Fast cell detection and distortion correction for outdoor electroluminescence images

Evgenii Sovetkin[1†], Bart E. Pieters[1], Andreas Gerber[1], Liviu Stoicescu[2] and Pascal Koelblin[3]
[1]IEK-5 Photovoltaics, Forschungszentrum Jülich, 52425 Jülich, Germany
[2]Solarzentrum Stuttgart GmbH, 70197 Stuttgart, Germany
[3]Institute for Photovoltaics and Research Center SCoPE, University of Stuttgart, 70569 Stuttgart, Germany

***Abstract*—This paper proposes a fast and robust method for electroluminescence image preprocessing, where lens and perspective distortions are corrected, and individual cells in the module are detected. Our approach works with low-resolution (640×512 pixels) images, uses an image-to-image translation neural network, and leverages the geometric properties of a photovoltaic module. The fast computational speed of the neural network allows us to complete image analysis in under 0.5 seconds, which is ten times faster than currently published methods. In addition, the geometry-based postprocessing makes our approach robust to small misdetections in the neural network output.**

***Index Terms*—electroluminescence, pix2pix neural network, distortion correction**

I. Introduction

The International Energy Agency (IEA) forecasts that solar energy will have at least 27% of the global share of energy production by 2050 [1]. With the increasing photovoltaic (PV) production capacity, there is a growing demand for monitoring and early-fault detection to minimize PV plant losses.

Time series data of inverters and strings may indicate a potential fault; however, for a thorough investigation, spatially-resolved measurements (imaging) provide a deeper understanding of the module's health and allows for diagnosing faults and defects, which is important for warranty or insurance claims.

With the increase in the installed PV capacity and the emergence of automated drone-based measurements, fast and real-time image processing algorithms for module and cell detection will become crucial for automated plant inspections.

In particular, camera and perspective distortion corrections and module and cell detection are essential routines for processing in-field-collected images. These algorithms are important as these processing steps are always required and applied before other analysis methods are used.

Previous studies of such algorithms are based on a series of image-processing filters. First, the original image undergoes a series of transformations, amplifying PV module characteristics such as grid lines and busbars. Those characteristics are then used to correct for lens and perspective distortions and detect individual cells. Sovetkin et al. [2] use Hough Transform to find short pieces of straight lines, followed by robust linear regression predicting the slope change of those lines hence estimating the parameters of the perspective distortion. Deitsch et al. [3] use tensor voting to amplify the line features that point in a similar direction and subpixel-accuracy routine to compute the location of cell corners. Kölblin et al. [4] detect cell gaps using an iterative linear Hough transform applied on an adaptive locally thresholded image.

This paper replaces image-processing filters with a trained image-to-image translation neural network, significantly reducing the runtime. Furthermore, we leverage the typical PV module's geometrical properties to overcome misdetections in the neural network and improve the algorithm's robustness.

II. Methodology

Our image correction approach consists of four steps. In the first step, we apply the trained image-to-image neural network that takes as input the EL image and outputs a binary image of a cell grid (see Section II-B and Figure 2). This grid hints at the location of individual cells, and the neural network is trained on the data obtained from the previously published method (Section II-A). In the second step, we apply several routines that rank cell corners and the corresponding cell locations within the module (Section II-C). The ranking of the cells corresponds to the quality of detection. In the third step, the accuracy of the cell corner coordinates is improved with a subpixel-accuracy method (Section II-D). Lastly, we use the best cell corners to either estimate the lens distortion parameters or correct the perspective distortion (Section II-E).

The flowchart in Figure 1 illustrates our approach pipeline. The original image can be used either to accumulate information to estimate the camera distortion matrix or to estimate homography and extract individual cell images.

A. Data

We develop our methods using the daylight electroluminescence (EL) images collected with the DaySy system of Solarzentrum Stuttgart, GmbH [5]. We use a low-resolution InGaAs sensor with a 640×512 pixels resolution. Our study uses data from several commercial PV fields in Germany containing several types of c-Si modules. Figure 2 (top) shows an example of an EL image.

