Chair of
Solar Technology

The present work was submitted to the
DLR - Institute of Solar Research, Chair of Solar Technology

Entwicklung eines geometrie-agnostischen Deep-Learning-Modells zur Erkennung von Heliostaten in Solarturmkraftwerken

Development of a Geometry-Agnostic Deep Learning Model for Detection of Heliostats in Solar Power Tower Plants

Master's Thesis

presented by
Tobias Wagner
Matr. Nr. 395391

Supervisor: Univ.-Prof. Dr.-Ing. Robert Pitz-Paal

Advisor: Rafal Broda, M.Sc.

Aachen, 05.05.2025

Abstract

This thesis investigates the generalization capabilities of deep learning models for the detection of heliostats with varying mirror geometries in aerial images of solar power tower plants. A geometry-agnostic modeling strategy is developed and evaluated in three structured scenarios, each based on different configurations of training and test geometries. The proposed methodology focuses on training models for object and keypoint detection using synthetic datasets and evaluating their performance based on the AP[1] and PCK[1] metrics. The results indicate that keypoint prediction generalizes more robustly under structural variation than bounding box detection, which is more sensitive to differences in geometric scale and design. Additional comparisons with class-aware models, i.e., models that were trained and tested on the same geometry, show that the geometry-agnostic models can achieve competitive performance in keypoint detection, but consistently underperform in the bounding box accuracy. Selected models trained on a diverse set of geometries, including the test geometry, achieve up to 2 percentage points higher PCK scores than the class-aware baseline while achieving comparable AP performance within a margin of 5 percentage points. Initial experiments on real-world data demonstrate that the sim-to-real transfer remains highly challenging, with performance degrading significantly in the absence of realism-enhancing factors such as soiling or contextual scene elements.

Keywords: Concentrated Solar Power, Deep Learning, Geometry-Agnostic, Object Detection, Keypoint Detection, Sim-to-Real Transfer, Pose Estimation

[1]average precision (AP), percentage of correct keypoints (PCK)

Acknowledgments

This thesis was prepared at the Institute of Solar Research of the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V., DLR) within the department Qualification. The work was carried out at the DLR office in Almería, Spain.

This research was funded by the German Federal Ministry for the Environment and Consumer Protection through the AuSeSol-AI project (grant agreement no. 67KI21007A), based on a decision by the German Bundestag. Additionally, we gratefully acknowledge the computational and data resources provided through the joint high-performance data analytics (HPDA) project terrabyte of the German Aerospace Center (DLR) and the Leibniz Supercomputing Center (LRZ). Lastly, we extend our thanks for the use of image data from the Plataforma Solar de Almería (owned and operated by CIEMAT).

I would like to express my sincere gratitude to my supervisor, Rafal Broda, for his valuable academic and scientific guidance throughout this thesis. His expertise helped me maintain focus, uphold methodological rigor, and navigate challenging phases with clarity and confidence. I particularly appreciated his constructive feedback, his support when needed, and the freedom he gave me to develop my own ideas. His thoughtful advice and consistent encouragement have been instrumental in shaping this work.

Contents

List of Figures

List of Tables

Acronyms

AI artificial intelligence.

AP average precision.

AUC area under the curve.

BCE binary cross entropy.

CIEMAT Centre for Energy, Environmental and Technological Research.

CNN convolutional neural network.

COCO Common Objects in Context.

CSP concentrated solar power.

CSV comma separated values.

CV computer vision.

DL deep learning.

DLR German Aerospace Center.

DNN deep neural network.

ENU east-north-up.

FN false negative.

FP false positive.

GCS global coordinate system.

GPU graphics processing unit.

HelioCon Heliostat Consortium for Concentrating Solar-Thermal Power.

ICS image coordinate system.

ID identifier.

IOU	intersection over union.
ISEGS	Ivanpah Solar Electric Generating System.
LED	light emitting diode.
LFR	linear fresnel reflectors.
mAP	mean average precision.
ML	machine learning.
MSE	mean squared error.
NN	neural network.
OCS	observer coordinate system.
P	precision.
PCK	percentage of correct keypoints.
PS10	Plantar Solar 10.
PS20	Plantar Solar 20.
PSA	Plataforma Solar de Almería.
PTC	parabolic trough collectors.
R	recall.
ReLU	rectified linear unit.
ResNet	residual network.
RSS	residual sum of squares.
SPD	solar parabolic dishes.
SPT	solar power tower.
TN	true negative.
TP	true positive.
UAV	unmanned aerial vehicle.
YOLO	you only look once.

Units

cm centimeter.

km kilometer.

m meter.

mrad milliradiant.

MW megawatt.

px pixel.

1. Introduction

Solar power tower (SPT) plants generate renewable electrical energy by utilizing solar radiation, thereby contributing to the reduction of greenhouse gas emissions. They primarily consist of heliostats that track the sun's movement on two axes and reflect the incoming solar radiation onto a central receiver. In the receiver, the sun's energy is transferred to a thermodynamic cycle for electricity generation. Excess solar energy can be stored as thermal energy and converted into electricity when required. This flexibility enables the plants to enhance grid stability through demand-oriented power generation [1–3]. To ensure a high level of plant efficiency, heliostat alignment must be monitored and calibrated regularly so that the incoming solar radiation is optimally focused on the receiver [4–7].

State-of-the-art calibration methods rely on conventional image processing techniques. Typically, heliostats are calibrated sequentially in order to achieve sufficient calibration accuracy. However, this sequential procedure results in long processing times, particularly in large-scale SPT plants [6]. To address this limitation, more recent approaches use drones equipped with high-resolution cameras that can quickly capture all heliostats in a field using aerial images. The current orientation of a heliostat is then determined by identifying the heliostat via conventional edge and corner detection algorithms. While these conventional image processing methods generally provide adequate results, they are not sufficiently fast and robust, as they primarily rely on handcrafted feature engineering [8–11]. Manual calibration work is often required, which makes the process increasingly costly and time-consuming [12, 13].

The emergence of artificial intelligence (AI) and machine learning (ML) is opening up new possibilities in various fields, with image processing being a key area of application [14, 15]. Initial research has explored the use of ML-based methods for heliostat detection in images. However, existing approaches either still partly use conventional image processing methods or ground-based imagery rather than aerial images [12, 16–18]. To the best of my knowledge, the work by Broda et al. [19] at the German Aerospace Center (DLR) Institute for Solar Research (Almería, Spain) is the only exception found in the existing literature. This work serves as the foundation for this master's thesis. The results are utilized to identify existing research gaps as well as to motivate the objectives of this study in the following.

Broda et al. [19] train a neural network (NN) to detect heliostats in aerial images. The model works exclusively with synthetically generated image data[2] during training to overcome the challenge of limited training data and to have improved control over the variables affecting

[2]Synthetic image data generated with the open-source 3D modeling software Blender [20], BlenderProc [21] and custom extensions [13].

the model performance. Figure 1.1 shows the model predictions of the trained model for a randomly selected test image. The model is class-aware, meaning it was trained on the same mirror geometry used in testing. The train set size is 20,000 images. To ensure clarity and distinguishability, the corresponding mirror geometry will be referred to as geometry A in the following. As it is common in object detection [22, 23], the position of the heliostats in the image is predicted using their bounding box (in red). The positions of the corner points (referred to as keypoints in the following) of the mirror facets are predicted using dots (in green). By visual inspection, all heliostats are detected by the model in this test image. The percentage of correctly predicted corner points is also high. Exceptions primarily include heliostats that are located far from the camera.

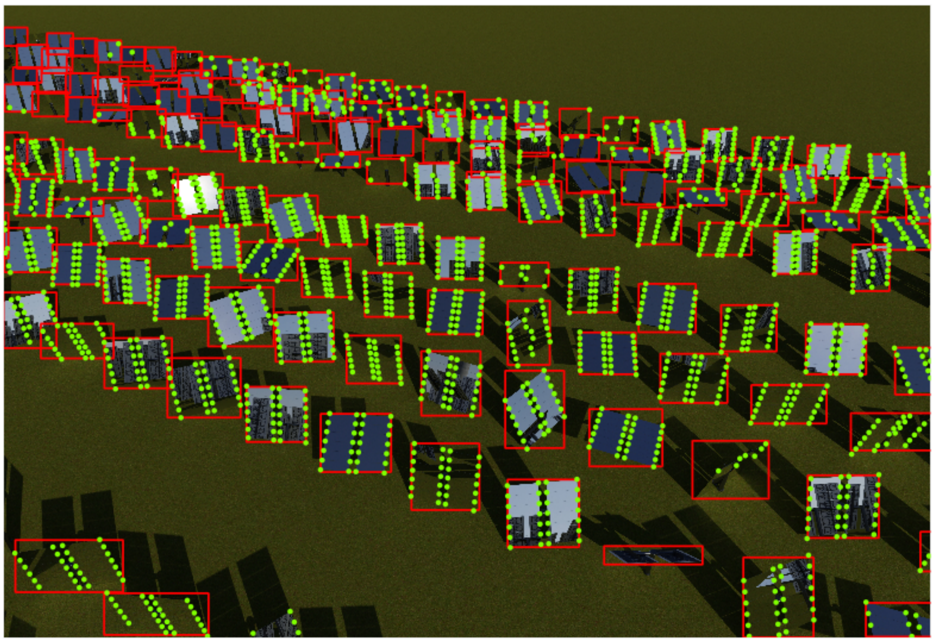


Figure 1.1.: Bounding box and keypoint predictions of a class-aware model at a model confidence score of 0.5. Model trained on 20,000 synthetic images and tested on seen geometry A.

For a mirror geometry not seen in training, referred to as geometry B in the following, the model predictions for another test image are shown in Fig. 1.2. For this image, it turns out that the model can only recognize about half of all heliostats. This detection rate indicates, that the model cannot directly transfer the learned patterns from geometry A to geometry B. To improve the model's performance on geometry B, Broda et al. [19] fine-tuned the existing model using an additional training set of 200 images from geometry B. The model predictions of the fine-tuned model, using the same camera orientation as in Fig. 1.2, are shown in Fig. 1.3

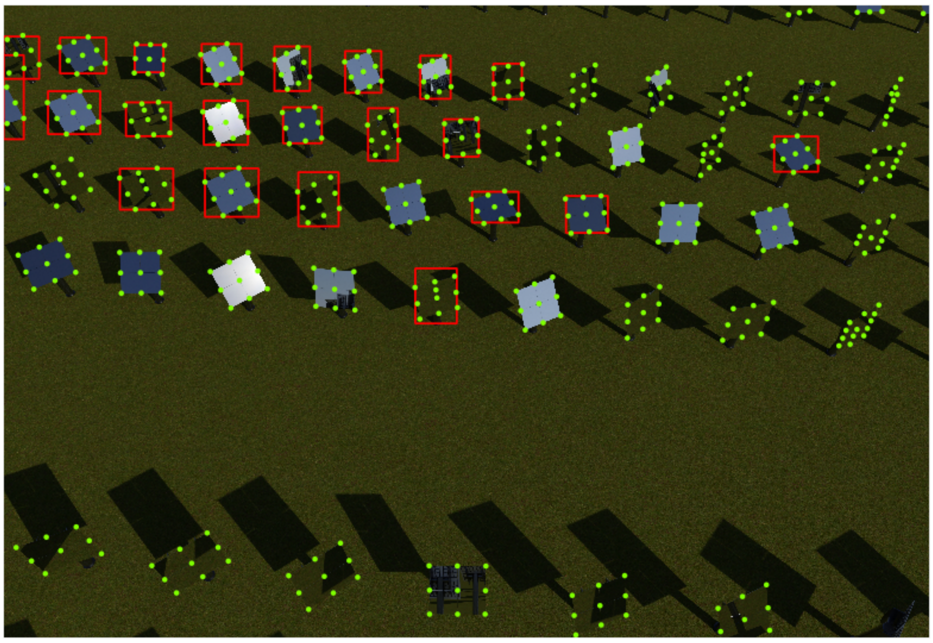


Figure 1.2.: Bounding box and keypoint predictions of a class-aware model at a model confidence score of 0.5. Model trained on 20,000 synthetic images and tested on unseen geometry B.

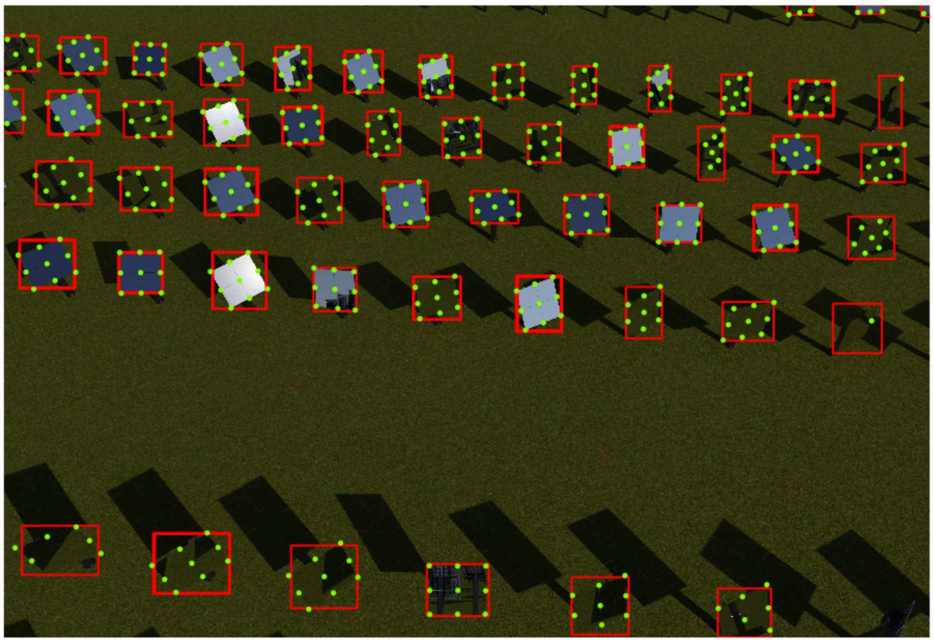


Figure 1.3.: Bounding box and keypoint predictions of a fine-tuned model at a model confidence score of 0.5. Model trained on additional 200 synthetic images and tested on fine-tuned geometry B.

Retraining the existing model with an additional reduced train set improves the model predictions significantly. This result indicates that fine-tuning constitutes a good strategy to enhance the model's adaptability to different geometries. However, testing on a third geometry, referred to as geometry C in the following, shows that the model's generalizability to further unseen geometries remains limited. The model predictions for an image showing heliostats with this geometry are shown in Fig. 1.4. Notably, a considerable number of heliostats are either not detected or are assigned by multiple bounding boxes. Moreover, fewer than half of all keypoints are identified successfully. In the current configuration, the model developed by Broda

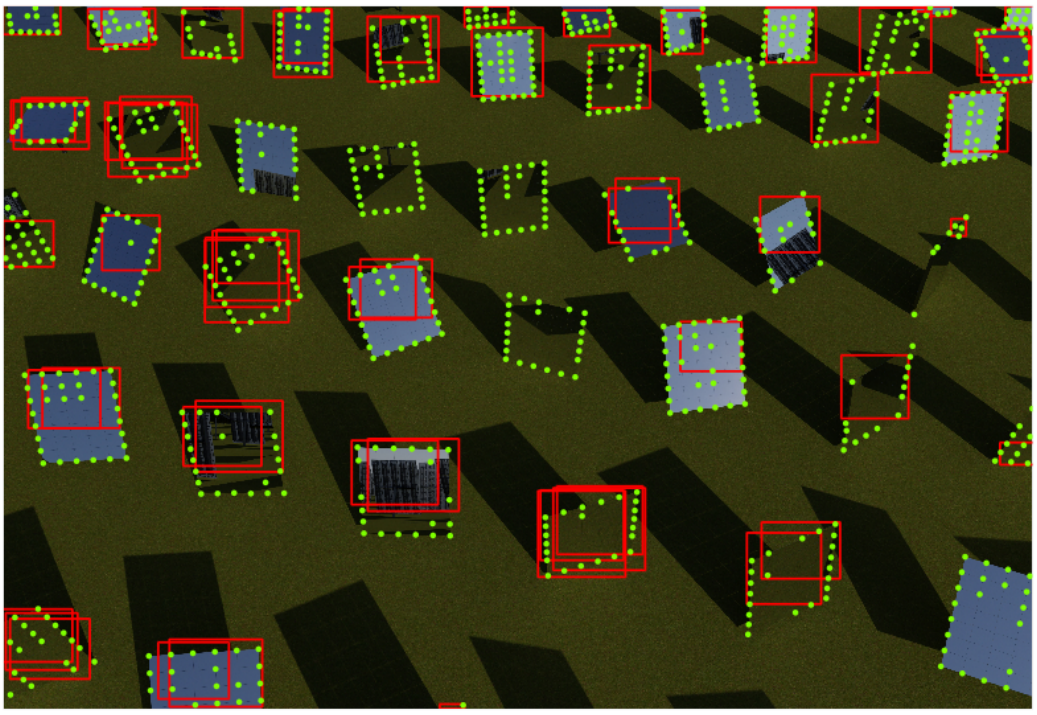


Figure 1.4.: Bounding box and keypoint predictions of a fine-tuned model at a model confidence score of 0.5. Model trained on additional 200 synthetic images and tested on an unseen geometry C.

et al. [19] is therefore only applicable to geometries on which it was trained. This finding implies a significant limitation for the application range as the heliostat geometry can vary with each plant [24]. Fine-tuning can improve the model performance for selected geometries. However, this process means additional work through retraining and also implies that the model would have to be tuned again for each additional geometry. Moreover, retraining requires the generation of new synthetic training data, which is time-consuming and represents a significant bottleneck in the tuning process.

Building on the work of Broda et al. [19], this master's thesis is therefore dedicated to developing a geometry-agnostic deep learning (DL) model that can detect heliostats of different geometries on aerial images without additional training. This model would then be capable of detecting heliostats on all types of SPT systems.

2. Fundamentals and Related Work

This chapter is subdivided into two main sections. Section one covers ML, including DL as its subarea. Section two considers technologies in the field of concentrated solar power (CSP) and focuses particularly on SPT plants. In both sections, fundamental terms and concepts are explained, and information is provided covering the current state-of-the-art.

2.1. Machine Learning

The following section begins by looking at the mathematical principles and techniques in the context of ML. Then, NNs are considered as a sub-area of ML. Convolutional neural networks (CNNs) are a special type of NNs and are therefore considered separately. Finally, state-of-the-art networks and their applications are reviewed.

2.1.1. Fundamental Terminology and Concepts

ML enables technical systems to learn from data and use this knowledge to make predictions for new, unknown data. The machine detects patterns, trends or key features in the provided data during the learning process without being explicitly programmed [14, 15]. ML techniques are commonly divided into four types: supervised, unsupervised, semi-supervised and reinforcement learning. In supervised learning, a function is learned by mapping given input data to corresponding output data, known as labels. The entirety of the data used in the learning phase is referred to as ground truth. This approach aims to learn a function that can generate output data for new input data that was not used during the learning process. Typical supervised learning tasks are regression and classification. Regression is a method for predicting continuous values, whereas classification assigns data points to discrete classes. Unsupervised learning deals with unlabeled data, meaning there is no output data for given input data during the learning phase. Unsupervised learning aims to detect hidden underlying patterns or groups, like in a clustering process [14, 25]. Semi-supervised learning and reinforcement learning are not considered further due to their lack of relevance to this thesis.

The key components of ML are a model, a strategy and an algorithm [26]. A linear regression with one input parameter is used as an example to illustrate these key components (Eq. 2.1). In this case, the model is a linear function $f(x)$. The function consists of two parameters, the weight w and the bias b. For a given input x, the function calculates a model output $f(x)$ based on the estimated model parameters [15, 27].

$$f(x) = wx + b \tag{2.1}$$

An optimization strategy is defined to determine which choice of the model parameters w and b is optimal. The corresponding objective is the minimization of a loss function, such as the residual sum of squares (RSS), also called l2 loss (Eq. 2.2).[3]

$$\min RSS = \min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.2}$$

The RSS sums over the squared differences between the ground truth labels y_i and the model predictions $\hat{y}_i$ for a given input value x_i. The number of data points is n. Using the squared loss ensures that deviations with different signs are not truncated. Therefore, a good model is one with a small RSS. It is convenient to multiply the RSS by the factor 0.5 to neutralize the factor of two in the derivative of the RSS. Closely related to the RSS is the mean squared error (MSE), which averages the RSS over the number of data points n [15, 27].
An algorithm is needed to compute the minimum of the loss function. A general approach in ML is using gradient descent [25, 28]. Gradient descent changes the parameters to be estimated iteratively based on the gradient of the loss function. The gradient for the linear regression problem with one input parameter is shown in Eq. 2.3. $\frac{\partial L}{\partial w}$ resp. $\frac{\partial L}{\partial b}$ is the partial derivative of the loss function L with respect to w resp. b.

$$\nabla L(w, b) = \begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix} \tag{2.3}$$

By definition, the gradient of the function $\nabla L(w, b)$ is a vector that points toward the steepest increase of the function at a given point (w, b). Changing the parameters of the current iteration t in the direction of the negative gradient aims to bring the parameter values closer to their minimum in the next iteration $t + 1$ (Eq. 2.4 and 2.5). The decisive factor here is a good choice of the learning rate η, which determines the step size with which the minimum is approached. The iterative optimization stops, for example, when it converges to a minimum or after a certain number of steps [27, 29, 30].

$$w_{t+1} := w_t - \eta \frac{\partial L}{\partial w} \tag{2.4}$$

$$b_{t+1} := b_t - \eta \frac{\partial L}{\partial b} \tag{2.5}$$

Optimizing the model parameters is the most crucial part of building a ML model. However, the typical ML workflow consists of four steps [15]: dataset collection, data pre-processing, model training (the learning phase) and model evaluation. The workflow is described in the following, with reference to Fig. 2.1.

[3] A detailed description of several state-of-the-art loss functions can be found in Appendix A.1.

The collected data is the fundamental factor for the effectiveness and efficiency of a ML model. Therefore, the data should be representative, relevant in terms of the measured features, of high quality and sufficient in quantity. However, real-world data is often unorganized and cannot be directly utilized in the subsequent steps. Before proceeding, the data needs to be pre-processed. Typical tasks include handling missing data, standardizing data features and eliminating outliers [14]. Model training comprises the mathematical operations previously discussed for optimizing the model parameters. Learning itself is realized through the combination of a ML model, an optimization strategy and the associated algorithm. In general, different model types t and variants v are trained, varying in their mathematical approaches or hyperparameters. For example, a regression model can be linear but also quadratic. The learning rate, an example of a hyperparameter, is also adjustable as part of the gradient descent optimization [15, 27]. For model training, only a part of the collected data is used. The data is divided into a train set and a test set, e.g., in an 80:20 ratio. A common strategy is to further divide the training set into a reduced training set and a validation set, allowing for the monitoring of model performance during training [15].

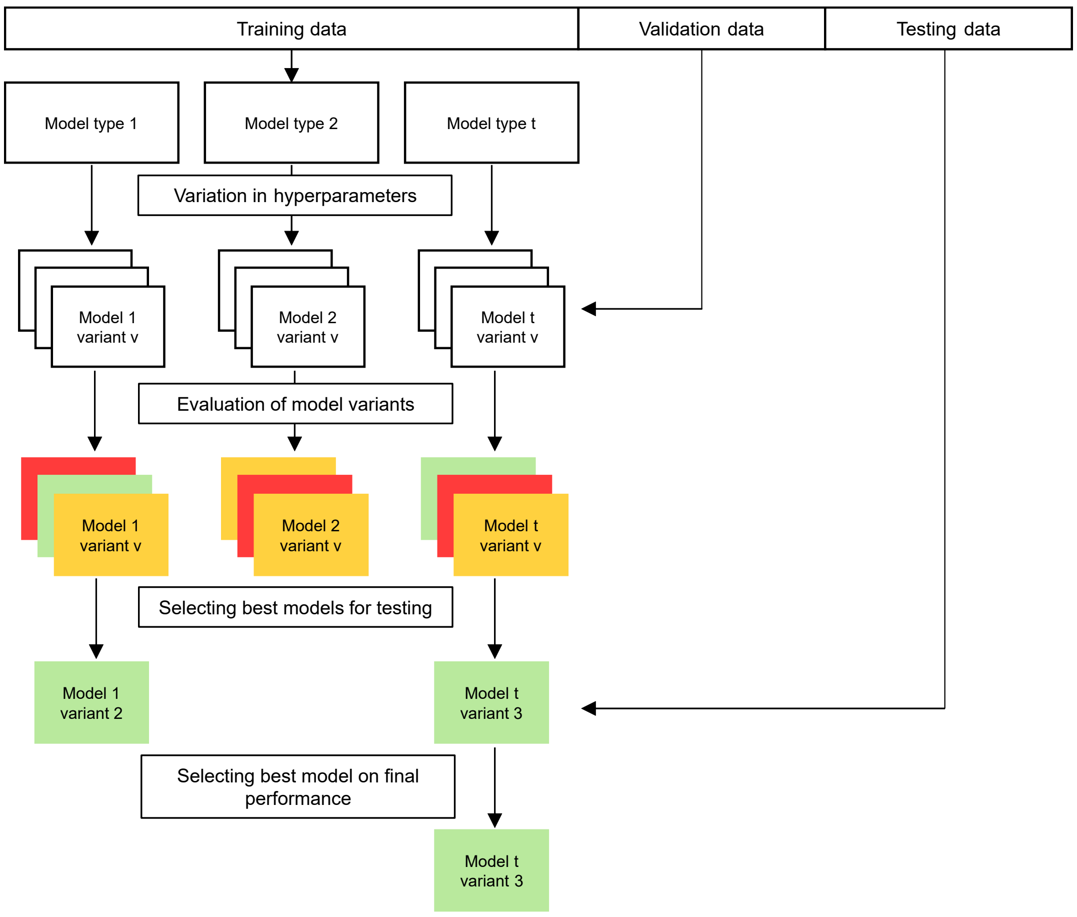


Figure 2.1.: Big picture: model training, validation, testing and selection [15, 31].

The monitoring provides a basis for excluding model variants or adjusting hyperparameters at an early stage. A final evaluation of the model's performance is conducted using the test set. This evaluation is necessary because a good model performance achieved during training and validation is not a sufficient indicator of its performance on unseen data. For example, a model with a large number of parameters and precisely adjusted hyperparameters can tend to approximate the given data (small training loss) and not learn the underlying function (high testing loss). The model is overfitted [14, 15, 32].

Having a sufficiently large quantity of data available for a ML project is often challenging. Reserving data aside for model evaluation can be critical to the remaining train set size. A solution to this problem is the use of resampling methods, where existing data is utilized in multiple iterations or to generate additional data artificially. One method is k-fold cross-validation, which is explained in the following [15, 28]. After splitting the data into a train set and a test set, the resulting train set is divided into K parts. In every iteration $k \in 1, \ldots, K$, the model is trained on all the folds except fold k, which is used as a validation set. The MSE is calculated, which averages the squared errors calculated in the k iterations.

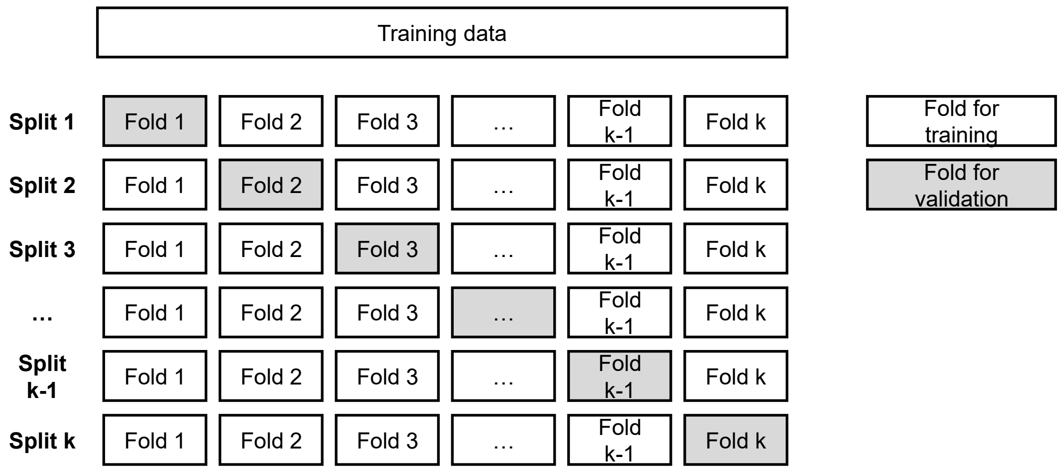


Figure 2.2.: k-fold cross-validation, own presentation based on [15].

After introducing fundamental concepts and terminology in the field of ML, the following chapter will focus on DL, which is a major subfield of ML.

2.1.2. Deep Learning

DL is a subfield of ML and is based on NNs. A NN is inspired by the information processing structure in the human brain. A neuron receives information, processes and forwards it to other neurons via synapses. Mathematically, this process is described as follows: A NN is organized in layers. Information is given to the network at the input layer. The corresponding neurons process the incoming information and forward it to the neurons in the next layer. The information is successively processed by the intermediate layers and finally output through the output layer. Due to the unidirectional flow of information from the input to the output layer, this type of network is also referred to as a feedforward network. Except for the input and output layer, an outside observer cannot observe the states and calculations. This is why these layers are referred to as hidden layers. Networks with many hidden layers are called deep neural networks (DNNs), and the study and application of these networks are collectively referred to as DL [14, 15, 27, 29, 32]. An exemplary structure of a small NN with two hidden layers is shown on the left in Fig. 2.3. The right side of Fig. 2.3 visualizes the information processing of an artificial neuron. In focus is a neuron of the first hidden layer, which receives information x_i weighted with w_i from all three neurons $i \in \{1, 2, 3\}$ of the input layer. The neuron sums up the incoming information, processes it with an activation function and forwards the output to the next neuron. The activation function used is the rectified linear unit (ReLU). ReLU has established itself as state-of-the-art and is defined as $ReLU(x) = max(0, x)$ [15, 27]. While applying other activation functions is possible, a closer examination of this subtopic is not addressed in this thesis.

The strength of a NN lies in the number of hidden layers, respectively, its depth. A neuron processes weighted input data, applies an activation function to reduce the input to a single value, and passes the information to a downstream neuron. Note that activation functions of hidden layers are in general non-linear. The composition of these non-linear functions enables the mathematical modeling of complex non-linear relationships. Thus, the deep architecture provides the capacity to learn highly complex data structures [29, 33].

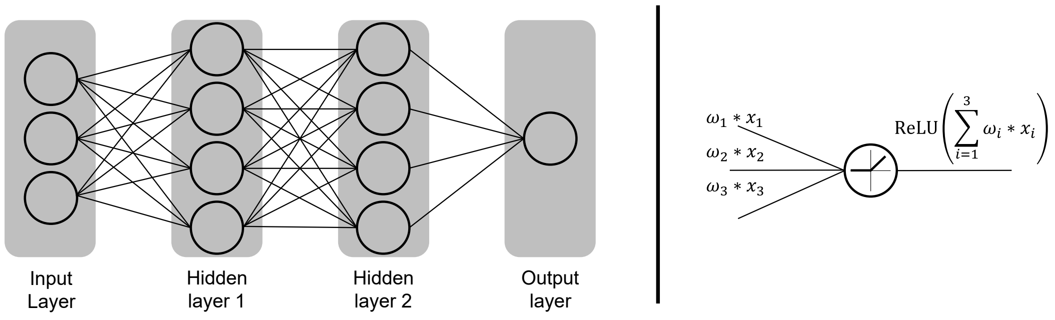


Figure 2.3.: Architecture of a NN (**left**) and structure of an artificial neuron with weighted input (**right**), based on [15].

Training a DNN is fitting its weights to minimize a certain optimization function by applying a training algorithm. For regression tasks, the l2 loss function and the gradient descent algorithm are also applicable within a DNN [28, 29]. However, due to the chaining of network layers and their associated weights, training a DNN becomes significantly more challenging. This difficulty is explained in the following. The change in the output of a function $f(x)$ is based on its derivative $\frac{df}{dx}$. If the input is first transformed by a function $z(x)$ before being passed into f, the total derivative is calculated by applying the chain rule (Eq. 2.6).

$$\frac{df}{dx} = \frac{dz}{dx} * \frac{df}{dz} \tag{2.6}$$

The optimization of the weights in a NN underlies this fundamental principle. During training, the gradient of the loss function with respect to the weights of any layer is propagated backwards from the output layer. This procedure is known as backpropagation. However, the complexity of backpropagation increases significantly with the depth of the NN because the gradient descent algorithm must compute derivatives of multiple non-linear transformations. One resulting problem is the vanishing gradient, where the gradients in the first layers of a DNN can have very small values, making it difficult to train the corresponding weights. The vanishing gradient is caused by the use of activation functions with a derivative in the range $[0, 1]$ in general, and their multiplication by the chain rule [27, 29, 30].

Although the small NN in Fig. 2.3 consists of only 12 neurons, 32 weights are required due to its full connectivity. A large NN can have more than 10^{12} parameters [27], which must be optimized during model training. Therefore, a large amount of training data and massive computing power are necessary. A high volume of training data enables the training algorithms to discover more hidden data features and dependencies, thereby improving the calculations of the weights [27, 29, 32, 34]. To train a DNN in a reasonable time, graphics processing units (GPUs) are used to provide the necessary computing power [27, 29]. GPUs are specialized hardware components originally developed as graphics processing hardware for the video gaming market. Their strength lies above all in their ability to parallelize tasks. This property is utilized in the calculation of the gradients, as the calculation of a gradient for a weight of a certain neuron connection can be carried out independently of other gradient calculations [27, 35].

After introducing fundamental concepts and terminology in the field of DL in this chapter, the next one focuses on CNNs as a special DNN architecture.

2.1.3. Convolutional Neural Networks

This section introduces the convolutional operation, its application in CNNs and particularly the strength of CNNs in image processing. Therefore, a short introduction to the structure of image data is necessary.

An image can be described by the number of pixels in both the horizontal and vertical directions, along with their corresponding color values. In a grayscale image, the color of a pixel is determined by a numerical value ranging from 0 to 255 (for an 8-bit image). A value of 0 corresponds to the color black, and a value of 255 corresponds to the color white. Color images, in turn, consist of three layered channels - red, green, and blue. Three values are then assigned to each pixel, corresponding to the intensity in the red, green and blue channels [27, 36].

