Semi-Automatic Detection and Tracking of Growing Mushrooms on Image Sequences

Gyula Simon
Alba Regia Technical Faculty
University of Óbuda
Székesfehérvár, Hungary
simon.gyula@amk.uni-obuda.hu

Gergely Vakulya
Alba Regia Technical Faculty
University of Óbuda
Székesfehérvár, Hungary
vakulya.gergely@amk.uni-obuda.hu

Sándor Tarsoly
Antal Bejczy Center for Intelligent Robotics
University of Óbuda
Budapest, Hungary
sandor.tarsoly@irob.uni-obuda.hu

Péter Galambos
Antal Bejczy Center for Intelligent Robotics
University of Óbuda
Budapest, Hungary
peter.galambos@irob.uni-obuda.hu

***Abstract*—The efficient management of autonomous mushroom production plants requires the model of growth rate of mushrooms. Photos of the plants are used as input for the growth models, which then predict the development of individual mushrooms. Recently machine learning techniques have been successfully applied to create such models. For the machine learning systems, however, large number of training samples are required. The training samples include photos of the plant and also ground truth markers indicating the position and size of the mushrooms on the photo. In this paper an image processing system is introduced, which is able to create good quality ground truth from sequences of images of the plant. The proposed system can automatically detect the mushroom positions and sizes on each of the pictures, but also allows user intervention to minimize the number of detection errors.**

Keywords—image processing, object detection, computer vision, mushroom cultivation

I. Introduction

In precision agriculture machine learning techniques are widely used to improve crop quality, e.g. using weather prediction, providing pest prevention, or maintaining optimal conditions in greenhouses [1]-[8]. Machine learning techniques have been successful to forecast various events based on time series, e.g. in weather and climate modelling [9], business [10], finance [11], and lately crop growth modelling [5]-[8] as well.

In mushroom production, technological greenhouses are used, where optimal parameters can be precisely adjusted [5]. Recently machine learning systems have been proposed to detect and measure mushrooms, and estimate their growth rate [5], [6]. To allow the training of machine learning systems, however, a large number of training samples are required, e.g. in the form of a series of photos showing the evaluation of the crop, with the synchronized timeline of various measured environmental parameters possibly affecting the growth rate (e.g. temperature or humidity). The training samples must include the ground truth, e.g. for the photos the mushrooms must be identified for the learning system, e.g. in the form of parameters center and radius. The generation of ground truth is burdensome, especially for large number of samples. In this paper an image processing system is proposed, which aids the creation of good quality ground truth data from sequences of images of the plant. The proposed system models mushroom as a circle, and can automatically detect the position and radius of each mushroom on each of the pictures. The system also allows user intervention to minimize the number of detection errors.

The outline of the paper is the following: Section II will provide the outline of the system and explains the mode of operation. In Section III the main components will be introduced. Section IV illustrates the operation of the system. Section V concludes the paper.

II. System Operation

The input of the system is a series of photos taken on the crop, as illustrated in Fig. 1. The photos are taken so that the growth of the mushrooms is observable, but the difference is not too large. In practice one photo in each hour gives good results. The operation of the system is based on the following trivial observation: large mushrooms are easier to detect than small ones. Thus, the processing of images is started from the last one, where the mushrooms are the largest. In the first phase all mushrooms with considerable size are detected on the latest image. In the second operation phase, stepping backwards, the mushrooms are tracked on each image. In the second phase information, found in the previous image, is used to locate the mushrooms.

The concept is illustrated in Fig. 1, where the input images are denoted by $I_1, I_2, \dots, I_N$. In phase #1 the last image I_N is processed. Notice that in this phase the only input is the image. The output of the processing is a vector $\boldsymbol{P}_N$ of estimated circle positions, where each position includes the center and radius of one circle, corresponding to a mushroom. In Fig. 1, four mushrooms are detected, the detections are shown as red circles.

