Machine learning for solar trackers

Cite as: AIP Conference Proceedings **2126**, 030012 (2019); https://doi.org/10.1063/1.5117524
Published Online: 26 July 2019

Jose A. Carballo, Javier Bonilla, Manuel Berenguel, et al.

Machine Learning for Solar Trackers

Jose A. Carballo [2, 3, b)], Javier Bonilla [1, 3, a)], Manuel Berenguel [2, 3, c)], Jesús Fernández-Reche, [1, d)] Ginés García [1, e)]

[1]*CIEMAT - Plataforma Solar de Almería, Ctra. de Senés s/n, 04200 Almería, Spain*
[2]*University of Almería, Ctra. Sacramento s/n, 04120 Almería, Spain,*
3CIESOL, Solar Energy Research Center, Joint Institute University of Almería - CIEMAT, Almería, Spain

a) Corresponding author: javier.bonilla@psa.es
b) jcarballo@psa.es c) beren@ual.es d) jesus.fernandez@psa.es e) gines.garcia@psa.es

Abstract. A new approach for solar tracking, based on deep learning techniques, is being studied and tested using Tensorflow, an open source machine learning framework. Tensorflow makes the implementation more flexible and increases the development capabilities. Tensorflow facilitates the neural network implementation on several kinds of devices (embedded and mobile devices, mini computers, etc.). Furthermore, Tensorflow supports different types of neural networks which can be tuned and retrained for particular purposes. The presented results are promising, since the retrained networks correctly identify the Sun and the target, with this information the system can be controlled to properly track the Sun's apparent trajectory without further information.

INTRODUCTION

In CSP technologies, the Sun Tracking Subsystem (STS) is one of the keystones to improve the performance of the whole system and reduce costs. Numerous researchers are attempting to change the future of STSs as revealed by the high number of publications related to this topic, especially in recent years [1-4]. Nowadays, several types of STSs can be found in the literature. They are commonly specific to the type of solar technology they were developed for and present some limitations regarding to the cost, operation or error correction.

With the main goal of mitigating the present limitations of current STSs, a new control approach of STSs based on computer vision can be carried out by detecting the Sun and the tower target area by means of a camera place at 0' and whose optical axis is arranged parallel to the optical axis of the collector (V_A). Once the elements have been identified, it is computed the midpoint (A') between S' and T' of the solar (V_s) and target (V_t) vectors intersections with the plane of an image (PC) taken by a camera aligned with the optical axis of the concentrator (V_A) which also intersect PC in A''. The differences between A' and A'' is called tracking error and it is employed as input for the STS controller. This approach can be used in any solar technology regardless of the type of solar tracker, as shown in "**FIGURE 1"**.

SolarPACES 2018
AIP Conf. Proc. 2126, 030012-1–030012-8; https://doi.org/10.1063/1.5117524
Published by AIP Publishing. 978-0-7354-1866-0/$30.00