We utilize the methods proposed in [4] to generate training data, which works well with our low-resolution images. First, we generate a binary grid image for every EL image using the computed cell data (see Figure 2, bottom). Then, we manually clear the training dataset, removing faulty images. For this

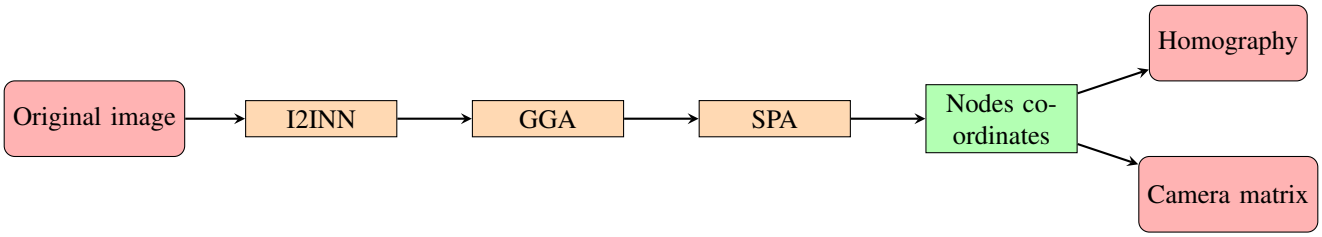

Fig. 1. A flowchart describing our approach pipeline. The original image is processed by an image-to-image neural network (I2INN, Section II-B). The graph-geometric analysis (GGA) is applied to rank the cell corner quality (Section II-C). The accuracy of the node coordinates is improved with the subpixel-accuracy method (Section II-D). The nodes resulting node coordinates are then used to estimate homography or a camera matrix (Section II-E)

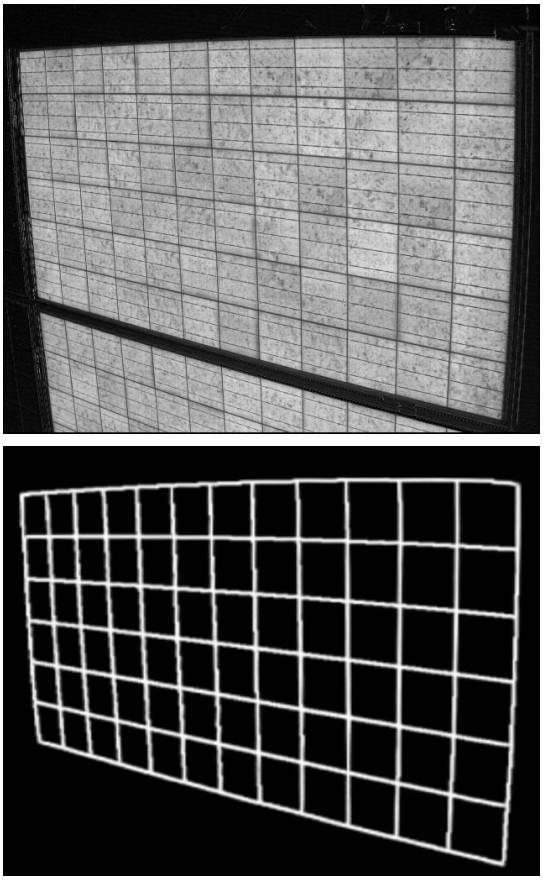

Fig. 2. Above: an EL image with 640×512 pixels resolution. Below: output of the neural network

paper, we use a sample 1800 number of modules to train the neural network.

B. Image-to-image neural network (I2INN)

We use the Pix2Pix [6] neural network for the image-to-image neural network (I2INN). First, we prepared a training dataset discussed in Section II-A. Then, the training images are augmented using a random perspective transformation, shift, symmetry reflection, contrast scaling, and random noise. Lastly, the resulting image is resized to have a 256×256 pixels resolution. From 1800 module images, we generate a training dataset with 126000 images. The training uses NVIDIA GeForce GTX 970 graphic card for 20 epochs. In the Pix2Pix architecture, we experimented with several different generators and discriminators, with all networks yielding similar results.