In phase #2, the remaining of the images are processed in reverse order, i.e. $I_{N-1}, I_{N-2}, \dots, I_1$. The result of each step is a vector of positions, similarly to phase #1. When image I_n is processed, the output is $\boldsymbol{P}_n$, and the input also includes the result vector $\boldsymbol{P}_{n+1}$ of the previous processing step.

Notices:

- As Fig. 1 illustrates, the processing is performed in reverse order.
- The lengths of the result vectors are the same, determined in phase #1. If the size of a mushroom becomes too small to detect, its position is still maintained in the next processing steps.

III. System Components

In this Section the image processing tools will be introduced.

A. Image Preprocessing

The input of phase #1 is the last image I_N. In the preprocessing step the image is converted to a grayscale image (I_{GRAY}). To highlight the edges on the image, the image gradient is computed (I_{GRAD}). Finally, the gradient image is binarized (I_{BIN}). The steps of the preprocessing are illustrated in Fig. 2.

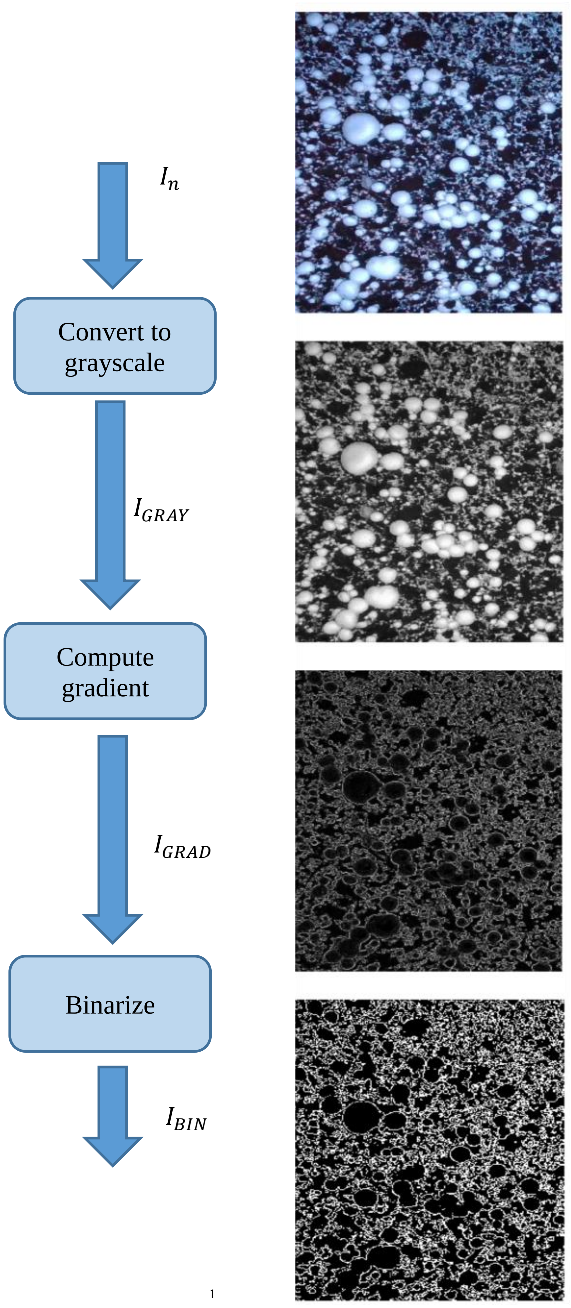

Fig. 2. The image preprocessing steps

B. Phase One

In phase #1 circles are searched for in the binary image I_{BIN}. For circle finding, the classical method is the Circle-Hough-transformation (CHT) [12], which is able to find circles with a given radius on binary images. Extensions of the Hough-transform allow the search of various shapes [13]. Also, circles with different radii can be searched, with iterative usage of the CHT. The Phase Coding Method (PCM) uses a scale-invariant kernel operator, thus it provides more efficient implementation in case the radius is not known [14]. The MATLAB Image Processing Toolbox contains implementations for both CHT and PCM [15]. The performance of the methods is illustrated in Fig. 3.

