptions.
3. **Scale errors** explain a stretching or compression of all four simulated corner points along the concentrator X- and Y-direction w.r.t. to their measured counterparts. Such errors can arise when the assumed concentrator geometry deviates from the actual physical geometry, e.g. due to inaccuracies in the modeled facet dimensions or gap sizes between facets.
4. **Residual errors** summarize all remaining errors not explainable by the above error components. They may arise due to facet mounting or canting errors [23].

During the point cloud comparisons, each error component is eliminated sequentially by applying a corresponding alignment step to the simulated points, fitting them to the measured/reference points for each concentrator. The following description only mentions the alignment with the measured points. However, the same steps are carried out w.r.t. the reference points. Recomputing the error metrics at each stage quantifies the contribution of individual components and provides a clearer understanding of how different error sources contribute to the overall deviation.

To eliminate translational errors, the simulated points are shifted, aligning the centroids of the simulated points with the centroids of the measured points for each concentrator. Rotational errors are then eliminated by rotating all simulated points around their centroids, such that they best fit the measured points for each concentrator. In a last step, scale errors are eliminated by scaling the simulated points along the X- and Y-axis of the corresponding CCS to best fit the measured points for each concentrator. While the order of these elimination steps is arbitrary, the chosen sequence is computationally convenient: First aligning simulated and measured centroids allows the rotation to be applied around this common centroid. Similarly, scaling can be performed assuming both point sets lie in the same CCS when they are first aligned rotationally.

After each error elimination step, the following error quantities are derived from the remaining error vectors $\boldsymbol{e} = (e_X, e_Y, e_Z)$, pointing from each simulated point to its measured counterpart: Since the mean of

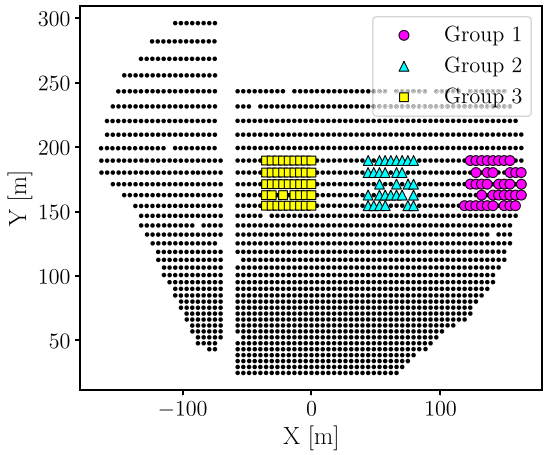


Fig. 7. Overview of the locations of each heliostat group for the three measurement images.

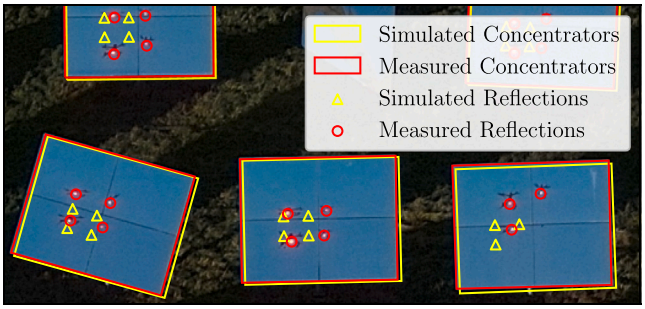


Fig. 8. Magnified view of validation image 1. For the validation of point light reflections, both concentrator corner points and LED reflections are measured (red rectangles and circles) and simulated (yellow rectangles and triangles) in the ICS and converted to their respective CCSs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the error distributions is zero for all components due the definition of the Kabsch algorithm [22], the standard deviations $\bar{\sigma}_{e_X}$, $\bar{\sigma}_{e_Y}$ and $\bar{\sigma}_{e_Z}$ are computed for each error component as defined in Eq. (12). In addition, the error magnitude $|\boldsymbol{e}|$ is characterized by the RMSE and MAE, as defined in Eqs. (13) and (14). The RMSEs after every error elimination step are denoted:

1. $\mathrm{RMSE}_{\mathrm{U}}$: RMSE obtained when leaving the simulated point cloud **U**nmodified (i.e. when no error elimination step is applied).
2. $\mathrm{RMSE}_{\mathrm{ET}}$: RMSE obtained after **E**liminating **T**ranslational errors.
3. $\mathrm{RMSE}_{\mathrm{ETR}}$: RMSE obtained after **E**liminating **T**ranslational and **R**otational errors.
4. $\mathrm{RMSE}_{\mathrm{ETRS}}$: RMSE obtained after **E**liminating **T**ranslational, **R**otational and **S**cale errors.

Since the individual error types are orthogonal (e.g. a rotation around the simulated concentrator centroid does not effect the translational error of this concentrator), the overall $\mathrm{RMSE}_{\mathrm{U}}$ is composed by the RMSEs of the individual error components:

$$\mathrm{RMSE}_{\mathrm{U}}^2 = \mathrm{RMSE}_{\mathrm{Trans}}^2 + \mathrm{RMSE}_{\mathrm{Rot}}^2 + \mathrm{RMSE}_{\mathrm{Scale}}^2 + \mathrm{RMSE}_{\mathrm{Res}}^2 \,. \tag{15}$$

Since each elimination step removes its respective component from its preceding RMSE, the individual RMSE components are obtained as

$$\begin{aligned} \mathrm{RMSE}_{\mathrm{Trans}}^2 &= \mathrm{RMSE}_{\mathrm{U}}^2 - \mathrm{RMSE}_{\mathrm{ET}}^2 \,, \\ \mathrm{RMSE}_{\mathrm{Rot}}^2 &= \mathrm{RMSE}_{\mathrm{ET}}^2 - \mathrm{RMSE}_{\mathrm{ETR}}^2 \,, \\ \mathrm{RMSE}_{\mathrm{Scale}}^2 &= \mathrm{RMSE}_{\mathrm{ETR}}^2 - \mathrm{RMSE}_{\mathrm{ETRS}}^2 \,, \\ \mathrm{RMSE}_{\mathrm{Res}}^2 &= \mathrm{RMSE}_{\mathrm{ETRS}}^2 \,. \end{aligned} \tag{16}$$

4.4. Point light reflection validation methodology

The measurement data for the validation of point light reflections is recorded using the camera described in Section 4.1 with a *HelioPoint* [4] measurement setup: An LED is mounted to the camera lens and both camera and LED are oriented toward a group of heliostats, pointing to the expected UAV position. This way, three measurement images are recorded, which are shown in Appendix B. Fig. 7 illustrates the locations of the recorded heliostat groups in the solar field for each measurement image.

Both LED reflections and concentrator corner points are detected in the ICS for each concentrator. The image coordinates of the reflections are transformed from pixel space to metric coordinates w.r.t. the CCS, using a homography defined by the four concentrator corner points and the known concentrator geometry. The simulated concentrators are aligned using the *Kinematic Model* and the camera EOR is estimated by means of a Perspective-n-Point algorithm [24], using the concentrator center points as correspondences. Concentrator corner points and point light reflections are then simulated (see Section 3.4) and expressed w.r.t. the CCS, using a homography as described above.

The simulation of point reflections makes various assumptions, potentially leading to deviations between simulated and measured reflections: As discussed in Section 4.3.3, the simulated concentrator orientations may deviate from the true concentrator orientations by rotational errors, leading to a shift of the simulated reflections w.r.t. the measured reflections in the ICS. For instance, misalignments between the GCS, the target coordinate system, or a true east-north-up frame [25] may go undetected during the determination of kinematic parameters using the camera-target method. In such cases, the kinematic parameters could converge such that the reflected solar beam still reaches the target for all calibration points, thereby compensating for the erroneous coordinate assumptions. However, the resulting concentrator normal vectors would deviate from the true physical normals, leading to discrepancies between simulated and measured reflections. Similarly, errors in the image-based camera pose estimation (and hence in the assumed LED position) may lead to shifted simulated reflections w.r.t. the measured reflections. Furthermore, the measured reflections are effected by distortions, caused by slope errors in the facet mirror surfaces. To account for these errors, the validation process is repeated for two cases:

1. In the raw orientations, ideal surfaces (ROIS) case, the reflections are simulated using the "raw" concentrator orientations obtained from the kinematic model and assuming the ideal surface case. Fig. 8 shows the magnified view of image 1 with reflections simulated under the ROIS assumptions.
2. In the fitted orientations, measured surfaces (FOMS) case, errors stemming from inaccurate concentrator orientations and noise in the camera pose estimate are eliminated by a fit of the concentrator orientations: Starting from the raw orientation, each concentrator is rotated around its X- and Y-axis, such that the deviations between measured and simulated reflections are minimized. During this fit, the point light reflections are simulated taking into account a measurement of the concentrator mirror surface: As described in Section 3.2.2 the concentrator's slope deviation map (obtained from a QDEC-H measurement [12]) is discretized into a 32 × 32 grid. Hence, a total number of 1024 potential reflections are computed for each concentrator in every orientation optimization step. If more than one reflection is classified as visible for a facet, the final reflection position is computed as the mean of all visible reflections for this facet.