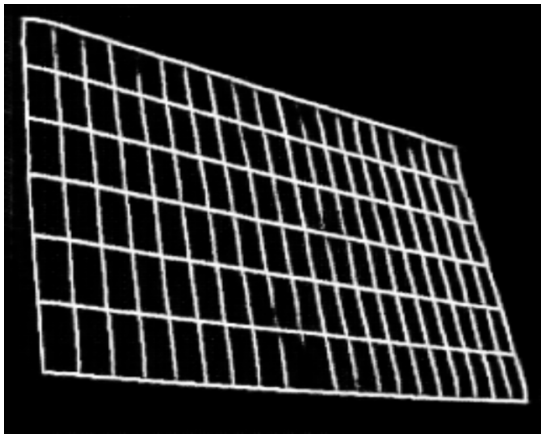

Fig. 3. An example of a faulty output of the neural network. The method is applied to an out-of-sample image of a half-cell PV module

The output of an I2INN is prone to small misidentification especially applied to new module types or modules with severe defects. For example, some grid lines may not be detected, or several cells may be merged to one area into the grid image (see Figure 3). Therefore, postprocessing is required to improve the robustness of the method.

C. Graph geometric analysis (GGA)

This paper utilizes the geometrical properties of the PV module structure to discover any potential misidentification from the neural network. Geometrically speaking, we expect a binary grid to consist of a grid of distorted rectangles, where each inner rectangle has precisely eight neighbors sharing a vertex, every rectangle on the module border has five neighbors, and every corner cell has three neighbors.

To analyze the structure of the binary grid, we focus on its graph-theoretic properties. To this end, we first apply the image thinning operator. Then, the resulting image can be treated as a pixel graph P, where nodes of the graph are non-zero pixels, and there is an edge between nodes if two pixels are neighbors. We compute pixel graph using the `pixel_graph` routine in skimage [7].

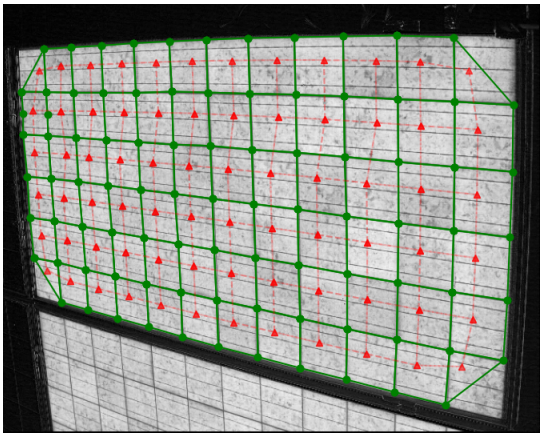

Fig. 4. An EL image with graph S (green nodes and edges) and graph $S^\star$ (red colour triangles with dashed edges)

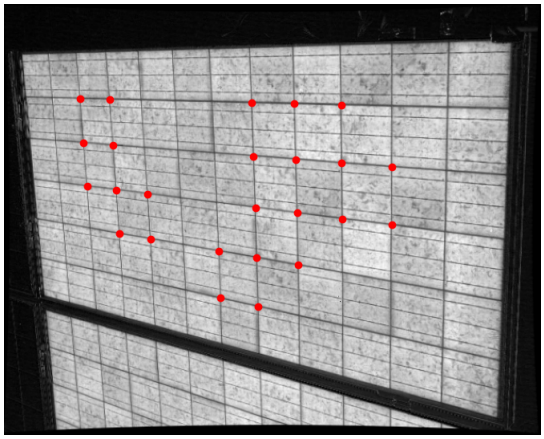

Fig. 5. Best points (red) are used to either the estimate camera matrix or correct homography (need at least 4 points)

The number of nodes in graph P can be further reduced, as most nodes have degree two with both edges oriented in the same direction. By applying a version of the breath-first-search algorithm, we obtain a weighted graph S, where every node has a degree of at least 3, and the weight of each edge corresponds to the length of the path connecting two nodes in the original graph P. We remark that the nodes of graph S are not a subset of graph P, as we average the coordinates of closely located nodes in P with a degree of at least 3. Such averaging guarantees that all our edges do not intersect one another, and graph S remains planar (with pixel coordinates being an embedding).

Further, since graph S is planar, the dual graph $S^\star$ is well-defined. Recall that the dual graph's nodes are the faces of the original graph and an edge in $S^\star$ connects two faces if they have a common edge in S. The dual graph is beneficial to us as it captures all the necessary geometric structures about the grid. Figure 4 depicts a module image and the corresponding graph S (green dots connected with lines) and dual graph $S^\star$ (red triangles connected with dashed lines).