The input of the circle search was the binary image I_{BIN}, but the figure also contains the grayscale image I_{GRAY} to ease visual evaluation. Both methods have a sensitivity parameter, where higher values close to 1 allow detection of more circles. The examples shown in Fig. 3 allow the following conclusions:

- Small sensitivity values provide fewer false detections but also may miss real targets.
- Large sensitivity values provide a great amount of false detections but the chance of mission a target is smaller.

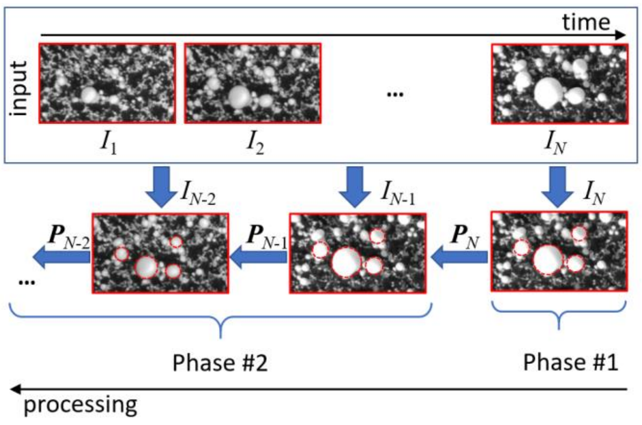

Fig. 1. The concept of image processing

Since the general circle search algorithms cannot provide robust detection by themselves, they are used to provide initial estimates, which are further evaluated. The sensitivity *S* is set high enough to provide a rich set of detections, with potentially no misses. The initial estimates filtered and pruned as follows:

The quality of the circle edge is measured by three quantities Q_{per}, Q_{plate}, and Q_{color}. The perimeter quality index Q_{per} of a circle with center *C* and radius *R* is calculated as follows:

$$Q_{per} = \frac{N_{white-per}}{2R\pi}, \quad (1)$$

where $N_{white-per}$ is the number of white pixels in I_{BIN} along the circle drawn at center *C* and radius *R*. Obviously, Q_{per} measures the ratio of white pixels of the current circle and the ideal circle (*C*,*R*) and shows the "fullness" of the circle. A full circle has $Q_{per} = 1$, while a circle with fewer white pixels along the ideal perimeter has $Q_{per} < 1$.

The plate quality of the circle measures the ratio of the black pixels and the total number of pixels inside (*C*,*R*) in I_{BIN}, as follows:

$$Q_{plate} = \frac{N_{black-plate}}{2R\pi}, \quad (2)$$

where $N_{black-plate}$ is the number of black pixels. High plate quality value close to 1 indicates that almost all of the pixels inside the circle are black, i.e. there is no significant change of color inside the circle. An ideal round mushroom would have $N_{black-plate} = 1$, while a false detection on the mycelium has small plate quality.

The color index measures the average brightness I_{mean} of the circle in image I_{GRAY}. The color quality is defined as follows:

$$Q_{color} = I_{plate}/I_{max}, \quad (3)$$

where I_{max} is the maximum possible value of the brightness (e.g. in an 8-bit picture $I_{max} = 255$). White mushrooms have high, while the ground and the mycelium have low brightness values.

Using quantities Q_{per}, Q_{plate}, and Q_{color}, each circle is tested. If all three quantities are higher than experimentally defined values L_{per}, L_{plate}, and L_{color}, respectively, then the circle is accepted, otherwise rejected. To each of the accepted circles a combined quality index Q is assigned:

$$Q_{comb} = \mu_{per}Q_{per} + \mu_{plate}Q_{plate} + \mu_{color}Q_{color}, \quad (4)$$

where weighting factors μ_{per}, μ_{plate}, and μ_{color} are set experimentally.

After the quality check the pruning is performed. Initially, the candidate set *A* contains all of the accepted circles with their quality indices, while the output set *B* is empty. The core of the pruning algorithm is the following:

Step 1. Select circle *C* with the highest * in *A*.

Step 2. Move *C* from *A* to *B*.