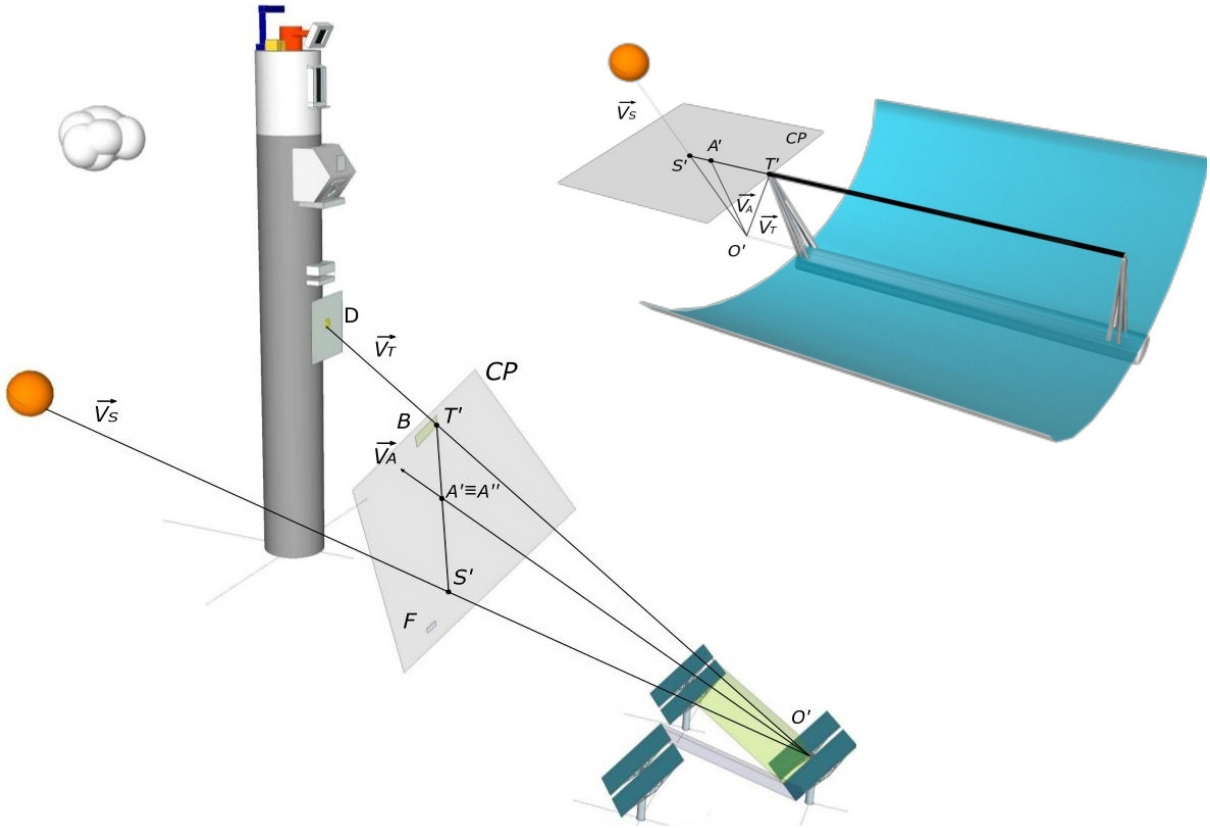

FIGURE 1. Central Receiver System (CRS) and Parabolic Trough-Collector (PTC) solar tracking

A small mockup heliostat, whose STS was based on traditional computer vision techniques, was successfully tested at Plataforma Solar de Almería [5]. The computer vision approach was improved by using low-cost open hardware and machine learning techniques (deep learning) [6] implemented using proprietary tools (Matlab [7] and Mathematica [8]). This last approach was tested at Plataforma Solar de Almería (PSA) and revealed the great potential of smart STSs, also the authors of the present work have patented the new approach for Sun tracking systems [9]. Preliminary results match with the main conclusion of an extended review by Al-Rousan [2], which indicates that the use of smart STSs is the most promising development path for concentrated solar power and photovoltaic technologies.

With the main goal of making the computer vision implementation more flexible, the approach for STSs based on machine learning was implemented using an open source machine learning framework, Tensorflow [10] developed by Google. Tensorflow facilitates on-device machine learning implementation on mobile and low-end embedded devices. Tensorflow also adds more flexibility, since different kinds of neural network models can be design, implemented and used. This work presents the preliminary results obtained with the latest approach.

NEURAL NETWORK MODELS

There are several pretrained models available in Tensorflow [11]. A pretrained model is a good starting point for creating new neural network models, since such models already learned a rich set of general features that will be useful for new networks. This approach also reduces the time required for the training process.

Several pretrained models were considered in this work, see **"TABLE 1"**. They were modified and tuned to suit our needs. That is to identify the Sun, target, heliostats and clouds. Guidelines on how to tune these models are given in [12].