For both cases, the distributions of the error vectors $\boldsymbol{e} = (e_X, e_Y)$ from simulated reflections to measured reflections is converted from millimeter units to milliradian units by considering the distance between

the UAV and each concentrator. The resulting distributions are then characterized by the estimated mean values $\bar{\mu}_{e_X}$ and $\bar{\mu}_{e_Y}$, as well as the estimated standard deviations $\bar{\sigma}_{e_X}$ and $\bar{\sigma}_{e_Y}$. In addition, the component-wise RMSE_X and RMSE_Y as well as the component-wise MAE_X and MAE_Y are computed.

Note that only those pairs of measured and simulated reflections are considered where a measured reflection is available. All simulated reflections without a measured counterpart are neglected. On the other hand, a pair of a measured reflection and its non-visible simulated counterpart (i.e. a simulated reflection that does not lie inside the facet frame on the ICS) is still considered, as the position of the simulated reflection outside of the facet frame can be computed.

4.5. Image data consistency

To enable the comparison between the concentrator corner point image coordinates computed by the *Ground Truth Image Feature Computation* with the images rendered by Unreal Engine, both components are initialized using the same *Camera State*. Specifically, the simulated camera is positioned at a height of $h = 30\,\mathrm{m}$ and oriented toward the solar field with a pitch angle of $\theta = 30°$. For simplicity, the camera IOR is modeled as a pinhole projection (see Appendix A) with a 4K resolution and a horizontal field of view of 60°. The choices of camera EOR and IOR reflect a realistic use case, in which the camera is operated in video mode and positioned and oriented considering the STJ solar field layout and concentrator dimensions. The rendered image is then overlaid by the simulated concentrator corner point image coordinates.

While Unreal Engine offers various reflection models with varying degrees of computational complexity, an analysis of the reflection accuracy exceeds the scope of this work. Initial experiments show that Unreal Engine can provide point light reflections matching the reflections of the *Ground Truth Image Feature Computation*. However, these experiments assume ideal facet surfaces and short distances ($< 20\,\mathrm{m}$) between simulated UAV and concentrator. At larger distances, the reflection accuracy decreases.

4.6. UAV flight dynamics validation methodology

To validate the UAV flight dynamics simulation, the responses of both real and virtual UAVs to a velocity step command are compared. The real UAV runs ArduPilot on a physical flight controller, while the virtual UAV operates ArduPilot in a SITL. Due to the symmetric configuration of the rotors, the system response in the X-direction is representative of any direction within the horizontal (XY-) plane and is therefore presented as the horizontal case. In contrast, the response in the Z-direction (vertical) is analyzed separately, as it exhibits different dynamics due to the influence of gravity. Each UAV begins in a hover state at $0\,\mathrm{m/s}$ before receiving a step velocity command. The command causes the UAVs to accelerate to a target velocity, which is then held constant at $1\,\mathrm{m/s}$ for a duration of $T = 18\,\mathrm{s}$, controlled by the ArduPilot velocity controller. This is done for both horizontal and vertical directions. The recorded flight logs of both simulated and real UAV are interpolated at a rate of $100\,\mathrm{Hz}$ for comparability. To reduce random effects such as sensor noise and disturbances due to wind and turbulences, the UAV is accelerated and stopped 10 times. The system response is computed as the mean over all responses.

Several performance metrics are derived from the averaged system responses to characterize the UAV flight dynamics: The rise time t_{Rise} is defined as the time required for the velocity to increase from $0.1\,\mathrm{m/s}$ to $0.9\,\mathrm{m/s}$. The settling time t_{Settling} denotes the time elapsed until the velocity remains within a specified tolerance band of $\pm 0.05\,\mathrm{m/s}$ around the target velocity of $1\,\mathrm{m/s}$. The steady state srror (SSE) is calculated as the deviation between the mean velocity within this tolerance band and the commanded velocity.

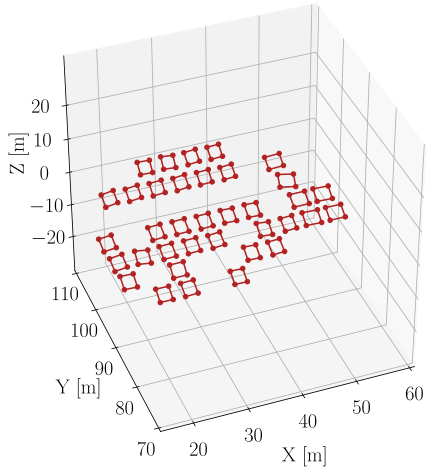


Fig. 9. The measured point cloud is obtained by following the process depicted in Fig. 5.

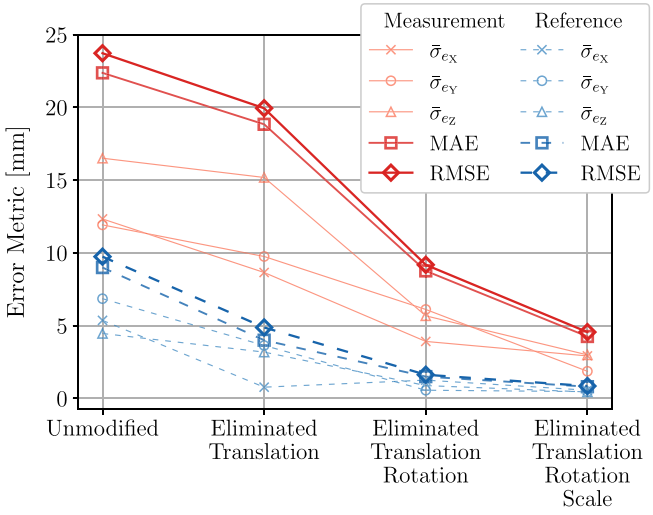


Fig. 10. Graphical representation of the error metrics $\bar{\sigma}_{e_X}$, $\bar{\sigma}_{e_Y}$, $\bar{\sigma}_{e_Z}$, MAE and RMSE remaining after each error elimination step applied to the simulated point cloud to fit both measurement and reference point cloud.

5. Validation results and discussion

5.1. Concentrator corners

5.1.1. Results

Following the process described in Section 4.3 a simulated, measured and reference point cloud are obtained. Each point cloud has a total number of $N = 36$ heliostats, which amounts to 144 concentrator corner points. The measured point cloud is depicted in Fig. 9. The simulated and reference point clouds can be found in Appendix B.

Fig. 10 shows the error metrics as a function of the error elimination steps: The "Measurement" graph shows the quantities for the comparison between the measured point cloud and the simulated point cloud. The "Reference" graph shows the quantities for the comparison between the synthetically generated reference point cloud and the simulated point cloud. The error distributions from which the error metrics are derived can be found in Appendix B, alongside the numeric data for Fig. 10.

In each error elimination step and for both measured and reference data, the RMSE is only slightly higher than the MAE, indicating that the error magnitude distributions are free of outliers and show symmetric

Table 1
RMSEs for each error component, derived for both measurement and reference from the data depicted in Fig. 10.

Comparison	RMSE [mm]			
	Trans	Rot	Scale	Res
Measured	12.8	17.7	8.0	4.6
Reference	8.4	4.6	1.4	0.8

behavior. Therefore, the discussion is limited to the RMSE, noting that the same behavior also applies to the MAE. The RMSEs for each error component are obtained by applying Eq. (16) to the data depicted in Fig. 10. The resulting RMSEs are listed in Table 1.