Now, a color image with size of $10^3 \times 10^3$ pixels is considered that is to be processed by a NN. Each pixel is handled as a separate input. This example results in 10^6 neurons in the input layer. For a fully connected feed-forward NN and, e.g., 10^3 neurons in the first hidden layer, such a network would have to learn $3 * 10^9$ weights only in the first layer. The factor three results from the three color channels. If this example were to be taken further, a NN would be created whose computing times and training data requirements would exceed a reasonable level [27]. CNNs, however, make use of the 2D grid-like structure of each channel of the input data [14]. Moreover, they take advantage of the fact that neighboring pixels have a certain correlation in their color when forming patterns and objects. This knowledge is used to process image data efficiently [27–29].

To illustrate the principle of the convolutional operation, a 5×5-pixel grid is convolved with a 3×3-pixel kernel as shown in Fig. 2.4. The kernel can be considered a small, two-dimensional grid, where each grid entry serves as a weight. The kernel takes as input only a rectangular region, called a receptive field, of its own size from the image.

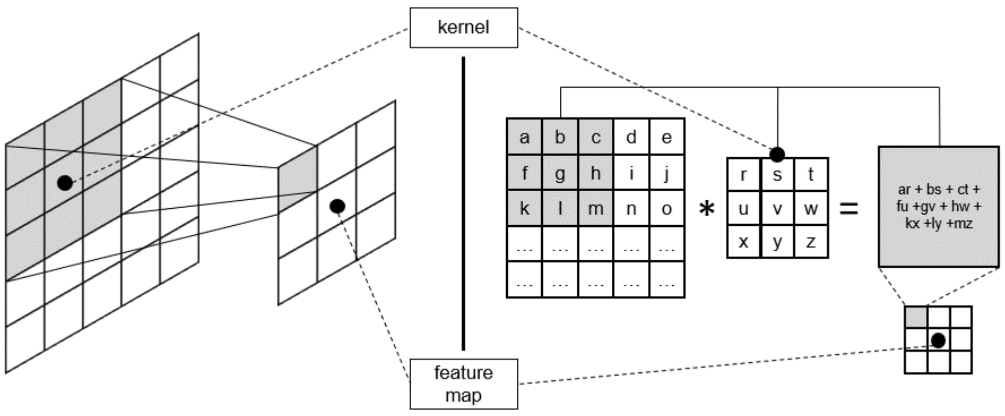


Figure 2.4.: Convolutional operation, visually **left**, mathematically **right**, own presentation based on [27].

In the convolutional operation, each input pixel of the receptive field is multiplied by the corresponding weight of the kernel, and the sum of these products is calculated. Before being passed to the next layer, the output is also transformed by a non-linear function, such as the ReLU [37]. In the following, the convolutional kernel slides along the input image until all entries from the input grid are convolved. Notice that the kernel uses the same weights during every convolutional operation. The weight sharing significantly reduces the number of trainable weights. By sliding the kernel over the input image, a transformed and compressed representation of the original image is created [27, 28].

The result of the convolutional operation is known as a feature map. The name originates from the fact that it is possible to emphasize specific image elements (features) by selecting the appropriate kernel. The feature selection is illustrated in Fig. 2.5. A 4×4 input feature map is taken, consisting of a black area on the left side and a white area on the right side. By choosing so, the input feature map has a vertical edge in the transition from the third to the fourth column, counted from left to right. A typical kernel for detecting vertical edges is a 3×3 kernel with a zero value in its middle column and values of minus one in its left column and values of one in the right column. Sliding this kernel over the input feature map calculates an output feature map of size 2×2. The kernel is visualized by a moving red rectangle that starts in the input's top-left corner. The kernel then slides to the right, then slides down left and finally to the bottom right corner. The output feature map is a compressed version of the input, emphasizing the vertical edge [27].

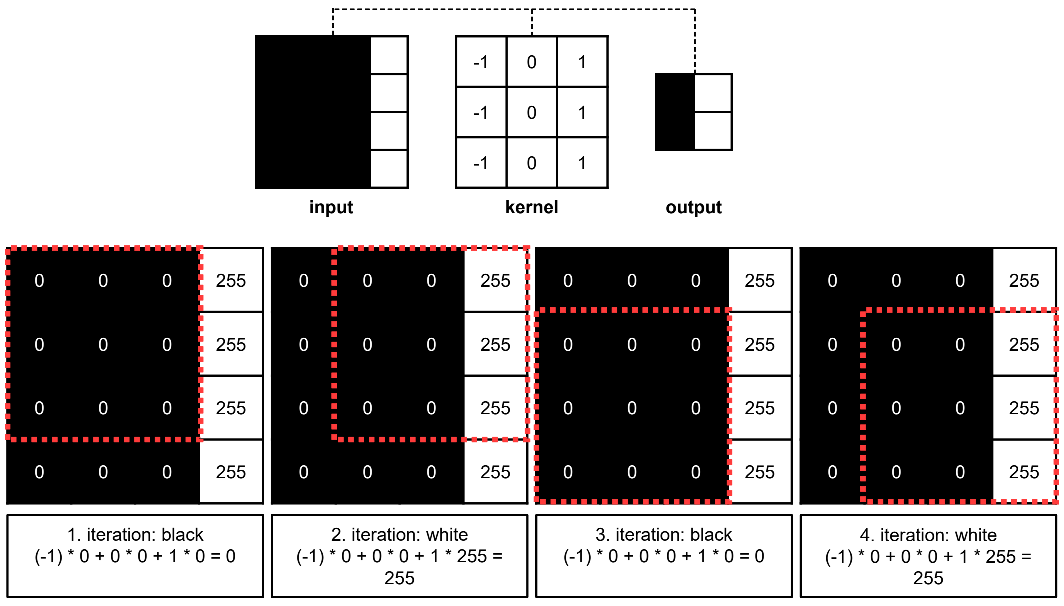


Figure 2.5.: Filtering vertical edges in a convolutional layer, own presentation based on [27].

The kernel shown in Fig. 2.5 is configured to detect vertical edges. Comparably, a 3 $\times$ 3 kernel can be defined for detecting horizontal edges. Obviously, edges in an input feature map do not have to be aligned horizontally or vertically. Implementing a kernel for every possible edge orientation is time-consuming and computationally expensive. However, traditional image processing techniques rely on these handcrafted features, thus making them less efficient and effective (see Appendix A.2) [33, 38]. This is where the true strength of CNNs lies: the convolutional kernels are not handcrafted but instead learned during the training process [27, 29].

Generally, several kernels convolve an input to detect all relevant features, resulting in multiple output feature maps. The feature maps collectively form the output of a convolutional layer. As a standard NN, a CNN consists of multiple of such layers. Each of the following layers has its own kernels and gets the output feature maps of the previous layer as input. The subsequent layers are trained to detect features, which are combinations of the features of the earlier layers. As the number of layers increases, so does the compression of the input data. At a certain point, a person can no longer extract useful information from the feature maps [27]. Local combinations of corners and edges form motifs, which are assembled into parts, and these, in turn, form larger objects. A hierarchy becomes recognizable in which convolutional layers pass on detected features in aggregated form to the subsequent layer. This concept is known as hierarchical feature extraction [25, 27, 30, 39].
CNNs were first applied with a notable performance by LeCun et al. (1998) for the task of recognizing handwritten digits [37]. The developed model architecture, known as LeNet-5, is shown in Fig. 2.6.

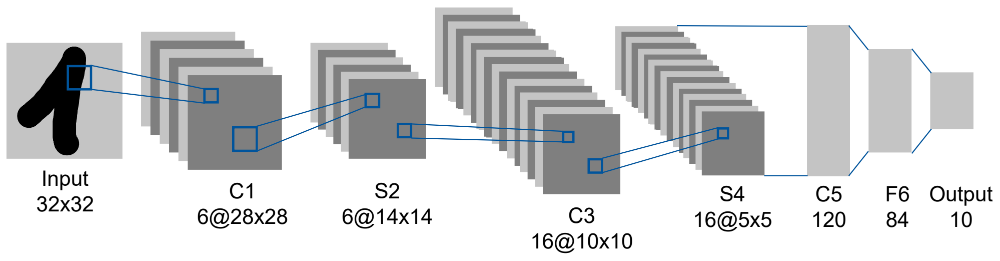


Figure 2.6.: LeNet-5 architecture, own presentation based on [37].

An input image of size 32×32 pixels is convolved with six kernels, resulting in six feature maps (also called channels) in the first convolutional layer C1. This layer is followed by a pooling layer S2, the second common building block in a CNN. The pooling operation summarizes the response of a receptive field into a single value. Common pooling operations are *max pooling* and *mean pooling*. Max pooling takes the maximum pixel value of a receptive field, and mean pooling takes the mean of the pixel values within a receptive field [33]. Thus, the pooling operation makes the output of a convolutional layer more robust, as variations in the input are reduced and only the most important information is processed in compact form.

The pooling layer is followed by another pair of a convolutional layer C3 and a pooling layer S4, both with 16 kernels. One channel in the convolutional layer C3 is generated by a kernel simultaneously sliding over all channels of the previous layer. Layer C5 is a convolutional layer with the special characteristic that the kernel size corresponds to the size of the channels in the pooling layer S4 (5×5). This means that each channel in the convolutional layer has a size of 1×1 and thus represents a simple neuron. Each neuron in layer C5 is connected to every neuron in layer S4. This type of connection is called a full connection, aiming to combine features learned in the previous layers. Layer C5 consists of 120 neurons, meaning the channels in layer S4 were convolved with 120 kernels. Layer F6 consists of 84 neurons that are fully connected to the neurons in layer C5. The last layer is the output layer, consisting of ten neurons fully connected to the neurons in layer six. Each of the ten neurons represents a digit between zero and nine. The output of each neuron represents the probability that the input digit is the digit associated with that neuron [37].

In this chapter, the convolutional operation, its application in CNNs and particularly the strength of CNNs in image processing were introduced. The LeNet-5 architecture was explained, which was a groundbreaking innovation at the time and the starting point for further developments. The following chapter will review selected model architectures with relevance for this thesis.

2.1.4. Review of Relevant Model Architectures

Since the availability of GPUs for high computing power and large amounts of labeled training data, intensive research has been conducted in the field of DL [33]. This research has led to the development of new and even more advanced network architectures [15, 32, 33, 40]. Models relevant to this thesis are reviewed in the following section.

The **U-net** was developed by Ronneberger et al. [41] for biomedical image segmentation purposes. Image segmentation refers to assigning a class label to every pixel on an image. The U-net architecture consists of a contracting path, called encoder, and an expanding path, called decoder. The architecture of the encoder is a typical CNN with alternating convolutional and pooling layers. The decoder consists of alternating upsampling and convolutional layers, aiming to reconstruct the encoded image. Upsampling can be understood as the counterpart to the pooling operation, where the image size is increased [27]. By first contracting and then expanding an image, the encoder-decoder learns to extract the relevant features of an image. During contraction, however, high-resolution features cannot be restored with sufficient accuracy, as would be necessary for instance segmentation. For this reason, feature maps with high resolution from the contracting path are combined via skip connections with the upsampled output at each stage of the expanding path. The successive convolutional layer can, therefore, generate more precise output [41].

The **residual network (ResNet)** was developed by He et al. [42] to tackle the problem of training very deep NNs with a large number of hidden layers. As discussed in Chapter 2.1.2, the vanishing gradient problem makes learning weights in the early layers of a DNN difficult. To overcome this problem, ResNet uses the approach of deep residual learning. The concept is illustrated in Fig. 2.7.

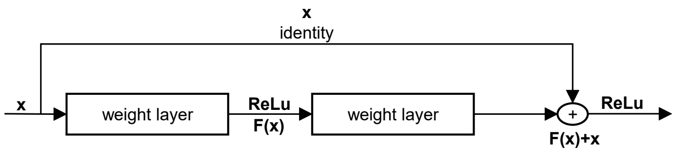


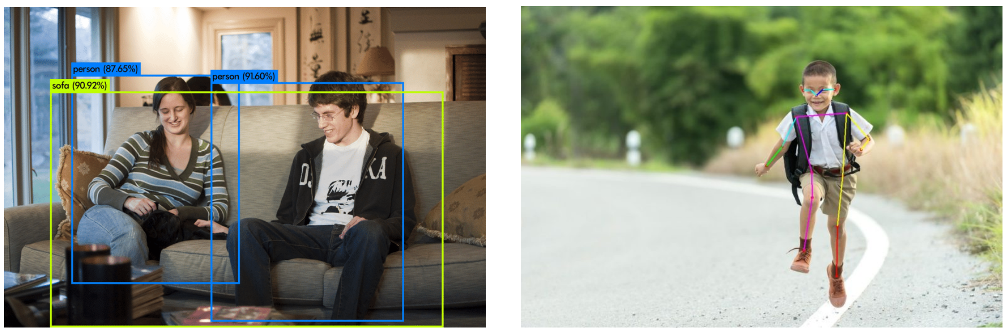
Figure 2.7.: Structure of a residual block in ResNet, own presentation based on [42].

A DNN should be able to learn deep features while also skipping unnecessary layers. In other words, allowing an input to pass through layers without transformation should be possible if no meaningful changes are needed. In this case, the concatenation of the non-linear transformation functions of several concatenated layers has to form an identity mapping $H(x) = x$. Here, x is the input of the first layer to be skipped, and $H(x)$ represents the desired concatenation of the non-linear functions of the skipped layers. He et al. [42] assume that learning $F(x) = H(x) = x$ is difficult, where $F(x)$ is the learned function. That is why the identity x is directly passed to the end of the skipped layers via skip connections, which allows $F(x)$ to be set to zero. Formally, $F(x) = H(x) - x = 0$, which means that $F(x)$ is set to the residuum between $H(x)$ and x. Learning the residuum is assumed to be easier to train. Since the gradient of the skipped layers increases by one (the derivative of the identity), model training becomes stable even when $\frac{dF}{dx} = F(x) = 0$. Thus, the problem of the vanishing gradient is also prevented. Due to the increasing number of layers, ResNets are structured in stages, which serve the purpose of clarity and an overview of the feature hierarchy. The core of ResNet34 (34 layers) described in [42] consists of four stages containing several residual blocks. The kernel size remains the same for every stage, but the number of kernels increases to extract more features in depth. Using the ResNet architecture, it was possible to train a DNN with 152 layers efficiently, showing outstanding performance and outperforming current state-of-the-art models [42].

The **R-CNN** was developed by Girshick et al. [22]. It is a DNN dedicated to the task of object detection. Object detection refers to estimating the location and type of objects in an image [33, 43]. The location of an object is predicted by determining its bounding box, a rectangle that closely fits the object's boundaries [27]. The model takes an image as input and extracts proposals from around 2000 object regions. Subsequently, a large CNN is trained to learn the relevant features to make different objects distinguishable. Based on the output of the CNN, a ML-based classifier generates class predictions for each region. The R-CNN achieved significantly better prediction results than other models available at the time of its introduction [22]. However, the bottleneck of the R-CNN is a high processing time due to the large number of region proposals. For this reason, the Fast R-CNN [44] and Faster R-CNN [45] were developed in the following years. A further development of Faster R-CNN is Mask R-CNN, which can provide pixel-accurate masks for the predicted objects. This task is known as object instance segmentation. [46]

The **you only look once (YOLO)** model architecture was developed by Redmon et al. [23] and is also designed for object detection tasks. In comparison to the R-CNN architecture and its subsequent developments, YOLO does not require a separate region proposal network but only employs a single CNN, resulting in a significant increase in processing speed. The basic idea of the YOLO model is to divide the image into a fixed grid (e.g., 7×7 [23]). In each cell of the image, objects are detected via their bounding boxes and assigned to a class. YOLO initially had problems identifying small objects due to the coarse grid size. However, this problem was reduced by further developments [47, 48].

Another major application for DNNs is keypoint detection. A keypoint is defined as a „local distinctive region" [49], and keypoint detection refers to the task of finding these regions in an image [49]. For instance, keypoint detection is used in human pose estimation. Here, the joints of the human body are considered keypoints for predicting human posture [50]. Figure 2.8 shows an example of object detection performed by YOLO on the left and of human pose estimation performed by R-CNN on the right. The YOLO model predicts bounding boxes and a probability for a classification. The R-CNN predicts the human pose by detecting the joints as keypoints.



(a) Example of object detection performed by YOLO. Image from [51].

(b) Example of keypoint detection and human pose estimation performed by R-CNN. Image from [52].

Figure 2.8.: Object detection and human pose estimation examples.

The **CenterNet**, developed by Zhou et al. [53], combines the approaches of object and keypoint detection. Instead of identifying objects by directly predicting the location and size of the corresponding bounding box, the center point of the bounding box is predicted in first stage. The model returns a heatmap where each peak corresponds to the location of an object center in the input image. The bounding box size is then regressed from the predicted bounding box center. Additional sub-pixel regression is necessary due to the different resolutions of the input image and output heatmap. Bounding box centers and keypoints are represented as 2D Gaussian heatmaps rather than single-pixel labels. This approach improves the training stability because the model can even learn from nearby pixels, providing smoother gradients and better localization performance [53].

As mentioned at the beginning of Chapter 2.2, a sufficiently large amount of data is required for effective training of DNNs and CNNs. Publicly accessible databases provide a large volume of labeled data. ImageNet is a well-known example of such a resource. It is an image database containing over 14 million hand-annotated images, which have been classified into more than 20,000 categories [54, 55]. Each image was hand-annotated (labeled) with the correct object class and bounding box position for every object it contains [56].

Note, that conventional object detection is class-aware, meaning models are applied to detect objects from classes seen during training. „However, these models do not generalize well to unseen object-types." [43] Class-agnostic models are not limited to the object types seen during training [43]. In comparison to class-aware models, they can be applied to other classes without retraining, which significantly reduces the volume of training data and cost of data annotation [57].

In this chapter, state-of-the-art model architectures relevant to this thesis were reviewed. Furthermore, use cases for DNNs, such as object and keypoint detection, were explained. The following chapter concludes by examining the special criteria that are used to evaluate the performance of detection models.

2.1.5. Performance Metrics for Model Evaluation

This section provides insights of metrics that are used to quantify the performance of detection models.

The **average precision (AP)** is the most common metric used to quantify the performance of an object detector [51]. To understand the AP, some basic concepts are reviewed beforehand. They refer to [15, 32].

- A true positive (TP) prediction corresponds to a correct detection of a ground truth bounding box.
- A false positive (FP) prediction corresponds to an incorrect detection of a non-existent object or an incorrect placement of an existing object.
- A false negative (FN) prediction corresponds to an undetected ground truth bounding box.
- A true negative (TN) prediction corresponds to a correct detection of a non-existing object. However, predicting non-existing objects is irrelevant for a detector as there is an infinite number of bounding boxes that should not be detected in an image.

The categorization of a prediction as correct or incorrect is based on the intersection over union (IOU). The IOU measures the area of intersection between the predicted bounding box and the ground truth bounding box divided by the area of the union between them. A detection is classified as correct if the IOU is equal to or greater than a given threshold. Otherwise, the detection is classified as incorrect [51]. From TP, FP and FN, the precision (P) (Eq. 2.7) and recall (R) (Eq. 2.8) are computed as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$

P is defined as the ratio of the TP detections to all detections. R is defined as the ratio of the TP detections to all ground truths [15, 32]. A good object detector should find all ground truth objects (FN = 0, high R) and identify only relevant objects (FP = 0, high precision). The precision and recall change when the threshold is varied. A trade-off is observed, where the FP decreases (as the precision increases) and the FN increases (as the recall decreases) as the threshold increases, and vice versa [51]. The area under the curve (AUC) is the area under the precision-recall curve and the final metric used to measure model performance [32]. However, the curve is discontinuous, which is why the precision values are computed at a fixed set of recall levels. In this thesis, the definition of the famous Common Objects in Context (COCO) detection challenge [51, 58] is followed by choosing 101 equally spaced recall values from zero to one. The precision at each point is derived by taking the maximum precision at each recall threshold. Choosing the maximum makes the calculation less susceptible to downward outliers [51, 59]. The AP (Eq. 2.9) is computed as the average over the derived precision values.

$$\text{AP} = \frac{1}{101} \sum_{r \in \{0.00, 0.01, \ldots, 1.00\}} \max_{\tilde{r} \geq r} \text{Precision}(\tilde{r}) \tag{2.9}$$

The mean average precision (mAP) then measures the accuracy of an object detector by summing the AP over all classes N and taking the mean value (Eq. 2.10) [51, 59].

$$\text{mAP} = \frac{1}{N} \sum_{n=1}^{N} \text{AP}(i) \tag{2.10}$$

The **percentage of correct keypoints (PCK)** is a popular metric for keypoint detection [57]. A keypoint prediction is considered correct if the normalized distance between the prediction $\hat{K}_i$ and the ground truth K_i is equal to or less than a certain threshold α [57, 60]. $\mathbf{1}(\cdot)$ is the indicator function. d is the normalization factor, e.g., the longest side of the ground truth bounding box. As a standard, the PCK is also calculated for different threshold values α [57, 60]. The PCK formula is shown in Eq. 2.11.

$$\text{PCK} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1} \left(\frac{\|\hat{K}_i - K_i\|_2}{d} < \alpha \right) \tag{2.11}$$

To normalize the distance between the prediction and ground truth via a bounding box edge, a correspondence between the keypoint and the object must be established. However, such correspondences are not always obtainable, as shown in [12, 19]. In these cases, α is set to a fixed distance in px.

Intermediate Results

In Chapter 2.1 of this thesis, the theoretical foundations from the fields of ML and DL were presented in detail. ML enables technical systems to learn from data and use this knowledge to make predictions for new data. In contrast to standard ML models, DNNs are capable of recognizing deep patterns in input data due to their highly complex network architecture. A major field of application is image processing due to the special structure of image data. CNNs architectures are used to process image data efficiently. They are based on the convolutional operation, taking into account the grid-like structure of images and the relation between neighboring pixels. The recognition and classification of objects as well as keypoints on images is a central application of CNNs. Furthermore, CNNs are the backbone of many state-of-the-art network architectures that were developed specifically for detection tasks. CenterNet, ResNet and U-Net in particular will play a role in the further course of this work. The following section lays the theoretical foundations for the CSP technology focusing on SPT.

2.2. Concentrated Solar Power

This chapter is structured as follows: First, a general overview of the four relevant technologies regarding CSP is provided. The focus of this work lies on the SPT technology, which is why this technology is considered in detail. After introducing the fundamentals, a literature review reveals the state-of-the-art heliostat monitoring and calibration approaches. Based on this, the use of DL in heliostat calibration techniques will be discussed.

2.2.1. Concentrated Solar Power Technologies

CSP systems consist of four core elements: solar reflectors, solar receivers, power conversion systems, and electric generators. The solar reflectors are used to concentrate solar irradiance on a receiver. A heat exchanger integrated into the receiver transfers the energy of the focused solar rays as thermal energy to a circuit fluid. The heated fluid then drives a thermodynamic circuit process to produce electric energy [1, 61–63].

CSP technologies can be categorized into four distinctive types: linear fresnel reflectors (LFR), solar power towers (SPT), solar parabolic dishes (SPD) and parabolic trough collectors (PTC). Figure 2.9 depicts the basic functionality of these four technologies. First, the technologies shown can be classified by focus and receiver types. Line focus technologies (Fig. 2.9: 1 and 4) focus the irradiance on a linear receiver, and point focus technologies (Fig. 2.9: 2 and 3) at a single point receiver. The receiver can be fixed (Fig. 2.9: 1 and 2) or integrated (Fig. 2.9: 3 and 4). In the case of a fixed receiver, the receiver is stationary. In the case of an integrated receiver, the receiver moves with the tracking reflector. Integrated receiver systems can produce more energy because they can better focus the irradiance on the receiver [2–4].

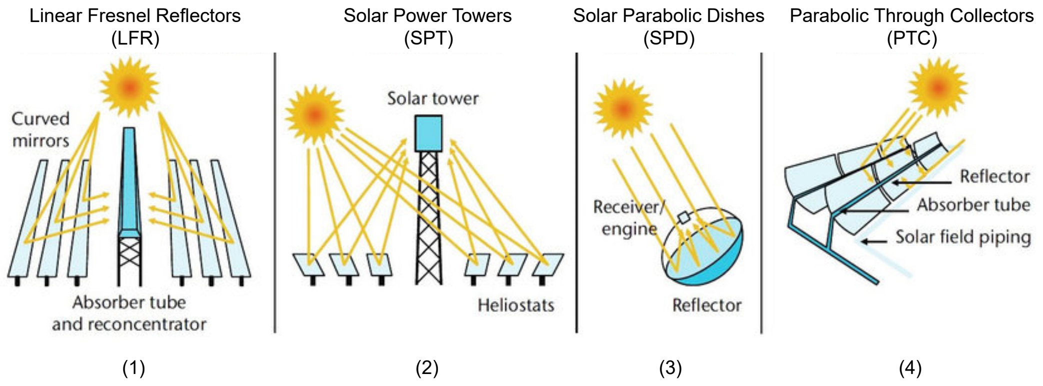


Figure 2.9.: CSP technologies [2].

In the following, the four technologies are explained in more detail.

1. LFR systems are line-focusing systems with a fixed receiver. The systems consist of parallel flat or slightly curved mirrors. The irradiance is focused on an absorber tube. As these tubes are fixed, only the mirrors can move, and only along the longitudinal axis. Due to their simple design, the LFR systems are cost-effective. However, this simplicity adversely affects the system's efficiency [2, 3, 63].

2. SPT systems are point-focusing systems with a fixed receiver. The systems consist of two-axis tracking mirrors, so-called heliostats, which concentrate the irradiance onto one solar receiver. The receiver is typically located at the top of a tower. The localization of the receiver at a certain height is necessary to prevent the reflected radiation of heliostats in the back rows from being blocked by those in the front rows. The functionality of a SPT plant is covered in detail in the next Chapter 2.2.2 [1–3, 63].

3. SPD systems are point-focusing systems with an integrated receiver. The systems consist of a parabolic two-axis tracking mirror that concentrates the solar irradiance at a focal point propped above the dish's center. The unique feature of this technology is the independent generator (e.g., a Stirling machine) at the focal point. As a Stirling machine directly converts heat into mechanical energy by compressing a gas (e.g., air) to generate electricity [64], a heat transfer fluid is not required. This direct power and heat cogeneration enables the highest efficiency of all the CSP systems [3]. However, the typical capacity of one dish is very low, which makes it necessary to co-locate hundreds to thousands of dishes to compete with other CSP technologies [2, 3, 63, 65].

4. PTC systems are line-focusing systems with an integrated receiver. The systems consist of parallel rows of reflectors curved in one dimension. The irradiance is concentrated onto absorber tubes, through which a heat transfer fluid (typically oil) flows, transferring the solar energy into a circuit process. The reflectors and the absorber tube are connected in their movement by following the sun during the day. Typically, movement is only possible by rotation along the longitudinal axis [2, 3, 63].

As discussed, CSP technology can be categorized into four main types. However, this thesis focuses exclusively on SPT technology, which reflects the scope defined in the thesis title.

2.2.2. Solar Power Tower Plants

In the following, the working principle of SPT systems is described. The description refers to Fig. 2.10 and [61, 64, 65]. SPT systems use a large number of two-axis tracking heliostats to concentrate solar irradiance onto a receiver. The sun's thermal energy is transferred to a working fluid via a heat exchanger in the receiver. The heated fluid then drives a thermodynamic circuit process to produce electric energy. There are two common concepts for transferring heat to the thermodynamic circuit process. The first option is to use the working fluid initially heated in the receiver for the whole circuit process. The second option is to use the initially heated fluid as an intermediate fluid. The intermediate fluid transfers its heat via a second heat-exchanger to another fluid in the power-block system, which runs the downstream circuit process. The commonly used intermediate fluids are oil and molten salt [3]. Although one-fluid systems require less investment (only one heat-exchanger in the receiver is needed), the strong dependence of fluid pressure on temperature, especially for water, makes it difficult to manage these systems. Controlling the process and preventing damages gets more challenging [4]. For this reason, these systems are generally less efficient.

The fluid choice in the power-block system is particularly dependent on the installed turbine. In the case of a steam turbine (Clausius-Rankine process), water and steam are used as the working fluid. The water is compressed in a pump and vaporized in a steam generator. The steam is then passed through a steam turbine and finally condensed back to its original state in a condenser. In the case of a gas turbine (Joule process), air is used as the working fluid. The hot air is first compressed and then passed through a turbine, where it expands. In both cases, the turbine is connected to a generator to produce electricity [61, 64, 65].

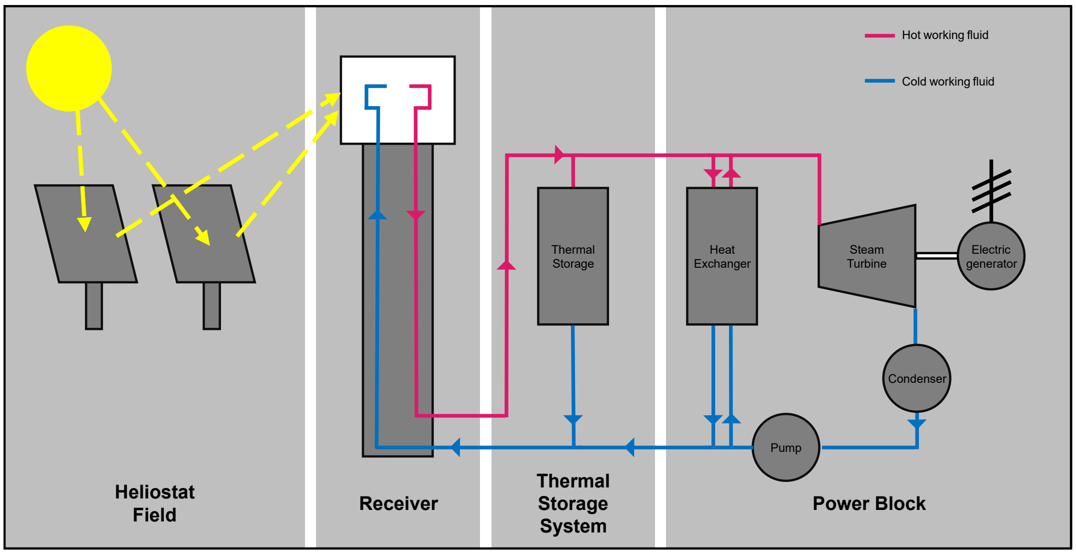


Figure 2.10.: Simplified working principle of a SPT plant with a Clausius-Rankine process and one fluid, own presentation based on [63, 66].

The ability to store heat is the most relevant advantage of SPT systems over other renewable energies (such as photovoltaic or wind energy). Excess thermal energy not needed to produce electrical energy in a specific period is stored in a thermal storage system. Another heat exchanger transfers the heat of the working fluid to the fluid in the thermal storage system, typically molten salt [66]. After sunset, the stored heat can be released into the power-block-system to produce electricity [2]. In this way, CSP systems function as a dispatchable energy source, and electrical energy can be produced on demand, contributing to a stable power grid [13, 61].

2.2.3. Heliostat and Field Designs

The basic design of a rectangular-shaped heliostat is shown in Fig. 2.11. The reflective surface of a heliostat typically consists of multiple small mirrors, so-called facets. Between the individual facets is a gap. Heliostats can track the sun by rotating around two axes: the azimuth and elevation axes [1]. With an even number of facets in the direction of the x-axis, it is possible to design the width of the middle gap larger so that the pedestal of the heliostat protrudes from this gap. The gap allows increased freedom of movement concerning rotation around the elevation axis (not shown in the figure).

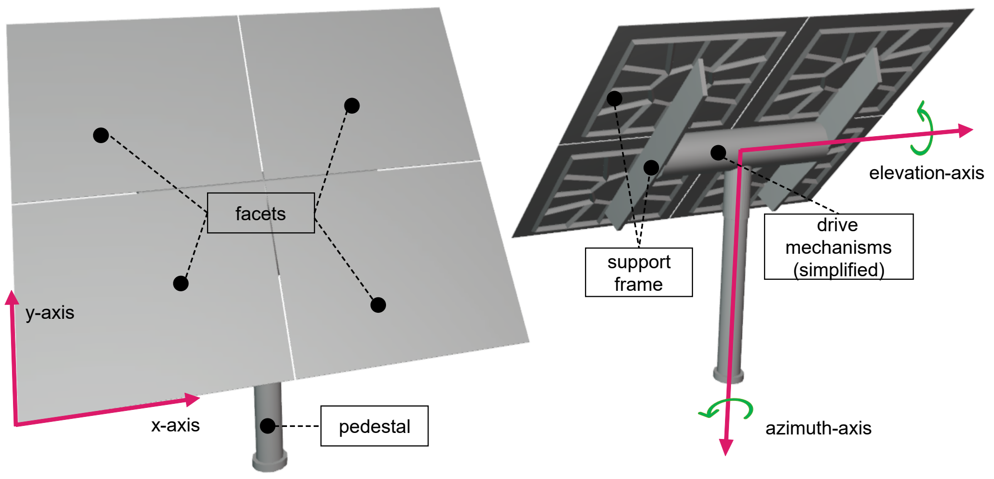


Figure 2.11.: Basic design of a heliostat, generated with the 3D graphics software Blender Blender, labeling of elements based on [67].

SPT systems offer flexibility in designing not only the heliostats but also the heliostat field. Rizvi et al. [62] distinguish between patterned heliostat field layouts and unpatterned ones. In this thesis, only patterned ones will be considered further. Patterned heliostat field layouts can be divided into the broader categories of rectangular and radial fields. In a rectangular field, heliostats are placed in straight rows and columns. In a radial (or circular) field, heliostats are placed in concentric circles around the tower. Furthermore, the field layout depends on the geographical latitude of the system location. For heliostat fields in the northern hemisphere, the heliostats are positioned to the north of the tower (north field). For heliostat fields in the southern hemisphere, the heliostats are positioned to the south of the tower (south field). For heliostat fields near the equator, it is worth placing heliostats around the tower (surrounding field) [4, 19, 62, 68]. The number of heliostats in the heliostat fields depends on the size of the heliostats and the installed and desired capacity of the power-block system, especially on the turbine and generator. The range of generating capacities of commercially operational SPT plants in 2022 was from 11 MW installed at the Plantar Solar 10 (PS10) in Spain to 377 MW installed at the Ivanpah Solar Electric Generating System (ISEGS) in the United States [1]. Figure 2.12 depicts a bird's eye view of these plants.

