COBEM 2021 26th International Congress of Mechanical Engineering
November 22-26, 2021, Virtual Congress, Brazil

COB-2021-0806
AUTONOMOUS NAVIGATION WITHIN CORN PLANTATION USING COMPUTER VISION

Gabriel Lima Araujo
Vitor Akihiro Hisano Higuti
University of São Paulo, Av Trabalhador São-Carlense, 400, São Carlos, São Paulo, Brazil
gabrielaraujo18@usp.br, vitor.higuti@usp.br

Marcelo Becker
University of São Paulo, Av Trabalhador São-Carlense, 400, São Carlos, São Paulo, Brazil
becker@sc.usp.br

Abstract. *Food shortages resulting from the population increase is a projection that has been of concern to a large number of researchers, generating a demand for more efficient methods of increasing food production while also minimizing environmental footprint. To increase efficiency and reduce costs and time in crop scouting, the application of autonomous robots for phenotyping research and monitoring of plantations is very attractive. In such context, this research investigates a method to allow vision-based navigation of a small robot within rows of a corn crop. The proposed method estimate the robot's pose from images treated by filters and color separation, binarization, edge detection and Hough transform, where the latter returns the possible lines to be chosen by the code as the desired path. The project stands out from the others by presenting navigation within the trails, as opposed to most researches that feature navigation over the top of the crops. The images used by the system to guide can also be used to study the phenotypes presented by the plants, making the system more efficient. The results obtained by the method were compared to manually annotated outputs and error standard deviation was 1.2035 degrees for earlier stages and 2.5295 degrees for later ones. Since its execution time is around 50 ms, such results seem satisfactory to be field implemented.*

Keywords: *Computer Vision, Autonomous Navigation, Corn Plantation*

1. INTRODUCTION

In recent years, population growth has been the object of study of researchers interested in not only its benefits but also in its complications, such as lack of space and food. Becoming popular in the 1950s, industrial automation was, and has been, of great importance for the growth of industries around the world. Given its effect, applying the concept of automation in other areas can prove to be quite profitable, with the agricultural field being one of them. To automate and increase production in the field, it is first necessary to understand and master how it works, so studying the behavior of plants under different conditions is an alternative. The field of genotype studies has been increasingly advancing, in part, due to computation; the field of phenotype studies, on the other hand, has been limited by the lack of technologies that can automate the process of collecting and analyzing data like described by Blackmore (2016). As a way to alleviate this problem, several laboratories in all continents have been researching ways to automate this process. In this context, researching and developing autonomous terrestrial robots since 2010, the LabRoM (Mobile Robotics Laboratory) is focused on creating an Full Path Autonomous Navigation (FPAN), that is, a system capable of dealing with any adversity that it encounters in its path while it autonomously navigates through the field.

The process of automating a robot involves a number of sensors and subsequent computation that can process their data. The most commonly used types of sensors are those for robot odometry fused with LiDAR type sensors (Light Detection And Ranging) or fused with cameras. While LiDAR-type sensors have the advantages of not suffering from variations in lighting, the cameras have the advantage of being used not only for the direction of the robot, but also for the analysis of plantations, since the same images captured can serve to both activities. In addition to the use of cameras, their placement not overhead, provides the benefit of being able to use smaller and lighter robots, allowing to follow the most advanced stages of the plant's development. It should be noted that as the robot has a small size and will move within the plantation, the GNSS signal (Global Navigation Satellite System), even in RTK mode (Real Time Kinematic), may not be available, or with a lot of noise due to the problem of obstruction and signal reflection ("Canions"). Therefore, even with an embedded GNSS sensor, the robot will use another configuration of sensors for navigation. The main robot platform implemented to develop the method was TerraSentia (joint research with the University of Illinois at Urbana-Champaign).

This paper describes the design process of the vision method capable of defining the robot position from images treated by filters and color separation, binarization, edge detection and Hough transform, tests and results. Section 2 describes the robotic platform, the field and the dataset. Section 3 explains how the method was developed and how it works. Section 4 shows the results obtained by the method and its errors.

2. ROBOTIC PLATFORM AND DATASET COLLECTION

The TerraSentia described in Kayacan *et al.* (2018) and Higuti *et al.* (2018) (Fig. 1) is an ultra-compact (0.31 m wide), ultra-light (6 kg) and low-cost robot developed by the University of Illinois at Urbana-Champaign and the American energy department focused on activities within the plantations, navigating within crop lines. It can be embedded with several types of sensors, and is equipped with a 64-bit minicomputer (Raspberry Pi 3 B), which is the main platform for conducting field tests.