Step 3. Remove circles from *A*, which are covered by *C*.

Repeat Steps 1-3 until *A* is empty.

The pruning is performed in two rounds. In the first round the pruning algorithm is run with * = Q_{comb}. Then the output of the first round is moved to the candidate set, and the pruning algorithm is repeated with * = *R*. The output set of the second round constitute the final output $\boldsymbol{P}_N$ of phase #1.

Notice that user intervention is possible at the end of phase #1. Here the found circles may be adjusted, deleted, or new circles may be added to $\boldsymbol{P}_N$.

C. Phase Two

The output of phase #1 provided an initial set of circle estimates $\boldsymbol{P}_N$ on image I_N, as shown in Fig. 1. In phase #2 images are processed in backward direction, and the search on image I_n is performed using the results $\boldsymbol{P}_{n+1}$. Since no new mushrooms can appear backwards in time, and the size and

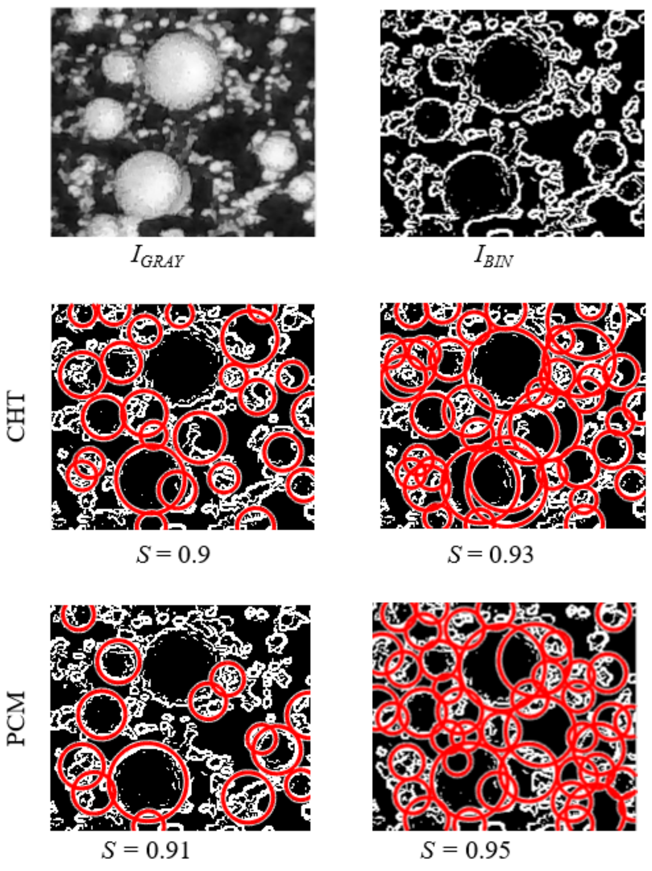

Fig. 3. The perfromance of the circle finder aloritms CHT and PCM, with various sensitivity values *S*.

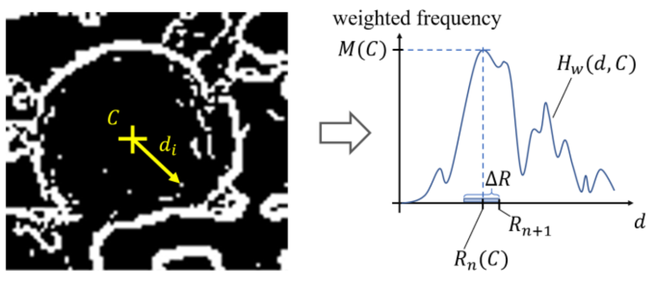

Fig. 4. Estimation of a circle's radius and quality at a given center (*x*,*y*).

position of existing mushrooms do not change significantly from one image to another, the search in image I_n is concentrated to positions and sizes $\boldsymbol{P}_{n+1}$ found in image I_{n+1}.