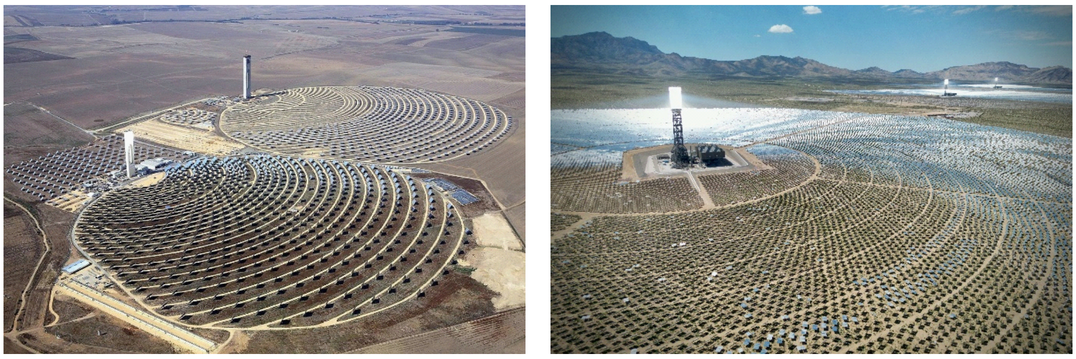


Figure 2.12.: Aerial view of the PS10 in the front and Plantar Solar 20 (PS20) in the back **(left)** [69], and the ISEGS **(right)** [70].

The picture on the left shows the circular field layout of the two plants PS10 and PS20. The right image shows the system ISEGS. The ISEGS is made up of a total of three separate, circular fields with a surrounding angle of 360°. Table 2.1 gives technical information about the respective power plants.

Table 2.1.: Technical data of the PS10 and ISEGS, data from [1, 71, 72].

Fact	PS10	ISEGS
Number of heliostats	624	173,500
Heliostat mirror area (m²)	120	15
Area of heliostat field (m²)[4]	75,000	2,600,000
Generating capacity (MW)[5]	11	377
Installation costs (Mio. €)	35	2200

[4]The area of the heliostat field of the PS10 corresponds to approximately 10 soccer fields, the area of the ISEGS even approximately 347 soccer fields [73].

[5]At full load, the PS10 can produce as much energy in one hour as five single-person households consume in approximately one year. The ISEGS can produce as much energy in one hour as 180 single-person households consume in approximately one year [74].

As can be gathered from Tab. 2.1, SPT systems do not only offer flexibility in terms of designing the heliostat field but also in terms of the heliostat design itself. Most of the heliostat fields around the world use heliostats with a rectangular surface design, but a pentagonal design is also possible [1, 75–77]. The range of possible heliostat sizes extends from very small heliostats (approximately 1 m^2) to large heliostats (approximately 180 m^2). However, it is noticeable that medium-sized heliostats (60-100 m^2) are not in use [75], which can be justified by the trade-off between the performance of a heliostat and its costs. Large heliostats generally have a lower irradiance concentration ratio than small heliostats, but have lower costs per square meter [75, 77, 78]. Furthermore, the optimal dimensions of a heliostat are influenced by the operating and environmental conditions [79].
Heliostats are a major cost driver for SPT plants (see Tab. 2.1) [3, 5, 78], and their precise alignment is a key factor for a high plant efficiency [5, 6, 75]. Precisely focusing the heliostats' reflected solar irradiance onto the receiver achieves high concentration ratios and temperatures and, thus, high efficiency [1, 4]. Regular condition monitoring of the heliostat field is therefore essential to ensure an efficient and reliable plant operation [6]. The following chapter thus addresses potential sources of errors in heliostats and their monitoring.

2.2.4. Heliostat Calibration

This chapter is divided into three sections. The first section explains the necessity for heliostat control. The second section analyses state-of-the-art methods for heliostat calibration. The last section focuses on using DL methods in this field.

2.2.4.1. Importance of Heliostat Calibration

Heliostats are a core element of a SPT plant and their precise alignment is essential to optimize the plant's efficiency [4–7]. However, many reasons can cause misalignment, like wind loads, a tilt of the heliostats' pedestal, or non-level terrain [12, 80]. There are several mathematical methods for analyzing the alignment accuracy of the heliostats. The calculation of the tracking, slope and canting errors are the most relevant ones [8, 75, 78]. These errors are defined as follows:

- The tracking error of a heliostat is defined as the deviation of the actual reflective surface orientation from the desired orientation [6]. The orientation is given by the normal vector of the reflective surface, called the optical axis [78].
- The slope error of a heliostat is defined as the difference between the desired design shape of the reflective surface at a certain point and the actual one [78]. The error is measured as the difference between the local normal vector at a given point and the optical axis of the heliostat [10].

- The canting error describes a systematic deviation of an entire mirror facet from its intended orientation. Since canting introduces a uniform bias across the facet, it is first determined as the average of the pointwise slope errors and then subtracted from the measured slope error values. In this way, the actual local deviations of the mirror surface are isolated [8].

The heliostat reflects incident solar irradiance based on the law of reflection. The angle of incidence equals the angle of reflection [81]. In other words, the angle of the incident radiation is mirrored on the (local) normal vector. This principle is illustrated in Fig. 2.13. In the case of correct alignment, the heliostat normal vector corresponds to the angle bisector between the vector of the incident solar radiation and the vector from the heliostat to the receiver. A minimal deviation of the heliostat alignment can significantly impact the solar focus on the receiver. For a heliostat at a distance of one km from the tower, a tracking error of one mrad - which equals 0.0057° - causes a deviation of around two meters between the desired aim point of the solar focus and the actual one [6]. As a result, the incident radiation on the heliostat is partially or entirely not reflected onto the receiver. This effect is called spillage [4, 80]. Besides reducing the power production of the SPT plant, the uneven focusing of the sunlight on the receiver can cause hotspots, which can cause damage [78, 80]. To reduce spillage and the risk of receiver damages, the angular accuracy should be in the range of 0.1 to 0.3 mrad [6].

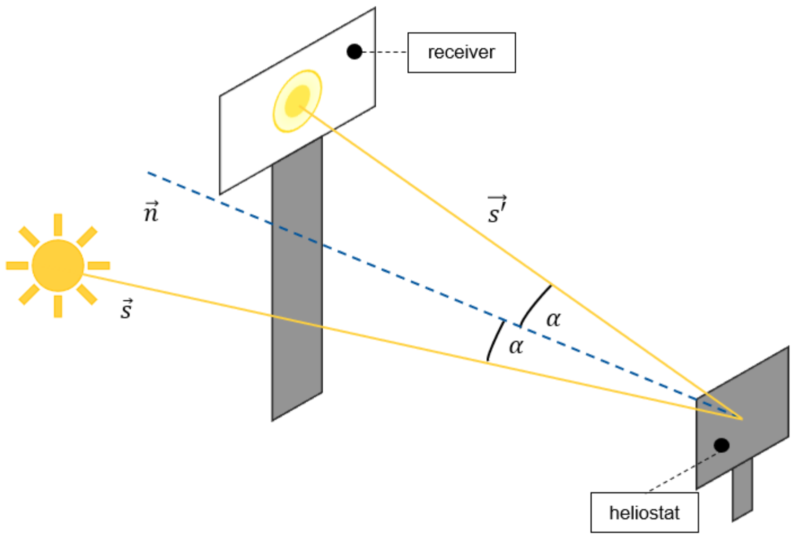


Figure 2.13.: Reflection of solar irradiance from a heliostat onto the receiver by law of reflection. Orange lines illustrate solar irradiance, a blue dashed line illustrates the optical axis of the heliostat. Illustration based on [82].

Heliostat control and calibration are necessary to ensure highly accurate heliostat alignment. Heliostat control refers to adjusting certain parameters, such as the drive positions (see Fig.

2.11), to obtain a desired heliostat orientation in which the incident solar radiation is reflected on the right target point on the receiver. Differences between the desired heliostat orientation given to the control system and the actual orientation are measured and compensated during heliostat calibration [6].

2.2.4.2. State-of-the-art Heliostat Calibration Methods

After the necessity of heliostat calibration was motivated, this section will cover state-of-the-art heliostat calibration methods. In a detailed study, Sattler et al. [6] review about 30 heliostat calibration and tracking control methods and categorize them into five groups. The general working principles of the control methods in these categories are explained in Appendix A.3. At this point, only one calibration method will be focused on in more detail, namely the camera-target method. Although the method was already developed in 1984 by Stone [83], it remains the most commonly used method in operating CSP plants. During a calibration, one heliostat is moved in such a way that the irradiance is pointing towards a white Lambertian target screen. The solar focus position on the target is then captured by a camera on the ground and compared to a reference position by using conventional image processing software. The identified difference between the actual and desired position of the reflection is used to adjust the heliostat's orientation in the heliostat control system. For a full calibration, this procedure has to be done for different sun positions [6, 83].

Although the camera-target-method has a sufficiently high accuracy, it has three significant weaknesses. First, the measurement depends on the sun because a specific heliostat orientation can only be calibrated with a specific sun position. Second, calibrating each heliostat one after another makes the calibration method time-consuming, which is especially a problem for large SPT plants. Third, the calibration is only possible if the tower and target are fully constructed. This circumstance prevents the heliostats from being calibrated during the construction phase [10].

A brief look at other calibration methods is provided in Appendix A.3. It can be stated that newly developed methods compete against the camera-target method in terms of measurement time and accuracy. However, if methods provide a sufficiently high accuracy, they suffer from a high measurement time [6, 8]. The reasons include individual calibrations [84, 85], the necessity of attaching markers or other sensors on each heliostat manually [86, 87] or applying solutions that must be fitted to the current heliostat field [88, 89]. Another major drawback of the majority of the respective methods is their applied approach of placing the camera and additional devices at a fixed position. As a result, the number of possible orientations for a heliostat that the camera can see is restricted. Furthermore, in scenarios where the camera is mounted on the tower [5, 85, 90], the mounting height must be sufficient to ensure that heliostats in the rear rows of the field are also visible. Additionally, it can be difficult to detect high-resolution reflections in the heliostat mirrors if the heliostat is far away from the tower [11]. For these reasons, the use of unmanned aerial vehicles (UAVs), or drones, is a promising alternative and has become a prominent field of research in recent years. Images taken by UAVs can supply a higher variation of camera positions and angles and can thereby reduce the distance between the camera and a heliostat. Two optical measurement methods are briefly introduced below before discussing approaches to using UAVs for heliostat monitoring.

- Photogrammetry is a general method for deriving an object's 3D shape and location based on one or more 2D images of this object. Part of the photogrammetric process is the image acquisition, image measurement, and object reconstruction based on physical models and mathematical formulas. For high-precision reconstruction, appropriate measurement systems are necessary. Moreover, photogrammetry allows for the calculation of the camera pose from known reference points. The pose includes the position and orientation [91].
- Deflectometry comprises methods to gain information about the shape and conditions of reflective surfaces by analyzing the reflected image of known objects. By analyzing the distortions (deflections) occurring in the reflected image, it is possible to obtain information about the surface properties, such as curvatures or defects [92].

These two methods form the basis for various research projects on the calibration of heliostats using UAV-based image acquisition. Building on these measurement principles, the studies in [8–11] present different approaches that utilize UAV-based imagery to accurately determine the orientation of heliostats. These approaches are summarized in Fig. 2.14 and explained in the following. Images are taken by a camera-equipped UAV. Conventional computer vision (CV) methods are used to process the images and derive the heliostat facet corner points from these images. By applying methods of the field of photogrammetry, the camera pose for every image and the approximate heliostat's optical axis for every heliostat on these images are calculated. Different approaches from the field of deflectometry are used to calculate the local normal vectors on the mirror surface for a heliostat at a certain point. Mitchell et al. [8] use the reflection of the tower. Jessen et al. [9] and Krauth et al. [10] use the reflection of a light emitting diode (LED)-equipped drone. Yellowhair et al. [11] use the reflection of a neighboring heliostat. By taking the average of the derived local normal vectors, the optical axis of each heliostat can then be calculated. Finally, the tracking and slope errors are derived, as explained above. Based on the measured errors, the heliostats' mirror surfaces can be adjusted to optimize the reflected solar irradiance onto the tower. This process step is shown in Fig. 2.14 with a dashed arrow to indicate that further intermediate steps are necessary at this point.

In all of the discussed approaches, the heliostat edge and corner detection is based on conventional image processing techniques, which have limitations and lack robustness[6]. For a heliostat pointing towards the ground due to a malfunction, its mirror surface is difficult to see from the air. Reflections of other objects (such as clouds or heliostats), varying lighting conditions or damaged and soiled mirrors influence the measurement negatively [8, 13]. Furthermore, the application of conventional image processing techniques requires prior knowledge of the heliostat's orientation within the image. The prior knowledge is needed since

[6]A separate research of selected conventional image processing techniques is provided in Appendix A.2.

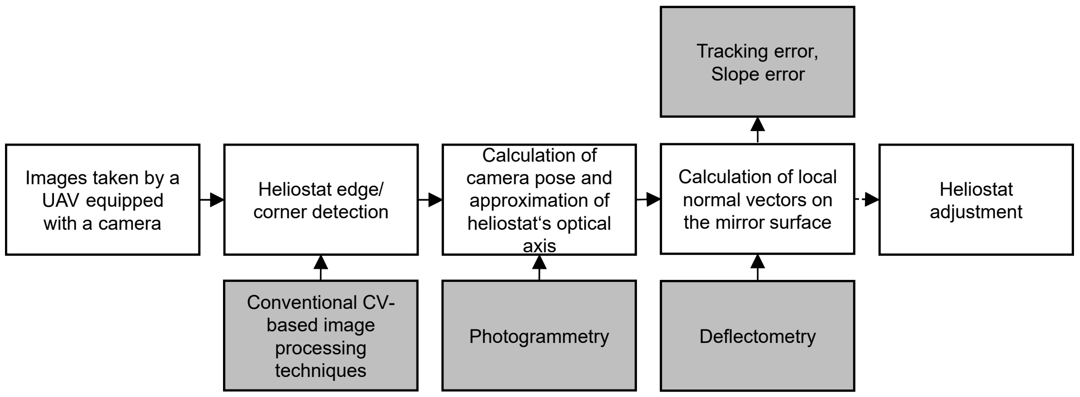

Tracking error, Slope error
Images taken by a UAV equipped with a camera
Heliostat edge/ corner detection
Calculation of camera pose and approximation of heliostat's optical axis
Calculation of local normal vectors on the mirror surface
Heliostat adjustment
Conventional CV-based image processing techniques
Photogrammetry
Deflectometry

Figure 2.14.: Heliostat calibration process based on [8–11]. Note that the canting error is also derived in [8, 11].

conventional edge and corner detection algorithms are typically optimized to identify features aligned with predefined directions, such as horizontal or vertical edges (see Appendix A.2). Consequently, the images often need to be orthogonalized to improve detection accuracy [91]. Moreover, many traditional approaches rely on so-called search windows or regions, within which specific image features such as corners or edges are searched for. These regions must be manually defined, introducing additional assumptions about the heliostat orientation and potential sources of error (see [12]). Overall, the tasks of orthogonalization and search window definition create a bottleneck in terms of the processing speed, which limits the applicability of the methods to commercial-scale power plants [13]. From the reasons stated, the integration of DL methods into the calibration process is currently being investigated. DL methods have already proven their wide range of use cases (see Chapter 2.1) and their „superiority over conventional image processing techniques in many domains." [13]. Therefore, the following section provides an overview of current approaches in this subfield.

2.2.4.3. Application of DL Methods in the Heliostat Calibration Process

Carballo et al. [16] use a computer vision (CV) approach to detect the sun, the tower target area and heliostats on images taken by cameras positioned at the heliostat mirror surfaces. Based on the gained information, the tracking error is calculated and the heliostat orientation is adjusted. The procedure is investigated for several available pretrained models. Liu et al. [17] use a CNN to obtain pixel-accurate segmentation masks of heliostats in the solar field. The CNN is supposed to enable robustness against two factors. Firstly, the images show varying lighting and soiling conditions. Secondly, the size of the heliostats varies in the images due to variable distances from the capturing camera. The results of the instance segmentation are used to estimate the optical axis of each heliostat by means of a not further specified algorithm. Although the results of Carballo et al. [16] show promising results, they rely on a fixed camera position. Kesseli et al. [12] use a combined DL and CV approach. In a first step, segmentation masks of PTCs are obtained using instance segmentation models. In the following, two distinct CV methods are used to detect the corner points of each PTC. Subsequently, the surface slope and offset of the absorber tube are computed. Even though the images of the solar field are taken by a camera-equipped UAV, this approach still relies on conventional image processing techniques. Xu et al. [18] developed a NN-architecture to detect heliostats in images. Results show an effective heliostat detection even with difficult lighting conditions. Nevertheless, the disadvantage of this approach is using a camera positioned at a fixed location. Broda et al. [13] propose model training with synthetic training data to overcome the problems of unbalanced and insufficiently large datasets. Preliminary results from testing the DL model with synthetic and real-world data demonstrate the plausibility of the approach. However, there is room for improvement regarding the application of real-world data. Also, the synthetic and real-world images are taken by an airborne camera. In a subsequent work by Broda et al., the influence of various parameters used to generate synthetic training images on the model's performance when applied to real-world data is analyzed [19]. Key parameters include ground texture, lighting conditions, heliostat soiling, and the placement and orientation of objects within the scene. The model is trained to detect heliostats along with their four outer mirror corners. With a suitable combination of simulation parameters, the results demonstrate good transferability from synthetic to real-world scenarios. Furthermore, the transferability of the results to other heliostat geometries is examined. In this case, however, the existing model must be retrained with the image data of the new geometry. As a qualitative analysis shows, the model cannot fully identify all heliostats and their four outer corners. Therefore, further work is recommended to improve transferability. In the following, Tab. 2.2 sums up the related work in the field of applying DL methods in the heliostat calibration process.

Table 2.2.: Overview of state-of-the-art literature covering DL-methods in the heliostat detection.

Author	Kesseli et al. [12]	Broda et al. [13, 19]	Carballo et al. [16]	Liu et al. [17]	Xu et al. [18]
Fixed camera position	✗	✗	✓	✓	✓
Conventional image processing techniques	(✓)	✗	✗	✗	✗
More than one heliostat geometry involved	✗	(✓)	✗	✗	✗
Model Architecture	Mask R-CNN	Mask R-CNN CenterNet	SSD MobileNet[7] SSD Inception[8] Mask R-CNN	Mask R-CNN	YOLO

As was elaborated, only [12, 13, 19] deal with a UAV and airborne image generation. However, [12] uses conventional image processing techniques that lack robustness and processing speed. Furthermore, it must be particularly emphasized that only [19] consider more than one heliostat geometry for the model used.

[7] See source [93] for more information
[8] See source [94] for more information

Intermediate results

Chapter 2.2 initially looks at the fundamentals of the CSP technologies. CSP comprises four technologies, whereby the SPT technology is focused in this thesis. It has been shown that the dimensioning of a heliostat field and the individual heliostats is flexible within certain limits. However, what all the SPT systems have in common is that continuous and highly precise monitoring of the alignment of the heliostats is necessary to ensure high system efficiency. Although it is already outdated, the camera-target method is widely used for heliostat calibration. Even though a high level of research is being carried out in this field, there is no other generally accepted method suitable for practical use regarding processing speed and accuracy. Using drones equipped with high-resolution cameras and optical measuring methods shows improvements in process speed and accuracy. However, using conventional image processing methods for the initial detection of heliostats still remains a weak point. Consequently, the entire calibration process becomes prone to errors in difficult lighting conditions, unfavorable positions of the camera concerning the heliostat or soiling. Recent research is therefore investigating the use of DL methods as part of the calibration process. Nevertheless, existing approaches either combine DL methods with conventional approaches or use a fixed camera, which is inferior to drone technology. All approaches, except for [19], focus only on detecting heliostats with one defined geometry. This focus implies a significant limitation for the application range because the heliostat geometry can vary with each plant. To detect heliostats with a second geometry, [19] trains an existing model with a further geometry. The evaluation shows promising results but also formulates a need for further investigation of the model transferability to other heliostat geometries and therefore other SPT plants.

Building on the work of Broda et al. [13, 19], this master's thesis is therefore dedicated to developing a geometry-agnostic DL model that can detect heliostats of different geometries on aerial images without additional training. This method would then be capable of detecting heliostats on all types of SPT systems.

3. Methodology

This thesis directly continues on the results of [13, 19]. Referring to Chapter 2.2.4.3, Broda et al. [19] examine the model performance by training and testing their detection model for one mirror geometry. Further, the model performance is analyzed when testing on another geometry. However, this investigation is carried out qualitatively and based on retraining. Under these conditions, the test results do not allow well-founded statements about the model's transferability to different heliostat geometries [19]. Therefore, this work aims to develop a geometry-agnostic DL model for detecting heliostats in SPT plants. The detection task includes the prediction of the heliostat bounding boxes as well as all heliostat keypoints. In particular, the developed geometry-agnostic model could recognize different heliostat geometries without having to be explicitly trained on each. As a result, the model could be applied to a large number of SPT plants.

Chapter 3 covers the steps of the methodology applied to develop a geometry-agnostic DL model to detect heliostats in SPT plants. As shown in Fig. 3.1, the designed methodology consists of five consecutive steps. A separate subchapter in this chapter is dedicated to the steps of synthetic data generation, data pre-processing and cluster-based model training. The evaluation of the results achieved by model training, testing and post-processing is done separately in the following chapter. The procedure is therefore only roughly outlined here. The model is trained with synthetic data due to the limited availability of real, labeled data. As a preparatory task, the synthetic data undergoes pre-processing to ensure compatibility with the further training pipeline. The model architecture developed by Broda et al. [19] is used for model training. The training is based on an iterative clustering approach. Various training and testing procedures are implemented and compared to ensure optimal results. In the model post-processing stage, an algorithm is proposed to further enhance model performance by applying geometric constraints to refine the keypoint predictions. In the following, section 3.1 begins by describing the model architecture.

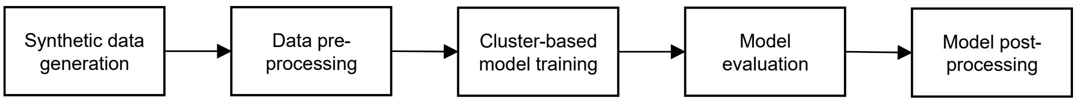


Figure 3.1.: Underlying methodology for developing a geometry-agnostic deep learning model in this thesis.

3.1. Model Architecture

This section provides a detailed overview of the architecture of the DL model employed in this work. The model was developed in preliminary work, which can be read in detail in [13, 19]. In the further course, the focus is on the model components used and how they interact.

The DL model is designed to detect heliostats and their mirror facet corner points. In the following, the mirror facet corner points are called keypoints. These two tasks can be executed simultaneously, but also function independently of one another. The fundamental structure of the model is based on an encoder-decoder architecture. The encoder is a ResNet with 50 layers, called ResNet50, which was pretrained on ImageNet[9] [56]. Similar to U-Net [41], skip connections are employed between the encoder and the decoder. Adapted from CenterNet [96], the model generates a heatmap in which each prediction corresponds to a location within the image's pixel-based coordinate system. It predicts both the center of the heliostat's bounding box and the positions of its keypoints. The output stride of the model, defined as the ratio between the input image and output heatmap resolution, is set to two. Thereby, the resulting heatmap has only half the resolution of the input image. Thus, a prediction on the heatmap does not directly correspond to a specific pixel in the input image. By applying sub-pixel regression, the coarse heatmap predictions are refined by estimating their precise positions in the original image space.

Combining an encoder-decoder structure with skip connections and implementing a ResNet50 enables the prediction of high-resolution features. Instead of relying on region proposals (see [22]), the prediction of the bounding box center, along with subsequent regression of its size, accelerates the object detection [53]. Another advantage of this architecture is its flexibility. The encoder can be easily exchanged for an even faster CNN such as MobileNet. Theoretically, this exchange enables real-time applications with a drone [19]. The model architecture is shown in Fig. 3.2. Within each encoder layer and within each decoder stage, the convolutional layers have the same number of kernels and the same kernel size. For a detailed description, reference is made to [19].

Minimizing the multi-task loss is the primary objective for the model training [44]. The multi-task loss is derived as the weighted sum of the losses of the individual model tasks. Keypoint detection involves predicting a keypoint heatmap and regressing the corresponding pixel offset. In object detection, the center of the bounding box is predicted by means of a heatmap, the bounding box size and the corresponding bounding box pixel offset. The focal loss is used for predictions on the heatmap [97] and the smooth l1 loss for the regressions.[10]

[9] Learned features can be transferred to new datasets. The transfer results in a reduction of computing time and data required for training. Furthermore, the model's performance increases compared to one that has been trained from scratch. In particular, the features of early layers (e.g., corners and edges) are well transferable. However, the transferability decreases with increasing differences between source and target objects [95].

[10] A detailed description of the individual loss functions can be found in Appendix A.1.

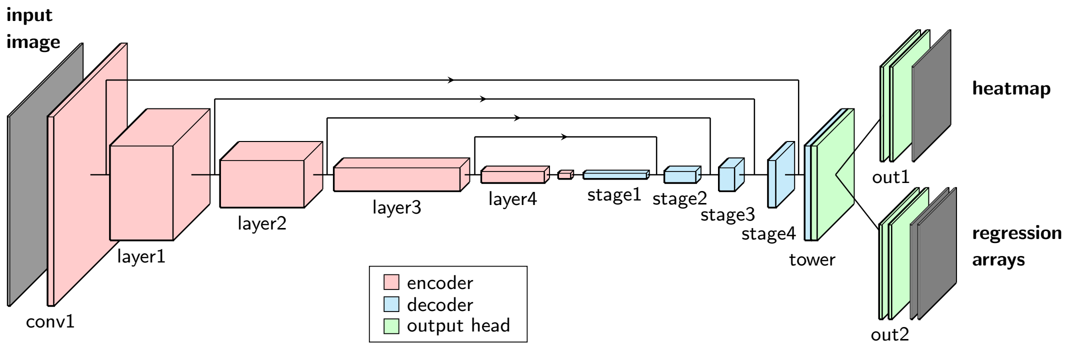


Figure 3.2.: Representation of the employed encoder-decoder model architecture [19].

In conclusion for this subchapter, Fig. 3.3 shows an exemplary bounding box heatmap for a sample image. A peak in the heatmap, indicating the bounding box center, is represented by a 2D Gaussian distribution. In the shown figure, the color of the peak correlates with the confidence for this prediction, with values close to one indicating a high degree of confidence.

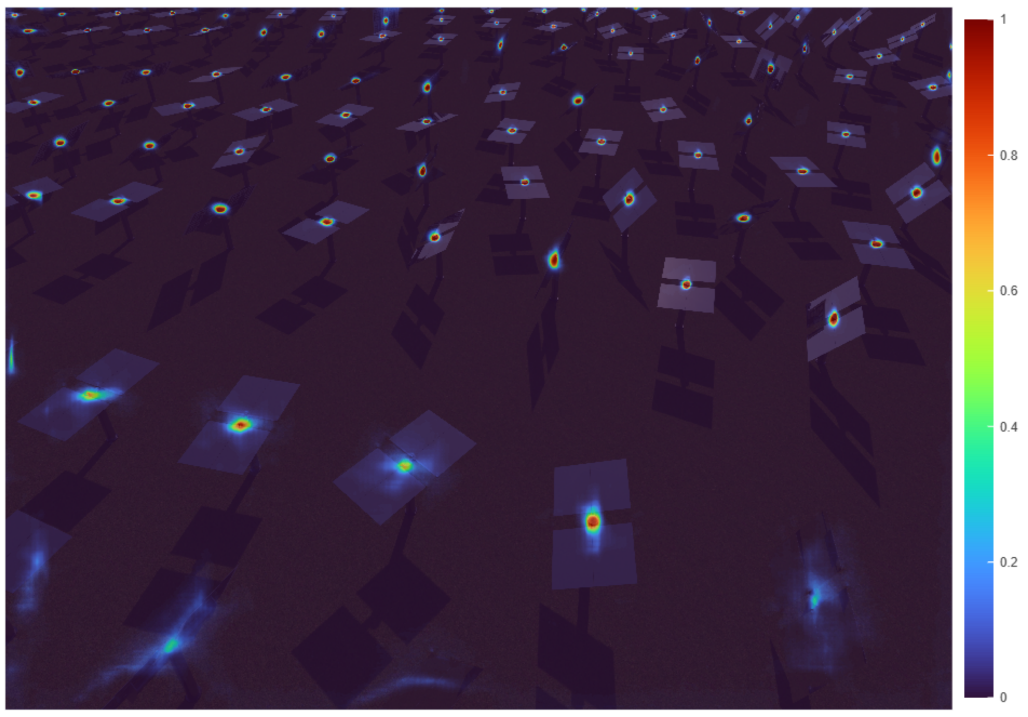


Figure 3.3.: Visualization of a bounding box heatmap for a sample image. The heatmap is overlaid on the original image with 50% transparency.

The model architecture was described in this subchapter. The following subchapter focuses on generating the artificial training data.

3.2. Data Generation

In this thesis, the study on the learnability of geometry agnosticism through a DL model relies entirely on the usage of synthetic data. This is a consequence of the limited availability of real image data from commercial or research SPT plants. Furthermore, even if data is available, manual annotation of e.g., keypoints or bounding boxes, is error-prone and time-intensive. In particular, the annotation of keypoints must be done very precisely [19]. During the work on this master thesis, real annotated data was only available from the CESA1[10] test plant as part of the Plataforma Solar de Almería (PSA) located in Almería (Spain).

The underlying idea in developing a geometry-agnostic model is to include a wide range of geometries in the training process. The central hypothesis is that the model can learn not only to detect heliostats and their keypoints within known geometries, but, with this training strategy, also to generalize to unseen geometries. For the purpose of generating various training data, the 3D modeling software Blender is used. Within Blender, an already existing parametric heliostat model is utilized to replicate different geometries. Furthermore, an existing pipeline [13] is used to create entire heliostat fields. The selection of parameters used to generate diverse heliostat geometries plays a crucial role. To ensure practical relevance, these parameters should not be chosen arbitrarily, but instead be based on real-world heliostat geometries currently deployed in SPT plants worldwide. Chapter 3.2.1 outlines the methodology and findings of the literature review conducted on existing heliostat geometries. Chapter 3.2.2 focuses the process of generating synthetic training data based on these real-world geometries.

3.2.1. Literature Research on Existing Heliostat Geometries

The Heliostat Consortium for Concentrating Solar-Thermal Power (HelioCon) maintains a database containing technical data on SPT plants worldwide and the installed heliostat designs [1, 24]. This database will be used to parameterize various heliostat models in Blender. In addition to [1, 24], other sources were also investigated. Further available studies give information on the state-of-the-art SPT technology, where a wide range of SPT plants are documented [3, 61, 62, 75–77, 98]. However, technical information regarding the heliostats is a rarity and is, if available, limited exclusively to the size of the whole mirror surface of a heliostat. From this information, the parameters of the heliostat model in Blender (see Tab. 3.2) cannot be derived.

Tabular 3.1 contains the heliostat geometries listed in the HelioCon database. Not all entries in the database could be transferred due to a lack of technical data. In some cases, missing data could be supplemented by taking the average of the available data. These entries are marked with a star (*). A detailed explanation of how the HelioCon database was analyzed can be found in Appendix A.3.

[10]CESA1 test plant owned and operated by Centre for Energy, Environmental and Technological Research (CIEMAT).

Table 3.1.: Summary of operational heliostat geometries worldwide based on [1, 24]. *Missing data was methodically complemented (see Appendix A.3). n_x, n_y denote the number of panels in x and y directions, s_x, s_y represent panel dimensions in m rounded to one decimal place. `middle_gap` is true if the heliostat type installed in the corresponding plant has a middle gap, false otherwise.

Id	Plant	n_x	n_y	s_x [m]	s_y [m]	`middle_gap`
1	Shouhang Dunhuang 100-MW Phase II (China)	7	5	1.5	2.2	false
3	Luneng Haixi 50-MW Molten Salt Tower (China)*	4	8	2.2	2.2	false
4	PowerChina Gonghe 50-MW CSP Plant (China)	2	2	2.9	1.8	true
5	SupCon Delingha 50-MW Tower (China)	2	2	2.9	1.8	true
6	Shouhang Dunhuang 10-MW Phase I (China)	7	5	1.5	2.2	false
7	Ashalim Plot B (Israel)	2	2	2.0	2.6	false
8	NOOR III (Morocco)	9	6	1.5	2.2	false
9	Khi Solar One (South Africa)	2	8	6.6	1.3	false
10	Gemasolar Thermosolar Plant (Spain)	7	5	1.6	2.1	false
11	Planta Solar 20 (Spain)	4	7	3.2	1.4	false
12	Planta Solar 10 (Spain)	4	7	3.2	1.4	false
13	Crescent Dunes Solar Energy Project (USA)	7	5	1.5	2.2	false
14	Ivanpah Solar Electric Generating System (USA)	2	1	2.3	3.0	true
15	ACME Solar Tower (India)*	1	1	1.0	1.0	false
16	Atacama I (Chile)*	4	8	2.2	2.2	false
17	Badaling Dahan (China)	8	8	1.3	1.3	false
19	CTGR Qinghai Golmud 100MW (China)*	5	7	2.2	2.2	false
21	Jemalong Solar Thermal Station (Australia)*	3	1	2.2	2.2	false
29	Sierra SunTower (USA)	1	1	1.0	1.0	false
33	Yumen Xinneng-Xinchen (China)	4	4	1.1	1.0	true
34	Sundrop CSP (Australia)	1	1	1.5	1.5	false

It should be noted that the identifier (ID) in the first column is an internal ID and does not correspond to the row number in the table. The existence of a middle gap in the mirror geometries of the heliostats was determined visually based on the existing image data in the respective entries of the HelioCon database. Moreover, note that the same (e.g., Planta Solar 10, Planta Solar 20) or very similar (e.g., Yumen Xinneng-Xinchen, Sundrop CSP) geometries are installed in some plants. At this point, these geometries are not deleted from the table to reflect the actual installations in the real world.

3.2.2. Scene Generation in Blender

Blender, an open-source 3D modeling and rendering software [20], serves as the backend for the entire data generation pipeline. In this work, Blender is used to model heliostats, build 3D heliostat fields (referred to as scenes), and render photorealistic images, where rendering denotes the creation of 2D images from 3D scenes [20]. Based on Blender, BlenderProc [21, 99] enables the automated generation of image labels, while a custom code library provides tools for heliostat modeling, scene construction, and coordinated rendering [13]. A model of a heliostat has already been developed in Blender, allowing for a flexible design using several parameters. Figure 3.4 shows four different specifications of the parametric heliostat model. Functionalities to create damaged mirrors (second from left) and soiling (third from left) are also implemented. The second and third heliostats (from the left) are distinguished from the others by a noticeable gap in their center.