Figure 1. TerraSentia in field. Source: EarthSense (2020)

For the tests of the method, two datasets were used, obtained in field expeditions prior to the research and made available to the laboratory, being one with plantation in early stages of development and other in middle stages of development. This field, from the University of Illinois, in Urbana-Champaign, presents a more regular terrain, a distance between maize rows of $(0.8 \pm 0.1)\ m$ and an average distance between plants of $0.1\ m$.

3. VISION CONTROL

3.1 Description of the methodology

The method, summarized in Fig. 2, was developed in C++ with the OpenCV library (C++) and tested on a computer running the Ubuntu 18.04 operating system, equipped with an Intel Core i7-7700HQ, a 4GB Nvidia Geforce GTX 1050 video card and 16GB of RAM memory.

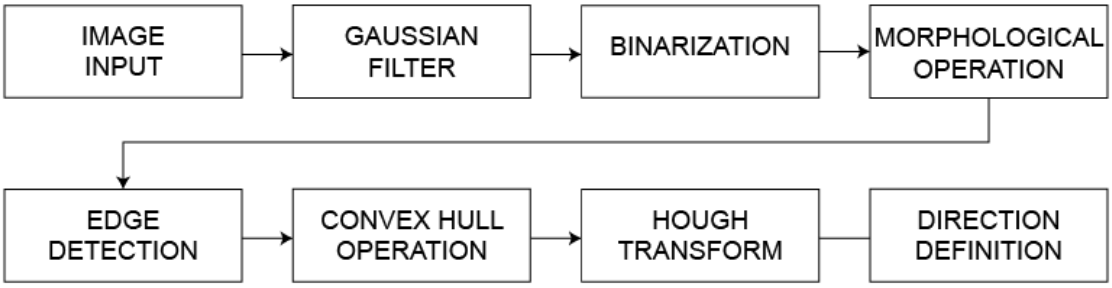

Figure 2. Method flowchart

3.2 Images obtaining

The images present in the dataset used, were obtained in experiments with the robot TerraSentia Higuti *et al.* (2018). This is equipped with 3 USB cameras (one in front and one on each side) capable of acquiring color images of up to 5 megapixels, these being the model ELP-USB500W02M-L21 (manufactured by ELP). For the development of the research, only images from the front camera were used.

3.3 Gaussian Filter

The Gaussian filter is applied to reduce noise present in the image. For this, it blurs the image, which is a result of the smoothing applied like reported by Jesus and Costa Jr (2015). The functioning of the Gaussian filter in image processing can be described as the convolution of a two-dimensional mask, $G(x, y)$, generated from predetermined dimensions and standard deviation. The values present in the mask are governed by the Eq. (1):

$$G(x, y) = \frac{1}{2 * \pi * \sigma^2} * e^{-\frac{x^2+y^2}{2*\sigma^2}} \tag{1}$$

For code implementation, the *GaussianBlur* function from the OpenCV library was used, and the mask size parameter was provided for it. The sigma parameter is determined by Eq. (2):

$$\sigma = 0.3 * ((ksize - 1) * 0.5 - 1) + 0.8 \tag{2}$$

Where *ksize* represents the mask size. This value was empirically determined by varying it within the range 1-101 through a slider, observing which provided the best results for the treated images. After this analysis, a value of 61 was obtained as ideal for the mask, as values below results in images with noise and above results in heavy blur. You can see the original image in the Fig. 3 and after applying the filter in the Fig. 4.

Figure 3. Original image.

Figure 4. Image with Gaussian filter applied.

3.4 Color and Binarization Treatments

After applying the filter, a color treatment was performed, where the image was converted from the RGB (Red Green Blue) color system to the color system HSV (Hue Saturarion Value). The maximum and minimum values of the parameters Hue, Saturarion and Value were defined using sliders, observing which values provided the best results for the zones of interest, the green of the plantations. The values obtained were:

$$15 \leq Hue \leq 80 \qquad 0 \leq Saturation \leq 160 \qquad 0 \leq Value \leq 165 \tag{3}$$

Once this separation was done, the image was then binarized (Fig. 5), using the *inRange* function that receives as parameters the image and the values defined in the color treatment, returning a black and white, with white being the area of interest.