TABLE 1. Neural network models

MODEL	DESCRIPTION	Reference
SSD MobileNet V1 optimized	Depth- wise separable convolutions to build light weight deep neural networks	[13]
SSD MobileNet V1 0.75 depth	Lighter SSD MobileNet V1	[13]
SSD MobileNet V1 quantized	Quantization scheme allows that detection can be carried out using integer-only arithmetic	[14]
SSDLite Mobilenet V2	New architecture, that improves the performance of models and benchmark marks	[15]
SSD Inception V2	Inception block replaces the extra layers in SSD Mobilenet V2	[16]
Mask RCNN Inception V2	Simple, flexible, and general framework for object instance segmentation	[17]

TRAINING

A small heterogeneous training image set (350 images) of the CESA central tower system located at PSA was used. Four object classes (heliostat, target, Sun and cloud) were labeled in the image set. **"FIGURE 2"** shows some images of the training image set.

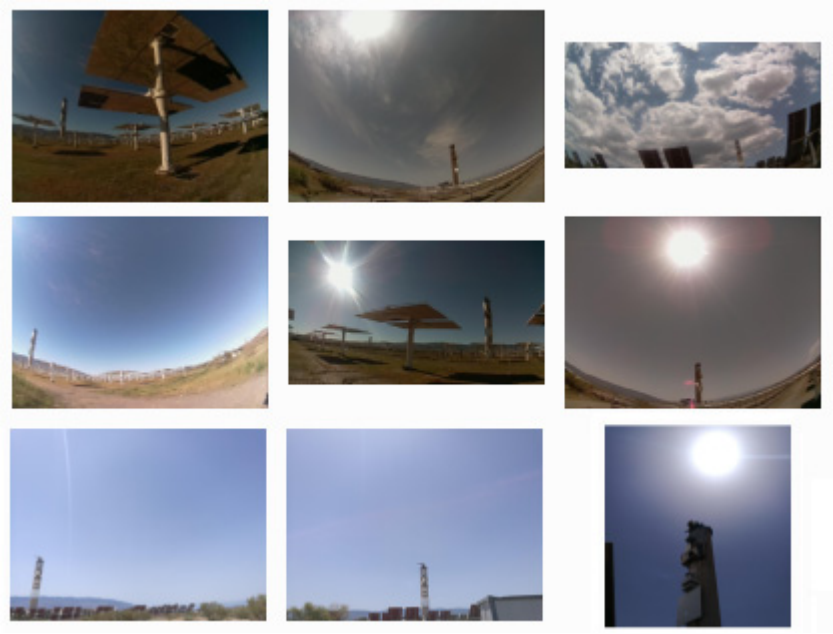

FIGURE 2. Some images from the training image set

Neural model training is a iterative process, among all the training strategies provided by Tensorflow, the RMSPropOptimizer has been employed in this study. It implements an adaptive version of Stochastic Gradient Descent called RMS Prop "Root Mean Squared Propagation. **"FIGURE 3"** shows the evolution of the neural model results, which was trained from 0 to 15000 iterations (epochs).