The search process for one particular mushroom is illustrated in Fig. 4. The position of this mushroom on image I_{n+1} is $\boldsymbol{P}_{n+1} = (C_{n+1}, R_{n+1})$. We assume that for position $\boldsymbol{P}_n = (C_n, R_n)$ the following constraints are true:

$$|C_{n+1} - C_n| \leq \Delta C, \tag{5}$$

and

$$0 \leq R_{n+1} - R_n \leq \Delta R. \tag{6}$$

Constraint (5) expresses the fact that the change of the center position of a mushroom is limited, the limit being ΔC. Notice that this change may happen because of the uneven growth of a mushroom, or a mushroom may be pushed by another touching mushroom, and estimation inaccuracies also can result a small shift.

Constraint (6) expresses the fact the radius of a mushroom increases in time. The limit of increase between two images is limited by parameter ΔR.

The core circle finding algorithm is the following:

Input: a central point C in the vicinity of C_{n+1}.

Step 1. On image I_n measure the distance of every white pixels from C. The distance of pixel i from C is d_i, as shown on the left-hand side of Fig. 4.

Step 2. Create the $H(C,d)$ histogram of distances, with bin size b equal to the required resolution (e.g. one pixel), so that $H(C,d)$ counts the number of white pixels on a circle with center C and radius between $d - b$ and $d + b$.

Step 3. Create a weighted histogram $H_w(C,d)$, as follows:

$$H_w(C,d) = \frac{H(c,d)}{2\Pi d}, \tag{7}$$

which defines the ratio of the actual number and the maximum number of white pixels on the circle. $H_w(C,d)$ is illustrated on the right-hand side of Fig. 4. The closer H_w is to 1 the "more perfect" the circle is.

Step 4. Search for the maximum of H_w in the region of $R_{n+1} - \Delta R \leq d \leq R_{n+1}$. Thus the radius of the best circle with center C is estimated as follows:

$$R_n(C) = \underset{R_{n+1}-\Delta R \leq d \leq R_{n+1}}{\operatorname{argmax}} H_w(C,d), \tag{7}$$

The quality Q of circle is defined as follows:

$$M(C) = \max_{R_{n+1}-\Delta R \leq d \leq R_{n+1}} H_w(d), \tag{8}$$

The algorithm defined by steps 1-4 provides the best radius, given the center C of the circle. During the estimation process these steps are repeated starting from different values of $C = C^k$, such that $|C_{n+1} - C^k|$, so that the region around C_{n+1} is searched with the required resolution (e.g. 1 pixel). The best estimate will define the new estimated circle:

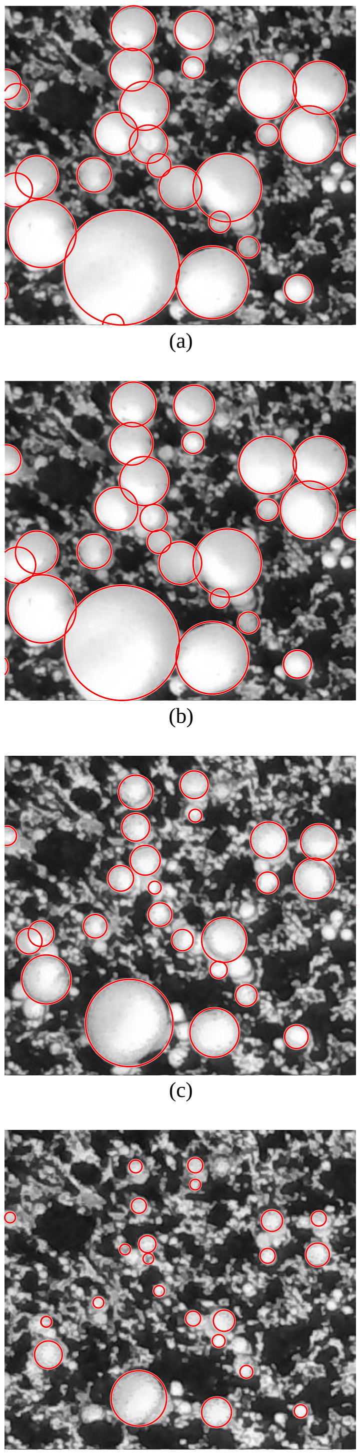
(a)
(b)
(c)
(d)

Fig. 5. Example processing. (a) input image I_{60} and phase #1 detection results for $10 \leq R \leq 60$, (b) manually corrected results, (c)-(d) phase #2 results for images I_{30} and I_1

$$C_n = \underset{|C_{n+1}-C^k| \leq \Delta C}{\operatorname{argmax}} M(C^k), \tag{9}$$

and the radius is $R_n(C_n)$, using (7).