The measured RMSE reduction after eliminating translational errors is $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Trans}} = 12.8\,\mathrm{mm}$, while the corresponding reference error amounts to $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Trans}} = 8.4\,\mathrm{mm}$. Subsequent elimination of rotational errors further reduces both the measured and reference RMSEs, with an impact of $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Rot}} = 17.7\,\mathrm{mm}$ and $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Rot}} = 4.6\,\mathrm{mm}$, respectively. Eliminating scale errors leads to an additional reduction in the measured RMSE, with only a minor reduction in the reference RMSE. The corresponding scale error components are $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Scale}} = 8.0\,\mathrm{mm}$ and $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Scale}} = 1.4\,\mathrm{mm}$. The residual error components after eliminating all error components are $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Res}} = 4.6\,\mathrm{mm}$ and $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Res}} = 0.8\,\mathrm{mm}$, respectively.

5.1.2. Discussion and impact on system development

The following discussion interprets the resulting error components identified through the stepwise error elimination process. While the physical origins of each error type have already been outlined in Section 4.3.3, the focus here lies on assessing the impact of the observed error magnitudes on the transferability of a system developed in simulation to the real world.

Since measured and reference translation RMSEs lie in the same order of magnitude, significant parts of the translational errors can be attributed to Aicon 3D Studio. Even if all translational deviation is attributed to inaccuracies in the field data, the heliostat positions are determined to be known with an accuracy of $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Trans}} = 12.8\,\mathrm{mm}$. However, the translational error quantities should be interpreted as indicators of field data *consistency*: The initial Helmert alignment is necessary to express the measured point cloud w.r.t. the GCS despite the lack of a reference frame in the Aicon 3D Studio evaluation. Potential misalignments of the field data w.r.t. a world coordinate system are not detectable in this measurement setup.

For systems that estimate concentrator orientations based on corner point positions, translational errors of the observed magnitude have minimal impact on orientation estimates in the OCS. This is because slight shifts of all corner points do not significantly alter their relative projections on the camera sensor at typical distances between camera and concentrators. However, these systems may also rely on camera pose estimates derived from field data, such as the Perspective-n-Point algorithm utilized in Section 4.4. In this case, translational errors may propagate into the final estimate. These errors can be accounted for during development by simulating translational errors at levels consistent with those identified in this work.

The observed $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Rot}} = 4.6\,\mathrm{mm}$ indicates that part of the rotational error originates from the photogrammetric reconstruction in Aicon 3D Studio. However, the significantly larger measured value of $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Rot}} = 17.7\,\mathrm{mm}$ suggests that the simulated concentrators are indeed rotated w.r.t. the real-world concentrators.

Rotational discrepancies may cause the *System Under Development* to yield different orientation estimates in simulation than in the real world when both the simulated field and the real system are initialized with identical motor positions. However, further experiments show that these rotational errors occur primarily around the concentrator surface normal vector. This normal vector, typically the output of corner-based orientation estimation algorithms, hence remains largely unaffected.

Table 2
Component-wise RMSE, MAE, mean and standard deviation in milliradian for the ROIS case.

Image	RMSE [mrad]		MAE [mrad]		$\bar{\mu}$ [mrad]		$\bar{\sigma}$ [mrad]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	2.08	2.43	1.71	1.32	1.50	0.47	1.45	2.39
Image 2	1.69	2.18	1.30	1.58	−1.22	−0.85	1.18	2.02
Image 3	2.76	1.56	2.34	1.33	−2.31	−0.85	1.52	1.32
Overall	2.25	2.09	1.81	1.39	−0.57	−0.35	2.18	2.07

This is supported by the fact that rotational deviations of the normal vectors remain below 3 mrad on average, as shown in Section 5.2.1. Corner-based methods are typically used for coarse calibration applications, aiming for accuracies around 3 mrad to 10 mrad [26]. The deviations identified in this work are not expected to be critical to the development of such systems. Even if the underlying kinematic model produces slightly incorrect orientations, the simulation environment still provides consistent corner point positions. As a result, an algorithm developed and tested in simulation can adapt to concentrators oriented differently in a real solar field without compromising its functionality.

For scaling errors, the fact that the measured scale RMSE is larger than the reference scale RMSE supports the interpretation that the assumed concentrator geometry deviates from the true physical geometry. In this case, fine-tuning the assumed geometry could improve algorithm performance when applied to real world data. Systematic errors introduced by the perception module, such as a consistent bias toward detecting corner points closer to the concentrator center, may also contribute to the observed discrepancies.

Random noise in the perception module also likely contributes to the remaining residual errors. These residuals may additionally be caused by physical imperfections not represented by the error model. In contrast to $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{Res}}$, $\mathrm{RMSE}^{\mathrm{Ref}}_{\mathrm{Res}}$ is significantly smaller, as the generation of the reference point cloud does not model noise in the perception module or physical imperfections (see Section 4.3.2).

To assess the practical significance of the remaining geometric errors, their projected size in the pixel space can be compared to the ground sampling distance (GSD), describing the physical distance covered by one pixel. The measured combined scaling and residual RMSE components amount to $\mathrm{RMSE}^{\mathrm{Meas}}_{\mathrm{ETR}} = 9.2\,\mathrm{mm}$. Assuming a flight height of $h = 30\,\mathrm{m}$, a pitch angle of $\theta = 30°$ and a DJI Mavic 3 Enterprise camera in video mode (pixel size of $s_{\mathrm{px}} = 3.3\,\mu\mathrm{m}$, focal length of $f = 8.8\,\mathrm{mm}$), the GSD amounts to $\mathrm{GSD} = (h \cdot s_{\mathrm{px}})/(f \cdot \cos\theta) = 12.86\,\mathrm{mm/px}$. This implies that the combined scaling and residual RMSE projects to less than one pixel on the ICS, indicating that these inaccuracies are unlikely to pose a significant limitation to an image-based processing pipeline.

5.2. Point light reflections

5.2.1. Results

An ROIS validation is possible for all heliostats with a visible reflection, i.e. 38 heliostats in group 1, 31 heliostats in group 2 and 43 heliostats in group 3. Fig. 11 shows the scatter plots of e_Y over e_X in milliradian units for all three validation images for the ROIS case. The error metrics of the distributions are summarized in Table 2 for each image/heliostat group. The scatter plots of e_Y over e_X in millimeter units as well as the corresponding error metric summary can be found in Appendix B.

The error distributions vary for the different images/heliostat groups: Especially the estimated mean value $\bar{\mu}_{e_X}$ strongly varies as a function of the measured group, with $\bar{\mu}_{e_X} = 1.50\,\mathrm{mrad}$ for group 1, $\bar{\mu}_{e_X} = -1.22\,\mathrm{mrad}$ for group 2 and $\bar{\mu}_{e_X} = -2.31\,\mathrm{mrad}$ for group 3. The image-wise estimated standard deviations lie below 2 mrad, which is the expected tracking accuracy at the STJ. An exception to this are the error Y-components for group 1 and 2 with $\bar{\sigma}_{e_Y} = 2.39\,\mathrm{mrad}$ and $\bar{\sigma}_{e_Y} = 2.02\,\mathrm{mrad}$, respectively. The overall RMSE_X lies at 2.25 mrad and

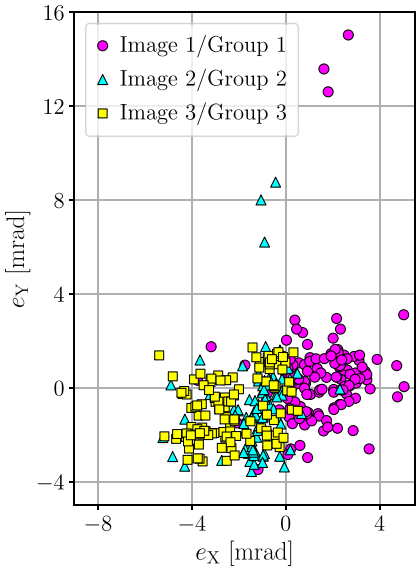


Fig. 11. Scatter plots of e_Y over e_X in milliradian units for the ROIS case. It can be seen that both image 1 and image 2 contain an outlier, resulting in increased error quantities.