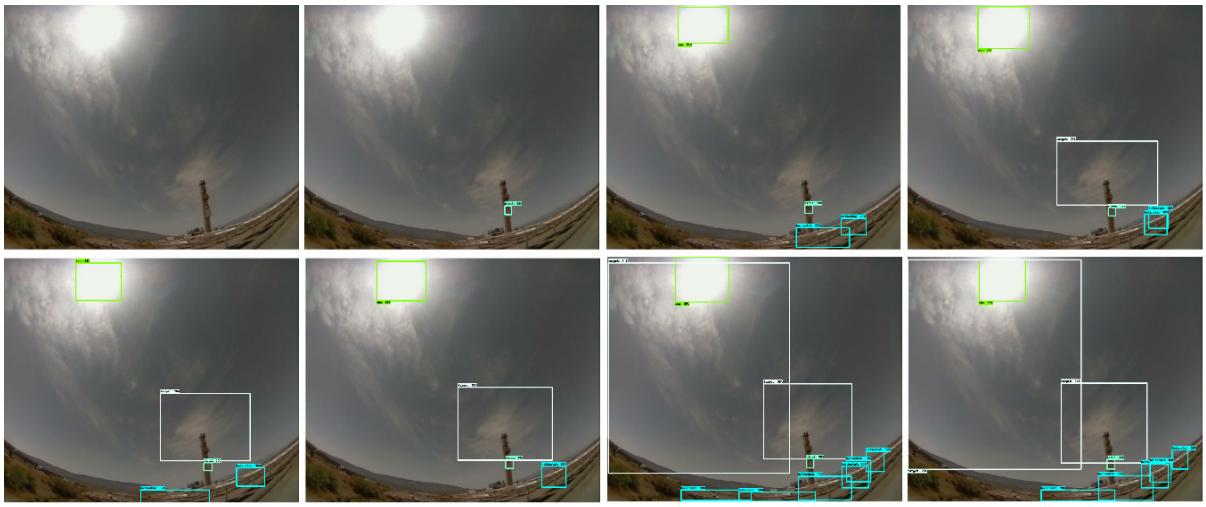

FIGURE 3. Result evolution with training

Neural network training is a computationally expensive process. In order to reduce the training time, the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT) were used. A Graphics Processing Unit (GPU) cluster with Compute Unified Device Architecture (CUDA) support was used for training the networks, see **"FIGURE 4"**. The average training time was around 24 hours.