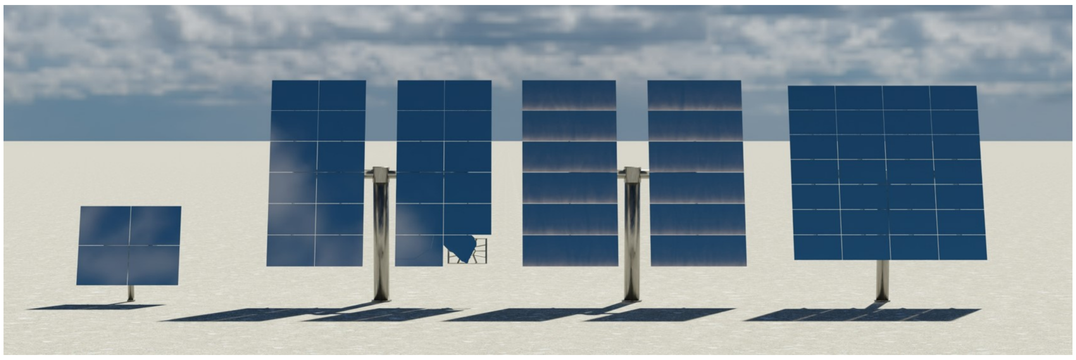


Figure 3.4.: Blender 3.5.0 parametric heliostat model [13].

In the following, Tab. 3.2 shows an extract of the parameters to be defined to configure the heliostat model in Blender.

Table 3.2.: Parameters of the heliostat model in Blender. Units: panel counts in panels, rotation angles in degrees, all lengths in m.

Parameter	Physical interpretation
`n_panels_x`	Number of panels of the heliostat mirror surface along the x-axis
`n_panels_y`	Number of panels of the heliostat mirror surface along the y-axis
`panel_size_x`	Width of a panel along its x-axis
`panel_size_y`	Width of a panel along its y-axis
`panel_thickness`	Thickness of a mirror panel
`gap_x`	Width of the gap between two mirror panels along the x-axis
`gap_y`	Width of the gap between two mirror panels along the y-axis
`middle_gap`	Width of the central gap between the adjacent panels to the left and right of the center along the x-axis (only greater than zero if the panel count is even)
`min_elevation_deg`	Lowest permissible value for the angle of rotation around the elevation axis
`max_elevation_deg`	Highest permissible value for the angle of rotation around the elevation axis

To recreate the real geometries from Tab. 3.1 in Blender, the number and size of the facets of each geometry can be used directly. Information on the gap size and panel thickness could not be determined during the literature research. However, to represent a good approximation for a realistic heliostat geometry, the values are chosen as follows: `gap_x` = `gap_y = 0.01` m, `middle_gap` = 0.75 m and `panel_thickness` = 0.004 m. These values were assumed constant for all geometries. For heliostats without a middle gap, the permissible rotation range around the elevation axis is set to the interval [0°, 80°]. Limiting the elevation angle prevents the mirror surface from cutting through the stand of the heliostat in the Blender model. This problem cannot occur with heliostats with a middle gap, which is why the permissible rotation range is set to the interval [0°, 180°]. Damaged or dirty mirror facets are excluded from this work.

In the following, heliostat fields are generated based on the parameterized heliostat models. To do so, an existing Python script is used, independently of Blender. Depending on the choice of the field layout (rectangular, circular), layout parameters (e.g., minimum and maximum distance of a heliostat to the field center) and other parameters, the positions of the heliostats are calculated and saved as 3D coordinates. Additionally, an azimuth and elevation angle is calculated for each heliostat based on the sun's position and the coordinates of the field location. The calculated data is exported as a comma separated values (CSV) file. An excerpt of an entry for a heliostat position is shown in the following Tab. 3.3.

Table 3.3.: Exemplary entry of a field.csv file (excerpt). x, y and z in m, azimuth and elevation in °.

x	y	z	Azimuth	Elevation
-11.92	-79.11	0.0	111.49	61.90

The entry shows the data for a heliostat at the position [-11.92, -79.11, 0.0]. The coordinate system is based on the east-north-up (ENU) standard where the x-axis points in the east, the y-axis in the north and the z-axis upwards [100]. According to this coordinate convention, the heliostat is located in the southwest of the tower in a circular heliostat field. In a second step, a random drone flight route over the simulated heliostat field is created. A function generates and saves 3D camera positions and orientations and also exports them as a CSV file. The flight routes are based on user-defined parameters such as altitude ranges and flight zones. An excerpt of an entry for a camera orientation is shown in the following Tab. 3.4.

Table 3.4.: Exemplary entry of a drone.csv (excerpt). x, y and z in m, Pitch ϕ, Roll θ and yaw ψ in °.

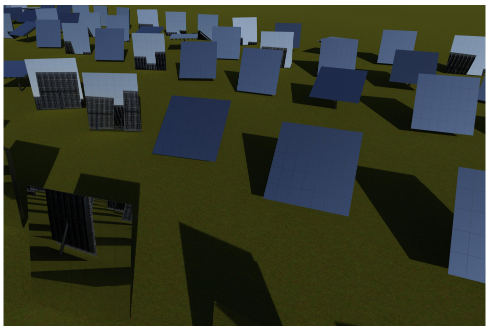
x	y	z	Pitch	Roll	Yaw
-169.59	28.12	22.66	41.46	0.0	67.38

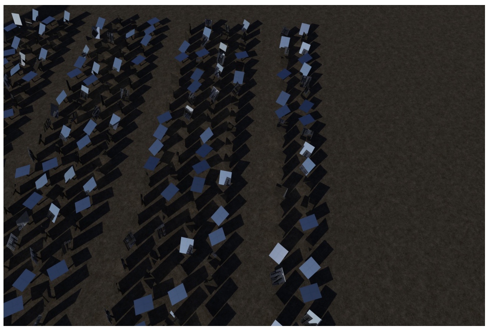
Roll θ, pitch ϕ and yaw ψ refer to rotations about the x-, y-, and z-axes, respectively. The specification of the rotation angle is based on the Tait-Bryan (also known as YRP) convention. The rotation angles are defined using a hierarchy in which the rotation axes change depending on the orientation of the coordinate system. The rotation takes place in the sequence yaw, roll, pitch. This means that the coordinate system is first rotated about the initial z-axis, then about the y-axis of the intermediate frame, and finally about the x-axis of the resulting frame [101–103]. The angles refer to a coordinate system according to the ENU standard, which is rotated 180 degrees around the x-axis. When taking aerial photographs with a drone, this convention reduces the roll angle as the z-axis (the direction of the camera) is already directed downwards.

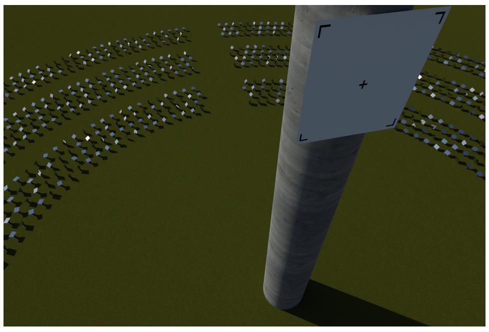
After explaining the relevant configurations for generating synthetic image data, the following subsection provides a detailed overview of the resulting synthetic datasets.

3.2.2.1. Presentation of Generated Synthetic Image Data

By determining the parameters of the heliostat model, the positioning and orientation of the heliostats in space and various camera poses, all the data required to create the artificial images is now in place. All of the Blender scenes created in this work are made up as follows: For each of the geometries listed in Table 3.1, four scenes were generated. Two of these scenes feature a rectangular field layout, while the other two follow a circular arrangement. Within each field layout, one scene uses grass and the other soil as the ground texture. The drone takes 125 pictures per scene from a varying altitude of between 20 and 100 meters. This procedure results in a dataset comprising 11,000 synthetic images in total at this point, each with a resolution of 6000 $\times$ 4000 px. Four sample images are shown below. Figures 3.5a and 3.5b show two images from different scenes with a rectangular field layout. Figure 3.5a shows a rendered image of geometry one (see Tab. 3.1) with grass as ground texture and a low drone flight altitude. Figure 3.5b shows a rendered image of geometry seven with soil as ground texture and a higher drone flight altitude. Figures 3.5c and 3.5d show two images from different scenes with a circular field layout. Figure 3.5c shows a rendered image of geometry 15 with grass as ground texture and a very high drone flight altitude. Also visible is the tower in the center of the field, which was included in every scene. Figure 3.5d shows a rendered image of geometry 19 with soil as ground texture and a medium drone flight altitude.


(a) Blender scene | geometry: 1, field layout: rectangular, ground: grass, image id: 18.


(b) Blender scene | geometry: 7, field layout: rectangular, ground: soil, image id: 113.


(c) Blender scene | geometry: 15, field layout: circular, ground: grass, image id: 114.

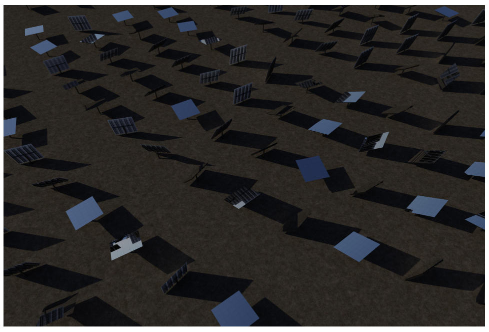

(d) Blender scene | geometry: 19, field layout: circular, ground: soil, image id: 112.

Figure 3.5.: Examples of rendered images of Blender scenes with rectangular (top) and circular (bottom) field layouts, grass ground (left) and soil ground (right). Resolution reduced to decrease file size.

The example images show that, despite the given position of the sun, even neighboring heliostats are aligned very differently from one another. The varying alignments were controlled by a randomness factor added to the elevation and azimuth angle of each heliostat. The aim was to introduce more variation into the poses of the heliostats, thereby varying the training data to a larger extent. This idea will become relevant again in the following Chapter 3.3.
In addition to Blender, the extension BlenderProc2 is used. Blenderproc2 can render and label photo-realistic images for the supervised training of NNs [21, 99]. In this thesis, Blenderproc2 is used to create the labels for the objects in a scene in the form of a COCO annotation file [58]. Figure 3.6 shows a sample image and corresponding ground truth labels for a scene created with the described simulation environment.

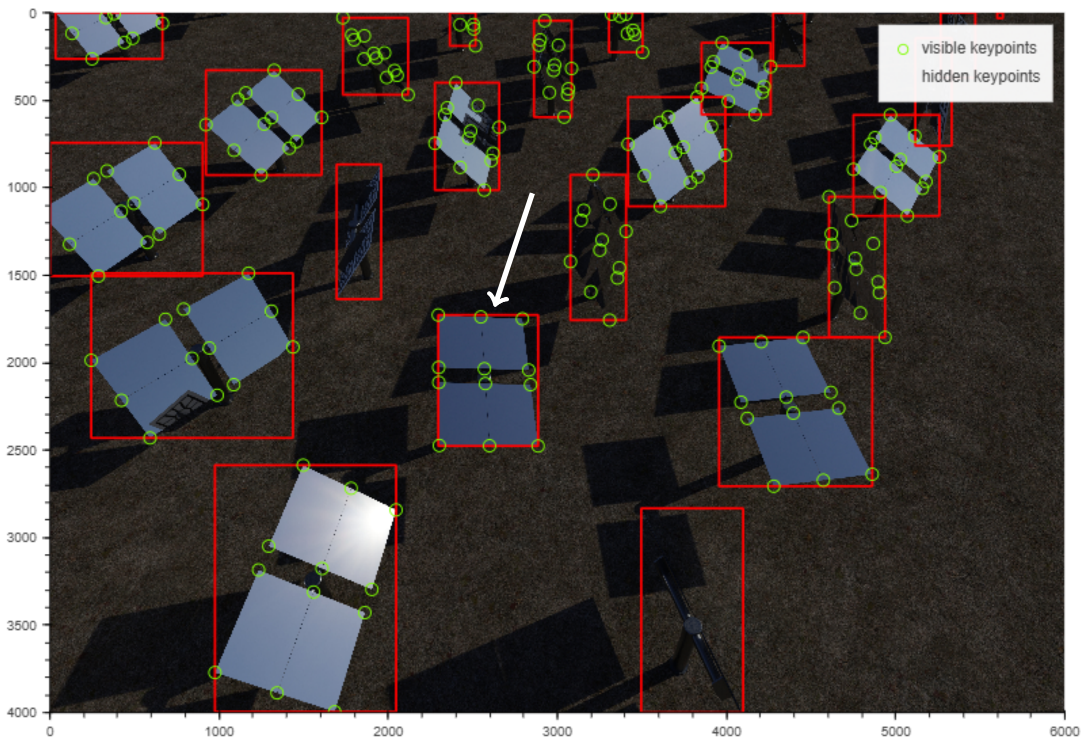


Figure 3.6.: Visualized ground truth labels for keypoints and bounding boxes on Blender scene | geometry: 4, field layout: rectangular, ground: soil, image id: 9. An excerpt of the COCO annotation of the heliostat marked with a white arrow is shown in the following.

The mirror surface shown consists of four facets and has a middle gap. Keypoints are marked by a green circle, bounding boxes by a red rectangle. Keypoints are not annotated for heliostats that are only visible from the rear, such as when the mirror surface is not visible due to heliostats' orientation to the camera. Listing 3.1 shows an excerpt of the corresponding COCO file, including the ground truth data for the heliostat marked with a white arrow.

```
{
  "id": 317,
  "image_id": 9,
  "category_id": 9,
  "bbox": [2295, 1727, 591, 750],
  "segmentation": [
    [2543.0, 2055.5, ..., 2543.0, 2055.5],
    [2885.0, 2476.5, ..., 2885.0, 2476.5]
  ],
  "keypoints": [
    2885.96, 2476.30, 2, 2793.14, 1749.52, 2, ...
  ],
  "num_keypoints": 12
}
```

Listing 3.1: Exemplary COCO entry for an annotated solar module (excerpt).

Following the COCO convention [58], every annotation in a rendered Blender scene has its own ID. Moreover, each image (`image_id`) and geometry type (`category_id`) has its own ID. Note that the `category_id` is not the ID from Tab. 3.1 but is a unique pre-defined ID for every geometry and is used consistently throughout all generated datasets. A bounding box in a COCO file is represented by four values: the x and y coordinates of its top left corner, followed by the width and height of the box. Mirror surfaces of `category_id` four consist of 12 keypoints. Each keypoint is defined by three values: the x and y coordinates and the visibility, in this order. A visibility of zero implies a keypoint that is outside the image frame. A visibility of one means that a keypoint is not visible in case of a heliostat facing away from the camera or overlap with other heliostats or objects. A keypoint with visibility two implicates a keypoint, which is visible in this image frame.

The synthetic COCO annotation files created with Blender and BlenderProc are pre-processed in the following. This is explained in the subsequent subchapter.

3.3. Data Pre-Processing

As part of this master's thesis, the existing model architecture will be expanded to include additional functionalities. Therefore, as a preparatory task, the synthetic data undergoes pre-processing to ensure compatibility with the further training pipeline. The tasks performed are explained in the following subsections.

3.3.1. Keypoint Accuracy Requirements for Reliable Calibration

Accurate detection of heliostat keypoints is essential, as they serve as the base for the heliostat calibration techniques discussed in Section 2.2.4.2. For an accurate heliostat calibration, Sattler et al. [6] state, that the angular accuracy for the detected keypoints should be in the range of 0.1 to 0.3 mrad (see Chapter 2.2.4). In their study, Kesseli et al. [12] investigate the maximum permissible distance between ground truth and model detection for the corner points of a PTC to ensure this angular accuracy. Since the requirements for keypoint detection accuracy are primarily determined by local image features, it is assumed that the results presented in [12] are transferable to heliostats and the SPT technology. Kesseli et al. [12] shift the four outer ground truth keypoints by a certain length in the image plane and measure the resulting angular accuracy. The length of the displacement was measured in cm rather than px by „using a known length-scale in the image." [12] The analyses made show that with 95% certainty, if the module corner points identified are within 1.18 cm of the manually marked ground truth points, the resulting angular accuracy remains within $\leq \pm 0.25$ mrad. Using a metric unit, such as cm, instead of px to determine the allowed tolerance, makes the corner identification sensitivity independent from camera parameters like the resolution or distance to the keypoints [12].

In preliminary works from Broda et al. [19], a fixed tolerance of three px was used to detect the four outer mirror corners. However, this does not account for the varying flight altitudes of the drone. Therefore, the fundamental idea of a keypoint tolerance based on a metric unit, such as cm, was adopted from Kesseli et al. [12] in this thesis. However, in contrast to Kesseli et al. [12], a fixed-length scale in the image is not known. Without going into further detail at this point, this is because at this stage of the detection process, no assignment is done between detected keypoints in the image plane and keypoints in 3D space. For this reason, a novel procedure was developed as part of this master's thesis to define a tolerance radius for a keypoint prediction in the image plane based on a metric unit without known length measurements in the image. As a theoretical foundation and for better understanding, the principles of camera models and projections are introduced in the following.

3.3.1.1. Excursus: Camera Projection Models

The simplest camera model, but good enough for the purposes of this work, is the pinhole camera model. It defines the projection of a point defined in $\mathbb{R}^{3\times3}$ onto an image. The three-step projection is shown in Fig. 3.7. In general, points in space are expressed in a Euclidean coordinate frame, in the following called global coordinate system (GCS) (see 3.7 on the left). The coordinate system of the camera (see 3.7 in the middle), here defined as observer coordinate system (OCS) as in [104], can be derived from the GCS by applying a rotation $\mathbf{R}^{3\times3}$ and a translation $\mathbf{t}^{3\times1}$. The transformation is shown visually by a curved dotted arrow connecting the origins of the GCS and OCS.

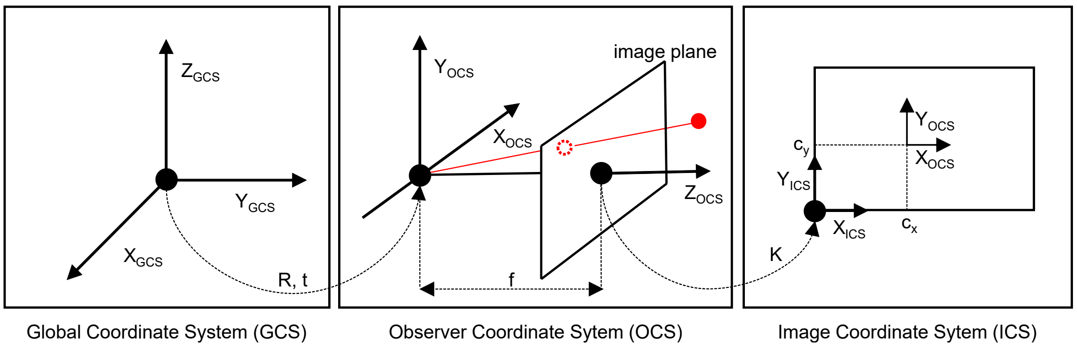


Figure 3.7.: Definition of GCS, OCS and ICS, based on [105].

Transforming a point $\mathbf{x}_{\text{global}}$ from the GCS into the OCS is mathematically defined by Eq. 3.1 [105].

$$\mathbf{x}_{\text{cam}} = \mathbf{R} \cdot \mathbf{x}_{\text{global}} + \mathbf{t} \tag{3.1}$$

By definition, the camera shows in the direction of the z-axis, also known as the optical axis. The optical axis has the important property that it is perpendicular to the image plane. In the following, the camera projects a point from the Euclidean 3-space $\mathbb{R}^3$ onto the Euclidean 2-space $\mathbb{R}^2$ image plane, defined by the image coordinate system (ICS) (see 3.7 on the right). The distance between the image plane and the origin of the OCS is defined by the focal length f of the camera. The position of the projection of a point $\mathbf{x}_{\text{cam}}$ in $\mathbb{R}^3$ space onto the image plane results from the intersection of the connecting line between the OCS origin and $\mathbf{x}_{\text{cam}}$, and the image plane. The projection is visualized in Fig. 3.7 with the red point and the dotted circle. Note, while coordinates in the GCS and OCS are expressed in metric units (e.g., cm, m), the ICS is based on pixels.

The projection is mathematically described by Eq. 3.2. In the intrinsic camera matrix $\mathbf{K}$, the focal length f is scaled by the image width (f_x) and image height (f_y), with the dimensions measured in pixels. The parameters c_x and c_y are the coordinates of the intersection of the image plane with the optical axis in the ICS.

$$\begin{aligned} \mathbf{x}_{\text{img}} &= \mathbf{K} \cdot \mathbf{x}_{\text{cam}} \\ \text{with } \mathbf{K} &= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.2}$$

Based on the derived knowledge about camera models and projection, the following shows how a keypoint detection tolerance based on a metric unit can be converted into a pixel tolerance.

3.3.1.2. Methodology for Local Keypoint-Individual Detection Tolerance

Based on the work of Kesseli et al. [12], a tolerance radius δ of 1.18 cm for potential model predictions is defined for each keypoint in an image. In [12], Kesseli et al. are able to project the given tolerance into the image plane by knowing a fixed length scale. This approach is possible because the images of the PTCs were taken without significant perspective distortion (from nearly orthogonal viewpoints). As a result, the distances between keypoints in the GCS could be directly related to pixel distances in the ICS, enabling the definition of a constant scaling factor per image. Importantly, this scaling factor was identical for all keypoints within a given image. The image dataset generated in this master thesis, however, differs fundamentally. Due to the varying orientations of the heliostats and the random camera positions, the heliostats appear perspectively distorted in the images. Only in rare cases, where the camera normal vector is aligned with the optical axis of a heliostat, a heliostat would appear undistorted and the application of the method from Kesseli et al. [12] would be valid. Consequently, a new method is required in this thesis that allows projecting a tolerance δ in a metric unit into the image plane even under perspective distortion. Therefore, a procedure was developed that first defines a keypoint tolerance zone in the GCS and then projects it into the ICS. The procedure is explained below.

Initially, the relevant matrices and coordinate systems from the previous subchapter are defined. The GCS is used to describe the position of the heliostats and the camera (see Tab. 3.3 and Tab. 3.4). The origin of the GCS is defined as the intersection point of the east-west and north-south symmetry axes of the heliostat field with the ground plane. By knowing the position (x, y, z) and orientation (pitch ϕ, roll θ, yaw ψ) of the camera, the transformation of a point $\mathbf{x}_{\mathrm{GCS}}$ from the GCS to the OCS is possible by using Eq. 3.1. The overall rotation matrix $\mathbf{R}$ is constructed using the Tait-Bryan convention (Chapter 3.2.2). In this convention, three successive rotations around the z, y, and x-axis are applied. The mathematical formulation is shown below in Eq. 3.3.

$$\mathbf{R} = \mathbf{R}_x(\phi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_z(\psi) \tag{3.3}$$

$\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$ and $\mathbf{R}_z(\psi)$ are defined as the standard rotation matrices in the 3D space [102, 103]. The rotation matrix is multiplied from the left to the $\mathbf{x}_{\mathrm{GCS}}$ so that the stepwise sequence of the rotations is in the correct order, although $\mathbf{R}$ is formally defined in a reverse order (see Eq. 3.1). The parameters of the intrinsic camera matrix $\mathbf{K}$ are set in the Blender settings and have the following values: $f_x = f_y$ = 4000, c_x = 3000 and c_y = 2000.

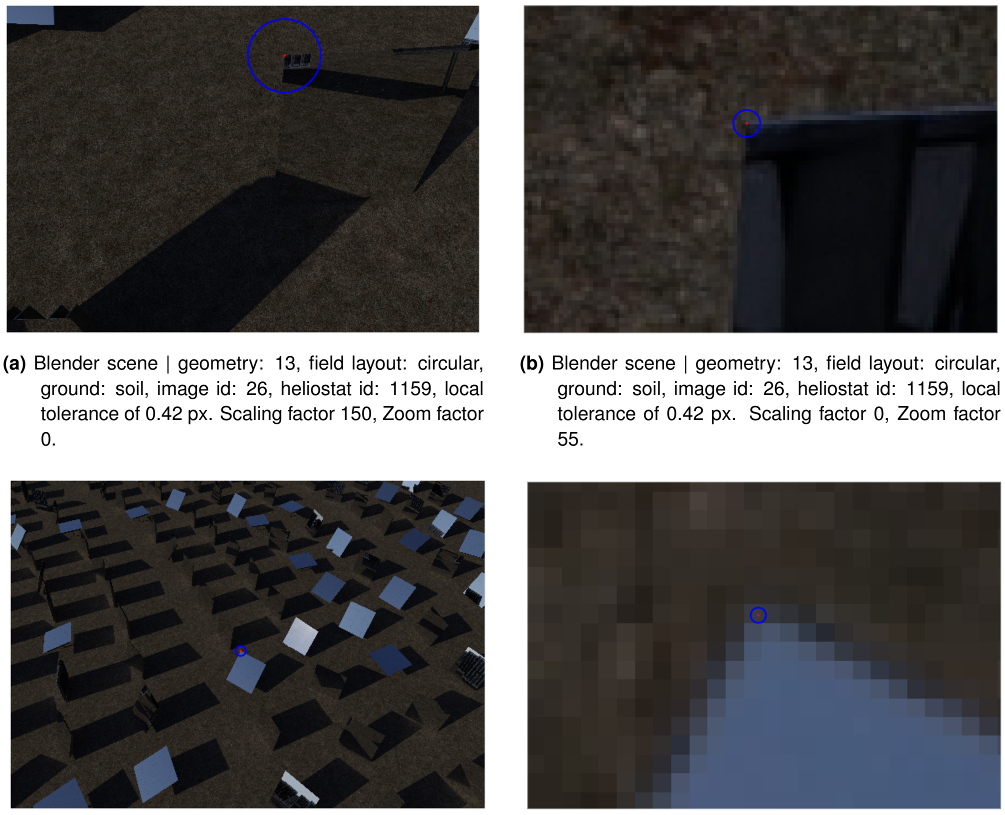
Based on the defined matrices and coordinate frames, Algorithm 1 provides the pseudocode to project the keypoint tolerance zone defined in the GCS into the ICS.

Algorithm 1 Computation of keypoint tolerance zone in the ICS for one image.

```
 1: Get camera position $(x, y, z)$ and orientation $(\psi, \theta, \phi)$ from drone.csv
 2: Transform camera's optical axis $\mathbf{x}_{\mathrm{opt,OCS}} = [0, 0, 1]$ from OCS to GCS using Eq. 3.1
 3: for each heliostat in the image do
 4:     for each keypoint of the heliostat do
 5:         Get keypoint position $\mathbf{x}_{\text{kp, GCS}}$ in GCS from Blender
 6:         Compute support point $\mathbf{x}_{\delta,\mathrm{GCS}}$ such that:
 7:             $|\mathbf{x}_{\delta,\mathrm{GCS}} - \mathbf{x}_{\text{kp, GCS}}| = \delta$            ▷ Defines the radius of the tolerance zone
 8:             $(\mathbf{x}_{\delta,\mathrm{GCS}} - \mathbf{x}_{\text{kp, GCS}}) \perp \mathbf{x}_{\mathrm{opt,GCS}}$     ▷ Ensures displacement lies in the image plane
 9:         Project $\mathbf{x}_{\delta,\mathrm{GCS}}$ into ICS using Eq. 3.2
10:         Get ground truth keypoint $\mathbf{x}_{\text{kp, coco}}$ from COCO file
11:         Compute pixel distance: $\delta_{\text{pixel}} = ||\mathbf{x}_{\delta,\text{proj}} - \mathbf{x}_{\text{kp, coco}}||$
12:         Write $\delta_{\text{pixel}}$ to COCO file
13:     end for
14: end for
```

The procedure is run through for each of the 11,000 images generated, setting $\delta = 1,18$ cm. Due to the unique combination of keypoint positions, camera positions and orientations, the algorithm results in a keypoint-individual tolerance radius in the image plane. Figure 3.9 shows the distribution of the generated tolerance radii and its statistics below. Compared to the previously used constant radius of three px, the newly computed, locally adaptive radii show significantly lower values. The median is only 0.355 px, with 50% of the values falling between 0.248 px and 0.507 px. Even the 95th percentile remains well below the former threshold, at just 0.848 px. Only a few outliers extend up to a maximum of 5.437 px. Overall, the reduced tolerance range may lead to a higher number of FP detections. As a result, the PCK metric could decrease, not necessarily because the model performs worse, but because the evaluation criterion has become stricter.

The following images in Fig. 3.8 illustrate how the local keypoint distance threshold behaves for two keypoints at different distances from the camera. This example shows that the keypoint tolerance zone varies significantly with the distance between the keypoint and the camera position.



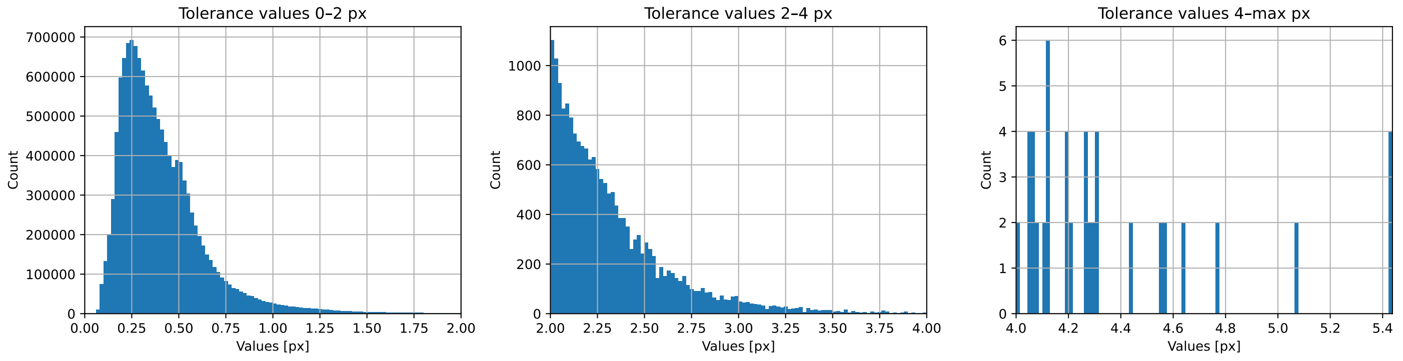
(a) Blender scene | geometry: 13, field layout: circular, ground: soil, image id: 26, heliostat id: 1159, local tolerance of 0.42 px. Scaling factor 150, Zoom factor 0.

(b) Blender scene | geometry: 13, field layout: circular, ground: soil, image id: 26, heliostat id: 1159, local tolerance of 0.42 px. Scaling factor 0, Zoom factor 55.

(c) Blender scene | geometry: 13, field layout: circular, ground: soil, image id: 121, heliostat id: 803, local tolerance of 3.09 px. Scaling factor 150, Zoom factor 0.

(d) Blender scene | geometry: 13, field layout: circular, ground: soil, image id: 121, heliostat id: 803, local tolerance of 3.09 px. Scaling factor 0, Zoom factor 330.

Figure 3.8.: Visualization of developed local keypoint tolerance for two keypoints with different distances to the camera. The tolerance radii have been enlarged 150 times on the left-hand images to increase their visibility. The zoom factor describes the optical enlargement of the image section, which includes the marked keypoint.



Min	5%	25%	Median	75%	95%	Max
0.067	0.158	0.248	0.355	0.507	0.848	5.437

Figure 3.9.: Distribution of the keypoint-individual distance threshold on generated image data and its' statistics (values in px).

3.3.2. Heliostat Pose Estimation for Improved Keypoint Detection

The introduction of this thesis (see Chapter 1) showed, that even a class-aware model that was trained on a large dataset with 20,000 images is still not able to detect all keypoints in an image. This chapter presents a post-processing method to partially complement missing keypoints. However, it includes learning further parameters that have to be additionally included in the generated COCO files. This method is therefore covered in the pre-processing section. The aim is to train the model on predicting the pose, defined by the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, of a heliostat in the OCS. Therefore, the two parameters have to be calculated and integrated into the training data. Their calculation is done separately in two steps below. The position and orientation of the GCS, the coordinates of the heliostats in the GCS, and the position and orientation of the OCS for each image can be assumed as given. The calculation of $\mathbf{t}_{\text{helio, OCS}}$ is shown in Algorithm 2 via pseudocode.

Algorithm 2 Computation of heliostat location in GCS and OCS.

```
 1: Get outer mirror corners $A$, $B$, $C$, $D$ in GCS for one heliostat from Blender (ordered: down-left, top-left,
    top-right, down-right)
 2: Compute diagonal midpoints:
 3:     $\mathbf{x}_{AC} \leftarrow \frac{1}{2}(A + C)$
 4:     $\mathbf{x}_{BD} \leftarrow \frac{1}{2}(B + D)$
 5: Compute heliostat center in GCS:
 6:     $\mathbf{t}_{\text{helio, GCS}} \leftarrow \frac{1}{2}(\mathbf{x}_{AC} + \mathbf{x}_{BD})$
 7: Load camera translation vector $\mathbf{t}_{\text{cam}}$ and rotation matrix $R_{\text{cam}}$ from drone.csv
 8: Transform to OCS:
 9:     Translation: $\mathbf{t}_{\text{helio, trans}} \leftarrow \mathbf{t}_{\text{helio, GCS}} - \mathbf{t}_{\text{cam}}$
10:     Rotation: $\mathbf{t}_{\text{helio, OCS}} \leftarrow R_{\text{cam}}^{\top} \cdot \mathbf{t}_{\text{helio, trans}}^{\top}$      ▷ $\mathbf{t}_{\text{helio, trans}}$ is a row vector
```

In a second step, the rotation matrix of the heliostat in the OCS is determined. The calculation is shown in Algorithm 3 via pseudocode.

Algorithm 3 Computation of heliostat orientation in GCS and OCS.

```
 1: Get outer mirror corners $A$, $B$, $C$, $D$ in GCS for one heliostat from Blender (ordered: down-left, top-left,
    top-right, down-right)
 2: Compute local axes of heliostat in GCS:
 3:     $\mathbf{x}_{\text{axis}} \leftarrow \frac{C-B}{\|C-B\|}$      ▷ horizontal edge (top)
 4:     $\mathbf{y}_{\text{axis}} \leftarrow \frac{B-A}{\|B-A\|}$      ▷ vertical edge (left)
 5:     $\mathbf{z}_{\text{axis}} \leftarrow \frac{\mathbf{x}_{\text{axis}} \times \mathbf{y}_{\text{axis}}}{\|\mathbf{x}_{\text{axis}} \times \mathbf{y}_{\text{axis}}\|}$      ▷ normal vector
 6: Construct heliostat rotation matrix in GCS:
 7:     $R_{\text{helio, GCS}} \leftarrow [\mathbf{x}_{\text{axis}} \mid \mathbf{y}_{\text{axis}} \mid \mathbf{z}_{\text{axis}}]$
 8: Load camera rotation matrix $R_{\text{cam}}$
 9: Transform heliostat rotation into OCS:
10:     $R_{\text{helio, OCS}} \leftarrow R_{\text{cam}}^{\top} \cdot R_{\text{helio, GCS}}$
```

A valid rotation matrix $R_{\text{helio, OCS}}$ must satisfy the condition that its column vectors are pairwise orthogonal and each has unit norm. However, representing a rotation using all three column vectors of a 3×3 matrix is redundant, as the third vector (e.g., $\mathbf{z}_{\text{axis}}$) can be computed as the normalized cross product of the first two (e.g., $\mathbf{x}_{\text{axis}} \times \mathbf{y}_{\text{axis}}$). To reduce redundancy and computational load, a six-dimensional representation of the rotation matrix, encoding only the first two column vectors, is learned. This idea is based on the works of Zhou et al. [106]. 3D rotations can also be represented using quaternions q. However, their representation of rotation is discontinuous, since the same rotation can be described by the same quaternion with a positive and a negative sign [106].