Notice that he circle finder solution used in phase #2 is not suitable for general circle finding, since it accepts any circular portion of a white are as a solution. Since the previous steps already provides a good estimate, it is used to fine-tune the solution.

IV. Experiments

A. Image collection

The data collection device contained a Raspberry Pi 4 mini-PC with 4GB RAM and two BME280 temperature, humidity, and barometer sensor modules. The images were taken with an Intel RealSense SR305 RGBD camera and saved to a flash drive. Every day a backup was created in a private cloud. The camera provided a VGA size depth map and a Full HD color image. The depth sensing of SR305 is based on structured light projection, where a known pattern is projected into the scene and by evaluating how this pattern deforms, the depth information is computed in the range of 0.2 - 1.5m. However, the compost albedo number is very low. Thus, most of the infrared light from the depth camera is absorbed by the compost. Since there is no reflection in these places, the depth map contains zeros there. The data collection device was placed ~700mm above the mushroom growing bed. The illumination was solved using neutral white LEDs. Two meters of LED stripe are used with 20W total power. A 50W power supply provided the energy supply, and an IP67-protected servomotor helped to open and close a camera protector cover. The data collection equipment had to withstand several environmental challenges: 100% relative humidity, 70 degrees Celsius temperature, and a corrosive environment during the 48-hour-long sterilization between cultivation cycles. Fig 6. Shows the image collection device installed in the mushroom house.

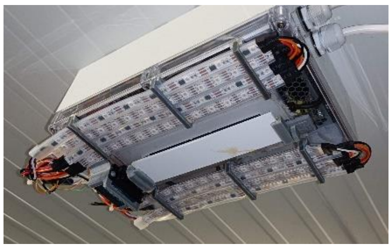

Fig. 6. Automatic image collection device mounted over the mushroom growing bed

B. Image Processing

An example processing is shown in Fig. 5. The pictures were taken in a mushroom cultivation facility over 60 hours. One picture was taken in each hour, producing input $I_1 \dots I_{60}$. The last picture I_{60} is shown in Figs. 5(a) and 5(b), while I_{30} and I_1 are shown in Figs. 5(c) and 5(d), respectively.

Phase #1 results are shown in Fig. 5(a) as red circles.. A few misdetections an inaccurate detections can be observed, which were manually corrected, as shown in Fig. 5(b). This was the input of phase #2. Two further frames are shown in the illustration: I_{30} was taken in the middle of the growing process, while I_1 is the first stage. The detections are shown with red circles.

The advantages of the proposed approach are clearly visible in Fig. 5(d). here the small mushrooms are hard to detect and distinguished from the background pattern. Using detection results from later frames makes the process much more reliable.

Conclusions

In this paper a semi/automatic image processing system was prosed to detect mushroom in a sequence of pictures taken on the plant. The proposed method contains two phases: in phase #1 the last image with the largest mushrooms is used and sufficiently large items are detected. In phase #1 classical circle-finding methods were used with specially tailored scoring and pruning mechanism. In phase #2, proceeding in backward direction in time, detection results of later images are utilized to find smaller items in the image, presumably nearly at the same position with almost the same size. In phase #2 a histogram-based detection was utilized to find items in the near vicinity of their presumed location.

The proposed method is intended to be utilized in the ground-truth generation phase of machine learning algorithms, which can provide growth models for mushroom crops.

Acknowledgment

Peter Galambos is a Bolyai Fellow of the Hungarian ´Academy of Sciences. Peter Galambos is supported by the UNKP-22-5 (Bolyai+) New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research Development and Innovation Fund.