We use a version of the depth-first-search algorithm to compute the dual graph $S^\star$. During the computation, we calculate various geometric quantities, such as the area of each face, circumference, minimal enclosing parallelogram, its edge ratio and area. We compute minimal enclosing parallelograms using the algorithm from [8].

The depth-first search algorithm identifies a face in graph S by walking around this face in a counter-clockwise direction. Apart from individual cell faces, there are outer faces when we walk around the module boundary in the clockwise direction. We define the area of the outer face to be a negative value, which equals to the module's area. We select the largest component when multiple modules are in a single image.

To avoid problems with the I2INN output inaccuracies (such as shown in Figure 3), we introduce a ranking of the nodes of the dual graph. This ranking aims to select the best faces with respect to some heuristic rule. We use geometric characteristics of the graph S faces and compare those values with the values of its neighbors. Our heuristic is based on the fact that in the PV module all cells are identical and similar in a distorted image.

To this end, every face in graph $S^\star$ comes with a set of qualitative characteristics $v_i \in \mathbb{R}^d$. Then, for each face $i \in S^\star$, we calculate the Mahalanobis distance d_M to the neighboring faces $N(i) \subset S^\star$ sharing a vertex with i:

$$w_i \coloneqq \{d_M(v_i, v_j)\}_{j \in N(i)}, \quad i \in S^\star,$$

where

$$d_M(x, y) \coloneqq \sqrt{(x-y)^T \Sigma^{-1} (x-y)}.$$

and the covariance matrix Σ is estimated using a set of training images.

In our current implementation, our vector v_i is three-dimensional and consists of the area of the face (normalized by the dimension of the image), the area ratio between the face and the area of the minimal enclosing parallelogram, and the ratio of the sides of the parallelogram.

A q-quantile, $w_i^{(q)} \in \mathbb{R}, i \in S^\star$ ($q > 50\%$), is computed on a sample of those distances. This quantile demonstrates how much face i differs from *most* of its neighbors. Note, here we consider two faces are neighbors when they share at least one node (i.e., each face inside a module has eight neighbors). We expect $w_i^{(q)}$ to be small if most neighbors are close in its characteristic to the face $i \in S^\star$. The ranking of faces is then performed according to the ordering of $w_i^{(q)}, i \in S^\star$.

We select the first half of the ordered list of faces. Then, the selected faces are reordered so that the first face has the highest rank; the second face is furthest away from the first, and so on. Such a strategy allows the distribution of the selected points throughout the image, improving homography estimation. Figure 5 demonstrates an example of selected faces.

Additionally, the dual graph determines the module *square lattice coordinates*. For that, we choose any node $i \in S^\star$ with degree 4 and assign to this face a square lattice coordinate $(0, 0)$. The nodes below and above the initial face

get coordinates $(0, -1)$ and $(0, 1)$, respectively, nodes on the left and the right get coordinates $(-1, 0)$ and $(1, 0)$. Walking over the complete grid allows us to determine a square lattice coordinate for each node in $S^\star$ with degree 4. Furthermore, the square lattice coordinates' range determines the detected module's size.

D. Subpixel accuracy correction (SPA)

The coordinate of the discovered nodes is accurate up to a few pixels. Improving the accuracy is beneficial for the camera distortion matrix and homography estimation. Therefore in the last step, we apply a subpixel-accuracy coordinate correction method, a procedure similar to the one used in [3].

To this end, for each node, we sample the original EL image in the direction of the edge. The image is then reduced to a vector, where each coordinate equals the sum of a column in the sampled image. Figure 6 shows the resulting vector with a red line. Since the image is sampled towards the direction of the graph S edge, the resulting vector is expected to contain a single peak corresponding to the cell border in the module. The x-axis is given relative to the original node coordinate.

To estimate the location of the shift, we use the following parametric model:

$$m(x; A, B, \mu, \sigma) = A + B \exp\left(\frac{(x-\mu)^2}{\sigma^2}\right). \qquad (1)$$

where parameter μ describes the desired shift value.

Unlike the subpixel-accuracy routine in [3], model (1) is a shifted Gaussian bell; hence fast Gaussian model fits like [9] are inapplicable in our scenario. Therefore, we rely on a generic curve-fitting method based on the trust-region optimization method [10]. We use the implementation from `curve_fit` in the scipy library [11]. The blue line in Figure 6 shows an example of the model fit.