The two algorithms shown are applied to derive the parameters $\mathbf{t}_{\text{helio, OCS}}$ and $\mathbf{R}_{\text{helio, OCS}}$ for every heliostat on the 11,000 rendered images. The parameters are then added to the corresponding annotation of the corresponding COCO file. With the right model configuration, the model theoretically can now also learn the pose of a heliostat. By adding a randomness factor to the elevation and azimuth angle of each heliostat during data generation (see Chapter 3.2), the model is able to learn various poses during training. The value of the pose in the context of post-processing and the benefit for keypoint detection is shown visually in Fig. 3.10.

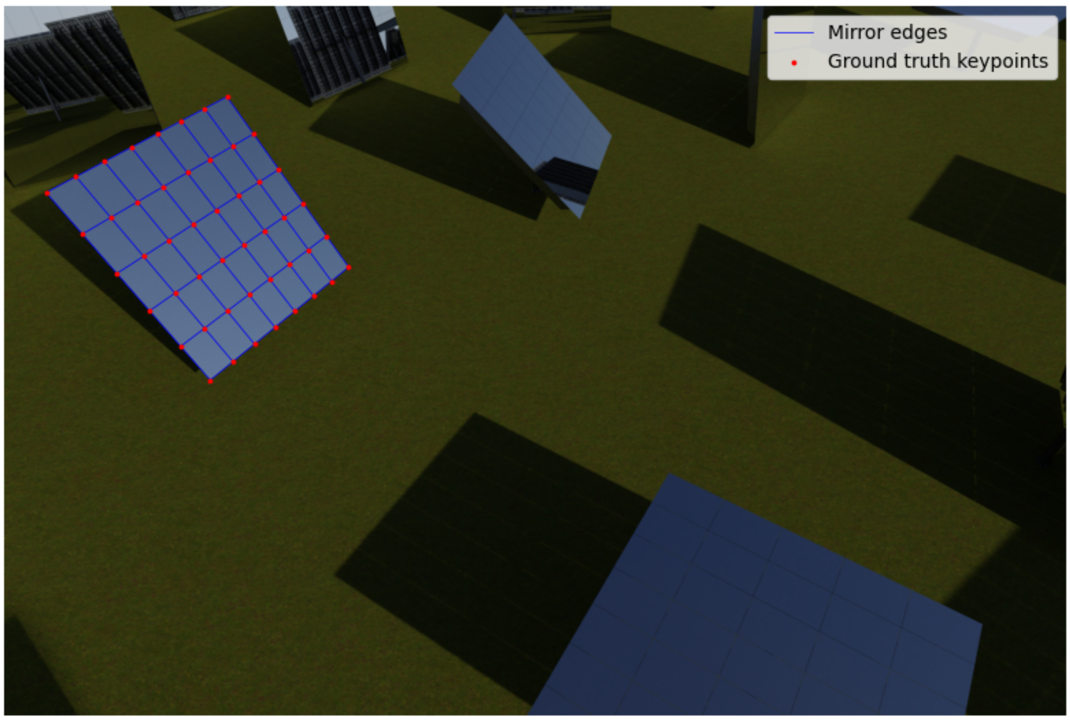


Figure 3.10.: Visualization of the ground truth pose | geometry: 1, field layout: circular, ground: grass, image id: 6, heliostat id: 149.

To visualize the ground truth pose for one heliostat, the mirror surface of the heliostat is approximated by a rectangular grid-like structure. Its dimensions are derived from known facet and gap sizes of the heliostat mirror surface. The grid is first defined in the OCS, then transformed using the calculated rotation $\mathbf{R}_{\text{helio, OCS}}$ and translation $\mathbf{t}_{\text{helio, OCS}}$, and finally projected onto the image plane using Eq. 3.2. The resulting projection overlays the actual mirror surface in the image, thereby providing a visual reference for the keypoint positions. In theory, if the pose is predicted accurately, the geometric constraints can be used to determine the position of unpredicted keypoints and interpolate the position of hidden ones. Furthermore, knowing the pose would enable a direct assignment of keypoints to individual heliostats, which is required for subsequent steps in the heliostat calibration. After supplementing the existing training data in two pre-processing steps, the following sections of Chapter 3 deal with the chosen cluster-based training strategy to develop a geometry-agnostic model.

3.4. Training Data Clustering

The underlying idea behind an agnostic detection model is, by definition, the ability to transfer features learned from one object class to others. To evaluate this transferability, it is therefore necessary to test the detector on unseen object classes. In the context of this master's thesis, each heliostat mirror design from Tab. 3.1, referred to as a geometry in the following, is categorized as a separate class. As already discussed in Chapter 3.2.1, certain geometries show remarkable similarities or differences due to variations in their number of facets and dimensions. It is therefore reasonable to cluster the geometries. Each cluster then comprises a set of geometries clearly distinguishable from those in other clusters. This approach enables the implementation of various training scenarios in which one cluster serves as the test dataset, while the remaining clusters are used to train the model. Also, the chosen approach is comparable to the k-fold cross-validation and thus has a theoretical foundation.

It is started with clustering the given geometries. The procedure is based on [107]. The parameters `n_panels_x`, `n_panels_y` and `middle_gap` are selected as cluster variables. The parameter `middle_gap` is a dichotomous parameter and can only take two values, true and false. By convention [107], it is converted into a binary variable by encoding, so that:

$$\texttt{middle\_gap} = \begin{cases} 1, & \text{if true} \\ 0, & \text{else.} \end{cases}$$

The euclidean distance is used as a measure to determine the distance between two geometries. The Ward method, a hierarchical agglomerative clustering algorithm, is used as the merging algorithm. The method is highly relevant in practice [107] and additionally aims to form clusters with the lowest possible internal variance. Minimizing the internal variance contributes to the formation of homogeneous clusters with high separability from other classes. The process of an agglomerative cluster algorithm can be illustrated graphically with a dendrogram. A dendrogram indicates the heterogeneity measure associated with a given cluster number. Figure 3.11 shows the dendrogram for the clustering of the heliostat geometries. The vertical axis shows the individual geometries. As a starting point, each geometry represents a separate cluster, with heterogeneity equal to 0. As the fusion progresses, the heterogeneity measure increases. The dendrogram then graphically connects the geometries that are fused at a particular stage [107].

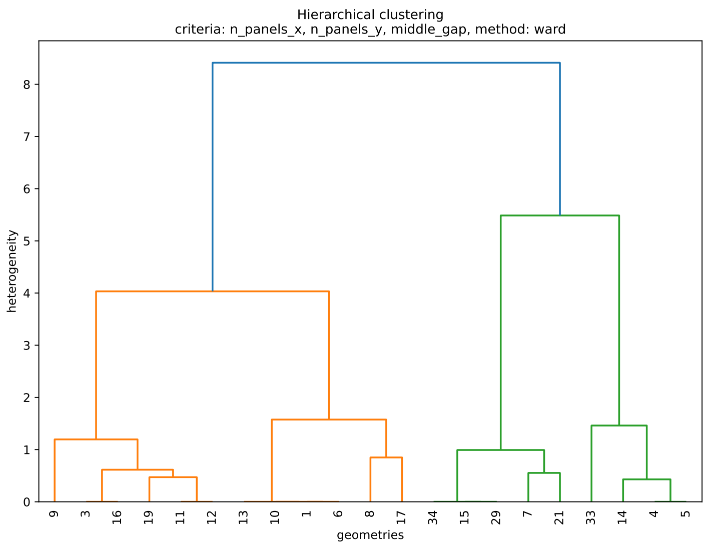


Figure 3.11.: Hierarchical clustering of heliostat geometries.

As a result, one must decide on ones own which cluster solution is considered the best. The elbow criterion is particularly useful for supporting this decision. According to this criterion, the optimal number of clusters k (the elbow) is the point at which the variance reduction decreases significantly as the number of clusters increases [107]. The application of the elbow method for the clustering of the geometries is graphically shown in Fig. 3.12. The elbow can be obtained for $k = 4$.

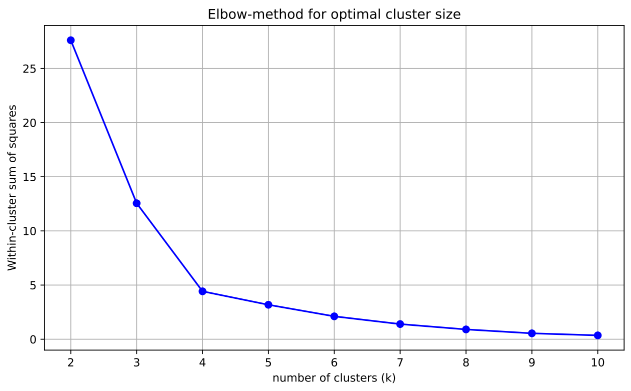


Figure 3.12.: Elbow-method for optimal cluster size.

With the determined cluster number, the assignment of the geometries to the respective clusters can be done using the dendrogram. For this purpose, an assumed horizontal cut is made in the dendrogram to cut exactly $k = 4$ branches. This cut is possible, e.g., for a heterogeneity of value three. The final assignment of the geometries to the clusters can be seen in Fig. 3.13.

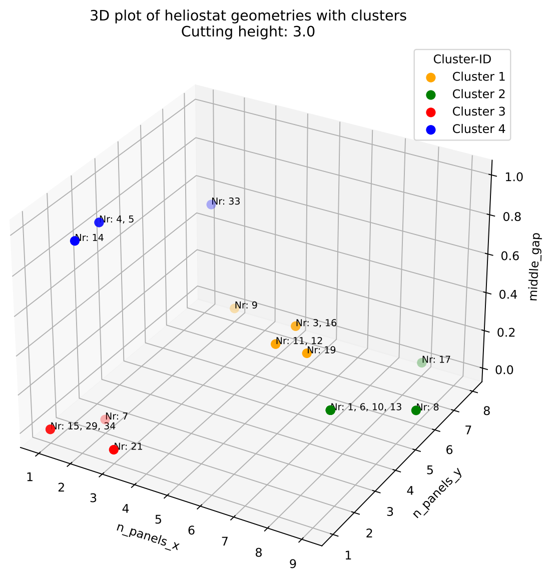
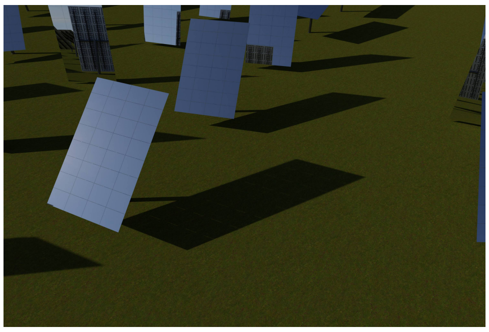
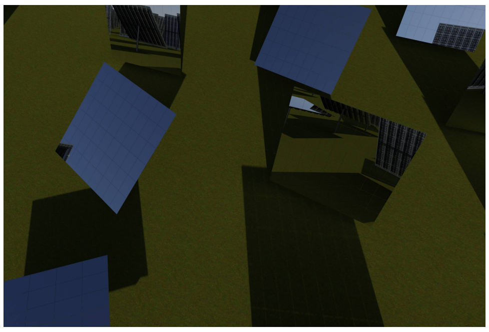


Figure 3.13.: 3D clustering of heliostat geometries.

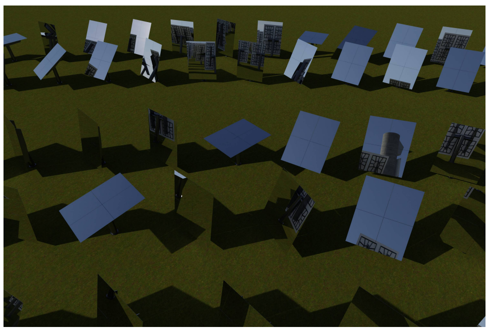
The individual clusters are well separated. Figure 3.14 below shows exemplary geometries from the developed clusters. Clusters one (Fig. 3.14a) and two (Fig. 3.14b) comprise geometries with a medium to large mirror design, with geometries in cluster two having more facets in the x-direction than those in cluster one. Furthermore, geometries 8 and 17 stand out in cluster two due to their large number of facets. Cluster three (Fig. 3.14c) contains geometries with a particularly small number of facets. Cluster four (Fig. 3.14d) comprises geometries with a middle gap. Based on the cluster constellation, various training scenarios are derived, evaluated, and compared with each other in the following section.



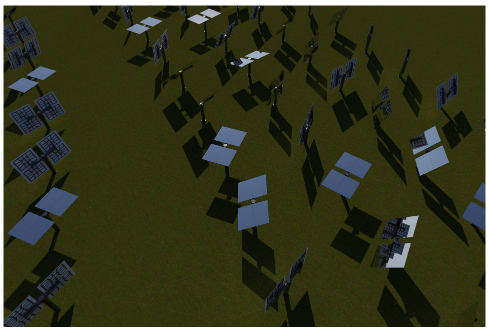
(a) Cluster 1 Blender scene | geometry: 3, field layout: circular, ground: grass, image id: 1.



(b) Cluster 2 Blender scene | geometry: 8, field layout: circular, ground: grass, image id: 9.



(c) Cluster 3 Blender scene | geometry: 7, field layout: circular, ground: grass, image id: 4.



(d) Cluster 4 Blender scene | geometry: 4, field layout: circular, ground: grass, image id: 2.

Figure 3.14.: Example geometries from the developed clusters. Resolution decreased to reduce file size.

3.5. Training Framework

A cluster-based training and evaluation approach is used to investigate geometry agnosticism. For this purpose, the existing geometries were divided into four clusters in the previous Section 3.4. Analogous to the k-fold cross-validation methodology, the following procedure is now used for model training: The set of clusters is defined as K with $K = \{1, 2, 3, 4\}$. In an iteration k with $k \in K$, cluster k is defined as the validation dataset. The three remaining clusters $K \setminus \{k\}$ are defined as the training dataset. In four iterations, the generalizability of the model is thus checked for different combinations of training and validation geometries. Furthermore, it is investigated how the integration of a specific cluster into the training dataset affects the model performance. Thus, in an iteration k, the model is not only trained with the clusters $K \setminus \{k\}$, but also in $m \in \{1, 2, 3\}$ iterations with one, two, and all three training clusters. The choice of the clusters in iteration m is determined based on the distance matrix between the cluster centers. Thus, for $m = 1$, the model is trained with the cluster whose cluster center has the smallest Euclidean distance to the center of the validation cluster. For $m = 2$, the cluster whose cluster center has the second smallest Euclidean distance to the center of the validation cluster is added to the training. For $m = 3$, all clusters $K \setminus \{k\}$ are included in the training.

At another level, it is examined how integrating or removing selected geometries from their clusters affects training and model performance. Three scenarios were defined for this purpose:

1. **Scenario *No Duplicates***: Duplicates are excluded from model development and evaluation. A geometry is defined as a duplicate if its parameters `n_panels_x` and `n_panels_y` match those of another geometry. The geometry with the smallest ID (see Tab. 3.1) among the duplicates is selected for this scenario. This scenario results in 13 geometries.

2. **Scenario *One Geometry per Cluster***: The cluster size is reduced to the extreme case of one geometry per cluster. For this purpose, the medoid of each cluster is selected.[11] Thus, this scenario includes four geometries.

3. **Scenario *All Geometries***: All 21 geometries listed in Tab. 3.1 are included in model development and evaluation.

Each model is trained (and tested) with two random seeds to reduce the influence of random effects (e.g., random weight initialization, data shuffling and data augmentation [109]) on model training and performance evaluation, thereby ensuring more robust and reliable results.

3.6. Testing Framework

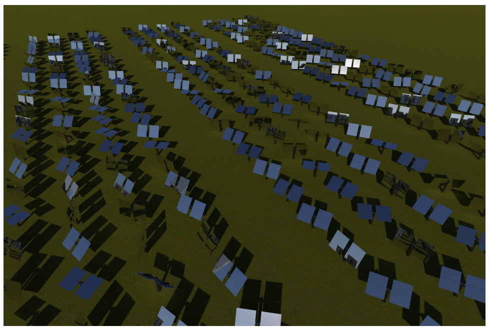
The model performance for a test dataset is evaluated based on the AP and the PCK. The PCK is evaluated for all predictions with a model confidence (i.e. heatmap peak value) higher than 0.5, and using a fixed tolerance radius of 3 px for ground truth to prediction assignment. This selection allows direct comparison with the studies by Broda et al. [19]. On the other hand, the PCK is determined with a projected variable tolerance radius of 1.18 cm. This method was derived in Chapter 3.3.1.2.

Significant discrepancies in model performance between the rectangular and circular scenes were found within the scope of the tests, without prejudging the results. Specifically, the tests for the circular scenes are significantly worse regarding the PCK. To present the test metrics for the rectangular scenes without distortion, tests for the rectangular and circular scenes were carried out separately. The separation allows representative test results to be presented for the rectangular scenes. The test metrics for the circular scenes were also collected for the sake of completeness. These and possible reasons for the discrepancies between the two field layouts can be found in Appendix C.1.

Drone images taken from a low altitude, combined with a flat camera angle, capture many heliostats and keypoints due to the strong depth effect. However, the bounding box centers and

[11] The medoid of a cluster is the data point for which the sum of distances to all other points in the cluster is the smallest. In contrast to the centroid, a medoid must actually exist. This means it is a real point from the data and not a calculated average [108].

keypoints are barely detected reliably at this image depth due to significant occlusions and overlaps. Such camera angles are rare in practical applications and can negatively influence the testing metrics. Therefore, the models will be further tested on datasets in which images with the above characteristics are filtered out. Based on visual inspection, as a quantitative filtering criterion, all samples with a camera pitch angle greater than 50 ° are excluded. Figure 3.15 shows two exemplary images, which are filtered out.


(a) Blender scene | geometry: 4, field layout: rectangular, ground: grass, image id: 49.

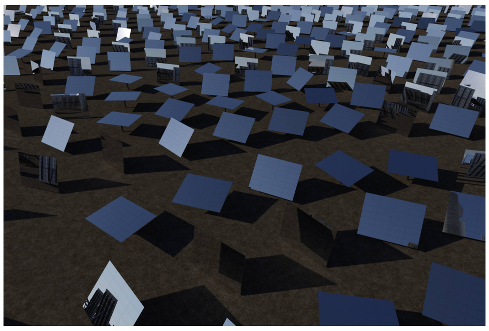

(b) Blender scene | geometry: 11, field layout: circular, ground: soil, image id: 44.

Figure 3.15.: Examples of filtered images for criterion pitch $\geq$ 50 °. Resolution decreased to reduce file size.

Testing is carried out using consistent test datasets across all scenarios. This strategy ensures comparability between the scenarios. The test datasets are based on the clusters defined in the *No Duplicates* scenario, as they do not include duplicates (as in scenario *All Geometries*) or are limited to a small size of geometries (as in scenario *One Geometry per Cluster*). Circular scenes and image data with a camera pitch angle of $\geq 50^\circ$ are filtered out. Within the *No Duplicates* scenario, using the same data for validation and testing is unproblematic, as no model tuning during training is performed based on the validation results. Table 3.5 shows the four test clusters and their corresponding dataset sizes.

Table 3.5.: Test datasets corresponding to the four validation clusters for all scenarios. Datasets are filtered for pitch $\geq$ 50 °. Only rectangular scenes are included. The dataset size indicates the number of synthetic images.

Test Cluster	Dataset Size Unfiltered	Dataset Size Filtered
1	1000	780
2	750	580
3	750	578
4	750	566

3.7. Hyperparameter Settings

By combining various elements, the model architecture has several hyperparameters. However, as the scope of this work lies on the geometry agnosticism and to ensure comparability, they are not tuned in this work but mostly adopted from the work of Broda et al. [19]. Table 3.6 shows the most important model and training hyperparameters.

Table 3.6.: Thematically grouped overview of selected hyperparameters in the model and training configuration. *Cropping is only used in model training. **Increased to 30,000 for model testing. Abbreviations: bbox = bounding box, kpt = keypoint, reg = regression. The settings were taken from the work of Broda et al. [19].

Hyperparameter	**Value**
1. Data and Pre-Processing	
batch size	8
crop width*	2048
crop height*	2048
max peaks**	2000
2. Loss and Training Objective	
reduction	sum
weights bbox heatmap	1.0
weights bbox reg offsets	1.0
weights bbox reg bounds	5.0
weights kpt heatmap	1.0
weights kpt reg offsets	5.0
weights pose estimation	5.0
bbox sigma	64
kpt sigma	16
3. Training Strategy and Stability	
max epochs	200
learning rate	0.001
optimizer	AdamW
lr scheduler	OneCycleLR

The model uses random cropping as a data augmentation technique to synthetically increase the training dataset variance and to improve computational efficiency [110]. To ensure tensor shape compatibility during computations, the maximum number of allowed model predictions per (cropped) image and heatmap must be pre-initialized by defining the parameter `max_n_peaks`. Keypoints and bounding box centers are counted separately, but share the same `max_n_peaks` limit. Tests have shown that a value of 2,000 for model training is sufficient. Increasing this value by a factor of ten did not lead to any significant improvement in model performance during testing that would have justified the additional computational cost. During testing, the PCK is only evaluated on visible keypoints, although the model might also predict occluded ones. For testing, `max_n_peaks` is set to 30,000 because unlike during training, the full image is evaluated. This value was verified to be sufficiently high to ensure that no potential predictions are lost due to limitations of the model hyperparameters. The multi-task loss to be minimized is the sum of the weighted losses of the individual tasks. Empirical pre-tests in previous works of Broda et al. [19] have shown that using a weight of five for the regression loss terms of the bounding box bounds, the keypoint offsets and the pose estimation task, improves model performance. Therefore, this value is adopted. All other loss terms are not amplified, but their weight is set to one. The bounding box sigma and keypoint sigma (both in px) indicate the 2D Gaussian standard deviation for ground truth heatmap creation. The training process uses the AdamW optimizer [111]. For fast model convergence, a OneCycle learning rate schedule is applied, where the learning rate is increased and then decreased over the course of training [112].

Intermediate Results

Chapter 3 forms the methodological core of this thesis. It considers all preparatory steps of model training. First, the existing model architecture was examined. Then, the focus was placed on the generation of artificial training data. The process pipeline and several sample images were examined in detail, demonstrating the quality and quantity of the generated image data. For the specific purposes of this master's thesis, the generated data was pre-processed in two steps. First, a procedure was presented that calculates an individual tolerance radius in the image plane for each keypoint. A post-processing algorithm based on the additionally added model task of pose estimation was proposed. The clustering of the derived heliostat geometries lays the direct foundation for the implementation of various training and testing scenarios. The developed framework for model training and testing, as well as the hyperparameter selection, was also presented. The detailed evaluation of the defined training scenarios begins in the following.

4. Model Evaluation

In the previous chapter, three experimental scenarios were introduced, in which different models are developed based on variations in training data volume and combinations of training and testing clusters. In this chapter, the results of these model evaluations are compared both across the scenarios and against class-aware baseline models, which were only trained on one heliostat geometry. Furthermore, the transferability of the models to real-world data is assessed. Moreover, the functionality of the developed pose estimation approach is examined as part of the post-processing pipeline.

4.1. Scenario *No Duplicates*

In this scenario, duplicates are excluded from model development and evaluation. The geometry with the smallest ID among the duplicates is selected for this scenario. This filters out eight duplicates distributed across five geometries. The selection of the geometry with the smallest ID is justified, as six out of the eight removed duplicates are not only duplicates in terms of the number of facets but also share the exact same facet dimensions (see Tab. 3.1 s_x and s_y). Thus, these geometries were parametrized identically in Blender and show no geometric differences. The remaining two duplicates exhibit only minor geometric differences compared to their respective reference geometries. Due to the negligible nature of these differences, the implementation of a more elaborate selection criterion was deemed scientifically unjustified for the purposes of this scenario. Overall, this scenario results in 13 geometries. The 13 geometries, their cluster assignment, as well as the cluster size are presented in Tab. 4.1.

Table 4.1.: Overview of clusters in scenario *No Duplicates*. Total amount of images: 6500.

Cluster	Geometries	Dataset Size
1	3, 9, 11, 19	2000
2	1, 8, 17	1500
3	7, 15, 21	1500
4	4, 14, 33	1500

The training cases are derived based on the distance matrix between the centers of the four clusters, as shown in Tab. 4.2. For iteration $k = 1$, with cluster 1 serving as the test cluster, the training scenarios are defined in ascending order of distance as follows: training with cluster 2 (distance: 4.4), training with clusters 2 and 4 (distance: 6.7), and training with clusters 2, 4, and 3 (distance: 7.8). The evaluation strategy is referred to as *Close First* (CF). This approach can be applied analogously to iterations with test clusters 2, 3, and 4. In total, this yields twelve distinct combinations of training and testing clusters. Figure 4.1 provides

Table 4.2.: Distance matrix between clusters for scenario *No Duplicates*.

Cluster	1	2	3	4
1	0.00	4.4	7.8	6.7
2	4.4	0.00	6.4	5.4
3	7.8	6.4	0.0	1.6
4	6.7	5.4	1.6	0.0

a visualization of the training and testing results for the twelve training cases. Each subplot corresponds to one test cluster and presents the results for different training cluster combinations. The performance metrics (AP, PCK@3px, and PCK@1.18cm) are plotted on the y-axis. The models were tested on the test clusters presented in Tab. 3.5. Complementary to Fig. 4.1, Tab. 4.3 provides the corresponding numerical values.

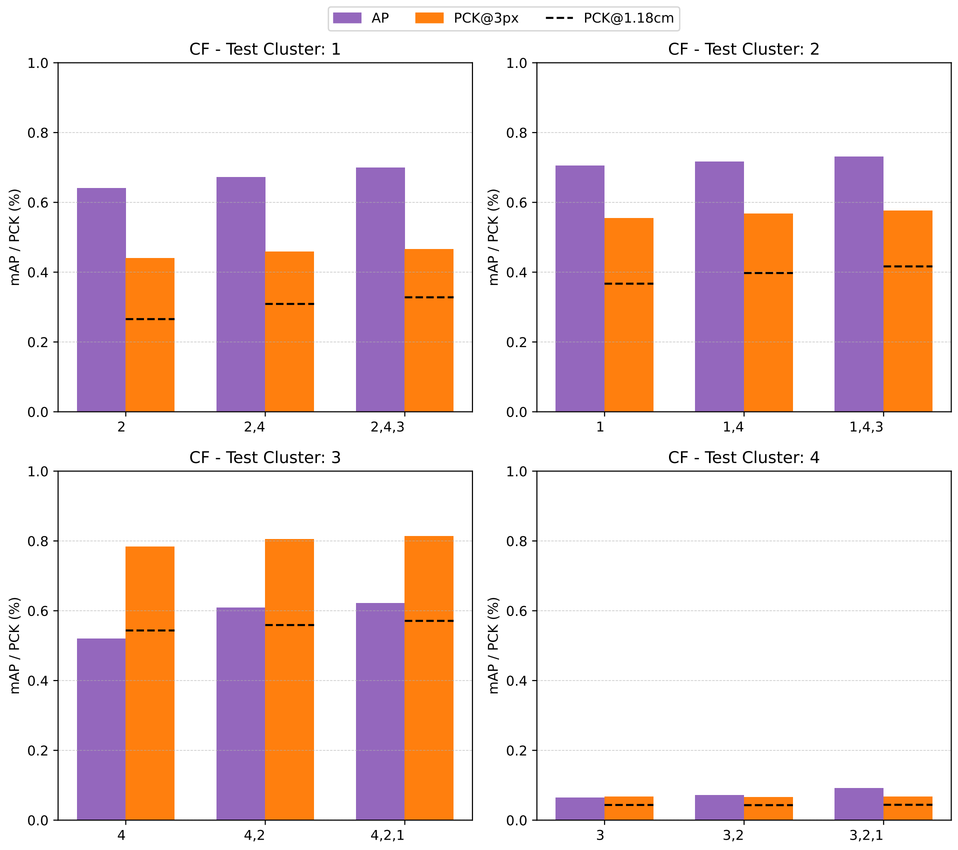


Figure 4.1.: Scenario *No Duplicates* | Performance metrics, grouped by test cluster and averaged for two seeds. AP and PCK are given in %. Order of training cluster merging: Close First (CF).

Test cluster 1 comprises medium to large geometries (see Fig. 3.14). The model was initially trained solely with cluster 2, which is most similar in terms of geometric structure. By adding further training clusters, the AP increases from 64.2% to 70.0%, while the PCK@3px rises from 44.1% to 46.7%. Including additional geometries has a stronger impact on the AP than on the PCK. The stricter metric PCK@1.18cm, which defines a narrower tolerance range, performs on average 15.5 percentage points worse than the PCK@3px.

Table 4.3.: Scenario *No Duplicates* | Numerical values of performance metrics, grouped by test cluster and averaged for two seeds. Order of training cluster merging: Close First (CF).

Test Cluster	Training Cluster	PCK@3px	PCK@1.18cm	AP
	2	44.1	26.6	64.2
1	2,4	45.9	30.9	67.3
	2,4,3	46.7	32.8	70.0
	1	55.5	36.7	70.5
2	1,4	56.9	39.8	71.6
	1,4,3	57.6	41.7	73.2
	4	78.5	54.4	52.1
3	4,2	80.6	56.0	60.9
	4,2,1	81.5	57.1	62.2
	3	6.7	4.4	6.4
4	3,2	6.6	4.3	7.1
	3,2,1	6.7	4.4	9.3

Test cluster 2 contains geometries that are similar in size or larger than those in cluster 1. As before, the model is initially trained with the most geometrically similar cluster (cluster 1). By including additional training clusters, the AP increases from 70.5% to 73.2%, while the PCK@3px rises from 55.5% to 57.6%. The stricter metric PCK@1.18cm again performs significantly lower than the PCK@3px (-17.3 percentage points on average). Notably, generalization from cluster 1 to cluster 2 results in better performance than vice versa.

Test cluster 3 contains small geometries and achieves the highest PCK@3px values (81.5% for training with all three clusters). In contrast, the AP remains below the levels observed for clusters 1 and 2. However, a significant performance gain in the AP can be observed when moving from training solely with cluster 4 to combinations including clusters 1 and 2 (+10.1 percentage points). Nevertheless, the AP stays lower overall compared to the tests on clusters 1 and 2.

Test cluster 4 contains geometries with a middle gap. The model performs significantly worse in both object and keypoint detection than in the previous cases across all three training stages (AP: 7,6%, PCK@3px: 6.7%, values averaged for three training stages).

Interpretation of Results

The results demonstrate the model's overall generalization capability across different geometric clusters. Training with geometrically similar clusters provides a stable foundation: even training with a single, structurally similar cluster (e.g., training cluster 1 and testing cluster 2, and vice versa) yields promising performance in the respective test scenarios. The successive inclusion of additional, less similar clusters then only leads to moderate improvements. This finding highlights the importance of structural similarity for successful model transfer.

Particularly noteworthy is the model's performance on small geometries (testing on cluster 3). In this case, the model achieves the best keypoint localization results (PCK). At the same time, the AP remains below the values observed in other clusters. One possible explanation for the increased PCK is the lower keypoint density per image caused by the smaller geometries, which leads to a less cluttered prediction context and thus simplifies the model's task. The lower AP, by contrast, may result from the fact that the training data contains geometries featuring middle gaps (cluster 4) or large bounding boxes (clusters 1 and 2), which differ significantly from the test geometries.

Interesting results are also evident with test cluster 4. The performances for object and keypoint detection consistently show the poorest test metrics, regardless of the configuration of the training data. Increasing the geometric diversity in the training set does not lead to any significant performance improvements for this cluster. This observation suggests that certain structural features, such as a middle gap, cannot be adequately approximated by other geometry types within the current model configuration. Furthermore, it is noteworthy that the generalization behavior is asymmetric: while generalization from cluster 4 to cluster 3 works relatively well, the reverse direction fails.

The above analysis shows in particular the influence of the training cluster closest to the testing cluster on the model performance. This means that the influence of the clusters added in subsequent iterations cannot be examined in isolation. **Therefore, in the following, the strategy *Far First (FF)* is considered**, in which the training cluster with the greatest Euclidean distance to the test cluster is used first for training. This is followed by the cluster with the second greatest distance, and finally all three training clusters are included. Figure 4.2 provides a structured visualization of the training and testing results for this scenario. The test metric values corresponding to training with all three clusters are the same as in the previous case and are therefore adopted from Tab. 4.3. The diagrams are not examined with the same granularity as in the preceding analyses. Instead, the emphasis is placed on extracting overarching insights by interpreting the broader training and test outcomes. The analysis therefore moves beyond individual metric values to identify general trends and underlying patterns. Concrete numerical values can be found in the complementary Tab. 4.4.

Test Cluster 1 & 2
Training begins with cluster 3, which is geometrically the most distant from the test clusters (small vs. large geometries). This strong discrepancy is reflected in low AP scores. The PCK performs better than the AP when trained on the more distant clusters 3 and 4. The test metrics increase significantly when the more similar clusters are added to the training set.

Test Cluster 3
The AP increases substantially when the more similar clusters 2 and 4 are added to the training set. In contrast, the improvement in the PCK remains moderate.