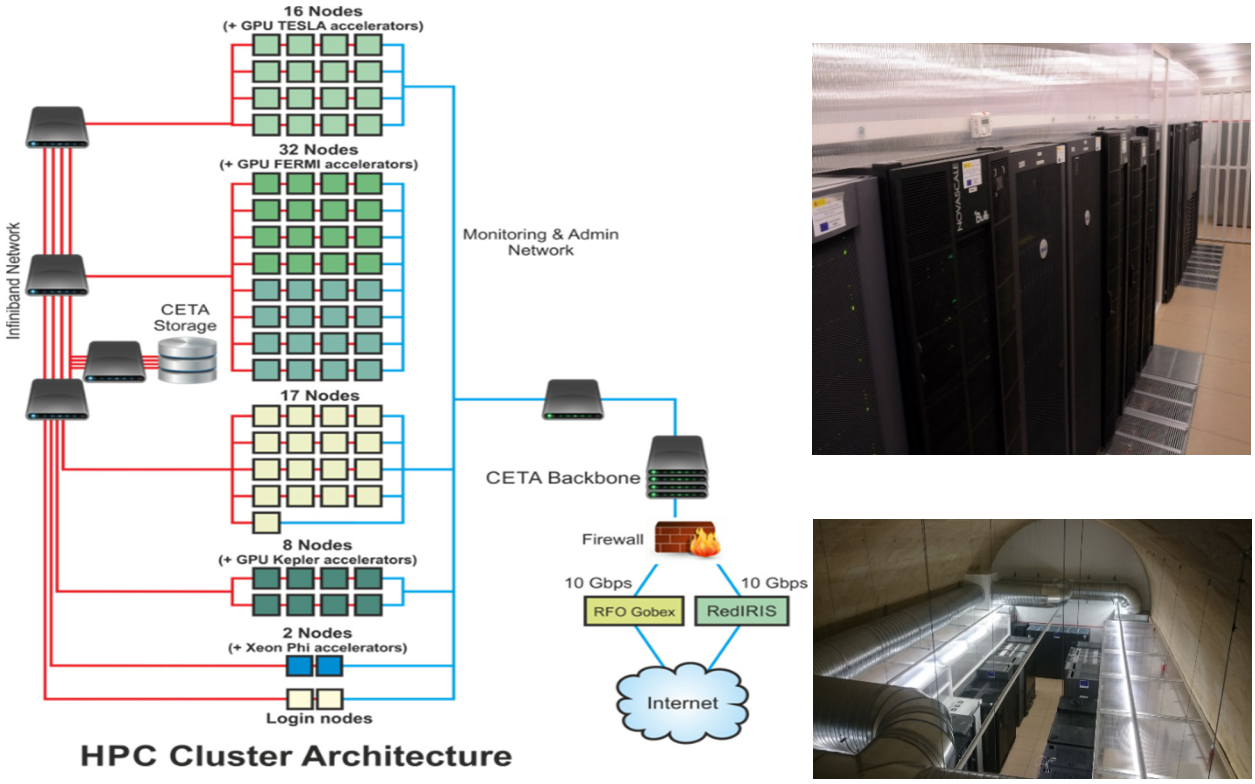

FIGURE 4. Computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT)

VALIDATION

Besides the training image set, there is an independent image set for validation (25 images). This set of images is used to validate the neural network results and provides information about the mean average precision (mAP) and inference time (speed of processing an image on CPU in this case). Mean average prediction is computed as the ratio of true object detections to the total number of objects that the classifier predicted, mAP is always calculated over a validation dataset. The predicted box is considered a true detection if a predicted box shares more than a percentage of its pixels with the ground-truth box (overlap criterion). In this work, the overlap criterion is defined as an Intersection-over-Union (IoU) greater than 0.5. Values of mAP close to 1.0 show high probability that whatever the classifier predicts as a positive detection is in fact a correct prediction. Inference time and mAP are common metrics to evaluate object detectors, for that reason they are shown for the evaluation of the neural networks shown in **"TABLE 2"**. These metrics are provided by a Tensorflow tool called Tensorboard, which provides detailed information about the neural network learning process.

TABLE 2. Neural network validation metrics.

MODEL	Speed (ms)	Precision (mAP@0.5IoU)
SSD MobileNet V1 optimized	100	0.63
SSD MobileNet V1 0.75 depth	90	0.78
SSD MobileNet V1 quantized	110	0.79
SSDLite Mobilenet V2	400	0.35
SSD Inception V2	350	0.41
Mask R-CNN Inception V2	1100	0.55

"TABLE 2" reveals that "SSD MobileNet V1 0.75 depth" is the fastest neural model, whereas "SSD MobileNet V1 quantized" is the most accurate. "SSD MobileNet V1 optimized" also show a good behavior. The rest of the models show similar results, although they are slightly slower or less precise. This fact reveals that better configuration of the model or training parameters may improve the results given by those networks.

It should be noted that "Mask R-CNN Inception V2" is not only an object detector, it is also a semantic segmentator, that is it provides object masks, not only bounding boxes. This could offer new capabilities to the smart STS approach, while achieving a good precision mark.

IMPLEMENTATION

The previously introduced smart STS approach has been mainly implemented for two kinds of devices: low-end embedded and mobile devices, although it also works in common computers. Tensorflow is the machine learning framework.

- **Embedded devices:** A Raspberry Pi 3 model B+ and Pi camera was the hardware used. Software was implemented using the open source programming language Python. This is an open-source and low-cost implementation ideal to test and evaluate this approach in real facilities, since this kind of devices can control and communicate with the solar collectors.

- **Mobile devices:** An app for iOS and Android devices was also implemented. This app is multiplatform and can be also executed in common computers (Linux, macOS and Windows operating systems). The app is programmed in the V-Play engine [18], which facilitates the app deployment to several devices and operating systems. V-Play is a Standard Developer Kit (SDK) based on the Qt framework [19]. The programming languages used were C++ for the computer logic and QML for the Graphical User Interface (GUI). This app is intended to be used for easily demonstrating the concept and as an educational resource. It might be also used for other purposes that must be studied, for instance for on-site testing and calibration.

EXAMPLES

"FIGURE 5" shows an augmented reality image taken from Raspberry Pi in the CESA solar field at PSA. The distance to the tower was more than 400 m. The objects detected are shown in colored boxes (Sun, target and heliostats). The numerical values represent the confidence levels of the detections, it is computed as the softmax probability (Neural Network output) of each of bounding boxes being on object.