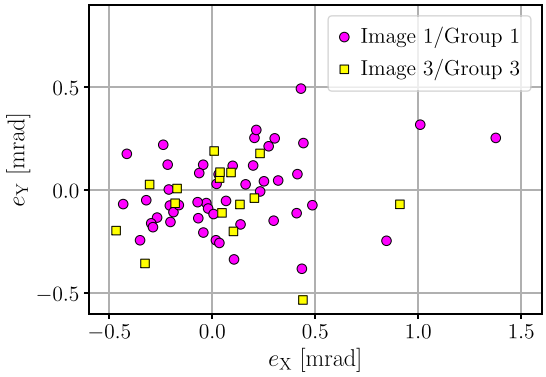


Fig. 12. Scatter plots of e_Y over e_X in milliradian units for the FOMS case.

Table 3
Component-wise RMSE, MAE, mean and standard deviation in milliradian for the FOMS case.

Image	RMSE [mrad]		MAE [mrad]		$\bar{\mu}$ [mrad]		$\bar{\sigma}$ [mrad]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	0.37	0.18	0.26	0.15	0.10	−0.01	0.36	0.19
Image 3	0.32	0.29	0.23	0.20	0.07	−0.13	0.32	0.26
Overall	0.35	0.22	0.25	0.17	0.09	−0.04	0.35	0.22

the overall RMSE_Y lies at 2.09 mrad. The MAE is smaller for both error components in all images.

The validation process is repeated for the FOMS case, additionally fitting the concentrator orientations and considering the discretized slope deviation maps. The resulting error vector scatter plots in milliradian units as well as the summary of error metrics are shown in Fig. 12 and Table 3 respectively. The scatter plots and error metric summary in millimeter units can be found in Appendix B. It is important to note that an FOMS validation is only possible for those heliostats with an available slope deviation map, leaving 14 concentrators in group 1, no heliostats in group 2 and five heliostats in group 3. An overview of all measured heliostats in the ROIS dataset, with those contained in the FOMS subset marked by a dot, is provided in Appendix B.

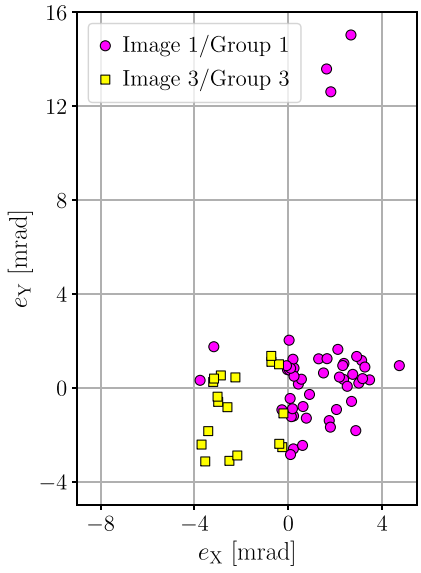


Fig. 13. Scatter plots of e_Y over e_X in milliradian units for the ROIS case, evaluated on the FOMS heliostat subset.

Table 4
Component-wise RMSE, MAE, mean and standard deviation in milliradian for the ROIS case, evaluated on the FOMS data.

Image	RMSE [mrad]		MAE [mrad]		$\bar{\mu}$ [mrad]		$\bar{\sigma}$ [mrad]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	1.99	3.60	1.53	1.78	1.23	0.90	1.58	3.52
Image 3	2.46	1.77	2.12	1.46	−2.12	−0.90	1.28	1.57
Overall	2.13	3.21	1.69	1.69	0.33	0.42	2.12	3.21

The errors are significantly reduced by the additional processing step: $\bar{\mu}$ is slightly shifted in positive X-direction but lies close to zero for both groups. The standard deviation is slightly higher in X- than in Y-direction. The overall error is reduced to $\text{RMSE}_X = 0.35$ mrad and $\text{RMSE}_Y = 0.22$ mrad and to $\text{MAE}_X = 0.25$ mrad and $\text{MAE}_Y = 0.17$ mrad.

Since the FOMS validation data is a subset of the ROIS validation data, the ROIS case is additionally evaluated on the FOMS heliostat subset only. This ensures that differences are not caused by random sampling effects of the FOMS subset. The resulting error vector scatter plots in milliradian units as well as the summary of error metrics are shown in Fig. 13 and Table 4 respectively. The scatter plots and error metric summary in millimeter units can be found in Appendix B.

The error metrics for the ROIS evaluated on the FOMS subset are generally comparable to those obtained from the full dataset. A notable exception is the Y-direction in image 1, where both the RMSE and the standard deviation $\bar{\sigma}_{e_Y}$ are increased by more than 1 mrad relative to the evaluation on the full dataset.

5.2.2. *Discussion and impact on system development*

As previously described in Section 4.4, errors in the ROIS case can originate from multiple sources. The spread observed in the error distributions can partly be attributed to tracking inaccuracies. Specifically, the increased spread along the Y-direction in groups 1 and 2 is likely caused by concentrator orientation outliers, as visible in Fig. 11. One such outlier causing three reflections in image 1 to largely deviate from the remaining reflections is included in both the ROIS and the FOMS datasets. This leads to an increased standard deviation $\bar{\sigma}_{e_Y}$ when validating the ROIS case on the FOMS dataset as compared to a validation on the ROIS dataset, because the reduced number of heliostats included in the FOMS dataset provides less opportunity for such outliers to be statistically compensated. In addition to heliostat

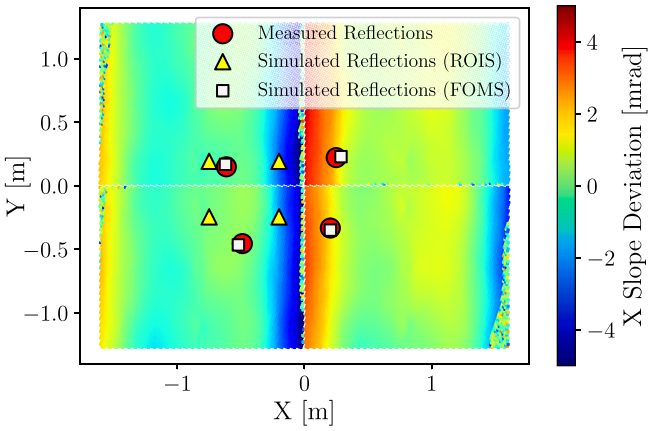


Fig. 14. An exemplary X-slope deviation map, overlaid by the measured reflections (red circles) and the simulated reflections for both the ROIS case (yellow triangles) and the FOMS case (white squares).

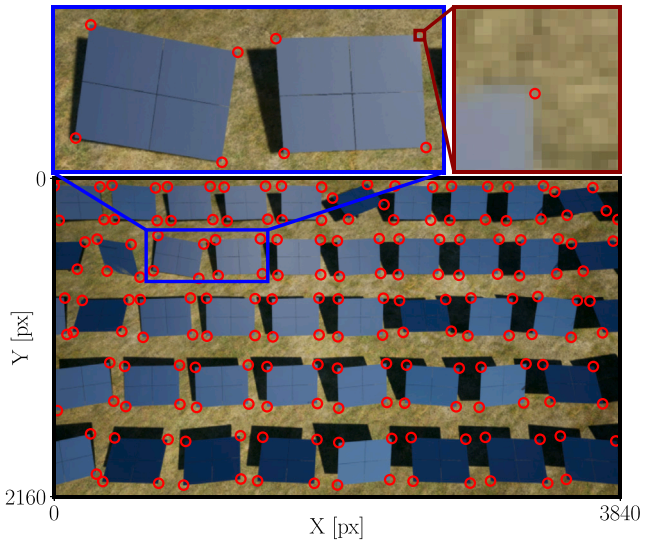


Fig. 15. Image rendered with Unreal Engine, overlaid by the concentrator corner point image coordinates obtained from the *Ground Truth Image Feature Computation*. The top left image and the top right image show zoomed-in sections of the bottom image.