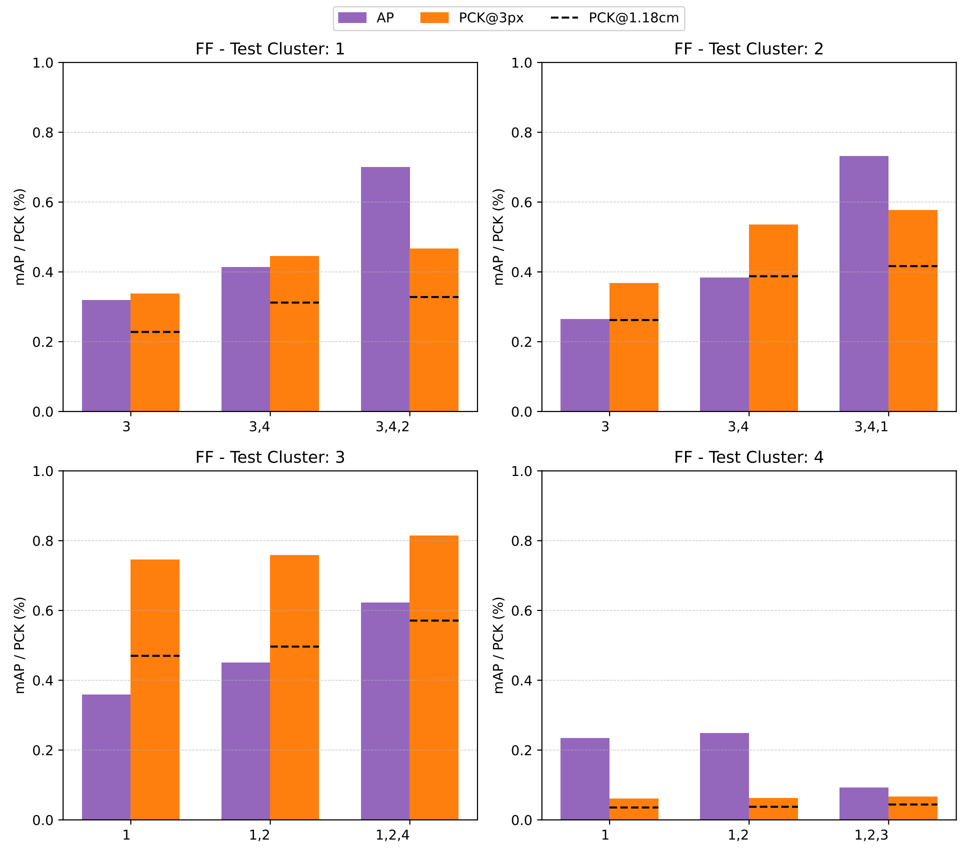


Figure 4.2.: Scenario *No Duplicates* | Performance metrics, grouped by test cluster and averaged for two seeds. Order of training cluster merging: Far First (FF).

Table 4.4.: Scenario *No Duplicates* | Numerical values of performance metrics, grouped by test cluster and averaged for two seeds. AP and PCK are given in %. Order of training cluster merging: Far First (FF).

Test Cluster	Training Cluster	PCK@3px	PCK@1.18cm	AP
1	3	33.7	22.8	31.8
	3,4	44.6	31.2	41.3
	3,4,2	46.6	32.8	70.0
2	3	36.8	26.2	26.4
	3,4	53.5	38.8	38.3
	3,4,1	57.6	41.7	73.2
3	1	74.6	47.0	35.8
	1,2	75.8	49.7	45.0
	1,2,4	81.5	57.1	62.2
4	1	6.1	3.6	23.4
	1,2	6.2	3.8	24.8
	1,2,3	6.7	4.4	9.3

Test Cluster 4
Both AP and PCK remain at relatively low levels. Notably, the integration of cluster 3 into the training process leads to a deterioration in the object detection performance. This is indicated by decreasing AP values. The PCK is largely unaffected by this change.

Interpretation of Results
The results confirm that object detection generalization is more effective between geometrically similar clusters (e.g., clusters 1 and 2), which suggests that bounding box prediction relies on global structural features. In contrast, keypoint detection, as a locally defined task, remains more robust under geometric variation, as reflected by consistently higher PCK values compared to AP when training and testing clusters differ significantly (testing cluster 1 and 2 on clusters 3 and 4). The choice of the training clusters has a major impact on model performance. With the exception of test cluster 4, integrating geometrically closer clusters leads to clear improvements across the test metrics, while more distant clusters contribute only marginal gains (see Fig. 4.1). Model performance declines due to adding another cluster only when training on clusters 1, 2, and 3 and testing on cluster 4. In this case, object detection performs worse than when the model is trained on clusters 1 and 2 alone. The findings highlight the relevance of the clustering strategy applied in this work, particularly the distinction between the *Close First* and *Far First* training sequences based on inter-cluster distances.

4.2. Scenario *One Geometry per Cluster*

In this scenario, the cluster size is reduced to the extreme case of one geometry per cluster. The aim is therefore to investigate how the significant reduction in training data and geometries affects model performance in testing. For this purpose, the medoid of each cluster is selected. This results in four geometries falling into this scenario. The four geometries, their cluster assignment, as well as the cluster size are presented in Tab. 4.5.

Table 4.5.: Overview of clusters in scenario *One Geometry per Cluster*. Total amount of images: 2000.

Cluster	Geometry	Dataset Size
1	11	500
2	8	500
3	7	500
4	4	500

With regard to the generation of training cases in this scenario, the procedure follows the same approach as described in the previous section. The order of the training clusters for iterative training is determined based on the *Close First* strategy, as previous results have shown that training with geometries similar to the test cluster leads to improved performance. The corresponding distance matrix is provided in Tab. 4.6 below.

Table 4.6.: Distance matrix between clusters for scenario *One Geometry per Cluster*.

Cluster	1	2	3	4
1	0.00	5.10	8.12	8.06
2	5.10	0.00	5.48	5.39
3	8.12	5.48	0.00	1.00
4	8.06	5.39	1.00	0.00

The evaluation of the model tests is based on the established metrics AP, PCK@3px, and PCK@1.18cm. The datasets listed in Tab. 3.5 are used again for model testing, allowing for a direct comparison of model performance across the different scenarios. Figure 4.3 provides a visualization of the testing results for the twelve training cases. Complementary to Fig. 4.3, Tab. 4.7 provides the corresponding numerical values.

This section focuses on comparing the test results of the current scenario with those obtained in the *No Duplicates (CF)* scenario. As can be computed from Tab. 4.7, the average decrease in the AP across test clusters 1 to 3 is 19.8 percentage points compared to the values in Tab. 4.3, while a smaller drop of 2.8 percentage points is observed for cluster 4. Regarding PCK@3px, the average decrease across all four clusters is 2.6 percentage points. Consistent with the *No Duplicates (CF)* results, the PCK@1.18cm metric yields lower values due to its in general stricter tolerance radius.

Interpretation of Results

The results indicate that, despite a significant reduction in the training dataset size from 2000 and 1500 images per cluster in the *No Duplicates* scenario to only 500 images per cluster in this scenario, the model is still able to generalize well to keypoints without substantial performance losses. However, the AP values show a much more notable drop. This contrast suggests that the prediction of bounding box centers is particularly sensitive to the amount of training data and geometries available. Test metrics for test cluster 4 also decline due to the reduced training set size. However, due to the already limited generalization capability observed for this cluster, further detailed comparisons are of limited interpretive value.

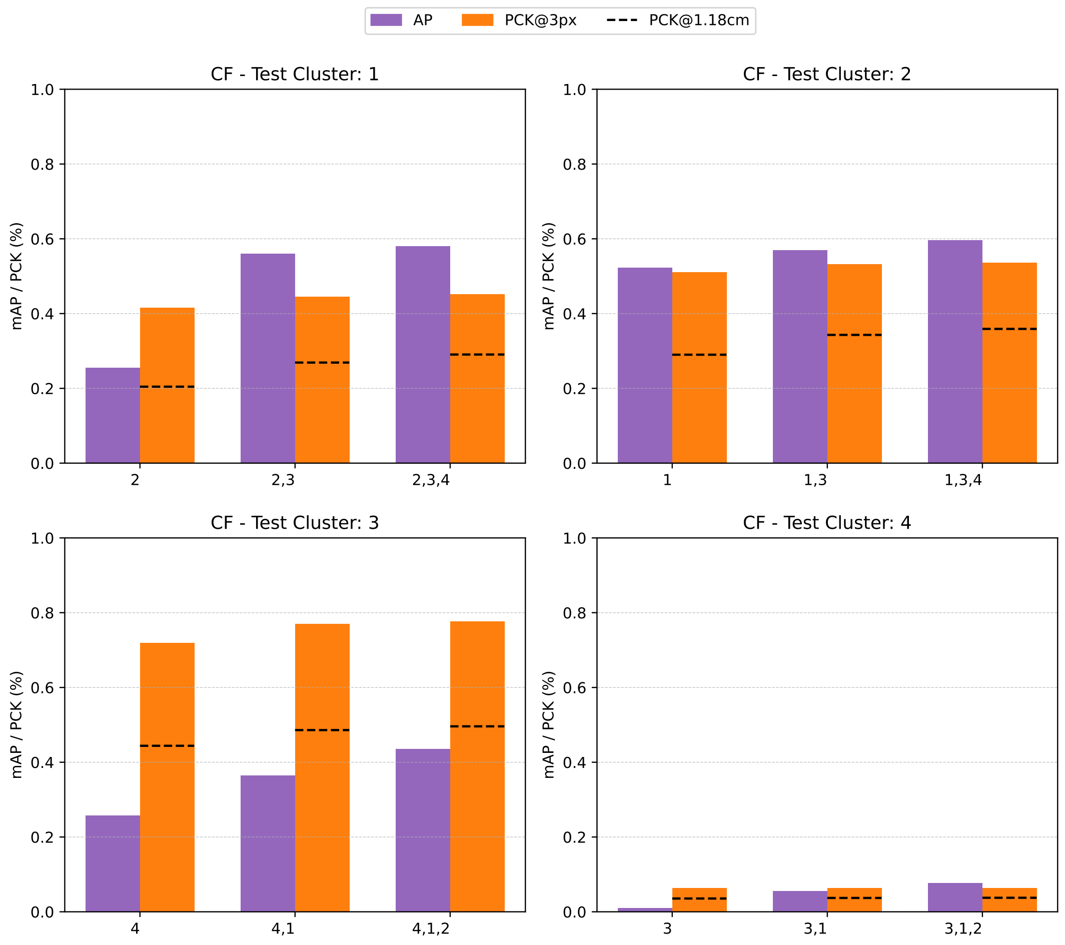


Figure 4.3.: Scenario *One Geometry per Cluster* | Performance metrics, grouped by test cluster and averaged for two seeds. Order of training cluster merging: Close First (CF).

Table 4.7.: Scenario *One Geometry per Cluster* | Numerical values of performance metrics, grouped by test cluster and averaged for two seeds. AP and PCK are given in %. Order of training cluster merging: Close First (CF).

Test Cluster	Training Cluster	PCK@3px	PCK@1.18cm	AP
	2	41.6	20.5	25.5
1	2,3	44.6	26.9	56.1
	2,3,4	45.2	29.1	58.1
	1	51.1	29.0	52.2
2	1,3	53.3	34.3	57.0
	1,3,4	53.7	36.0	59.6
	4	72.0	44.4	25.8
3	4,1	77.0	48.6	36.4
	4,1,2	77.7	49.6	43.6
	3	6.4	3.6	1.1
4	3,1	6.4	3.7	5.6
	3,1,2	6.4	3.8	7.7

4.3. Scenario *All Geometries*

In the *All Geometries* scenario, all 21 geometries listed in Tab. 3.1 are included. The 21 geometries, their cluster assignment, as well as the cluster size are presented in Tab. 4.8. Compared to scenario *No Duplicates*, eight additional geometries are included in the training and evaluation. Except of geometries 10 and 34, these additional entries are not only duplicates but also share the same facet dimensions (s_x, s_y) as their corresponding counterparts in the *No Duplicates* scenario. The reason for this is the missing technical data for some geometries, which was supplemented by averaging values from the available geometries (see Chapter 3.2). The aim is therefore to investigate how the increase in training data and geometries affects model performance in testing.

Table 4.8.: Overview of clusters in scenario *All Geometries*. Total amount of images: 10500. *Marked geometries are new compared to scenario *No Duplicates*.

Cluster	Geometries	Dataset Size
1	3, 9, 11, 12*, 16*, 19	3000
2	1, 6*, 8, 10*, 13*, 17	3000
3	7, 15, 21, 29*, 34*	2500
4	4, 5*, 14, 33	2000

Compared to the *No Duplicates* scenario, the training set in the current setting includes 1000 additional images for cluster 1, 1500 for cluster 2, 1000 for cluster 3, and 500 for cluster 4. With regard to the creation of training cases in this scenario, the procedure follows the same approach as in the previous sections. The order of the training clusters for iterative training is determined based on the *Close First* strategy. The corresponding distance matrix is shown in Tab. 4.9.

Table 4.9.: Distance matrix between clusters for scenario *All Geometries*.

Cluster	1	2	3	4
1	0.0	4.1	6.7	5.5
2	4.1	0.0	7.4	6.1
3	6.7	7.4	0.0	1.7
4	5.5	6.1	1.7	0.0

The datasets listed in Tab. 3.5 are used again for model testing. Figure 4.4 provides a visualization of the testing results for the twelve training cases in this scenario. Complementary to Fig. 4.4, Tab. 4.10 provides the corresponding numerical values.

Comparable to the previous section 4.2, this section focuses on comparing the test results of the current scenario with those obtained in the *No Duplicates (CF)* scenario. As can be computed from Tab. 4.10 and Tab. 4.3, the average increase in AP across the test cases of test clusters 1 to 3 is 2.4 percentage points while a smaller increase of 0.1 percentage points is observed for cluster 4. Regarding the PCK@3px, the average increase across all four clusters is 0.6 percentage points. Cluster 4 is not an outlier in this case. Analogous to the other scenarios, the PCK@1.18cm metric yields lower values.

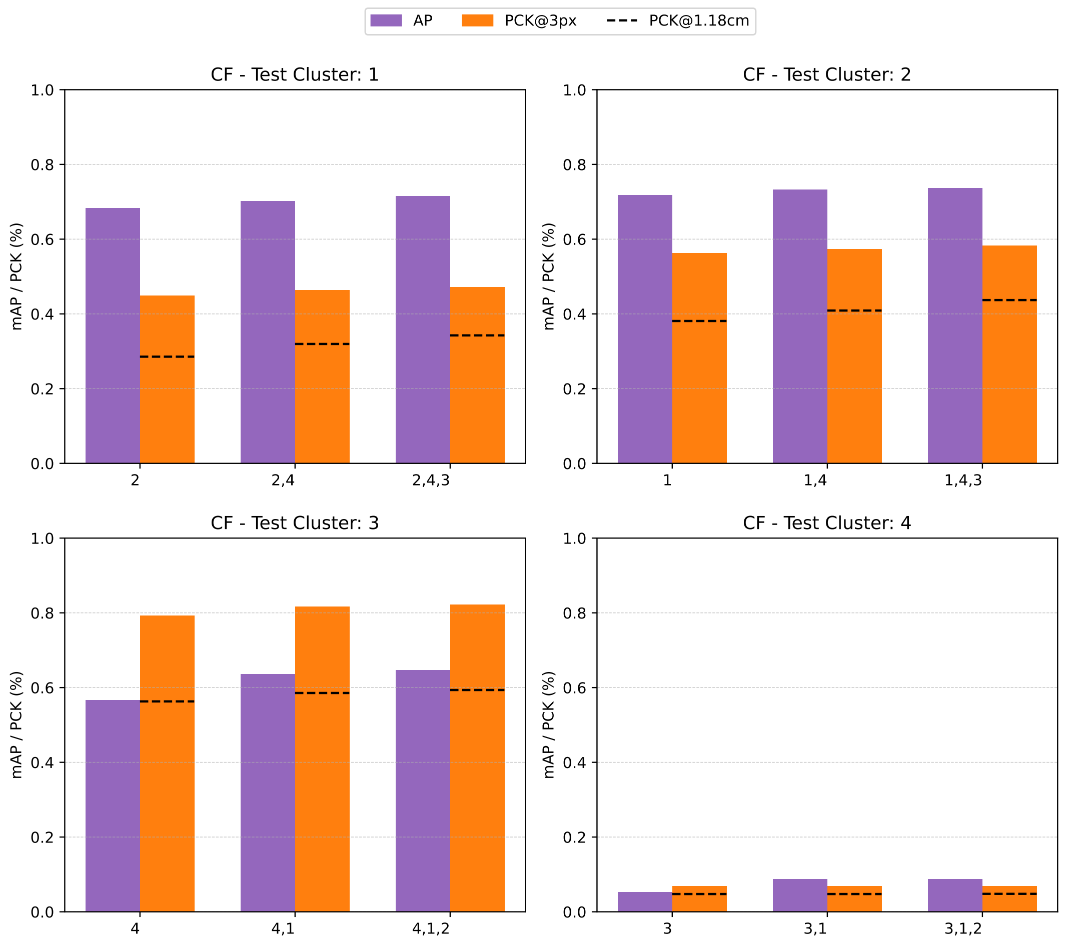


Figure 4.4.: Scenario *All Geometries* | Performance metrics, grouped by test cluster and averaged for two seeds. Order of training cluster merging: Close First (CF).

Interpretation of Results

As expected, increasing the size of the training dataset leads to improved model performance during testing. Thus, this scenario yields the best overall results across all test clusters observed in this study. However, the diagram also indicates that the model performance approaches a saturation point: the effect of adding further geometries is much smaller compared to the performance increase observed between the *One Geometry per Cluster* and *No Duplicates* scenarios. Due to the design of this scenario, the effect of adding additional geometries (IDs 10 and 34) cannot be assessed independently from the simultaneous increase in the overall size of the training dataset. While a further increase of the training dataset by including new geometries instead of duplicates could allow for additional insights, such additions would deviate from practical relevance. This is because the additional geometries would be geometries that are not currently used in real-world applications.

Table 4.10.: Scenario *All Geometries* | Numerical values of performance metrics, grouped by test cluster and averaged for two seeds. AP and PCK are given in %. Order of training cluster merging: Close First (CF).

Test Cluster	Training Cluster	PCK@3px	PCK@1.18cm	AP
1	2	44.9	28.6	68.3
	2,4	46.3	32.0	70.2
	2,4,3	47.2	34.3	71.6
2	1	56.3	38.1	71.8
	1,4	57.4	40.9	73.3
	1,4,3	58.3	43.7	73.6
3	4	79.3	56.3	56.7
	1,4	81.7	58.6	63.7
	1,4,2	82.2	59.4	64.8
4	3	6.9	4.8	5.4
	1,3	6.9	4.8	8.8
	1,3,2	6.9	4.8	8.8

4.4. Comparison to Baseline Models

In addition to comparing the performance of the geometry-agnostic models across the scenarios, the model performance is also evaluated in relation to class-aware models, which were trained on one mirror geometry. This comparison is conducted exclusively for the test results with rectangular fields. For this purpose, three class-aware models from previous work by Broda et al. [19] are used as so-called baseline models, each of which was trained on a different dataset size. These baseline models were trained with synthetic data of the CESA1[12] mirror geometries. The heliostat designs are listed in Tab. 4.11.

Table 4.11.: Data of mirror geometries of the CESA1 heliostats. n_x, n_y denote the number of panels in x and y directions. Both heliostat types have a middle gap.

Id	Plant	n_x	n_y	middle_gap
47	Cesa1 (Spain)	2	6	true
48	Cesa1 (Spain)	4	6	true

Since the class-aware baseline models are designed for geometry-specific learning, they were trained and tested on synthetic data from the CESA1 geometries. The test data was generated in this work following the procedure described in Chapter 3. The test dataset comprises 368 images and consists of two rectangular fields including heliostats of IDs 47 and 48. It only includes images with a camera pitch angle of $\leq 50°$. Table 4.12 lists the three baseline models and the numerical values of their test metrics.

In the three scenarios *No Duplicates*, *One Geometry per Cluster* and *All Geometries*, twelve training cases were developed respectively and evaluated on consistent test datasets. This

[12]CESA1 test plant owned and operated by CIEMAT

Table 4.12.: Class-aware baseline Models for three different train dataset sizes. Test dataset size: 368. Evaluation only on rectangular fields. AP and PCK are given in %.

Baseline Model	Train Dataset Size	AP	PCK@3px
B2	2,000	80.52	53.50
B5	5,000	84.12	55.20
B20	20,000	86.13	57.20

setup allows for a direct comparison of the model performances across the scenarios. However, these results do not permit an adequate comparison with the baseline models, as the baseline models are evaluated on the CESA1 dataset. To ensure comparability, the trained models are now evaluated on the CESA1 dataset. The evaluation is based on the AP and PCK@3px metrics. It is important to note that the size of the training datasets varies both between and within the previously evaluated scenarios. For this reason, each trained model is compared with the baseline model that was trained with the closest larger amount of training data. The results of these comparisons are presented in Tab. 4.13 - Tab. 4.15.

It is acknowledged that the selection of baseline models may be suboptimal to some extent, as they were trained and evaluated on geometries with a middle gap. Previous results have shown that generalization to such geometries is particularly challenging. Nevertheless, the baseline models were used for practical reasons, as they had already been trained and published in the work by Broda et al. [19]. Developing additional baseline models, especially those trained on large datasets containing 5,000 or 20,000 images, was beyond the scope of this thesis. Furthermore, testing on these geometries remains valuable, since corresponding real-world data is available (see subsequent Section 4.5).

Negative values in Tab. 4.13 - Tab. 4.15 indicate a lower performance compared to the baseline model. Gray-shaded cells refer to models whose training included cluster 4. When tested on the CESA1 geometries, these models cannot be considered entirely geometry-agnostic, as clusters 47 and 48 also contain a middle gap. A green color indicates a better performance than the corresponding baseline model. A yellow color indicates deviations within 10 percentage points of the baseline performance, while red denotes larger discrepancies.

Table 4.13.: Comparison of developed models with baseline model B2 on the CESA1 test set. AP and PCK are given in %. $\Delta_{AP,2}$ and $\Delta_{PCK,2}$ in percentage points.

Scenario *All Geometries*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,2}$	$\Delta_{PCK@3px,2}$
4	54.33	55.60	2000	-26.19	2.10

Scenario *No Duplicates*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,2}$	$\Delta_{PCK@3px,2}$
1	17.58	52.75	2000	-62.94	-0.75
2	15.29	49.30	1500	-65.23	-4.20
3	6.71	46.90	1500	-73.81	-6.60
4	48.32	55.00	1500	-32.20	1.50

Scenario *One Geometry per Cluster*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,2}$	$\Delta_{PCK@3px,2}$
1	11.05	49.25	500	-69.47	-4.25
2	7.07	45.95	500	-73.45	-7.55
3	0.76	46.55	500	-79.76	-6.95
4	5.50	48.90	500	-75.02	-4.60
1,3	12.30	53.05	1000	-68.22	-0.45
1,4	64.82	53.35	1000	-15.70	-0.15
2,3	8.65	50.65	1000	-71.87	-2.85
1,2,3	12.93	53.15	1500	-67.59	-0.35
1,2,4	69.41	53.95	1500	-11.11	0.45
1,3,4	70.81	53.80	1500	-9.71	0.30
2,3,4	69.22	51.00	1500	-11.30	-2.50

Table 4.14.: Comparison of developed models with baseline model B5 on the CESA1 test set. AP and PCK are given in %. $\Delta_{AP,5}$ and $\Delta_{PCK,5}$ in percentage points.

Scenario *All Geometries*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,5}$	$\Delta_{PCK@3px,5}$
1	19.95	52.95	3000	-64.17	-2.25
2	20.03	49.75	3000	-64.09	-5.45
3	6.07	47.70	2500	-78.05	-7.50
4	54.33	55.60	2000	-29.79	0.40

Scenario *No Duplicates*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,5}$	$\Delta_{PCK@3px,5}$
1,4	77.83	56.35	3500	-6.29	1.15
2,3	11.34	52.20	3000	-72.78	-3.00
2,4	75.85	55.50	3000	-8.27	0.30
2,3,4	77.67	56.40	4500	-6.45	1.20

Table 4.15.: Comparison of developed models with baseline model B20 on the CESA1 test set. AP and PCK are given in %. $\Delta_{AP,20}$ and $\Delta_{PCK,20}$ in percentage points.

Scenario *All Geometries*

Training Cluster	**AP**	**PCK@3px**	Train Set Size	$\Delta_{AP,20}$	$\Delta_{PCK@3px,20}$
1,3	15.31	55.45	5500	-70.82	-1.75
2,4	77.18	56.00	5500	-8.95	-1.20
1,2,3	14.27	55.85	8500	-71.86	-1.35
1,2,4	80.92	57.05	8000	-5.21	-0.15
1,3,4	78.46	57.80	7500	-7.67	0.60
2,3,4	78.50	57.20	7500	-7.63	0.00

Evaluation of geometry-agnostic models

None of the geometry-agnostic models outperforms the baseline models in terms of overall performance. However, several models approximate the PCK of the baseline models within a few percentage points. The closest result is achieved by the model trained on clusters 1, 2 and 3 in the *One Geometry per Cluster* scenario. Furthermore, it is important to note that the models referred to in Table 4.15 are compared against baseline model B20, although their training set consists of only up to 8,500 images. Given this significant difference in training volume, the PCK deviation of just 1.35 percentage points appears comparatively low. Interestingly, the PCK values on the CESA1 test set here are higher than previously observed for test cluster 4 across all three scenarios. This performance difference could indicate that individual geometries within test cluster 4 may act as outliers and negatively impact the model performance in the previous scenarios. Thus, the assessment that keypoint detection generalizes poorly to test cluster 4 requires a more differentiated analysis. Future work should examine whether certain geometries within a cluster exert a stronger influence on the evaluation results. As observed in the three scenarios, the generalization of object detection to geometries with a middle gap remains a challenge. Thus, the geometry-agnostic models show substantial object detection deficits compared to their respective baseline models.

Evaluation of non-agnostic models

The non-agnostic models are trained on data consisting of or including cluster 4. In several cases, these models achieve equal or slightly higher PCK values than their baseline counterparts. Nevertheless, the AP remains lower in all comparisons. The evaluation of the PCK suggests that training a model not only on geometries included in the test set, but also on additional, diverse geometries can enhance keypoint detection. However, the geometry-agnostic training is potentially at the cost of the object detection performance. These AP deficits could potentially be tackled by increasing the number of training samples for the test geometries. This approach would, however, tend towards a class-aware approach.

4.5. Evaluation of the Simulation-to-Real Transferability

The focus of this thesis lies on the training, testing, and evaluation of geometry-agnostic models trained on synthetic data. Nevertheless, the transferability of the results to real-world data is also briefly examined and related to the previously defined baseline models. In order to enable a meaningful comparison between the baseline models and the geometry-agnostic models presented in this work, it is essential to understand the underlying principles according to which the synthetic data used in the work by Broda et al. [19] was generated. Therefore, the Blender settings, used for the training data generation for the baseline models from [19], and the geometry-agnostic models in this work are compared in Tab. 4.16. The configuration of the baseline models serves as a reference, as it is the result of an in-depth study conducted. These settings were specifically chosen to maximize the transferability of the models to real-world data [19].

Table 4.16.: Comparison of Blender settings for synthetic data generation between baseline models from [19] and geometry-agnostic models.

Criterion	**Baseline Model Settings** [19]	**Geometry-Agnostic Model Settings** (this work)
Appearance		
Ground texture	Procedural with randomized 3D surface features	✓
Lighting	Realistic lighting aligned with test conditions	✓
Mirror soiling	Realistic simulation of dirt on mirror surfaces	✗
Content		
Distractor objects	Realistic objects (e.g., vehicles, buildings)	✗
Object orientation	Physically plausible (e.g., aligned to sun, no extreme angles)	(✓)
Object placement	Randomized, but plausible (not chaotic or physically impossible)	✗
Camera & Flight Setup		
Camera positions and altitudes	Realistic drone perspectives and altitude variation across field	✓
Camera calibration	Intrinsics matching real hardware (e.g., 6000×4000 px, fx = fy = 4000)	✓

The comparison reveals that some of the criteria were not applied to the synthetic data generated in this study. This difference is primarily because the corresponding study results became available after the training data had already been created. In addition, the focus on the geometric variation of the synthetic data and the omission of certain aspects (e.g., soiling) helped to simplify the data generation process.

Table 4.17 shows the numerical values of the test metrics for the three baseline models. It should be noted that the real test images were manually annotated with considerable effort, which explains the small test set size of only six images.
To evaluate the transferability to real-world data, three models from the *All Geometries* scenario are selected as they overall show the highest test performances. The first model (M1)

Table 4.17.: Class-aware baseline Models for three different train dataset sizes. Tested on real data from CESA1. Test dataset size: 6. The PCK was calculated for a fixed tolerance radius of 3px. AP and PCK are given in %.

Baseline Model	Train Dataset Size	AP_{real}	$\text{PCK@3px}_{\text{real}}$
B2	2,000	56.68	36.6
B5	5,000	59.51	37.0
B20	20,000	65.76	42.6

was trained on clusters 1, 2 and 4. The second model (M2) was trained on clusters 1, 3 and 4. These are the models with the highest overall AP (M1) and PCK@3px (M2) across all scenarios. The third model (M3) is the one trained on clusters 1,2 and 3 and is therefore considered a geometry-agnostic model. Table 4.18 presents the evaluation results of the test metrics for the aforementioned models. The results indicate that, under the current configuration used for generating the synthetic images, the learned patterns do not transfer well to real-world data. These observations hold true for both the class-aware models and the geometry-agnostic model. This can be partially explained by the fact that several parameters relevant for transferability (e.g., soiling, random distractor objects) were not included in the training data generation process. However, the findings presented in [19] demonstrate significantly better results even with suboptimal generated synthetic data, compared to what could be achieved within the scope of this thesis. Further analysis of this discrepancy could be part of future research efforts.

For the sake of completeness and to provide additional insight for the reader, the predictions on the test dataset are visualized in Fig. 4.5. For the predictions, however, the model confidence threshold used to identify a keypoint or bounding box center on the heatmap was lowered from 0.5 to 0.1. This adjustment allows at least some predictions to be visible. Nevertheless, these results do not hold any practical relevance.

Table 4.18.: Overall best models tested on real data. Models from scenario *All Geometries*. Real data from CESA1. Test dataset size: six real-world images. The PCK was calculated for a fixed tolerance radius of 3px. AP and PCK are given in %.

Test Model	Train Dataset Size	Train Clusters	AP_{real}	$\text{PCK@3px}_{\text{real}}$
M1	8,000	1,2,4	0.06	0.2
M2	7,500	1,3,4	0.016	0.4
M3	8,500	1,2,3	0.3	0.3

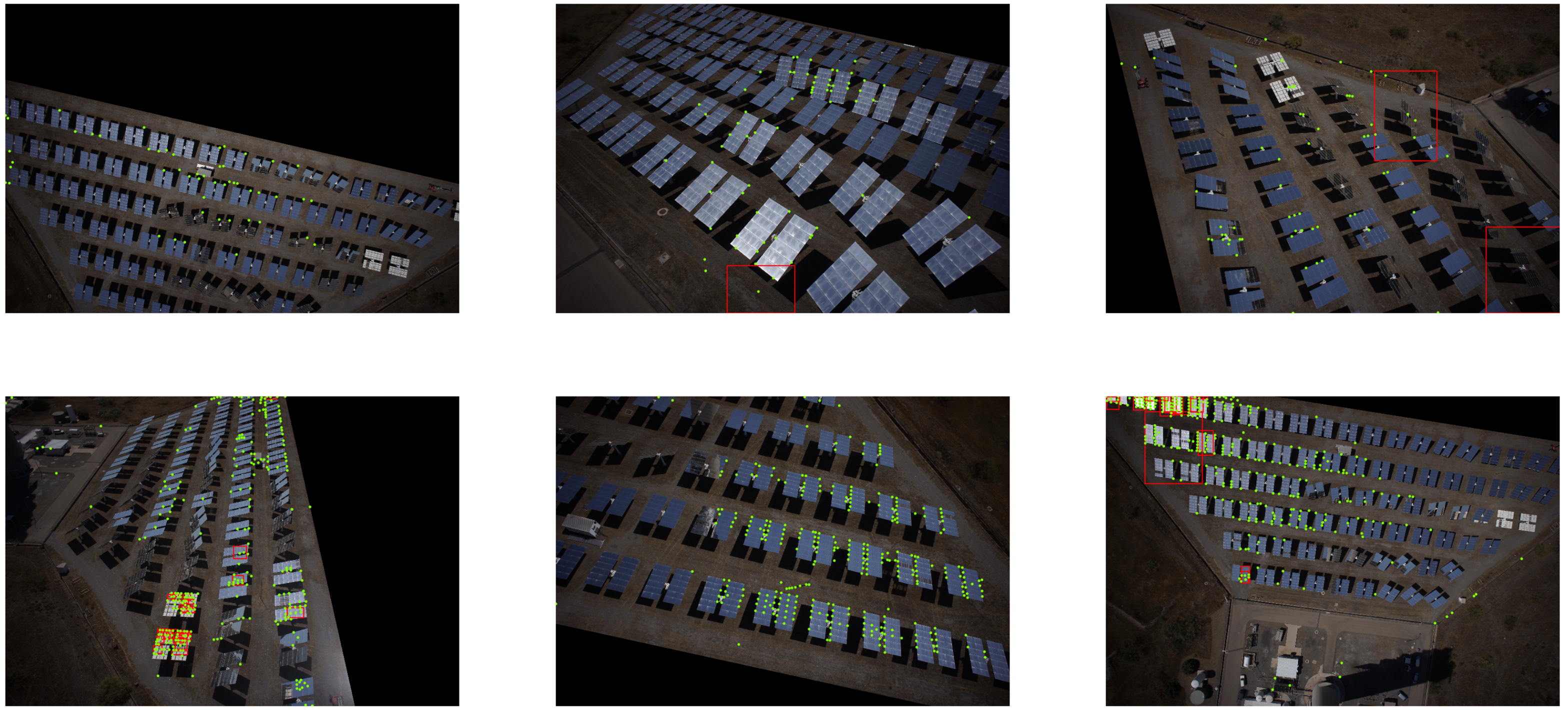


Figure 4.5.: Visualization of model predictions on real-world data. Real world data from Jessen et al. [9]'s measurement fight at the PSA (owned and operated by CIEMAT. Compare Broda et al. [19]). The model confidence threshold for the model predictions was reduced to 0.1 to allow visualization of model predictions. Note that this visualization does not correspond to the test metrics reported in Tab. 4.17.

4.6. Model Post-Processing

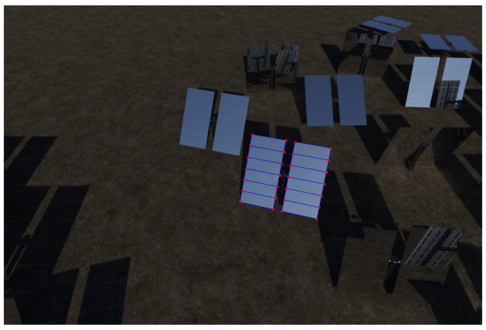
In Chapter 3.3.2, a methodology for estimating the heliostat pose was presented. This approach is based on estimating the translation and rotation of a heliostat relative to the observer (i.e., camera) coordinate system OCS. In this section of the evaluation, the model's performance with respect to pose estimation is analyzed. For a precise assessment of the pose estimation performance, one could compute the average difference between the estimated rotation and the ground truth rotation, as well as between the estimated and ground truth translation. However, since estimating the heliostat pose represents a novel and exploratory approach at the DLR Institute of Solar Research (Almería, Spain), a visual inspection is considered sufficient at this stage. The model's ability to learn heliostat pose estimation is initially demonstrated by the low loss values for translation and rotation, as presented in Table 4.19.