As commented before, from the Sun (S_0) and target (T_0) box detections the black cross and gray circle positions can be computed. The black cross represents the solar collector aiming point (A'), whereas the gray circle represents the correct Sun tracking point A''. The Sun tracking error can be computed as the distance between the black cross and the gray circle. This error is used to compute the control signal [6] and to properly aim the solar collector to follow the Sun's apparent trajectory.

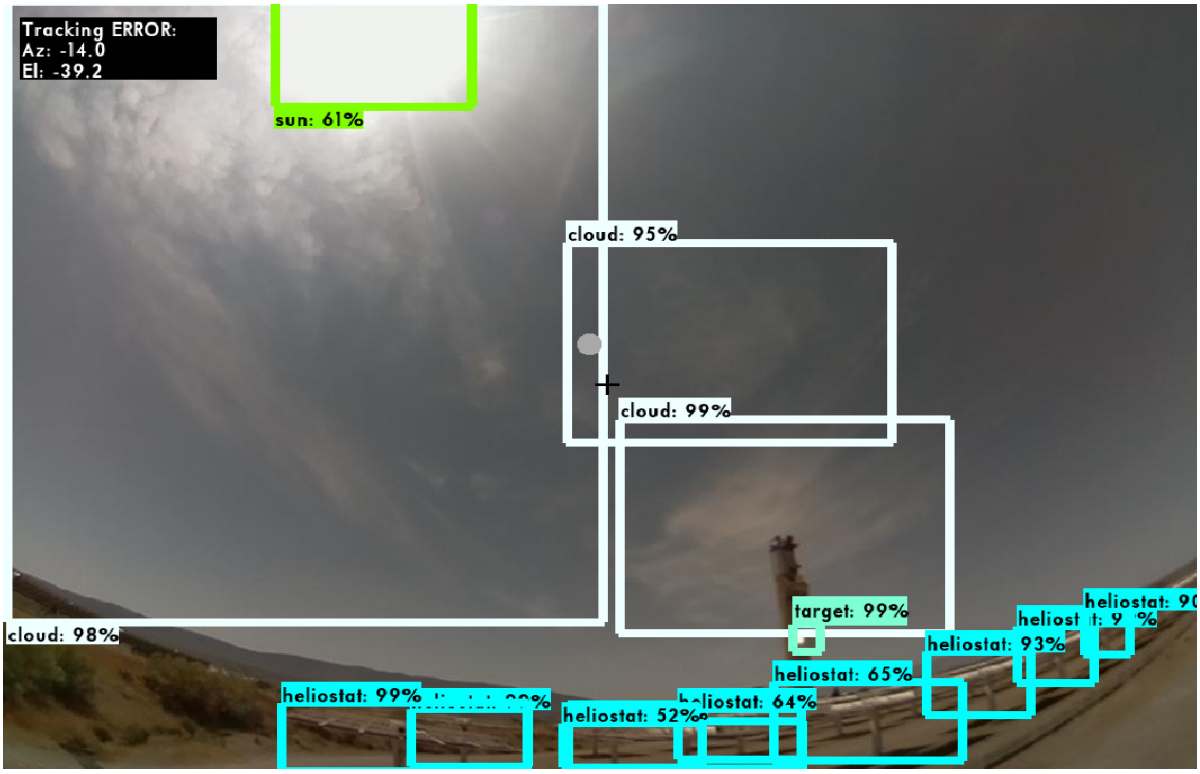

FIGURE 5. Augmented reality image taken and processed from Raspberry Pi at CESA solar field

"FIGURE 6" shows an augmented reality image taken from a mobile device in the CESA solar field at PSA. This app presents the same behavior than the developed software for embedded devices. Several objects were detected in colored boxes, the semi-transparent gray circle represents the correct Sun tracking point, its size matches the target size. The red dot represents the current heliostat aiming point (center of the screen), therefore the red dot must be placed inside the gray circle to reflect the Sun irradiance onto the target.