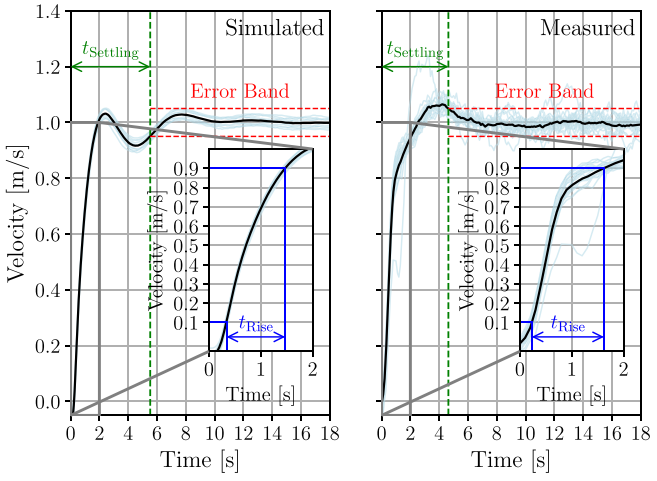


Fig. 16. Horizontal step responses for both the simulated UAV (left side) and the real UAV (right side), alongside the derived metrics. The thin light blue lines show the responses for each repetition, the black thick lines show the averaged responses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

orientation errors, surface slope deviations contribute to the spread across all error distributions.

The systematic variation in the estimated mean values could be explained by an inaccurate camera EOR estimate, as discussed in Section 4.4. However, Monte-Carlo simulations using the field data uncertainty derived in Section 5.1 suggest that the effect of camera EOR inaccuracies is not large enough to fully explain the observed effects. This indicates that additional factors are contributing to these systematic deviations. As discussed in Section 4.4, erroneous assumptions in the camera-target method could also lead to systematic heliostat tracking errors, depending on the concentrator's position or orientation. However, the presence of such errors stays speculative and further investigation is required to explain the cause of the observed systematic deviations.

To illustrate how the concentrator orientation fit and the incorporation of measured slope deviation data in the surface model reduces the observed errors, an example is shown in Fig. 14. The figure shows the distribution of simulated reflections for both the FOMS and ROIS cases for a representative heliostat from the FOMS dataset. For reference, the corresponding measured reflections are also included. The figure highlights the significant reduction in both spread and systematic shift of the error achieved in the FOMS case. The remaining inaccuracies lie in the order of magnitude necessary to develop fine-calibration systems [26].

The ROIS validation indicates that the simulation does not perfectly reproduce the true solar field state, potentially due to noise in the camera EOR estimation, inaccuracies in the tracking angles caused by coordinate system mismatches or other error sources. However, the FOMS dataset demonstrates that the surface properties are accurately modeled. Even if the heliostats are oriented differently in simulation compared to the real system, the same argument as in the corner point validation applies (see Section 5.1.2): Since point light reflections are modeled in fine-calibration accuracy, the system's ability to estimate the solar field state is still meaningfully developed and tested under realistic conditions.

5.3. Image data consistency

Fig. 15 shows the image rendered by Unreal Engine, overlaid with the concentrator corner point image coordinates computed by the *Ground Truth Image Feature Computation*. The alignment of the computed points and the corner points in the rendered image confirms the consistency of the two components. This indicates that the rendered image can be used within the simulation loop, allowing a perception module under test to extract image features directly, rather than relying on the *Ground Truth Image Feature Computation*.

Table 5
Comparison between the metrics derived from the simulated and measured velocity step responses in horizontal and vertical direction.

Direction	t_{Rise} [s]		t_{Settling} [s]		SSE [%]	
	Sim	Meas	Sim	Meas	Sim	Meas
Horizontal	1.13	1.39	5.53	4.64	−0.45	0.26
Vertical	0.66	0.72	7.96	1.72	3.04	0.80

5.4. UAV flight dynamic

Fig. 16 shows the horizontal step responses and the derived metrics for both simulated and real UAVs. The step responses for the vertical direction can be found in Appendix B. Table 5 shows the comparison of the metrics derived from the simulated and the measured step responses in horizontal and vertical direction.

The measured step velocity responses exhibit a higher noise level compared to the simulation. While the rise times t_{Rise} are similar in both directions for simulation and measurement, the simulated vertical settling time t_{Settling} is noticeably longer than the measured one. A similar pattern is observed for the SSE: In the horizontal direction, it remains below $1\,\%$ for both simulation and measurement, whereas in the vertical direction, only the measured response meets this threshold.

The increased noise level in the measured responses is likely caused by sensor noise as well as environmental influences such as wind or turbulence during the measurement flight. The observed qualitative differences in the step responses can be explained by different factors. In this work the same UAV parameters as in Shah et al. [8] were used as a proof of concept, since an adjustment of mass, dimensions and other parameters to match the real UAV was considered beyond the scope of this work.

As a result, the flight dynamics of the real UAV deviate from those observed in simulation. In addition, the ArduPilot velocity controller may be tuned differently in simulation compared to the real system. This discrepancy is particularly evident in the vertical direction, where velocity must be controlled against the gravitational force. Despite these differences in the step responses of the simulated and measured flight velocities, velocity-based UAV control logic developed in simulation is expected to be transferable to real-world systems, as controllers are typically designed to tolerate variations in system dynamics. However, fine-tuning of the transferred algorithm may be necessary to account for differences between simulation and real world.

6. Conclusion and outlook

In this work, we have presented a simulation environment for the development and testing of UAV-based CSP condition monitoring systems, aiming to bridge the gap between existing UAV simulators and simulation tools used in the CSP community. Validation results show that the concentrator corner points at the STJ can be simulated with an accuracy of up to RMSE $= 23.7\,\text{mm}$. By eliminating translational and rotational errors, the RMSE is reduced to $9.2\,\text{mm}$, corresponding to sub-pixel accuracy under typical camera configurations. Additionally eliminating scale errors results in a remaining RMSE of $4.6\,\text{mm}$. The origin of the observed scale errors remains an open question for future investigation.

Using a *HelioPoint* measurement setup, we determined that the simulated reflections match the measured reflections with an accuracy of up to $\text{RMSE}_\text{X} = 2.25\,\text{mrad}$ and $\text{RMSE}_\text{Y} = 2.09\,\text{mrad}$. The results assume raw motor positions from the field operator, ideal mirror surfaces, and a camera EOR estimated using a Perspective-n-Point algorithm with field data as 2D–3D point correspondences. In addition to statistical heliostat tracking and surface slope errors, we observed systematic deviations between simulated and measured reflections. These deviations may be attributed to inaccuracies in the camera EOR estimation, misalignments between the GCS and the world coordinate system or other unknown factors. Further investigation is required to clarify their origin. After compensating for these effects by fitting the simulated concentrators to the measured data while considering measured slope deviation maps, the RMSE is reduced to $\text{RMSE}_\text{X} = 0.35\,\text{mrad}$ and $\text{RMSE}_\text{Y} = 0.22\,\text{mrad}$.

We simulated and measured a UAV's response to velocity control commands and compare the responses using three derived metrics. While differences between simulation and measurement are observed, the responses are similar in both the vertical and horizontal directions. This indicates that the simulation environment provides a sufficiently accurate dynamic model for developing and testing UAV control logic. Remaining deviations may stem from parameter mismatches and sensor noise in the real-world system.

The results show that the simulation environment cannot replicate the optical measurement data with perfect accuracy: Perception noise, field data uncertainties, misalignments between coordinate systems and other factors inevitably introduce discrepancies. However, these differences can be quantified and their impact understood. The algorithm's ability to estimate the *Solar Field State* from camera observations can still be developed and tested meaningfully, since the simulated camera observations are consistent with the simulated *Solar Field State* and the simulation captures all aspects critical to the *System Under Development's* functionality.

We conclude that the presented simulation environment offers the added value of safe, fast and cheap prototyping to the development and testing process of new UAV-based CSP condition monitoring algorithms. Some adaptation of system parameters may be required when transferring a system from simulation to real world.

Further investigation is needed to assess the suitability of Unreal Engine for accurately simulating reflections, and to explore whether measured surface maps can be incorporated into the rendering process. Currently, rendered images are used primarily as a qualitative reference to help developers assess whether the simulated observations behave as expected. Since the projected concentrator corner points align with the rendered images, we conclude that the images can be passed to a perception module as part of the simulation loop. However, future research is required to evaluate how well such modules perform when applied to fully simulated image data.