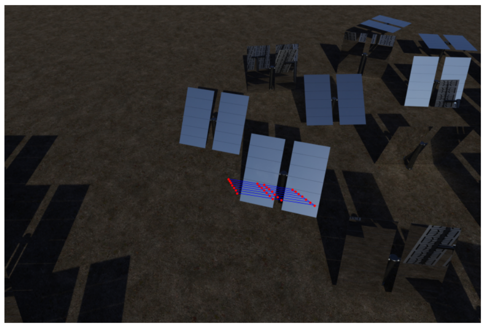
Table 4.19.: Pose Estimation | Evaluation of translation and rotation loss. For proper loss weighting, the norm of the translation vectors, representing the distance between the observer and an object in meters, is divided by a constant factor of 100.

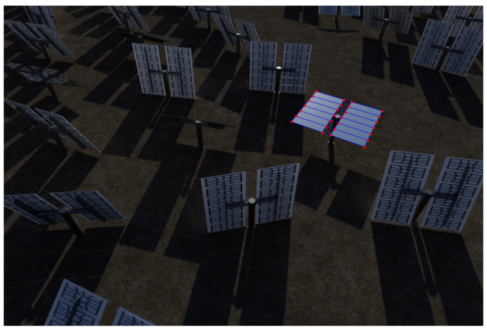
Models	Training Data	Translation Loss	Rotation Loss
B2	2,000	0.06	0.08
B5	5,000	0.03	0.03
B20	20,000	0.06	0.008

To evaluate the feasibility of the proposed training approach, the baseline model B20 is used, as it achieves the lowest combined translation and rotation loss. The estimated pose of two test heliostats is compared to their ground truth pose in Fig. 4.6.

The results indicate that, with the current training setup and model configuration, the overall pose cannot be estimated accurately. While the translation of the grid generally points in the correct direction (especially in Fig. 4.6b), the estimated rotation angles show larger deviations. The deviation is particularly evident for rotation around the horizontal axis. Although the loss values for pose estimation are low (see Table 4.19), they do not necessarily indicate accurate results. This finding suggests that the current loss formulation may not sufficiently guide the model, and increasing the corresponding weights could help improve overall pose estimation performance. Enhancing this approach or exploring alternative methods remains a subject for future investigation.


(a) Ground truth | geometry: 47, field layout: rectangular, ground: soil, image id: 1, heliostat id: 380.


(b) Pose estimation | geometry: 47, field layout: rectangular, ground: soil, image id: 1, heliostat id: 380.


(c) Ground truth | geometry: 47, field layout: rectangular, ground: soil, image id: 9, heliostat id: 126.

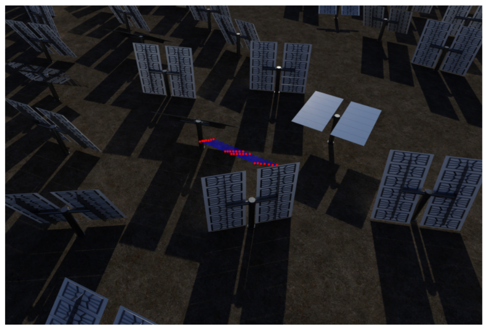

(d) Pose Estimation | geometry: 47, field layout: rectangular, ground: soil, image id: 9, heliostat id: 126

Figure 4.6.: Pose estimation for exemplary Blender scenes. Left: Ground Truth. Right: Model Prediction.

5. Conclusion and Outlook

In Chapter 4 of this thesis, the extent to which learned patterns for detecting objects and keypoints can be transferred across different mirror geometries was investigated. To this end, twelve different combinations of training and testing clusters were generated in each of the three developed scenarios. The evaluation of the test results reveals a number of valuable insights, which are summarized below.

Summary

- The evaluation of the geometry-agnostic modeling approach revealed a generally good generalization capability across different geometric clusters. A key finding is the importance of structural similarity between training and testing clusters.
- Object detection is more sensitive to geometric variation, as it seems to rely on global structural features and shows noticeable performance drops when training and testing clusters differ significantly.
- Keypoint detection is more robust under geometric variation. As a locally defined task, in most of the analyzed cases, it yields good performance values even when the training and testing clusters differ significantly.
- The generalization behavior itself is not necessarily symmetric. While models trained on large geometries or geometries with a middle gap can generalize to smaller ones, the reverse is not true. Especially geometries with a middle gap cannot be adequately approximated by other geometry types within the current model configuration.
- Training data reduction has a noticeably stronger negative effect on object than on keypoint detection. This comparison supports the robustness of keypoint-based approaches under data scarcity and structural variation.

Furthermore, the trained geometry-agnostic models were evaluated on the CESA1 geometry to assess their performance compared to a class-aware baseline model.

- In terms of object detection, the geometry-agnostic models perform noticeably worse. However, for keypoint detection, their results approach those of the baseline models, with the best model showing a minimal difference of only 0.35 percentage points.
- Interestingly, some non-agnostic model variants outperform the baseline models in keypoint prediction, although being partly trained on smaller datasets. While using these non-agnostic models also leads to improvements in the AP, the overall AP values remain consistently below those of the baseline models.

- These findings further support the observation that keypoint detection generalizes more robustly than bounding box prediction across geometric variations.

In addition, the study briefly examined the extent to which learned patterns from synthetic data can be transferred to real-world data.

- The transferability of results from the geometry-agnostic study based on synthetic data could not be confirmed for real-world data. The test metrics AP and PCK for the three best-performing models were close to zero.
- The poor model performances can be partially explained by the fact that several parameters relevant for transferability (e.g., soiling and random distractor objects) were not included in the training data generation process. However, the findings presented by Broda et al. [19] demonstrate significantly better results even with suboptimal generated synthetic data, compared to what could be achieved within the scope of this thesis.
- Given the limited focus on this aspect, the present work can only conclude that the learned patterns from synthetic data do not transfer effectively to real-world data, neither for class-aware nor for geometry-agnostic models.

Finally, the pose estimation was qualitatively evaluated. Initial results for randomly selected heliostats show that the translation vector can be roughly approximated, whereas the prediction of the rotation remains challenging for the model.

Recommendations for applications of the developed geometry-agnostic models
The generalization performance of the geometry-agnostic models was shown to be highly dependent on the specific combination of training and testing clusters. Therefore, any application of these models requires case-specific evaluations based on the underlying testing geometry to determine whether a geometry-gnostic model is suitable, and if so, which one performs best.

- Generalization from clusters 2, 3, and 4 to cluster 1, as well as from clusters 1, 3, and 4 to cluster 2, yielded promising results in testing. The results suggest that a model trained on all four clusters can likely be applied effectively to geometries from clusters 1 and 2. Since these clusters would also be included in the training set, further performance improvements can be expected, though this effect would require additional evaluation. If training resources are to be reduced, one could consider training only on geometries from clusters 1 and 2, as clusters 3 and 4 were shown to contribute only minor performance gains.
- Generalization from clusters 1, 2, and 4 to cluster 3 was also found to be reasonably effective. In this case, PCK scores were the highest, while AP values remained lower compared to other test cases. Therefore, adding training data from cluster 3 is assumed to further improve performance. In combination with the findings related to the point before, it can be concluded that a model trained on all four clusters should be applicable to geometries from clusters 1 to 3.

- Generalization to geometries with a middle gap was found to be poor, both in terms of object and keypoint detection. Based on these findings, it is recommended to use a dedicated model trained exclusively on geometries with a middle gap when applied to such geometries.

- The results indicate that PCK is comparatively robust with respect to variations in training data volume. This insight may support the decision to reduce the number of geometries per cluster in scenarios where training effort needs to be constrained.

Outlook
The key findings of this thesis have identified several relevant directions for future research. Some of these aspects were already addressed during the analyses in Chapter 4. The following concludes this thesis by proposing further studies and research tasks.

- The analyses suggest, that the transferability from large to small geometries is less effective than in the opposite direction. One potential explanation is the low density of keypoints and bounding box centers in fields with small geometries. To verify this hypothesis, fields with large geometries but reduced heliostat density could be generated to assess whether this parameter has a measurable impact on model performance.

- Interestingly, the developed geometry-agnostic models achieved better PCK values on the CESA1 test dataset than on test cluster 4, although CESA1 geometries are structurally similar to those in that cluster. This difference suggests the presence of an outlier geometry within the test cluster. Further training experiments could be conducted by iteratively excluding individual geometries from cluster 4 to investigate this effect in more detail.

- It would be valuable to train baseline models on geometries without a middle gap, where geometry-agnostic models are expected to perform better. A promising candidate is the geometry used in the SPT plant in Jülich (Germany), for which real-world data is also available. However, this study did not utilize these data due to currently missing annotations.

- Further investigations into the simulation-to-real transferability are recommended. This includes incorporating additional realism factors such as soiling, random objects, and randomized heliostat placement, as proposed in [19].

- A more in-depth examination of the pose estimation process is needed, as current results are not yet satisfactory.

Appendix A

A.1. Comparison of Selected Loss Functions

In this chapter, selected loss functions are introduced mathematically, their fields of application and their benefits are explained. This starts with the l1, l2 and smooth l1 loss, which are used for training models dedicated to regression tasks. The l1 loss (Eq. A.1) computes the absolute difference between a model prediction $\hat{y}$ and the corresponding ground truth value y [113].

$$\mathcal{L}_{\text{L1}}(\hat{y}, y) = |\hat{y} - y| \tag{A.1}$$

The l2 loss (Eq. A.2) computes the squared difference between a model prediction $\hat{y}$ and the corresponding ground truth value y [113].

$$\mathcal{L}_{\text{L2}}(\hat{y}, y) = (\hat{y} - y)^2 \tag{A.2}$$

The smooth l1 loss (Eq. A.3) behaves like the l2 loss for values smaller than one. For values equal to or greater than one, the loss function behaves like the l1 loss [44]. The factor 0.5 neutralizes the factor 2 from the derivative. The summand 0.5 is then additionally required to ensure continuity in the point $|\hat{y} - y| = 1$.

$$\mathcal{L}_{\text{SmoothL1}}(\hat{y}, y) = \begin{cases} 0.5(\hat{y} - y)^2 & \text{if } |\hat{y} - y| < 1 \\ |\hat{y} - y| - 0.5 & \text{otherwise} \end{cases} \tag{A.3}$$

Fig. A.1 shows the functions of the l1, l2 and smooth l1 losses plotted over the prediction error. Due to its quadratic error behavior, the l2 loss is sensitive to outliers and leads to strongly increasing loss values for a prediction error greater than one. This quadratic increase can impact the stability of the training. The l1 loss, on the other hand, has a linear error term, but is not differentiable for $|\hat{y} - y| = 0$. The smooth l1 loss combines the advantages of both functions. It is overall differentiable and less sensitive to outliers than the l2 loss [15].

Another standard use case in supervised learning is classification. A typical loss function for classification tasks is the cross-entropy, which evaluates the difference in the probability distributions of the ground truth and the model prediction. The binary cross entropy (BCE) loss function for a classification problem with two classes (C=2) is stated in Eq. A.4. y is the ground truth label, $\hat{p}$ the predicted probability that a given input belongs to class one. Note that y is in $\{0, 1\}$ and $\hat{p}$ is in $[0, 1]$.

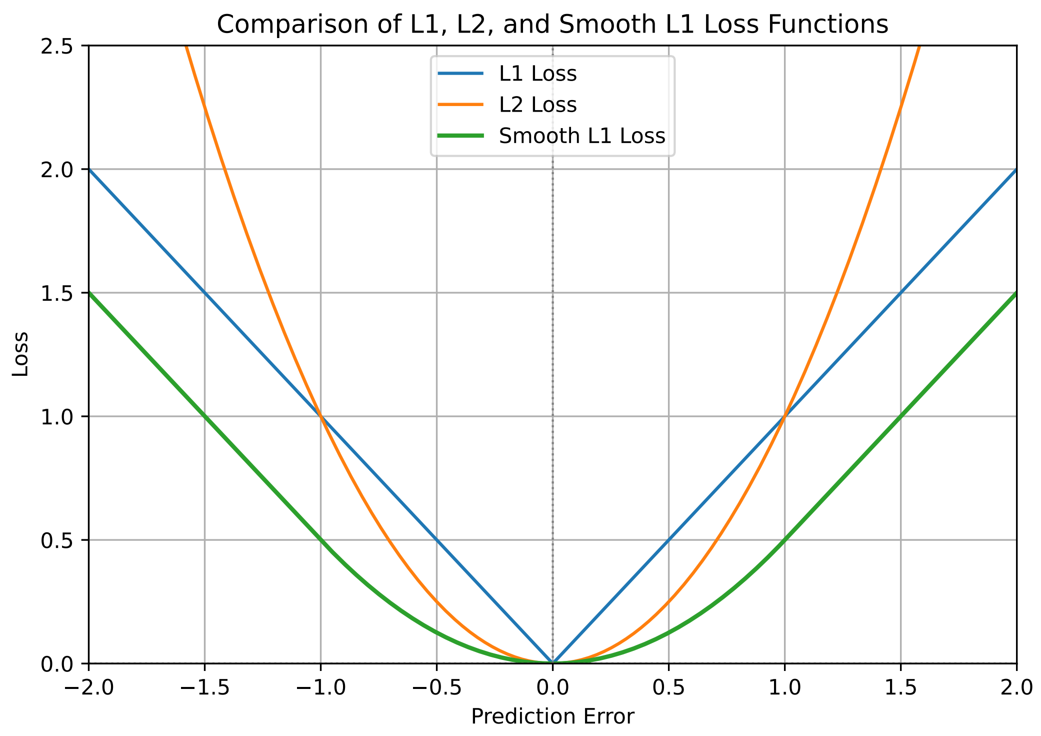


Figure A.1.: Comparison of l1, l2, and smooth l1 Loss functions, own presentation.

$$\mathcal{L}_{\text{BCE}} = \begin{cases} -\log\left(\hat{p}\right), & \text{if } y = 1 \\ -\log\left(1-\hat{p}\right), & \text{if } y = 0 \end{cases} \tag{A.4}$$

For a perfect correct prediction (e.g., $y = 1$ and $\hat{p} = 1$), the loss is equal to zero. The loss increases as the prediction probability for the correct class decreases [28, 29, 39]. A shortcoming of cross-entropy is that for unbalanced datasets, the model simply learns to select the dominant class. Although the total loss is low as a result, the model has not learnt from the data. This is a particular problem in object detection. The CenterNet [49, 53] architecture, for example, predicts bounding box centers as single pixels on a heatmap, which only have a very small proportion of all pixels in the image. The focal loss (Eq. A.5) exactly takes this aspect into account [97].

$$\mathcal{L}_{\text{Focal}} = \begin{cases} -\alpha \cdot (1-\hat{p})^{\gamma} \cdot \log(\hat{p}) & \text{if } y = 1 \\ -(1-\alpha) \cdot \hat{p}^{\gamma} \cdot \log(1-\hat{p}) & \text{if } y = 0 \end{cases} \tag{A.5}$$

The focal loss scales the loss by a factor of $(1-\hat{p})^{\gamma}$ respectively $\hat{p}^{\gamma}$. If the model is very reliable and the prediction is correct, the loss tends to be suppressed. In contrast, the factor for incorrectly or uncertainly classified data becomes dominant. γ controls the strength of the focus and α optionally compensates for the imbalance between the classes [96, 97].

A.2. Conventional Image Processing Techniques for Edge and Corner Detection

The intermediate conclusion at the end of Subchapter 2.2.4.2 shows the limitations of conventional image processing methods regarding edge and corner detection for heliostats in aerial images compared to DL methods. At this point, investigations into this topic are also being carried out as part of this work. Four well-known mathematical methods for edge and corner detection are considered below. The information provided is based on Gonzalez et al. [114] unless otherwise stated.

Conventional edge detection in images is a gradient-based method. A vertical respectively horizontal edge can be seen as a significant local change in the intensity of neighboring pixels in the horizontal respectively vertical plane [27]. This is illustrated in Fig. A.2. On the left side, an excerpt of an image with a local vertical edge is shown. This edge causes a linear change in the horizontal intensity profile. The profile is increasing due to the change of a dark color with low pixel values and intensity to a brighter light-gray with higher pixel values and intensity (see Fig. 2.5). The first derivative of this intensity profile thus shows the vertical edge as a local maximum.

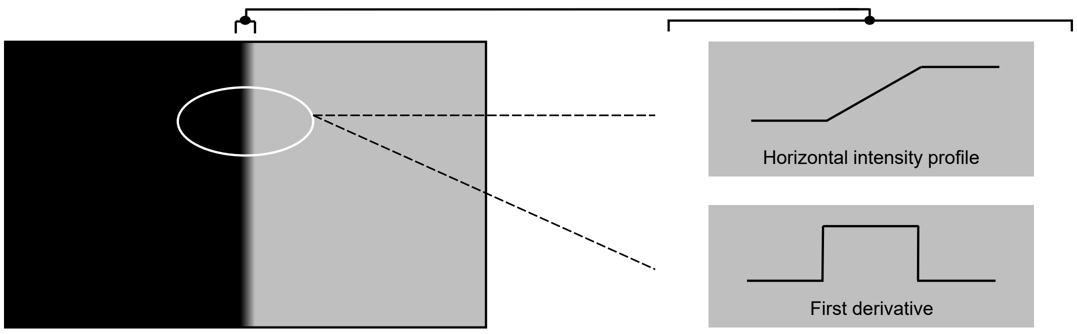


Figure A.2.: Intensity profile of a vertical edge, own figure based on [114].

The local gradients can be derived by applying the **Sobel operators**. The Sobel operators are mathematical operators based on the convolutional operation. The operator consists of two 3×3 kernels for combined detection of vertical and horizontal edges. They can be seen in Fig. A.3. The factor of two in the center position provides image smoothing because neighboring pixels are strongly weighted (not considered further). Convolving the two kernels with an image obtains every pixel's local gradients in the vertical and horizontal directions. Based on these gradients, the edge strength and direction are derived. Due to the specific choice of the kernels, the Sobel operators are designed to find horizontal and vertical edges. However, edges in other directions can also be detected via the local gradients but only with a lower intensity. Since only detected edges are considered as such if their intensity is higher than a threshold value, the Sobel operators neglect edges that run, for instance, diagonally and do not detect them.

-1	0	1
-2	0	2
-1	0	1

kernel for vertical edges

-1	-2	-1
0	0	0
1	2	1

kernel for horizontal edges

Figure A.3.: Sobel kernels, own figure based on [114].

Canny edge detection is a more advanced method to detect edges. In a first step, the input image is smoothed with a Gaussian filter. This task corresponds with the assumption that noise - random color and intensity deviation - is distributed by a Gaussian normal distribution. Suppressing this noise improves the quality of the detected edges. In the following, edges are detected with the Sobel kernels and their strength and direction are computed. A feature of the Sobel operators is that the edges are strengthened but also widened by using a 3 $\times$ 3 kernel. Canny edge detection avoids this problem of wide edges by using a nonmaxima suppression. This means that the edge strength of a pixel is compared with the neighboring pixels along the normal vector of the edge. If the current pixel value is not the local maximum in this direction, the pixel is set to 0. This process results in thin edges. In the final step, a method with two threshold values checks whether an identified pixel represents an edge point or noise. Reliable edge points are pixels with an intensity higher than the upper threshold. Pixels with a low intensity - but still higher than the lower threshold - are only categorized as edge points if they are adjacent to a pixel with a high intensity. Pixels with an intensity lower than the lower threshold are ignored. In this way, it is ensured that only connected edges are recognized.

The **Harris corner detector** is a gradient-based method to detect corners in images. Comparable to the Canny edge detection procedure, a Gaussian filter suppresses noise. In the next step, the local gradients for each pixel in the horizontal and vertical directions are computed with the Sobel operators. For every pixel, the Harris matrix is calculated, which contains information about the intensity changes in all directions in 2D. A parameter R is introduced, describing the pixel's squareness. R is calculated based on the Harris matrix's eigenvalues. A pixel is classified as a flat area, edge or corner depending on R. An edge is characterized by a significant change in intensity in one direction, while a corner is characterized by a change in intensity in all directions. This is illustrated in Fig. A.4. The figure shows a black corner on white ground detected by a kernel. The intensity of the pixel values changes from low (black) to high (white) in the directions of bottom-top and right-left. A corner's greater change in intensity is reflected in the Harris matrix's eigenvalues and the squareness parameter R. High values of R classify a pixel as a corner, and values near zero classify a pixel as an edge. Values smaller than zero classify a pixel as a flat area. Nonmaxima suppression allows the detection of thin edges respectively corners.

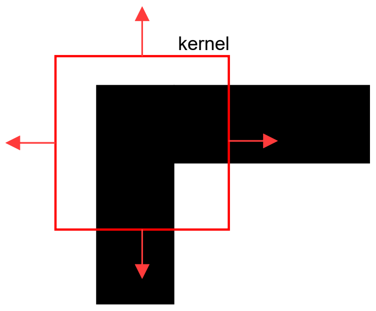


Figure A.4.: Intensity changes at a corner in an image, own figure based on [114].

Methods such as the Canny edge detector or the Harris corner detector provide specific features in the image, such as edge points or corner points. These features provide local information about structural properties but do not allow direct conclusions about complete edge lines. Gaining this information requires higher-quality methods, such as the Hough transformation.

The **Hough transformation** is a method that allows global geometric structures such as straight lines, circles, or ellipses to be reconstructed from a set of point features. Instead of searching directly for connected edge lines in the image space, the Hough transformation transfers the problem to a so-called parameter space. There, possible shapes are represented by their characteristic parameters (e.g., slope and axis intercept for straight lines). For each edge point, the parameter space determines which geometric shapes could pass through that point. A single image point generates a curve in the parameter space. If several of these curves meet at one point, this means that many image points support a common geometric structure, such as a line. This accumulation is made visible by a so-called accumulator field. In this way, it is possible to make robust estimates about the presence of geometric objects even in noisy or incomplete image data.

Conclusions

The classical methods for edge and corner detection presented here rely on handcrafted features and fixed parameter settings. This fundamental characteristic introduces several limitations: Firstly, methods such as the Sobel operators are based on predefined filter kernels primarily designed to detect vertical and horizontal edges. Edges with different orientations are detected with reduced intensity or may be missed entirely. While other kernels can be used, they must be specifically tailored to the edges of interest, requiring prior knowledge and additional implementation effort. Furthermore, both the Canny edge detector and the Harris corner detector rely on fixed threshold values to classify pixels as relevant features (e.g., edge or corner points). These thresholds are often image-dependent and typically require manual tuning or heuristic approaches, making the process inflexible and labor-intensive. Another challenge lies in the limited robustness to image noise, despite the use of smoothing techniques such as the Gaussian filter. In addition, the lack of scale invariance is a critical weakness: thick edges or features in images with varying resolutions are not consistently detected, as the fixed filter sizes fail to adapt accordingly. Non-maximum suppression, while providing thin, one-pixel-wide edges, may also be limiting in applications where the full width of an edge carries relevant information. The Hough transformation, though useful for detecting global geometric structures, requires a prior definition of the desired shape (e.g., line, circle), which restricts flexibility and further increases the need for manual parameter selection. Overall, these techniques demand manual adjustment and in-depth knowledge of the image content and context to produce reliable results. Their performance is thus highly context-dependent and lacks adaptability to varying image conditions.

A.3. State-of-the-art Heliostat Calibration Methods

This section corresponds to chapter 2.2.4 and delivers additional information about state-of-the-art heliostat calibration methods. In a detailed study Sattler et al. [6] review about 30 heliostat calibration and tracking control methods and categorize them into five groups. The general working principles of the control methods in these categories are explained below.

Group one includes variations of the camera-target method, which was developed in 1984 and still is state-of-the-art in many CSP plants [6, 83]. During a calibration, one heliostat is moved so that the irradiance is not focused any longer onto the receiver but onto a white target screen underneath the receiver. The solar focus position on the target is then captured by a camera, which is placed on the ground, and finally compared to a reference position. For a full calibration, this process has to be done for different points in time with varying sun positions [6, 83]. Further developments like the one from Bern et al. [115] allow the calibration of multiple heliostats at one time. This is done by coding the heliostats' reflected sun arrays (beams) through movement of the heliostat itself.

In group two, calibration methods process images of the heliostat instead of the heliostat beam. Röger et al. [5] presents an approach in which an individual image of a heliostat and an edge detection algorithm are used to calculate the heliostat's position relative to a fixed camera. An alternative approach is the attachment of special markers on each heliostat whose reflection patterns provide information about the heliostat orientation relative to the camera [90]. Group two also includes methods using multiple images of one heliostat to calculate its position. Photogrammetric[13] processes are used to merge the multiple orientation information of each heliostat [116]. Another approach that differs from those mentioned is to use reflected images of an object in the heliostat mirror like a LED. By the position and size of the reflection on the heliostat mirror, the heliostat orientation can be determined [117]. Another option is to use an airborne camera mounted on an UAV to get a higher variation of camera positions and angles [9, 117].

Calibration methods in group three use central laser or radar-based approaches to determine the heliostat orientation. Dabrowski et al. [85] propose a laser calibration system where a laser, oriented towards the heliostat mirror, emits a short laser pulse that is reflected into the sky. Parts of the laser beam are scattered by molecules in the atmosphere, making it visible to cameras. The normal vector of the heliostat mirror can be calculated based on the vector of the laser beam. The method proposed by Klimek et al. [118] determines the orientation of a heliostat by using radar technology to analyze the reflections from three markers mounted on the heliostat. The orientation of the mirror is calculated from the returned radar signals.

Group four includes a small collection of calibration methods that detect the heliostats' reflection with cameras mounted on the tower or around the receiver. A method researched in depth can be traced back to Yogev et al. [88]. Their approach is based on comparing the brightness of a reflection in a heliostat mirror between four photos. The photos are taken by four cameras mounted around the receiver. An alignment is considered accurate if the brightness is identical on all four images.

Cameras or sensors on each heliostat are used in the calibration methods of group five. The orientation of a heliostat mirror is computed via the recognition of objects and their position in the camera image. Therefore, the location of the reference objects has to be known. The sun, the tower, or the receiver can be used as a reference [86, 87].

The amount of available calibration methods shows the intensive research that is done in this field. Despite that, the authors Zhu et al. [1] and Sattler et al. [6] come to the following conclusion: „Although the high number of published calibration methods and unique concepts show that much work in this field is being done, it is still uncertain which ones might be successful." [6] That is because all the listed methods have to deal with significant disadvantages, as shown in the Tab. A.1 below.

Table A.1.: Limitations of state-of-the-art heliostat calibration methods based on [5].

Group	Author	Significant Limitations
(1) Camera on ground	[6, 83]	High process time due to individual heliostat calibration via camera target method.
	[115]	Distant heliostat coding beams are difficult to measure.
(2) Camera on tower or UAV	[5, 9, 90, 116, 117]	Heliostats with extreme angles to the camera orientation are difficult to measure. Reflections of the ground are complex to detect and process.
(3) Central laser or radar	[85]	High process time due to individual heliostat calibration via laser.
	[118]	Effort for manual installation of markers.
(4) Central solar focus position detection	[88, 89]	Individual software for error analysis, heliostat identification and correction calculation for every solar field
(5) Camera or sensor on each heliostat	[86, 87]	High investment costs and manual effort for manual installation

The limitations identified show that none of the state-of-the-art calibration methods fulfill entirely the requirements regarding efficiency and effectivity.

[13]The term *Photogrammetry* is defined in Chapter 2.2.4

Appendix B

B.1. Derivation of Heliostat Parameters from Real Geometries

Zhu et al. [1] summarizes technical data of commercially operational heliostat designs around the world (see [1] p. 8). Relevant columns are the heliostat dimensions, the heliostat facets in columns and rows of the heliostat mirror surface and the heliostat mirror area. The parameters `panel_size_x` and `panel_size_y`, which are needed for the parametric heliostat model in Blender (see Tab. 3.2), are computed by dividing the size of the heliostat in a dimension by the number of facets in this dimension. The size of the middle gap is neglected in this calculation in order to simplify the calculation. In addition, the aim is to cover the variety of geometries relevant in practice and not to reproduce them exactly. The heliostat dimensions for the entry `Luneng Haixi 50-MW Molten Salt Tower` are missing. The procedure to compute these values is described below. At the present time, a parametric model in Blender is only available for heliostats with a rectangular mirror surface. Thus, the entry `Hami 50-MW CSP Project` and its pentagonal heliostat geometry is neglected.

Table B.1.: Heliostat panel configurations of various CSP plants. n_x, n_y denote the number of panels in x and y directions, s_x, s_y represent panel dimensions in meters.

Plant	n_x	n_y	s_x [m]	s_y [m]
Shouhang Dunhuang 100-MW Phase II (China)	7	5	1.5	2.2
Luneng Haixi 50-MW Molten Salt Tower (China)	4	8	–	–
PowerChina Gonghe 50-MW CSP Plant (China)	2	2	2.9	1.8
SupCon Delingha 50-MW Tower (China)	2	2	2.9	1.8
Shouhang Dunhuang 10-MW Phase I (China)	7	5	1.5	2.2
Ashalim Plot B (Israel)	2	2	2.0	2.6
NOOR III (Morocco)	9	6	1.5	2.2
Khi Solar One (South Africa)	2	8	6.6	1.3
Gemasolar Thermosolar Plant (Spain)	7	5	1.6	2.1
Planta Solar 20 (Spain)	4	7	3.2	1.4
Planta Solar 10 (Spain)	4	7	3.2	1.4
Crescent Dunes Solar Energy Project (USA)	7	5	1.5	2.2
Ivanpah Solar Electric Generating System (USA)	2	1	2.3	3.0

HelioCon maintains two databases including technical heliostat data [24]. The database `Power Tower Plant Database` contains a list of SPT plants, which give information about the installed heliostats. The database `Heliostat Database` offers essentially the same content. Here, an entry is identified by the model code of the heliostat. The entry contains, among others, the information on the plants in which this heliostat is installed. This database therefore offers no added value for the purposes of this master's thesis.

Only a fraction of the data available in the databases can be used directly and has added value. These are shown in the following Tab. B.2.

Table B.2.: Heliostat panel configurations of additional CSP plants. n_x, n_y denote the number of panels in x and y directions, s_x, s_y represent panel dimensions in meters.

Plant	n_x	n_y	s_x [m]	s_y [m]
Badaling Dahan (China)	8	8	1.25	1.25
Jülich Solar Tower (Germany)	2	2	1.41	1.41
Sierra SunTower (USA)	1	1	1.00	1.00
Yumen Xinneng-Xinchen (China)	4	4	1.10	1.00
Sundrop CSP (Australia)	1	1	1.48	1.48

The majority of the data entered in the `Power Tower Plant Database` is duplicated from the data from [1] or does not provide any technical data on the installed heliostats. However, a few entries provide enough information to calculate or estimate the missing values, which is done below. Also, the missing values for the entry `Luneng Haixi 50-MW Molten Salt Tower` from Tab. B.1 are calculated. For the entries in Tab. B.3 below, the number of panels in the x-axis and y-axis and the area of the heliostat mirror surface are given. Entries in the database, where only the number of panels or only the mirror surface size is given, were not taken into account due to insufficient data.

Table B.3.: CSP plants without available panel size data but with known number of panels and total surface area.

Plant	n_x	n_y	Surface area [m^2]
Luneng Haixi 50-MW Molten Salt Tower (China)	4	8	138.0
ACME Solar Tower (India)	1	1	1.0
Atacama I (Chile)	4	8	140.0
CTGR Qinghai Golmud 100MW (China)	5	7	115.8
Jemalong Solar Thermal Station (Australia)	3	1	3.6

To complete the missing values for the parameters `panel_size_x` and `panel_size_y`, the following steps are taken. First, the mean value $\bar{n}_x$ and the mean panel ratio $\overline{\frac{\texttt{panel\_size\_x}}{\texttt{panel\_size\_y}}}$ are calculated for the merged Tab. B.1 and Tab. B.2. $\bar{n}_x$ is set as `panel_size_x` for the entries in Tab. B.3. Missing values of `panel_size_y` are obtained by scaling `panel_size_x` using $\overline{\frac{\texttt{panel\_size\_x}}{\texttt{panel\_size\_y}}}$. The entry `Jülich Solar Tower` is not included in this table because its geometry is intended for model tests using real data, rather than for generating synthetic data with Blender.[14]

[14] $\bar{n}_x$ = 2.2 and $\overline{\frac{\texttt{panel\_size\_x}}{\texttt{panel\_size\_y}}}$ = 1 for the data shown in Tab. B.1 and Tab. B.2. However, due to the incorrect inclusion of duplicates at that time, the mean and panel ratio were slightly distorted. The Blender models were therefore developed with different parameters than those given here ($\bar{n}_x$ = 1.76 and $\overline{\frac{\texttt{panel\_size\_x}}{\texttt{panel\_size\_y}}}$ = 1.3). However, as Chapter 3.4 shows, this miscalculation does not influence the clustering, since the cluster variables are the number of panels and not their dimensions. Furthermore, the overall task is not to exactly reproduce existing geometries, but to use them as a reference.

Appendix C

C.1. Evaluation of Circular Scenes

In the main part of the analysis, evaluations were conducted exclusively on rectangular Blender scenes. In this section, the corresponding evaluations for the circular scenes are presented. The AP values across all scenarios and trained models in the circular scenes are similar to those from the rectangular scenes. Nonetheless, a separate evaluation was performed, as the PCK in the circular scenes is significantly lower and would have distorted the overall interpretation if included in the same analysis. Furthermore, since the PCK@1.18cm is consistently lower than PCK@3px, no additional insights could be gained from the stricter threshold. Therefore, the PCK@1.18cm was not computed.

The following tables Tab. C.1 - Tab. C.4 show the numerical values of the test metrics AP and PCK@3px for each of the three scenarios. As in the main evaluation, the metrics are grouped by the test clusters and averaged for two seeds. All metric values are given in %. Images with a camera pitch angle of $\geq 50°$ are filtered out. The parameter `n_max_peaks` is set to 2,000 in training and to 30,000 in testing. The abbreviation *circ* stands for *circular*.

Table C.1.: Performance evaluation of models from scenario *No Duplicates*, strategy *Close First* (CF). Tested on circular scenes.

Test Cluster	**Training Cluster**	**mAP**$_{\mathrm{circ}}$	**PCK@3px**$_{\mathrm{circ}}$
	2	66.38	0.10
1	2,4	68.21	0.10
	2,4,3	71.31	0.10
	1	69.95	0.10
2	1,4	71.16	0.10
	1,4,3	72.52	0.10
	4	40.53	1.40
3	4,2	49.49	1.40
	4,2,1	50.04	1.40
	3	5.69	0.60
4	3,2	6.67	0.60
	3,2,1	8.68	0.60

Table C.2.: Performance evaluation of models from scenario *No Duplicates*, Strategy *Close First* (FF). Tested on circular scenes.