Among the parameters of the model (1), the σ and its variance of the estimator can be utilized to identify problematic fits. Namely, we ignore any nodes with $\sigma^2 > 0$ and $\mathrm{Var}(\hat{\sigma}^2) > 10$. Such thresholding allows ignoring curves where peaks are not prominent or, in the scenario when a dark cell makes the bell asymmetric.

Optimization is performed twice for each node: in vertical and horizontal directions. The resulting parameters μ and the direction of the edges define the correction shift vector, with which the node's coordinate is adjusted.

E. Camera and homography corrections

The resulting cell corner and square lattice coordinates are used to estimate the camera matrix and perform a lens distortion correction or image homography. In addition, the square lattice coordinates allow us to determine the number of rows and columns in the module and extract those cells according to the square lattice grid.

For camera calibration and homography estimation, we use routines implemented in the OpenCV library [12]. Namely, `calibrateCamera` [13] and `findHomography`. The

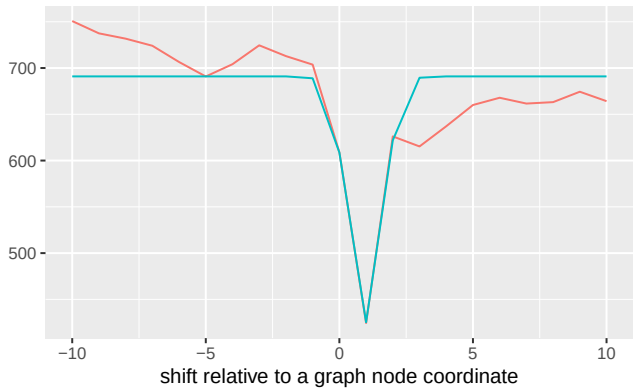

Fig. 6. Subpixel-accuracy coordinate correction method to improve point locations. Sample average pixel intensity of the original image in the direction on graph edges (red) and fit a parametric model (blue)

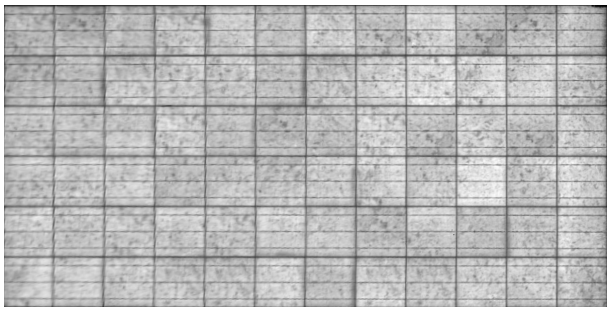

Fig. 7. Corrected image

camera matrix is estimated from identified coordinates from several images.

Lastly, since we use only a subset of good nodes for estimating homography, we can verify the detection with the remaining nodes. Namely, we apply the corresponding homography transformation and compute the root-mean-square error (RMSE) with the excepted square lattice coordinates. If the RMSE is above a chosen value, we report an error for the given image.

III. Results

Figure 7 depicts the desired corrected image for the module in Figure 2. Here we selected each cell to have a dimension of 100×100 pixels as those modules have square cells. The resulting image dimension equals 1200×600. We can easily extract individual cell images by shifting a window within an image.

Several factors contribute to the robustness of our method. Firstly, the ability for neural networks to generalize to new data, our I2INN step can be applied to new types of modules. Furthermore, graph-based geometric property analysis allows filtering potential misdetections in the grid images. Again, our approach can handle images with multiple modules, as all our routines are applied for each component. Moreover, the thresholding of covariances of the estimator in the SPA routines allows us to ignore nodes where cells are defective.

TABLE I
I2INN: TIME REQUIRED DEPENDS ON THE DEVICE

device	time
gpu:jetson	70 ms
cpu:jetson	400 ms
gpu:nvidia gv 102	5 ms
cpu:intel xeon w-2123	70 ms

TABLE II
OTHER STEP: RUNS ON 1-CPU PER IMAGE, SUBPIXEL CORRECTION PER NODE (MEASURED ON INTEL XEON W-2123). IO STANDS FOR INPUT/OUTPUT IMAGE OPERATION

procedure	time
IO	60 ms
GGA	89 ms
SPA	7 ms
lens correction	5 ms
homography estimation	4 ms