CRediT authorship contribution statement

Alexander Schnerring: Writing – review & editing, Visualization, Investigation, Conceptualization, Writing – original draft, Software, Data curation, Validation, Methodology, Formal analysis. **Rafal Broda:** Writing – review & editing, Methodology, Software, Conceptualization. **Adrian Winter:** Writing – review & editing, Methodology, Validation, Investigation, Visualization, Software. **Michael Nieslony:** Writing – review & editing, Supervision, Conceptualization, Methodology. **Julian J. Krauth:** Supervision, Writing – review & editing, Conceptualization, Methodology. **Marc Röger:** Supervision, Conceptualization, Writing – review & editing, Methodology, Project administration, Funding acquisition. **Sonja Kallio:** Supervision, Writing – review & editing. **Robert Pitz-Paal:** Funding acquisition, Supervision.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT (GPT-4, OpenAI) to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

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.

Acknowledgments

Financial support from the Ministry of Economic Affairs, Industry, Climate Action and Energy of the State of North Rhine-Westphalia, Germany (5hine, contract 005-2108-0068) is gratefully acknowledged. We also thank CSP Services for providing the image data for the reflection validation and the support during the measurement campaign. Furthermore, we thank our DLR colleagues Niels Algner, Oliver Kaufhold, Moritz Wirger, Marcel Sibum, Max Pargmann, Felix Göhring and Peter Schwarzbözl at the STJ.

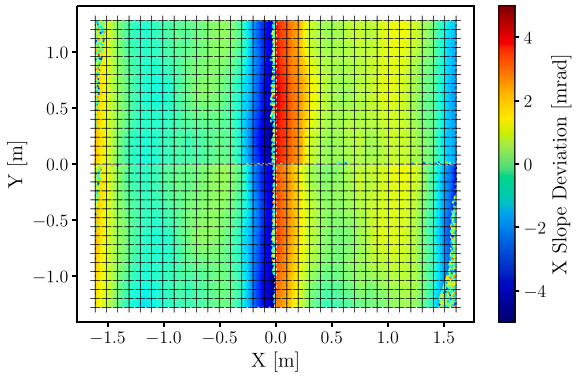


Fig. B.1. Exemplary slope deviation map, overlaid by the discretization grid. In this work, the slope deviation of a concentrator is discretized in a 32 × 32 grid of tiles with the same aspect ratio as the concentrator. Each tile is treated as a perfectly flat mirror surface and orientated according to the slope deviations measured inside the tile.

Appendix A. Camera model

The projection of a point $\boldsymbol{p}^{\mathrm{GCS}}$ onto the ICS is denoted by

$$\boldsymbol{p}^{\mathrm{ICS}} = \mathrm{projection}(\boldsymbol{p}^{\mathrm{GCS}} | \mathrm{IOR}, \mathrm{EOR}). \tag{A.1}$$

The projection can be split into two subsequent steps, considering the EOR and the IOR separately: First, the point is expressed w.r.t. the OCS, using the camera EOR which is equivalent to applying the coordinate transform $\boldsymbol{T}_{\mathrm{GCS}}^{\mathrm{OCS}}$:

$$\boldsymbol{p}^{\mathrm{OCS}} = \boldsymbol{T}_{\mathrm{GCS}}^{\mathrm{OCS}} \cdot \boldsymbol{p}^{\mathrm{GCS}} = \begin{pmatrix} x^{\mathrm{OCS}} \\ y^{\mathrm{OCS}} \\ z^{\mathrm{OCS}} \end{pmatrix}. \tag{A.2}$$

The projection of $\boldsymbol{p}^{\mathrm{OCS}}$ onto the camera sensor used in OpenCV is described by a pinhole projection, followed by the Brown-Conrady lens distortion model [27,28], where the camera IOR is comprised of the focal lengths $f_{\mathrm{X}}, f_{\mathrm{Y}}$, the principal point $(c_{\mathrm{X}}, c_{\mathrm{Y}})$ and the distortion parameters k_1, k_2, k_3, p_1, p_2. The normalized coordinates are then computed as

$$x' = x^{\mathrm{OCS}}/z^{\mathrm{OCS}}, \quad y' = -y^{\mathrm{OCS}}/z^{\mathrm{OCS}}. \tag{A.3}$$

To account for camera distortions, the normalized coordinates are extended:

$$\begin{aligned} x'' &= x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) \\ y'' &= y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 x' y' + p_1(r^2 + 2y'^2), \end{aligned} \tag{A.4}$$

where

$$r^2 = x'^2 + y'^2. \tag{A.5}$$

Using the focal lengths $f_{\mathrm{X}}, f_{\mathrm{Y}}$ and the principal point $(c_{\mathrm{X}}, c_{\mathrm{Y}})$, the projected point in the image plane is expressed in pixel units as

$$\boldsymbol{p}^{\mathrm{ICS}} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_{\mathrm{X}} \cdot x'' + c_{\mathrm{X}} \\ f_{\mathrm{Y}} \cdot y'' + c_{\mathrm{Y}} \end{pmatrix}. \tag{A.6}$$

Analogously to Eq. (A.1), a point cloud $\boldsymbol{P}^{\mathrm{GCS}}$ can be projected:

$$\boldsymbol{P}^{\mathrm{ICS}} = \mathrm{projection}(\boldsymbol{P}^{\mathrm{GCS}} | \mathrm{IOR}, \mathrm{EOR}). \tag{A.7}$$

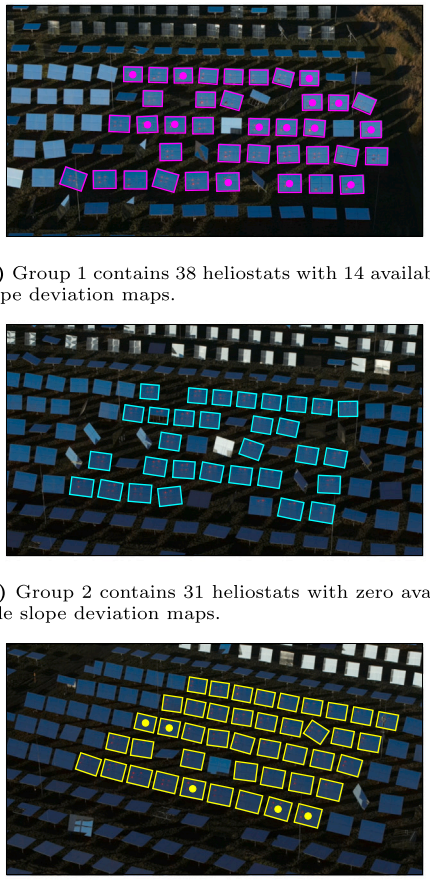
Appendix B. Supplementary material

B.1. Supplementary material for Section 3

See Fig. B.1.

B.2. Supplementary material for Section 4

See Fig. B.2.



(a) Group 1 contains 38 heliostats with 14 available slope deviation maps.

(b) Group 2 contains 31 heliostats with zero available slope deviation maps.

(c) Group 3 contains 43 heliostats with five available slope deviation maps.

Fig. B.2. Overview of the three heliostat groups measured with the *HelioPoint* [4] setup. Heliostats with available slope deviation maps are depicted with a dot in the corresponding color.

Table B.1
Numeric values for the error statistics shown in Fig. 10. The acronyms in the first column denote the error elimination steps applied to the simulated data in order to best fit the measured data, where U=Unmodified, ET=Eliminated Translational Errors, ETR=Eliminated Translational & Rotational Errors and ETRS=Eliminated Translational, Rotational & Scale Errors. Each column denoted by Meas (=Measured) shows the quantities for the comparison between measured point cloud and the simulated data. Each column denoted by Ref (=Reference) shows the quantities for the comparison between synthetically generated reference point cloud and the simulated data.