Test Cluster	Training Cluster	mAP$_{\mathrm{circ}}$	PCK@3px$_{\mathrm{circ}}$
1	3	30.84	0.10
	3,4	43.68	0.10
2	3	25.61	0.10
	3,4	39.97	0.10
3	1	23.75	1.30
	1,2	30.40	1.40
4	1	21.44	0.50
	1,2	22.31	0.50

Table C.3.: Performance evaluation of models from scenario *All Geometries*, strategy *Close First* (CF). Tested on circular scenes.

Test Cluster	Training Cluster	mAP$_{\mathrm{circ}}$	PCK@3px$_{\mathrm{circ}}$
	2	68.81	0.10
1	2,4	58.90	0.10
	2,4,3	72.51	0.10
	1	71.59	0.10
2	1,4	72.73	0.15
	1,4,3	73.01	0.10
	4	45.82	1.40
3	4,1	52.73	1.40
	4,1,2	52.49	1.40
	3	5.16	0.60
4	3,1	8.48	0.60
	3,1,2	8.43	0.60

Table C.4.: Performance evaluation of models from scenario *One Geometry per Cluster*, strategy *Close First* (CF). Tested on circular scenes.

Test Cluster	Training Cluster	mAP$_{\mathrm{circ}}$	PCK@3px$_{\mathrm{circ}}$
	2	25.59	0.1
1	2,3	56.41	0.1
	2,3,4	58.51	0.1
	1	51.33	0.1
2	1,3	56.89	0.1
	1,3,4	59.83	0.1
	4	19.71	1.2
3	4,1	30.88	1.35
	4,1,2	31.8	1.4
	3	1.22	0.5
4	3,1	6.11	0.5
	3,1,2	7.57	0.5

The parameter `max_n_peaks` limits the number of possible predictions the model can make on a heatmap. Predictions for keypoints and bounding box centers are handled separately. As part of the model evaluation, it was investigated whether the low PCK values observed in circular field layouts could be attributed to this parameter being set too low. Specifically, the concern was that a value of 2,000 may prevent the model from detecting all keypoints in densely populated scenes with many heliostats. To test this hypothesis, all models in the *No Duplicates* scenario were retrained and evaluated with an increased value of `max_n_peaks` set to 20,000. However, no significant improvements were observed that would justify a more in-depth investigation in this direction.

The significant difference in the test scores of the PCK metric between the circular and rect-angular Blender scenes was finally attributed to the following: All circular fields were created with a larger number of heliostats than the rectangular ones. This is because in rectangular fields, the number of heliostats can be increased more gradually by adding a single row of heliostats. However, the circumference of a circular field grows with each additional row of heliostats, making it difficult to precisely match the size of rectangular fields. As a result of the higher number of heliostats in the circular fields, drone imagery captures more heliostats and thus more keypoints. The higher density of keypoints and their potential overlap can pose a challenge for the model. Additionally, even with the same number of heliostats in the field, more heliostats are visible in the image in the circular layout due to the curved arrangement of the rows of heliostats, compared to the straight rows in the rectangular layout.

Bibliography

[1] G. Zhu et al. *Roadmap to Advance Heliostat Technologies for Concentrating Solar-Thermal Power*. 2022. URL: https://www.nrel.gov/docs/fy22osti/83041.pdf (visited on 11/15/2024).

[2] International Energy Agency. *Technology Roadmap - Concentrating Solar Power*. 2010. URL: https://www.iea.org/reports/technology-roadmap-concentrating-solar-power (visited on 11/29/2024).

[3] M. Alam et al. “Conventional and Emerging CSP Technologies and Design Modifications: Research Status and Recent Advancements”. In: *International Journal of Thermofluids* 20 (2023). ISSN: 2666-2027. DOI: 10.1016/j.ijft.2023.100406. URL: https://www.sciencedirect.com/science/article/pii/S2666202723001234 (visited on 11/08/2024).

[4] M. Kaltschmitt, W. Streicher, and A. Wiese. *Erneuerbare Energien: Systemtechnik, Wirtschaftlichkeit, Umweltaspekte*. 5., erw. Aufl. Berlin and Heidelberg: Springer Vieweg, 2013. ISBN: 978-3-642-03248-6. DOI: 10.1007/978-3-642-03249-3. URL: https://link.springer.com/book/10.1007/978-3-642-03249-3 (visited on 12/10/2024).

[5] M. Röger, C. Prahl, and S. Ulmer. “Heliostat Shape and Orientation by Edge Detection”. In: *Journal of Solar Energy Engineering* 132.2 (2010). ISSN: 0199-6231. DOI: 10.1115/1.4001400. URL: https://asmedigitalcollection.asme.org/solarenergyengineering/article/132/2/021002/455822/Heliostat-Shape-and-Orientation-by-Edge-Detection (visited on 11/11/2024).

[6] J. Sattler et al. *Review of heliostat calibration and tracking control methods*. 2020. DOI: 10.1016/j.solener.2020.06.030. URL: https://www.sciencedirect.com/science/article/pii/S0038092X20306447 (visited on 11/05/2024).

[7] C.. Xu et al. “Energy and exergy analysis of solar power tower plants”. In: *Applied Thermal Engineering* 31.17 (2011), pp. 3904–3913. ISSN: 1359-4311. DOI: 10.1016/j.applthermaleng.2011.07.038. URL: https://www.sciencedirect.com/science/article/pii/S135943111100398X (visited on 11/18/2024).

[8] R. A. Mitchell and G. Zhu. “A non-intrusive optical (NIO) approach to characterize heliostats in utility-scale power tower plants: Methodology and in-situ validation”. In: *Solar Energy* 209 (2020), pp. 431–445. ISSN: 0038092X. DOI: 10.1016/j.solener.2020.09.004. URL: https://research-hub.nrel.gov/en/publications/a-non-intrusive-optical-nio-approach-to-characterize-heliostats-i-3 (visited on 02/20/2025).

[9] W. Jessen et al. “A Two-Stage Method for Measuring the Heliostat Offset”. In: *AIP Conference Proceedings*. Ed. by American Institute of Physics. Vol. 2445. 2022, p. 070005. DOI: 10.1063/5.0087036. URL: https://pubs.aip.org/aip/acp/article/2445/1/070005/2823711/A-two-stage-method-for-measuring-the-heliostat (visited on 12/03/2024).

[10] J. Krauth et al. “HelioPoint – A Fast Airborne Calibration Method for Heliostat Fields”. In: *Journal of Solar Energy Engineering* 146.6 (2024). ISSN: 0199-6231. DOI: 10.1115/1.4065868. (Visited on 01/19/2025).

[11] J. Yellowhair et al. “Development of an aerial imaging system for heliostat canting assessments”. In: *SOLARPACES 2020: 26th International Conference on Concentrating Solar Power and Chemical Energy Systems*. AIP Conference Proceedings. AIP Publishing, 2022, p. 120024. DOI: 10.1063/5.0087057. URL: https://pubs.aip.org/aip/acp/article/2445/1/120024/2824316/Development-of-an-aerial-imaging-system-for (visited on 02/20/2025).

[12] D. Kesseli et al. “A Combined Computer Vision and Deep Learning Approach for Rapid Drone-Based Optical Characterization of Parabolic Troughs”. In: *Journal of Solar Energy Engineering* 145.2 (2023). ISSN: 0199-6231. DOI: 10.1115/1.4055172. URL: https://asmedigitalcollection.asme.org/solarenergyengineering/article/145/2/021008/1145346/A-Combined-Computer-Vision-and-Deep-Learning (visited on 12/04/2024).

[13] R. Broda et al. *Towards deep learning based airborne monitoring methods for heliostats in solar tower power plants*. 2023. DOI: 10.1117/12.2676821. URL: https://www.researchgate.net/publication/374467206_Towards_deep_learning_based_airborne_monitoring_methods_for_heliostats_in_solar_tower_power_plants (visited on 11/15/2024).

[14] I. Sarker. “Machine Learning: Algorithms, Real-World Applications and Research Directions”. In: *SN computer science* 2.3 (2021), p. 160. DOI: 10.1007/s42979-021-00592-x. URL: https://link.springer.com/article/10.1007/s42979-021-00592-x (visited on 11/28/2024).

[15] E. Hossain, ed. *Machine Learning Crash Course for Engineers*. 1st ed. 2024. Cham: Springer International Publishing and Imprint Springer, 2024. ISBN: 978-3-031-46989-3. DOI: 10.1007/978-3-031-46990-9. URL: https://link.springer.com/chapter/10.1007/978-3-031-46990-9_1 (visited on 11/06/2024).

[16] J. Carballo et al. “Machine learning for solar trackers”. In: *SOLARPACES 2018: International Conference on Concentrating Solar Power and Chemical Energy Systems*. Ed. by American Institute of Physics. AIP Conference Proceedings. AIP Publishing, 2019, p. 030012. DOI: 10.1063/1.5117524. URL: https://www.researchgate.net/publication/334727667_Machine_learning_for_solar_trackers (visited on 12/04/2024).

[17] B. Liu et al. “Deep Learning Method for Heliostat Instance Segmentation”. In: *SolarPACES Conference Proceedings* 1 (2023). DOI: 10.52825/solarpaces.v1i.735. URL: https://www.tib-op.org/ojs/index.php/solarpaces/article/view/735 (visited on 12/04/2024).

[18] F. Xu and C. Li. “An Improved YOLO Model for Automatic Detection of Heliostats in a Concentrated Solar Power Plant”. In: *2024 IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS)*. 2024, pp. 1412–1416. URL: https://ieeexplore.ieee.org/document/10606608 (visited on 12/04/2024).

[19] R. Broda et al. *Bridging the Sim2Real Gap: Training Deep Neural Networks for Heliostat Detection with Purely Synthetic Data*. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12671/1267108/Towards-deep-learning-based-airborne-monitoring-methods-for-heliostats-in/10.1117/12.2676821.short (visited on 04/26/2025).

[20] Blender. *Blender 4.4 Manual: About Blender*. 2025-03-23. URL: https://docs.blender.org/manual/en/latest/getting_started/about/index.html (visited on 03/23/2025).

[21] M. Denninger et al. *BlenderProc*. 2019. URL: https://github.com/DLR-RM/BlenderProc (visited on 03/24/2025).

[22] R. Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2013. URL: http://arxiv.org/pdf/1311.2524 (visited on 03/16/2025).

[23] J. Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. URL: https://arxiv.org/pdf/1506.02640 (visited on 03/16/2025).

[24] Heliostat Consortium for Concentrating Solar-Thermal Power. *HelioCon Database: Power Tower Plant Database, Heliostat Database*. URL: https://www.heliocon.org/resources/plant_information_overview.html (visited on 03/25/2025).

[25] X. Tan. *Neural Text-to-Speech Synthesis*. 1st ed. 2023. Artificial Intelligence. Singapore: Springer Nature Singapore and Imprint Springer, 2023. ISBN: 978-981-99-0826-4. DOI: 10.1007/978-981-99-0827-1. URL: https://link.springer.com/book/10.1007/978-981-99-0827-1 (visited on 11/06/2024).

[26] H. Li. *Statistical learning methods*. Tsinghua University Press, 2019. ISBN: 9787302517276. URL: http://www.tup.tsinghua.edu.cn/en/book_08132901.html (visited on 03/13/2025).

[27] C. Bishop and H. Bishop. *Deep learning: Foundations and concepts*. Cham, Switzerland: Springer, 2024. ISBN: 978-3-031-45467-7. DOI: 10.1007/978-3-031-45468-4. URL: https://link.springer.com/book/10.1007/978-3-031-45468-4 (visited on 11/05/2024).

[28] G. Cerulli. *Fundamentals of supervised machine learning: With applications in Python, R, and Stata*. Statistics and Computing. Cham: Springer, 2023. ISBN: 978-3-031-41336-0. DOI: 10.1007/978-3-031-41337-7. URL: https://link.springer.com/book/10.1007/978-3-031-41337-7 (visited on 01/13/2025).

[29] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. 2nd ed. 2023. Cham: Springer International Publishing and Imprint Springer, 2023. ISBN: 978-3-031-29641-3. DOI: 10.1007/978-3-031-29642-0. URL: https://link.springer.com/book/10.1007/978-3-031-29642-0 (visited on 01/17/2025).

[30] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://www.nature.com/articles/nature14539 (visited on 01/17/2025).

[31] S. Trimpe. *Fundamentals of machine Learning: Lecture 8: Linear Models for Regression*. 2023. URL: https://moodle.rwth-aachen.de/mod/folder/view.php?id=1466995 (visited on 04/24/2025).

[32] Y. Geng et al. *Practical Machine Learning Illustrated with KNIME*. 1st ed. 2024. Singapore: Springer Nature Singapore and Imprint Springer, 2024. ISBN: 978-981-97-3953-0. DOI: 10.1007/978-981-97-3954-7. URL: https://link.springer.com/book/10.1007/978-981-97-3954-7 (visited on 11/06/2024).

[33] Z. Zhong-Qui et al. *Object Detection with Deep Learning: A Review*. 2018. URL: http://arxiv.org/pdf/1807.05511 (visited on 11/18/2024).

[34] R. I. Mukhamediev et al. “From Classical Machine Learning to Deep Neural Networks: A Simplified Scientometric Review”. In: *Applied Sciences* 11.12 (2021), p. 5541. DOI: 10.3390/app11125541. URL: https://www.mdpi.com/2076-3417/11/12/5541 (visited on 01/17/2025).

[35] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/ (visited on 01/18/2025).

[36] A. Nischwitz et al. *Bildverarbeitung*. 3., neu bearb. Aufl. Vol. Bd. 2. Studium. Wiesbaden: Vieweg + Teubner, 2011. ISBN: 978-3-8348-1712-9. DOI: 10.1007/978-3-8348-8300-1. (Visited on 04/26/2025).

[37] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791. URL: https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition (visited on 03/11/2025).

[38] H. Fujiyoshi, T. Hirakawa, and T. Yamashita. “Deep learning-based image recognition for autonomous driving”. In: *IATSS Research* 43.4 (2019), pp. 244–252. ISSN: 0386-1112. DOI: 10.1016/j.iatssr.2019.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0386111219301566 (visited on 03/17/2025).

[39] G. Aurelién. *Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. First edition, fifth release. Beijing et al.: O'Reilly, 2018. ISBN: 9781491962299. URL: https://www.clc.hcmus.edu.vn/wp-content/uploads/2017/11/Hands_On_Machine_Learning_with_Scikit_Learn_and_TensorFlow.pdf (visited on 01/04/2025).

[40] A. Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial Intelligence Review* 53.8 (2020), pp. 5455–5516. ISSN: 0269-2821. DOI: 10.1007/s10462-020-09825-6. URL: https://link.springer.com/article/10.1007/s10462-020-09825-6 (visited on 01/05/2025).

[41] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. URL: http://arxiv.org/pdf/1505.04597 (visited on 01/09/2025).

[42] K. He et al. *Deep Residual Learning for Image Recognition*. 2015. URL: http://arxiv.org/pdf/1512.03385 (visited on 01/09/2025).

[43] A. Jaiswal et al. *Class-agnostic Object Detection*. 2020. URL: http://arxiv.org/pdf/2011.14204 (visited on 11/20/2024).

[44] R. Girshick. *Fast R-CNN*. 2015. URL: http://arxiv.org/pdf/1504.08083 (visited on 03/16/2025).

[45] S. Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. URL: http://arxiv.org/pdf/1506.01497 (visited on 03/16/2025).

[46] K. He et al. *Mask R-CNN*. 2017. URL: http://arxiv.org/pdf/1703.06870 (visited on 03/16/2025).

[47] J. Redmon and A. Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. URL: http://arxiv.org/pdf/1612.08242 (visited on 03/18/2025).

[48] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. URL: http://arxiv.org/pdf/1804.02767 (visited on 03/18/2025).

[49] L. Cuiyin, X. Jishang, and W. Feng. "A Review of Keypoints' Detection and Feature Description in Image Registration". In: *Scientific Programming* 2021 (2021), pp. 1–25. ISSN: 1058-9244. DOI: 10.1155/2021/8509164. URL: https://onlinelibrary.wiley.com/doi/full/10.1155/2021/8509164 (visited on 03/18/2025).

[50] S. Wei et al. "Convolutional Pose Machines". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4724–4732. DOI: 10.1109/CVPR.2016.511. URL: https://ieeexplore.ieee.org/document/7780880 (visited on 03/18/2025).

[51] R. Padilla, S.. Netto, and E. Da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. Ed. by A. Paiva. Piscataway, NJ: IEEE, 2020, pp. 237–242. ISBN: 978-1-7281-7539-3. DOI: 10.1109/IWSSIP48289.2020.9145130. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9145130 (visited on 11/27/2024).

[52] S. Rath. *Human Pose Detection using PyTorch Keypoint RCNN*. Ed. by Debugger Cafe, Machine Learning and Deep Learning. 2020. URL: https://debuggercafe.com/human-pose-detection-using-pytorch-keypoint-rcnn/ (visited on 03/03/2025).

[53] X. Zhou, D. Wang, and P. Krähenbühl. *Objects as Points*. 2019. URL: http://arxiv.org/pdf/1904.07850 (visited on 01/05/2025).

[54] D. Toumpanakis and A. Liao. "ImageNet dataset". In: *Radiopaedia.org*. Radiopaedia.org, 2005. DOI: 10.53347/rID-88649. URL: https://radiopaedia.org/articles/imagenet-dataset?lang=us&utm_source=chatgpt.com (visited on 03/30/2025).

[55] J. Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. (Visited on 03/30/2025).

[56] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y. URL: https://link.springer.com/article/10.1007/s11263-015-0816-y (visited on 03/30/2025).

[57] L. Xu et al. *Pose for Everything: Towards Category-Agnostic Pose Estimation*. 2022. URL: http://arxiv.org/pdf/2207.10387 (visited on 11/27/2024).

[58] T. Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. URL: http://arxiv.org/pdf/1405.0312 (visited on 03/24/2025).

[59] J. Hui. *mAP (mean Average Precision) for Object Detection*. 2019. URL: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173 (visited on 03/31/2025).

[60] H. Zhang et al. "Open-Vocabulary Animal Keypoint Detection with Semantic-Feature Matching". In: *International Journal of Computer Vision* 132.12 (2024), pp. 5741–5758. ISSN: 0920-5691. DOI: 10.1007/s11263-024-02126-3. URL: https://link.springer.com/article/10.1007/s11263-024-02126-3 (visited on 04/26/2025).

[61] R. P. Merchán et al. "High temperature central tower plants for concentrated solar power: 2021 overview". In: *Renewable and Sustainable Energy Reviews* 155 (2022), p. 111828. ISSN: 1364-0321. DOI: 10.1016/j.rser.2021.111828. URL: https://www.sciencedirect.com/science/article/pii/S1364032121010923 (visited on 11/08/2024).

[62] A. Rizvi et al. “A review and classification of layouts and optimization techniques used in design of heliostat fields in solar central receiver systems”. In: *Solar Energy* 218 (2021), pp. 296–311. ISSN: 0038092X. URL: `https://www.researchgate.net/publication/350143614_A_review_and_classification_of_layouts_and_optimization_techniques_used_in_design_of_heliostat_fields_in_solar_central_receiver_systems` (visited on 11/05/2024).

[63] S. alexopoulos and B. Hoffschmidt. “Solar tower power plant in Germany and future perspectives of the development of the technology in Greece and Cyprus”. In: *Renewable Energy* 35.7 (2010), pp. 1352–1356. ISSN: 0960-1481. DOI: `10.1016/j.renene.2009.11.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0960148109004790` (visited on 11/11/2024).

[64] M. Dehli, E. Doering, and H. Schedwill, eds. *Grundlagen der Technischen Thermodynamik: Für eine praxisorientierte Lehre*. 10. Auflage. Wiesbaden and Heidelberg: Springer Vieweg, 2023. ISBN: 978-3-658-41250-0. DOI: `10.1007/978-3-658-41251-7`. URL: `https://link.springer.com/chapter/10.1007/978-3-658-41251-7_7` (visited on 11/15/2024).

[65] H. Watter. *Regenerative Energiesysteme: Grundlagen, Systemtechnik und Analysen ausgeführter Beispiele nachhaltiger Energiesysteme*. 6. Auflage. Lehrbuch. Wiesbaden and Heidelberg: Springer Vieweg, 2022. ISBN: 978-3-658-35867-9. DOI: `10.1007/978-3-658-35868-6`. URL: `https://link.springer.com/chapter/10.1007/978-3-658-35868-6_11` (visited on 11/05/2024).

[66] D. Enescu et al. “Thermal Energy Storage for Grid Applications: Current Status and Emerging Trends”. In: *Energies* 13.2 (2020), p. 340. DOI: `10.3390/en13020340`. URL: `https://www.researchgate.net/publication/338524770_Thermal_Energy_Storage_for_Grid_Applications_Current_Status_and_Emerging_Trends` (visited on 11/15/2024).

[67] E. Salgado-Plasencia et al. “SCADA-Based Heliostat Control System with a Fuzzy Logic Controller for the Heliostat Orientation”. In: *Applied Sciences* 9.15 (2019), p. 2966. DOI: `10.3390/app9152966`. URL: `https://www.mdpi.com/2076-3417/9/15/2966` (visited on 01/11/2025).

[68] G. Weinrebe. “Technische, ökologische und ökonomische Analyse von solarthermischen Turmkraftwerken”. Promotion. Stuttgart: Universität Stuggart, 2000. URL: `https://www.researchgate.net/publication/33787965_Technische_okologische_und_okonomische_Analyse_von_solarthermischen_Turmkraftwerken` (visited on 01/06/2025).

[69] Wikipedia. *PS10 Solar Power Plant*. URL: `https://en.wikipedia.org/wiki/PS10_solar_power_plant` (visited on 12/08/2024).

[70] Green City Times. *ISEGS - A Shining Example of Concentrated Solar Power (CSP) in California*. URL: `https://www.greencitytimes.com/ivanpah-solar-electric-generating-system/` (visited on 12/08/2024).

[71] National Renewable Energy Laboratory. *Planta Solar 10 -PS10 CSP Project*. 2022. URL: https://solarpaces.nrel.gov/project/planta-solar-10-ps10 (visited on 12/10/2024).

[72] National Renewable Energy Laboratory. *Ivanpah Solar Electric Generating System CSP Project*. 2022. URL: https://solarpaces.nrel.gov/project/ivanpah-solar-electric-generating-system (visited on 12/10/2024).

[73] Deutscher Fußball Bund. *Fussball-Regeln: 2023/2024*. URL: https://www.dfb.de/fileadmin/_dfbdam/287914-AU2300707_PL_Broschuere.pdf (visited on 01/24/2025).

[74] Statistisches Bundesamt. *Stromverbrauch der privaten Haushalte nach Haushaltsgrößenklassen*. 2023. URL: https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Umwelt/UGR/private-haushalte/Tabellen/stromverbrauch-haushalte.html#fussnote-2-133562 (visited on 01/13/2025).

[75] A. Yerudkar et al. "Economically feasible solutions in concentrating solar power technology specifically for heliostats – A review". In: *Renewable and Sustainable Energy Reviews* 189 (2024). ISSN: 1364-0321. DOI: 10.1016/j.rser.2023.113825. URL: https://www.sciencedirect.com/science/article/pii/S1364032123006822 (visited on 11/08/2024).

[76] A. Pfahl et al. "Progress in heliostat development". In: *Solar Energy* 152 (2017), pp. 3–37. ISSN: 0038092X. DOI: 10.1016/j.solener.2017.03.029. URL: https://www.sciencedirect.com/science/article/pii/S0038092X17301895 (visited on 11/08/2024).

[77] F. Téllez et al. *State of the Art in Heliostats and Definition of Specifications: Survey for a low cost heliostat development*. 2015. URL: https://www.stage-ste.eu/deliverables/STAGE_STE_Deliverable_12_1.pdf (visited on 11/08/2024).

[78] I. Les et al. "Techno-Economic Analysis of Central Receiver Plants According to the Optical Error of the Solar Field". In: *SolarPACES Conference Proceedings* 1 (2023). DOI: 10.52825/solarpaces.v1i.697. URL: https://www.tib-op.org/ojs/index.php/solarpaces/article/view/697 (visited on 11/19/2024).

[79] C. Jones et al. *Heliostat Cost Reduction Study*. URL: https://api.semanticscholar.org/CorpusID:111289300 (visited on 12/06/2024).

[80] J. Freeman, E. U. Kiranlal, and S. R. Rajasree. "Study of the errors influencing heliostats for calibration and control system design". In: *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*. Ed. by Institute of Electrical and Electronics Engineers. USA: Curran Associates, Inc., 2014, pp. 1–8. DOI: 10.1109/ICRAIE.2014.6909113. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6909113&tag=1 (visited on 11/19/2024).

[81] U. Harten. *Physik: Eine Einführung für Ingenieure und Naturwissenschaftler*. 9. Auflage. Berlin and Heidelberg: Springer Vieweg, 2024. ISBN: 978-3-662-68483-2. DOI: 10.1007/978-3-662-68484-9. URL: https://link.springer.com/book/10.1007/978-3-662-68484-9 (visited on 03/01/2025).

[82] M. Sarr, A. Thiam, and B. Dieng. “ANFIS and ANN models to predict heliostat tracking errors”. In: *Heliyon* 9.1 (2023). ISSN: 2405-8440. DOI: 10.1016/j.heliyon.2023.e12804. URL: https://www.cell.com/heliyon/fulltext/S2405-8440(23)00011-7 (visited on 03/01/2025).

[83] K. Stone. “Automatic heliostat track alignment method”. US4564275A. URL: https://patentimages.storage.googleapis.com/52/e2/7d/867bb217876b81/US4564275.pdf (visited on 12/03/2024).

[84] M. Röger et al. *Verfahren zur Vermessung und Kalibrierung von Heliostaten*. Ed. by Deutsches Zentrum für Luft- und Raumfahrt. 2018. URL: https://elib.dlr.de/122783/1/20180704_Soko18_Vermessung_Kalibrierung_Helios_FINAL.pdf (visited on 11/20/2024).

[85] J. Dabrowski et al. “Verfahren zur Positionsbestimmung oder zur Antriebsregelung eines eine Spiegelfläche aufweisenden Heliostaten sowie System zur Positionsbestimmung oder zur Antriebsregelung des Heliostaten”. DE102013207022B3. 2013. URL: https://patentimages.storage.googleapis.com/ca/76/54/2b876df78f75fd/DE102013207022B3.pdf (visited on 02/19/2025).

[86] K. Hickerson and D. Reznik. “Heliostat with integrated image-based tracking controller”. US20080236568A1. URL: https://patentimages.storage.googleapis.com/b1/ce/31/20ade751127bfc/US20080236568A1.pdf (visited on 02/19/2025).

[87] A. Pfahl, R. Buck, and K. Rehschu. “Method for controlling the alignment of a heliostat with respect to a receiver, heliostat device and solar power plant”. US8651100B2. URL: https://patentimages.storage.googleapis.com/15/7a/23/955d9ceebb50af/US8651100.pdf (visited on 02/19/2025).

[88] A. Yogev and V. Krupkin. “Control of a heliostat field in a solar energy plant”. URL: https://patentimages.storage.googleapis.com/98/4d/e8/0099fa8c0a5820/US5862799.pdf (visited on 02/19/2025).

[89] J. Fitch et al. “Heliostat control scheme using cameras”. US20110120448A1. URL: https://patentimages.storage.googleapis.com/e5/7b/36/94a974004f19c0/US20110120448A1.pdf (visited on 02/19/2025).

[90] B. Hines and R. Johnson. “Appartus and method for pointing light sources”. US20140110560. URL: https://patents.google.com/patent/US20140110560A1/en?oq=+US20140110560 (visited on 02/19/2025).

[91] T. Luhmann et al. *Close-range photogrammetry and 3D imaging*. 3rd edition. De Gruyter STEM. Berlin and Boston: De Gruyter, 2020. ISBN: 9783110607253. DOI: 10.1515/9783110607253. URL: https://www.degruyter.com/document/doi/10.1515/9783110607253/html#contents (visited on 02/24/2025).

[92] S. Werling. *Deflektometrie zur automatischen Sichtprüfung und Rekonstruktion spiegelnder Oberflächen*. Schriftenreihe Automatische Sichtprüfung und Bildverarbeitung. Erscheinungsort nicht ermittelbar: KIT Scientific Publishing, 2011. ISBN: 9783866446878. DOI: 44647. URL: `https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A` (visited on 02/04/2025).

[93] A. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. URL: `http://arxiv.org/pdf/1704.04861` (visited on 04/01/2025).

[94] C. Ning et al. "Inception Single Shot MultiBox Detector for object detection". In: *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. 2017, pp. 549–554. DOI: `10.1109/ICMEW.2017.8026312`. URL: `https://ieeexplore.ieee.org/document/8026312` (visited on 04/04/2025).

[95] J. Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems 27* (). URL: `http://arxiv.org/pdf/1411.1792` (visited on 03/30/2025).

[96] K. Duan et al. *CenterNet: Keypoint Triplets for Object Detection*. 2019. URL: `http://arxiv.org/pdf/1904.08189` (visited on 01/09/2025).

[97] T. Lin et al. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007. DOI: `10.1109/ICCV.2017.324`. URL: `https://ieeexplore.ieee.org/document/8237586` (visited on 03/03/2025).

[98] T. Pavlovic, I. Radonjic, and L. Milosavljevic. "A review of concentrating solar power plants in the world and their potential use in Serbia". In: *Renewable and Sustainable Energy Reviews* 16.6 (2012), pp. 3891–3902. ISSN: 1364-0321. DOI: `10.1016/j.rser.2012.03.042`. URL: `https://www.sciencedirect.com/science/article/pii/S1364032112002250` (visited on 11/08/2024).

[99] M. Denninger et al. "BlenderProc2: A Procedural Pipeline for Photorealistic Rendering". In: *Journal of Open Source Software* 8.82 (2023), p. 4901. DOI: `10.21105/joss.04901`. URL: `https://joss.theoj.org/papers/10.21105/joss.04901` (visited on 03/24/2025).

[100] T. Foote and M. Purvis. *Standard Units of Measure and Coordinate Conventions*. 2010. URL: `https://www.ros.org/reps/rep-0103.html` (visited on 04/04/2025).

[101] H. Goldstein, C. Poole, and J. Safko. *Classical Mechanics: Third Edition*. Addison-Wesley, 2002. URL: `https://poincare.matf.bg.ac.rs/~zarkom/Book_Mechanics_Goldstein_Classical_Mechanics_optimized.pdf` (visited on 04/04/2025).

[102] J. Vries. *Learn OpenGL - Graphics programming: Learn modern OpenGL graphics programming in a step-by-step fashion*. Kendall & Welling, 2020. ISBN: 978-90-90-33256-7. URL: `https://learnopengl.com/book/book_pdf.pdf` (visited on 01/24/2025).

[103] J. Craig. *Introduction to Robotics: Mechanics and Control*. 4th ed. United Kingdom: Pearson Education Limited, 2022. URL: https://api.pageplace.de/preview/DT0400.9781292164953_A42125292/preview-9781292164953_A42125292.pdf (visited on 04/08/2025).

[104] M. Röger. *SolarPACES Guideline for Heliostat Performance Testing*. Ed. by IEA Technology Collaboration Programme SolarPACES. 2023. URL: https://elib.dlr.de/199338/ (visited on 05/23/2023).

[105] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. 2. edition, 17.printing. Cambridge: Cambridge Univ. Press, 2018. ISBN: 9780521540513. DOI: 10.1017/CBO9780511811685. URL: https://doi.org/10.1017/CBO9780511811685 (visited on 01/09/2025).

[106] Y. Zhou et al. *On the Continuity of Rotation Representations in Neural Networks*. 2018. URL: http://arxiv.org/pdf/1812.07035 (visited on 02/05/2025).

[107] K. Backhaus et al. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 17., aktualisierte Auflage. Wiesbaden: Springer Gabler, 2023. ISBN: 978-3-658-40464-2. DOI: 10.1007/978-3-658-40465-9. URL: https://link.springer.com/book/10.1007/978-3-658-40465-9 (visited on 11/28/2024).

[108] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* New York, New York: John Wiley & Sons Inc., 1990. URL: https://download.e-bookshelf.de/download/0000/5714/40/L-G-0000571440-0015244442.pdf (visited on 04/26/2025).

[109] D. Picard. *Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision*. 2021. URL: http://arxiv.org/pdf/2109.08203 (visited on 04/28/2025).

[110] R. Takashi and T. Matsubara. “Data Augmentation Using Random Image Cropping and Patching for Deep CNNs”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.9 (2020), pp. 2917–2931. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2019.2935128. URL: http://arxiv.org/pdf/1811.09030 (visited on 04/09/2025).

[111] I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. 2017. URL: http://arxiv.org/pdf/1711.05101 (visited on 04/19/2025).

[112] L. Smith. “A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay”. In: *US Naval Research Laboratory Technical Report* (2018). URL: http://arxiv.org/pdf/1803.09820 (visited on 04/09/2025).

[113] H. Li. *Machine learning methods*. Singapore: Springer, 2024. ISBN: 9789819939176. DOI: 10.1007/978-981-99-3917-6. URL: https://link.springer.com/book/10.1007/978-981-99-3917-6 (visited on 01/12/2025).

[114] R. Gonzalez and R. Woods. *Digital image processing*. Fourth, global edition. New York, New York: Pearson Education, 2018. ISBN: 9781292223070. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5832133 (visited on 03/02/2025).

[115] G. Bern et al. "Novel imaging closed loop control strategy for heliostats". In: *Proceedings of the 22nd SolaPACES 2016 International Conference, Abu Dhabi, UAE*. Ed. by SolarPACES. AIP Conference Proceedings. AIP Publishing, 2017, p. 030005. DOI: 10.1063/1.4984348. URL: https://www.researchgate.net/publication/317984003_Novel_imaging_closed_loop_control_strategy_for_heliostats (visited on 02/19/2025).

[116] C. Prahl, M. Röger, and W. Kiewitt. *Advances in optical measurement techniques for solar concentrators*. 2009. URL: https://www.researchgate.net/publication/224990464_Advances_in_Optical_Measurement_Techniques_for_Solar_Concentrators (visited on 02/19/2025).

[117] F. Göhring et al. "Verfahren zur Vermessung von Heliostaten". DE 10 2015 217 086 A1. URL: https://patentimages.storage.googleapis.com/cd/3f/17/74d5325e3cf9c8/DE102015217086A1.pdf (visited on 02/19/2025).

[118] J. Klimek et al. *Radar technology for heliostat position control*. 2012. URL: https://www.researchgate.net/publication/259896664_Radar_Technology_For_Heliostat_Position_Control (visited on 02/19/2025).