References

[1] C.-H. Chen, H.-Y. Kung, and F.-J. Hwang, "Deep Learning Techniques for Agronomy Applications," Agronomy, vol. 9, no. 3, paper 142, Mar. 2019, doi: 10.3390/agronomy9030142.

[2] M.D. Bah, A. Hafiane, R. Canals, "Deep learning with unsupervised data labeling for weed detection in line crops in UAV images," Remote Sens. vol. 10, no. 11, paper. 1690, Oct. 2018.

[3] M. Mehra, S. Saxena, S. Sankaranarayanan, R.J. Tom, M. Veeramanikandan, "IoT based hydroponics system using deep neural networks," Comput. Electron. Agric. vol. 155, pp. 473–486, Dec. 2018, doi: 10.1016/j.compag.2018.10.015.

[4] L. Zhong, L. Hu, H. Zhou, "Deep learning based multi-temporal crop classification," Remote Sensing of Environment, vol. 221, pp. 430-443, Feb. 2019, doi: 10.1016/j.rse.2018.11.032.

[5] C.-P. Lu, J.-J. Liaw, T.-C. Wu, and T.-F. Hung, "Development of a Mushroom Growth Measurement System Applying Deep Learning for Image Recognition," *Agronomy*, vol. 9, no. 1, paper 32, Jan. 2019, doi: 10.3390/agronomy9010032.

[6] T.M. Gundoshmian, S. Ardabili, C. Mako, A. Mosavi, "Modeling and optimization of the oyster mushroom growth using artificial neural network: Economic and environmental impacts," Mathematical Biosciences and Engineering, vol. 19, no. 10, pp. 9749-9768, July 2022, doi: 10.3934/mbe.2022453

[7] C. Folberth, A. Baklanov, J. Balkovič, R. Skalský, N. Khabarov, M. Obersteiner, "Spatio-temporal downscaling of gridded crop model yield estimates based on machine learning," Agricultural and Forest Meteorology, vol. 264, pp. 1-15, Jan. 2019, doi: 10.1016/j.agrformet.2018.09.021.

[8] D. Paudel, et al., "Machine learning for large-scale crop yield forecasting," Agricultural Systems, vol. 187, paper 103016, Feb. 2021, doi: 10.1016/j.agsy.2020.103016.

[9] M.G. Schultz et al., "Can deep learning beat numerical weather prediction," Phil. Trans. R. Soc. A., vol. 379, paper 20200097, Feb. 2021, doi: 10.1098/rsta.2020.0097

[10] A. Kolková, M. Navrátil, "Demand Forecasting in Python: Deep Learning Model Based on LSTM Architecture versus Statistical Models," Acta Polytechnica Hungarica vol. 18, no. 8, pp., 123/141, 2021, doi: 10.12700/APH.18.8.2021.8.7.

[11] P. Gogas, T. Papadimitriou, "Machine Learning in Economics and Finance," Comput Econ, vol. 57, 1–4 (2021). https://doi.org/10.1007/s10614-021-10094-w

[12] E.R. Davies, "A Modified Hough Scheme for General Circle Location," Pattern Recognition Letters, vol. 7, no. 1. pp. 37–43, Jan. 1988, doi: 10.1016/0167-8655(88)90042-6.

[13] Priyanka Mukhopadhyay, Bidyut B. Chaudhuri, "A survey of Hough Transform," Pattern Recognition, vol. 48, no. 3, pp. 993-1010, 2015, doi: 10.1016/j.patcog.2014.08.027.

[14] T.J. Atherton, D.J. Kerbyson, "Size invariant circle detection," Image and Vision Computing, vol. 17, no. 11, pp. 795-803, Sep. 1999, doi: 10.1016/S0262-8856(98)00160-7.

[15] MathWorks: “imfindcircles. Find circles using circular Hough transform.” online: https://www.mathworks.com/help/images/ref/imfindcircles.html. Accessed: Sept 1., 2022.