Elimination step	RMSE [mm]		MAE [mm]		$\bar{\sigma}_{e_{\mathrm{X}}}$ [mm]		$\bar{\sigma}_{e_{\mathrm{Y}}}$ [mm]		$\bar{\sigma}_{e_{\mathrm{Z}}}$ [mm]	
	Meas	Ref	Meas	Ref	Meas	Ref	Meas	Ref	Meas	Ref
U	23.7	9.7	22.4	9.0	12.3	5.4	11.9	6.8	16.5	4.5
ET	19.9	4.9	18.8	4.0	8.6	0.8	9.7	3.6	15.2	3.2
ETR	9.2	1.6	8.8	1.5	3.9	1.3	6.1	0.6	5.7	0.9
ETRS	4.6	0.8	4.3	0.8	2.9	0.6	1.9	0.5	3.0	0.4

B.3. Supplementary material for Section 5

B.3.1. Concentrator corner point validation

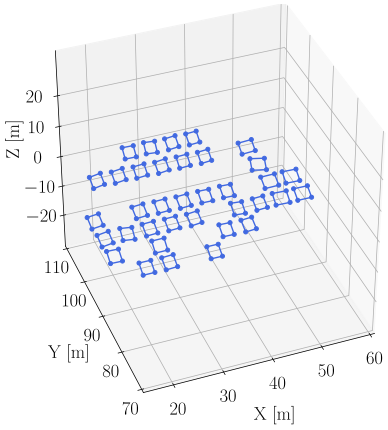
See Figs. B.3 and B.4 and Table B.1.

B.3.2. Point light reflection validation

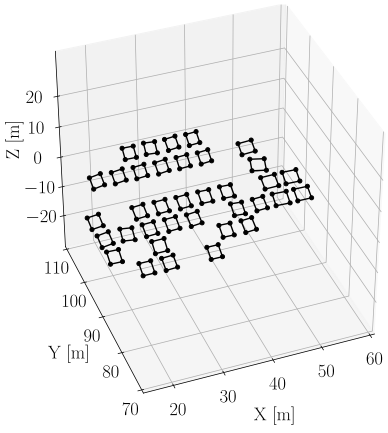
See Figs. B.5–B.7 and Tables B.2–B.4.

B.3.3. UAV Flight Dynamics Validation

See Fig. B.8.



(a) The reference point cloud is obtained by following the process depicted in Fig. 6.



(b) The simulated point cloud is obtained by initializing the solar field state with the tracking angles at the image recording time, obtained by the field operator.

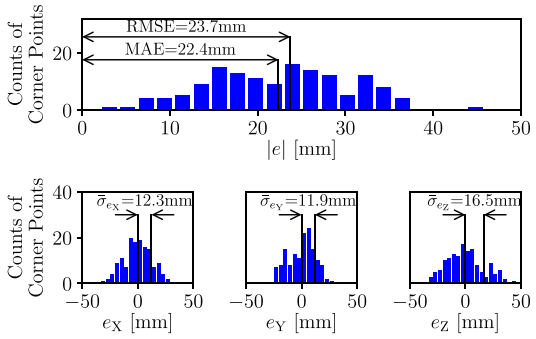
Fig. B.3. Scatter plots of the reference point cloud and simulated point cloud. A total number of $N = 36$ concentrators are evaluated, amounting to 144 concentrator corner points.

Table B.2
Component-wise RMSE, MAE, mean and standard deviation in millimeter units for the ROIS case.

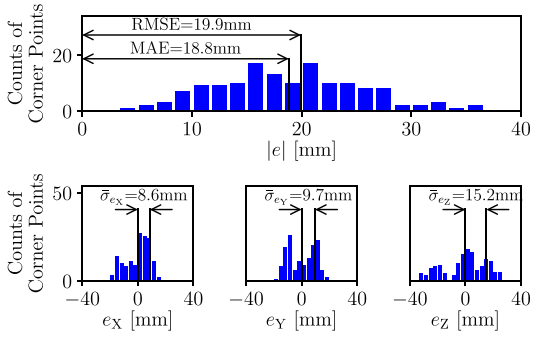
Image	RMSE [mm]		MAE [mm]		$\bar{\mu}$ [mm]		$\bar{\sigma}$ [mm]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	307	338	253	193	221	67	215	332
Image 2	254	317	196	234	−184	−130	175	290
Image 3	416	232	350	198	−347	−125	231	196
Overall	337	299	271	206	−88	−54	326	294

Table B.3
Component-wise RMSE, MAE, mean and standard deviation in millimeter units for the FOMS case.

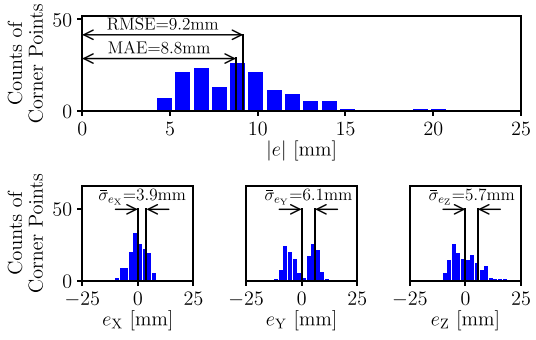
Image	RMSE [mm]		MAE [mm]		$\bar{\mu}$ [mm]		$\bar{\sigma}$ [mm]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	57	28	41	23	15	−1	55	28
Image 3	43	39	31	27	9	−17	43	36
Overall	53	31	38	24	13	−5	52	31



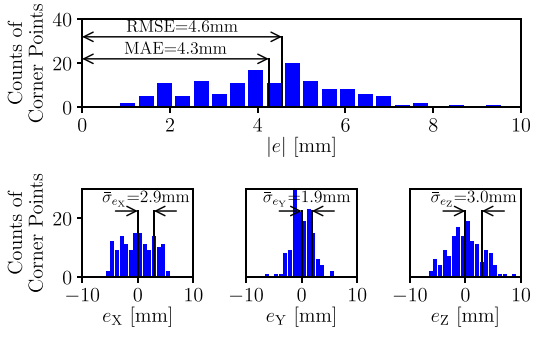
(a) Error histograms for the case of unmodified simulated points.



(b) Error histograms after the elimination of translational errors.



(c) Error histograms after the elimination of translational and rotational errors.



(d) Error histograms after the elimination of translational, rotational and scale errors.

Fig. B.4. Histograms for the error magnitude (top) and the error components e_X (bottom left), e_Y (bottom center) and e_Z (bottom right) after each error elimination step described in Section 4.3.3.

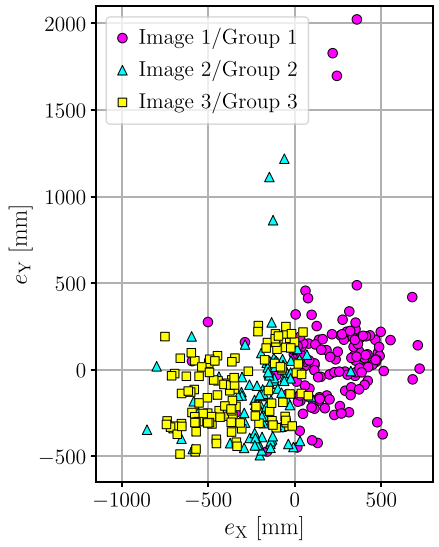


Fig. B.5. Scatter plots of e_Y over e_X in millimeter units for the ROIS case.

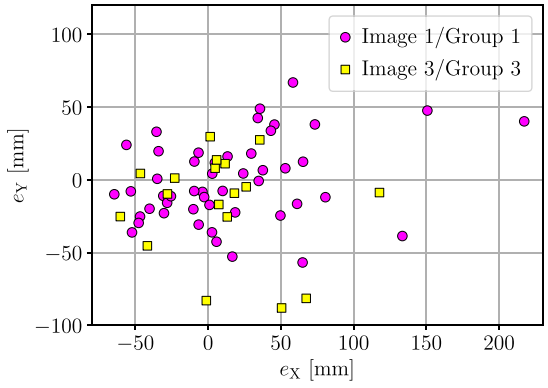


Fig. B.6. Scatter plots of e_Y over e_X in millimeter units for the FOMS case.