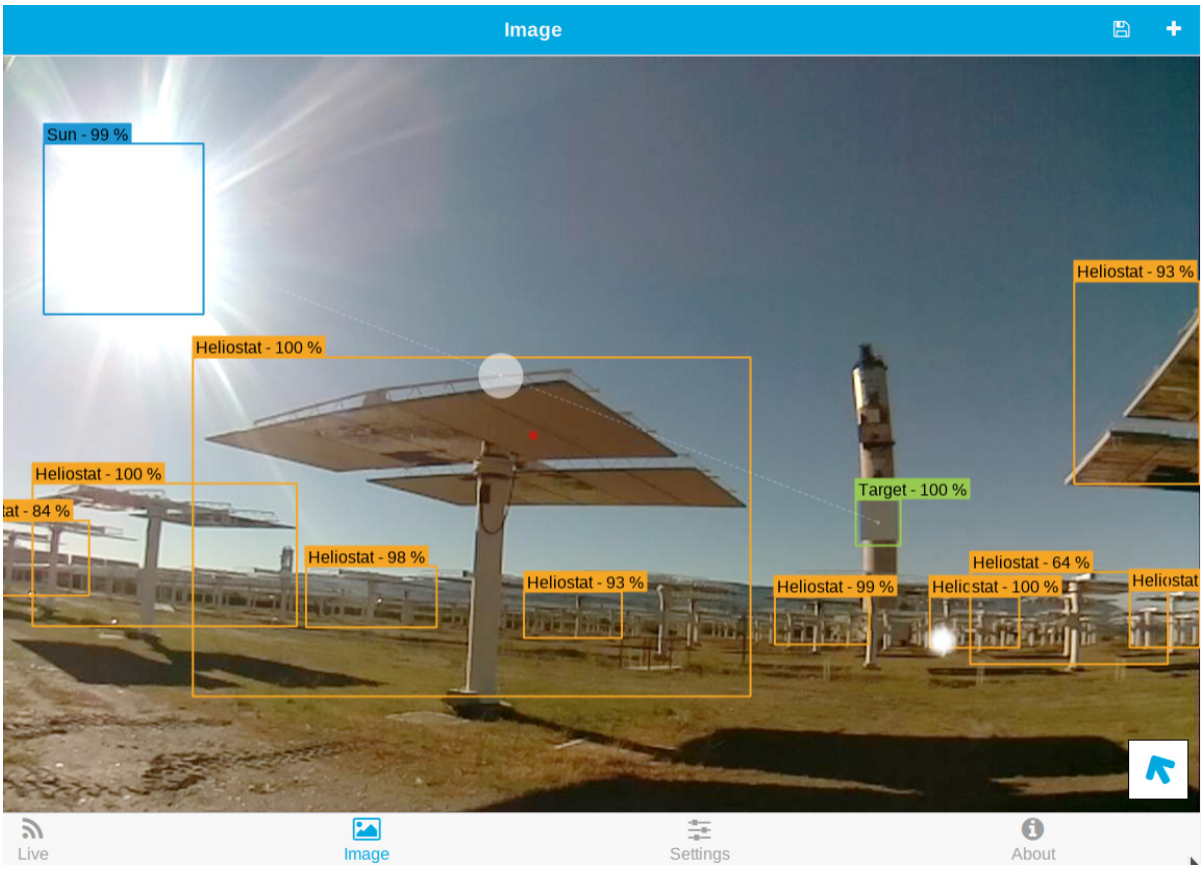

FIGURE 6. Augmented reality image processed on a mobile device at CESA solar field

CONCLUSSION AND FUTURE WORK

The new machine learning framework is faster and more accurate than the original implementation [6] according to results from **"TABLE 2"**. The rapid growth of Tensorflow and the large number of contributions from the open community augurs a substantial improvement of the marks obtained in a short time.