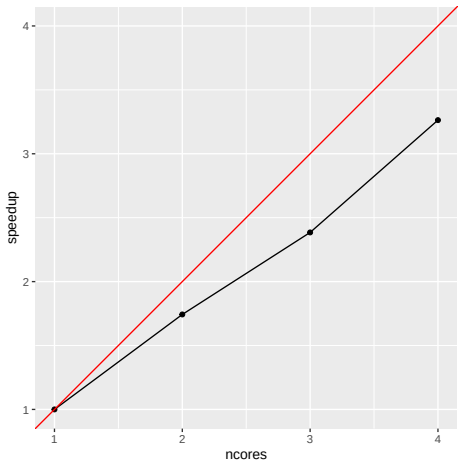

Fig. 8. Comparison of pipeline speedup with the theoretical one (red line). Benchmark using Intel Xeon W-2123

Lastly, even after all those steps, the module is not accurately detected; computation RMSE with all the nodes allows for identifying such failures.

We benchmark our pipeline on several devices to demonstrate our approach's computational performance. First, we use the NVIDIA Jetson machine, a lightweight computer with a GPU module for potential in-field applications. Further, we use the Intel Xeon W-2123-based machine for large-scale processing with the NVIDIA GV 102 graphic card. For all results, we process images in maximum batches, device memory permits.

Table I compares the performance of the I2INN step for different devices. Here, we have an option of running the routine on CPU or GPU devices for each of the machines. On the Jetson, we used batches of size 16; on a larger machine, we used batches of size 256.

Table II shows the running time of the GGA step. This step has no parallel implementation; hence each image is processed on a single CPU. Here the subpixel correction routine timing is given per node. The homography requires calling routine at least four times, whereas, for the camera distortion matrix computation, we need 20–30 nodes.

The complete pipeline can be computed on multiple CPUs for processing many images in batches. Figure 8 demonstrates how our parallel implementation of batch processing deviates from the ideal linear scaling measured on the Intel Xeon W-2123 machine. Table III provides timings for the complete pipeline on small and large machines. The memory requirement of the pipeline depends on the batch size. For instance, a batch size 256 requires up to 5.6 GB of memory on an Intel Xeon W-2123 machine.

TABLE III
COMPLETE PIPELINE USING 4 CPU

machine	time/image
intel xeon + gpu	125 ms/image
nvidia jetson	527 ms/image

These performance benchmark values demonstrate a significant improvement upon the previously published model, for which the running time ranges from 6 s [4] to 360 s [3, 2] per image. The performance of our method can be further improved by implementing graph algorithms in a fast-compiled language (the current implementation is in Python), and the input/output can be further optimized. However, we expect a running time to be at most 60 ms per image.

To estimate the accuracy of the proposed approach, we compute the RMSE values for the camera calibration and homography correction routines. First, we estimate the camera distortion matrix with the iterative aruco-based calibration method suitable for low-resolution images [14]. Then we calculate the camera distortion matrix with our approach based on the 320 EL images. Then we project every pixel coordinate of the image using those two camera matrices and compute an RMSE of 2.9 pixels between the two projections.

For homography accuracy, we compare the cell coordinates between our method and the results of the method published in [4]. Based on the data from 1500 images, we obtained an RMSE of 3.2 pixels. We remark that the method in [4] does not perform subpixel-accuracy coordinate correction, hence neither of the methods can be considered as ground truth. Figure 9 shows that visually, the SPA routine gives a more accurate position of the cell coordinate (red points) compared to the green points obtained from [4].

We remark that the proposed approach is not restricted to EL images, for example, can be applied to drone-based infrared images or EL for complete module identification. Figure 10 depicts individual module identification in low-resolution IR images, where the camera distortion matrix is estimated from a sample of 13 images.

IV. SUMMARY

Image data volumes will grow with the emergence of drone-based automatic PV plant inspection. Therefore, automatizing module inspection and defect detection require fast and reliable image processing algorithms.