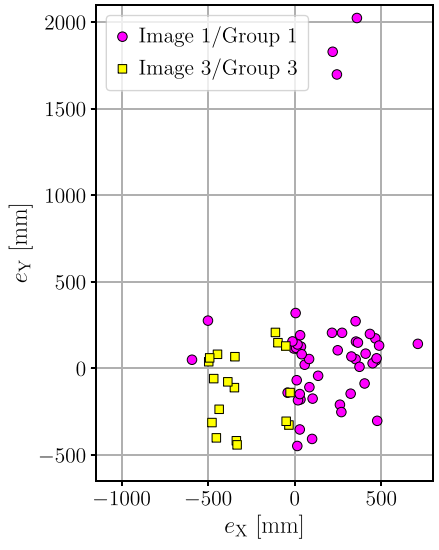


Fig. B.7. Scatter plots of e_Y over e_X in millimeter units for the ROIS case, evaluated on the FOMS dataset.

Table B.4
Component-wise RMSE, MAE, mean and standard deviation in millimeter units for the ROIS case, evaluated on the FOMS dataset.

Image	RMSE [mm]		MAE [mm]		$\bar{\mu}$ [mm]		$\bar{\sigma}$ [mm]	
	X	Y	X	Y	e_X	e_Y	e_X	e_Y
Image 1	301	492	231	258	184	125	241	481
Image 3	347	239	299	198	−299	−117	181	214
Overall	314	439	250	242	54	60	312	438

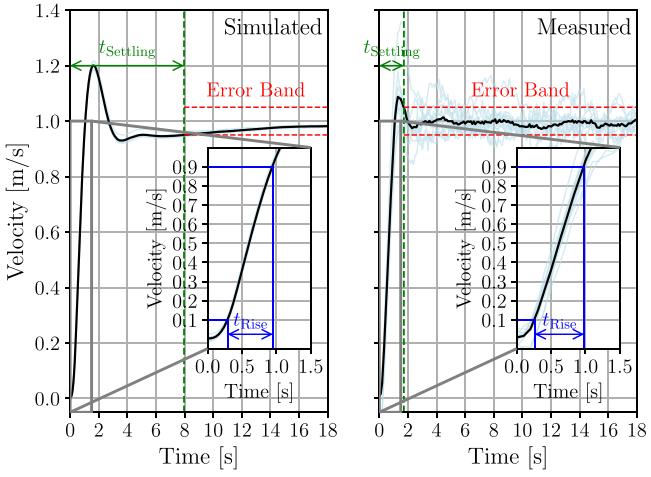


Fig. B.8. Vertical step responses for both the simulated UAV (left side) and the real UAV (right side), alongside the derived metrics. The thin light blue lines show the responses for each repetition, the black thick lines show the averaged responses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] R.A. Mitchell, G. Zhu, A non-intrusive optical (NIO) approach to characterize heliostats in utility-scale power tower plants: Methodology and in-situ validation, Sol. Energy 209 (2020) 431–445, http://dx.doi.org/10.1016/j.solener.2020.09.004.

[2] W. Jessen, M. Röger, C. Prahl, R. Pitz-Paal, A two-stage method for measuring the heliostat offset, AIP Conf. Proc. 2445 (1) (2022) 070005, http://dx.doi.org/10.1063/5.0087036.

[3] J. Yellowhair, P.A. Apostolopoulos, D.E. Small, D. Novick, M. Mann, Development of an aerial imaging system for heliostat canting assessments, AIP Conf. Proc. 2445 (1) (2022) 120024, http://dx.doi.org/10.1063/5.0087057.

[4] J.J. Krauth, C. Happich, N. Algner, R. Broda, A. Kämpgen, A. Schnerring, S. Ulmer, M. Röger, HelioPoint – A fast airborne calibration method for heliostat fields, J. Sol. Energy Eng. 146 (6) (2024) 061005, http://dx.doi.org/10.1115/1.4065868.

[5] C. Dimmig, G. Silano, K. Mcguire, C. Gabellieri, W. Hoenig, J. Moore, M. Kobilarov, Survey of simulators for aerial robots, IEEE Robot. Autom. Mag. PP (2023) http://dx.doi.org/10.1109/MRA.2024.3433171.

[6] M. Pargmann, D. Maldonado Quinto, P. Schwarzbözl, R. Pitz-Paal, High accuracy data-driven heliostat calibration and state prediction with pretrained deep neural networks, Sol. Energy 218 (2021) 48–56, http://dx.doi.org/10.1016/j.solener.2021.01.046.

[7] OpenCSP Team, OpenCSP: An Environment for Collaborative CSP Optical Technology Development. URL https://opencsp.sandia.gov.

[8] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles, 2018, pp. 621–635, http://dx.doi.org/10.1007/978-3-319-67361-5_40.

[9] Epic Games, Unreal Engine, 2019, URL https://www.unrealengine.com.

[10] M. Röger, T. Schlichting, K. Blume, Guidelines for Heliostat Testing, 2023, p. 6, http://dx.doi.org/10.1117/12.2682675.

[11] K.W. Stone, Automatic heliostat track alignment method, U.S. Patent 4,564,275, 1986, URL https://patents.google.com/patent/US4564275A.

[12] S. Ulmer, T. März, C. Prahl, W. Reinalter, B. Belhomme, Automated high resolution measurement of heliostat slope errors, Sol. Energy 85 (4) (2011) 681–687, http://dx.doi.org/10.1016/j.solener.2010.01.010, SolarPACES 2009.

[13] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi,

C. Gohlke, T.E. Oliphant, Array programming with NumPy, Nature 585 (7825) (2020) 357–362, http://dx.doi.org/10.1038/s41586-020-2649-2.

[14] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nature Methods 17 (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.

[15] G. Bradski, The OpenCV library, Dr. Dobb's J. Softw. Tools (2000).

[16] C.-W. Huang, T.-Y. Shih, On the complexity of point-in-polygon algorithms, Comput. Geosci. 23 (1) (1997) 109–118, http://dx.doi.org/10.1016/S0098-3004(96)00071-4.

[17] ArduPilot Team, ArduPilot, 2025, URL https://ardupilot.org/.

[18] Colosseum Team, Colosseum. URL https://github.com/xcloudplatform/Colosseum.

[19] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, M. Khalgui, Micro Air Vehicle Link (MAVLink) in a nutshell: A survey, IEEE Access 7 (2019) 87658–87680, http://dx.doi.org/10.1109/ACCESS.2019.2924410.

[20] T.O. Hodson, Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geosci. Model. Dev. 15 (14) (2022) 5481–5487, http://dx.doi.org/10.5194/gmd-15-5481-2022.

[21] R. Broda, A. Schnerring, D. Schnaus, M. Nieslony, J.J. Krauth, M. Röger, S. Kallio, R. Pitz-Paal, Bridging the sim2real gap: Training deep neural networks for heliostat detection with purely synthetic data, Sol. Energy 300 (2025) 113728, http://dx.doi.org/10.1016/j.solener.2025.113728.

[22] W. Kabsch, A discussion of the solution for the best rotation to relate two sets of vectors, Acta Crystallogr. Sect. A 34 (5) (1978) 827–828, http://dx.doi.org/10.1107/S0567739478001680.

[23] R. Mitchell, K. Sperber, M. Grabel, G. Zhu, A nonintrusive optical approach to characterize heliostats in utility-scale power tower plants: Camera position sensitivity analysis, J. Sol. Energy Eng. 146 (6) (2024) 061009, http://dx.doi.org/10.1115/1.4066496.

[24] E. Marchand, H. Uchiyama, F. Spindler, Pose estimation for augmented reality: A hands-on survey, IEEE Trans. Vis. Comput. Graphics 22 (12) (2016) 2633–2651, http://dx.doi.org/10.1109/TVCG.2015.2513408.

[25] F. Gross, M. Balz, Potentially confusing coordinate systems for solar tower plants, AIP Conf. Proc. 2303 (1) (2020) 030017, http://dx.doi.org/10.1063/5.0028942.

[26] Review of heliostat calibration and tracking control methods, Sol. Energy 207 (2020) 110–132, http://dx.doi.org/10.1016/j.solener.2020.06.030.

[27] D. Brown, Decentering distortion of lenses, 1966, URL https://api.semanticscholar.org/CorpusID:117271607.

[28] A.E. Conrady, Decentred lens-systems, Mon. Not. R. Astron. Soc. 79 (5) (1919) 384–390, http://dx.doi.org/10.1093/mnras/79.5.384.