"SSD MobileNet V1 optimized", "SSD MobileNet V1 quantized" and "SSD MobileNet V1 0.75 depth" show the best behavior in terms of speed and precision of all the evaluated models.

The new framework is open and more flexible, it allows implementing the smart STS approach in almost every kind of device. Embedded and mobile implementations have been carried out in this work.

The good results obtained in terms of speed and precision, make it possible to use this approach to design new control strategies based on artificial intelligence and computer vision that can take into account key factors for solar collector such as block, shadow and cloud detections.

Ongoing work includes training and validating the neural networks with larger image sets, evaluate new neural networks and further optimize the approach implementations.

Future work includes the autonomous control of a heliostat using the embedded device implementation in order to estimate the control accuracy in terms of mrad, as well as determine the optimal image resolution for minimizing such error.

ACKNOWLEDGMENTS

This work has been funded by the National R+D+i Plan DPI2014-56364-C2-2-R Spanish Ministry of Economy, Industry and Competitiveness funds. This work was also partially supported by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT belongs to CIEMAT and the Government of Spain.

REFERENCES

1. Hafez, A. Z., Yousef, A. M., & Harag, N. M. (2018). Solar tracking systems: Technologies and trackers drive types – A review. Renewable and Sustainable Energy Reviews, 91(November 2017), 754–782. https://doi.org/10.1016/j.rser.2018.03.094
2. AL-Rousan, N., Isa, N. A. M., & Desa, M. K. M. (2018). Advances in solar photovoltaic tracking systems: A review. Renewable and Sustainable Energy Reviews, 82(September), 2548–2569. https://doi.org/10.1016/j.rser.2017.09.077
3. Nsengiyumva, W., Chen, S. G., Hu, L., & Chen, X. (2018). Recent advancements and challenges in Solar Tracking Systems (STS): A review. Renewable and Sustainable Energy Reviews, 81(August 2017), 250–279. https://doi.org/10.1016/j.rser.2017.06.085
4. Camacho E.F., Berenguel M., Rubio F. R., Martínez D., Control issues in solar systems, in: Control of Solar Energy Systems, Springer, 2012, pp. 25–47.
5. Carballo, J. A., Bonilla, J., Roca, L., Berenguel, M. New low-cost solar tracking system based on open source hardware for educational purposes. Under Review.
6. Carballo, J. A., Bonilla, J., Berenguel, M., Fernández-Reche, J., García. G. New approach for solar tracking systems based on computer vision, low cost hardware and deep learning. Under Review.
7. The MathWorks Inc. (2018) MATLAB R2018a, http://www.mathworks.es/products/matlab
8. Wolfram (2018). Mathematica 11, http://www.wolfram.com/mathematica
9. J. A. Carballo, J. Bonilla, J. Fernandez, G. Garcia, (2018). Sistema de captación solar mediante técnicas de visión artificial ES P2018300773.
10. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Zheng, X. (2016). Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. https://doi.org/10.1038/nn.3331
11. Google (2018). Tensorflow detection model zoo, https://gitub.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md
12. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017–January, 3296–3305. https://doi.org/10.1109/CVPR.2017.351
13. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T. Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. https://doi.org/arXiv:1704.04861
14. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A. Kalenichenko, D. (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. https://doi.org/10.1109/CVPR.2018.00286
15. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. https://doi.org/10.1134/S0001434607010294
16. Ning, C., Zhou, H., Song, Y., & Tang, J. (2017). Inception Single Shot MultiBox Detector for object detection. 2017 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2017, (July), 549–554. https://doi.org/10.1109/ICMEW.2017.8026312
17. He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision, 2017–Octob, 2980–2988. https://doi.org/10.1109/ICCV.2017.322
18. V-Play GmbH (2018). V-Play Engine, http://www.v-play.net
19. The Qt Company (2018). Qt – Cross-platfrom software development for embedded & desktop, https://www.qt.io