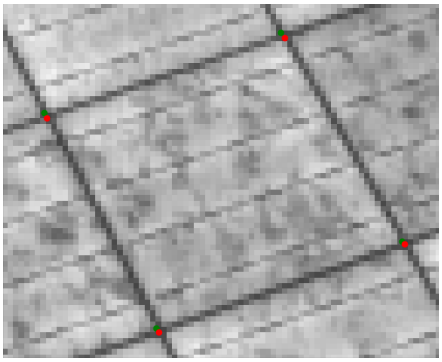

Fig. 9. Comparison of cell corners between [4] (green points) and our method (red points). The SPA in our approach visually improves the node estimation

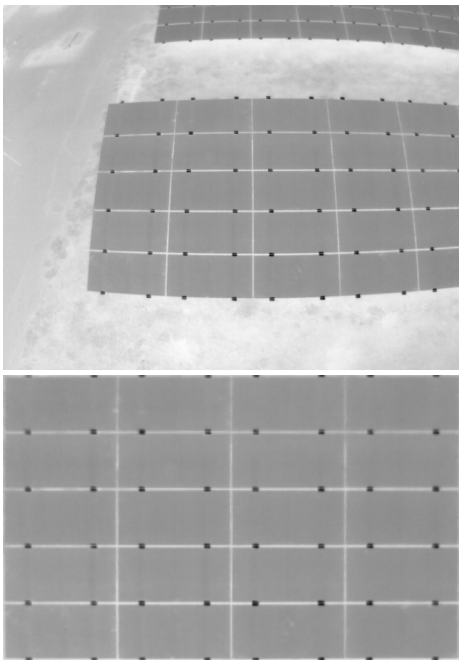

Fig. 10. The approach can be applied to IR/EL drone-based images. Image source: Aerial PV Inspection GmbH

This paper proposed a method for camera and homography distortion corrections and cell extraction in EL images. By leveraging the computational speed of neural networks and the geometric properties of a PV module, we improved the running times of such preprocessing algorithms by a factor of 10. Generally, our approach applies to any images containing regular patterns.

Acknowledgment

This work is supported by the "PARK3" project (Förderkennzeichen: 03EE1104) and "ReliaREN-Pro" project (Förderkennzeichen: 03EI4052B). The authors would like to thank Solarzentrum Stuttgart GmbH for providing the data.

References

[1] IEA. *Net Zero by 2050*. 2021. URL: https://www.iea.org/reports/net-zero-by-2050.

[2] E. Sovetkin and A. Steland. "Automatic processing and solar cell detection in photovoltaic electroluminescence images". In: *Integrated Computer-Aided Engineering* 26.2 (2019), pp. 123–137. DOI: 10.3233/ICA-180588.

[3] S. Deitsch et al. "Segmentation of photovoltaic module cells in uncalibrated electroluminescence images". In: *Machine Vision and Applications* 32.4 (2021), p. 84. DOI: 10.1007/s00138-021-01191-9.

[4] P. Kölblin, A. Bartler, and M. Füller. "Image Preprocessing for Outdoor Luminescence Inspection of Large Photovoltaic Parks". In: *Energies* 14.9 (2021), p. 2508. DOI: 10.3390/en14092508.

[5] L. Stoicescu, M. Reuter, and J. Werner. "DaySy: luminescence imaging of PV modules in daylight". In: *29th European Photovoltaic Solar Energy Conference and Exhibition*. 2014, pp. 2553–2554. DOI: 10.4229/EUPVSEC20142014-5DO.16.2.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1125–1134.

[7] S. van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. DOI: 10.7717/peerj.453.

[8] C. Schwarz, J. Teich, E. Welzl, and B. Evans. "On finding a minimal enclosing parallelogram". In: *International Computer Science Institute, Berkeley, CA, Tech. Rep. tr-94-036* (1994).

[9] E. Pastuchová and M. Zákopčan. "Comparison of algorithms for fitting a gaussian function used in testing smart sensors". In: *Journal of Electrical Engineering* 66.3 (2015), pp. 178–181. DOI: 10.2478/jee-2015-0029.

[10] M. A. Branch, T. F. Coleman, and Y. Li. "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems". In: *SIAM Journal on Scientific Computing* 21.1 (1999), pp. 1–23. DOI: 10.1137/S1064827595289108.

[11] P. Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[12] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015.

[13] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[14] S. Schramm, J. Ebert, J. Rangel, R. Schmoll, and A. Kroll. "Iterative feature detection of a coded checkerboard target for the geometric calibration of infrared cameras". In: *Journal of Sensors and Sensor Systems* 10.2 (2021), pp. 207–218. DOI: 10.5194/jsss-10-207-2021.