Figure 5. Binarized image.

3.5 Morphological Operation

After the binarization process, a morphological opening operation was applied to the image to smooth the present contours, facilitating the next step. This operation consists of two other operations: one for erosion, followed by one for dilation.

Erosion can be defined by the convolution of a kernel where the pixel in the original image will only be considered 1 (white) if all the pixels over the kernel are 1, if otherwise, this pixel is considered to be zero (black), that is, being eroded. This way, all pixels near the edges are eroded, depending on the size of the kernel, as well as isolated pixels.

Mathematically, the erosion process can be defined as A eroded by B:

$$A \ominus B = \{z|(B)_z \cap A^c \neq \phi\} \tag{4}$$

Where ϕ represents the empty set, B the *kernel* and A^c the supplementary set of A defined in Raid *et al.* (2014).

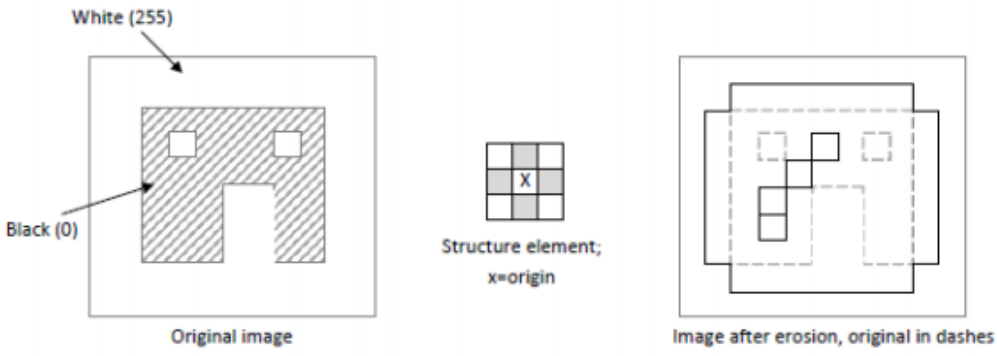

Figure 6. Process of erosion of an image. Original on the left and eroded on the right. Source: Raid *et al.* (2014)

Dilation, on the other hand, is treated as a dual operation to erosion, that is, it works in the opposite way. Thus, a pixel is set to 1 if during the convolution of the kernel over the original image, at least one pixel over it is 1 (white). In this way, there is a dilation of the image, increasing its area.

Similarly, the dilation process can be defined mathematically as A dilated by B:

$$A \oplus B = \{z|(\hat{B})_z \cap A \neq \phi\} \tag{5}$$

Where ϕ represents the empty set, B the *kernel* and $\hat{B}$ the reflected set of B.

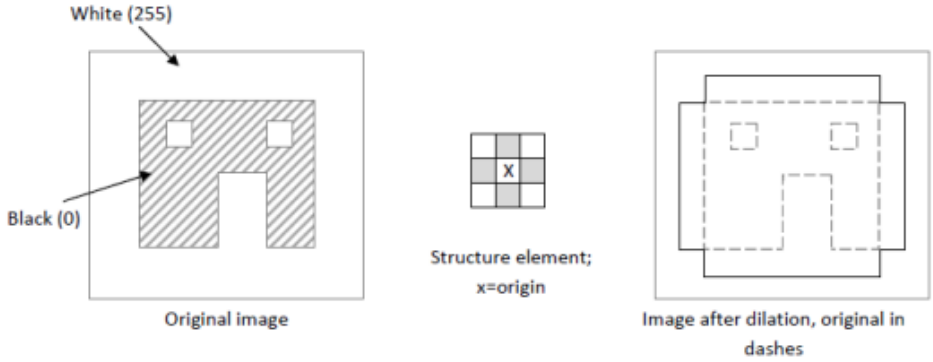

Figure 7. Process of dilation of an image. Original on the left and dilated on the right. Source: Raid *et al.* (2014)

In the opening process, the same kernel is used for both the erosion and expansion operations. Thus, an elliptical kernel of 15x15 was applied, resulting in the image of the Fig. 8. The opening process can be described mathematically as:

$$A \circ B = (A \ominus B) \oplus B \tag{6}$$

Where A represents the set to be opened and B the kernel.

Figure 8. Image after opening operation.

3.6 Edge detection

Once the image was operated morphologically, an edge detection function (Canny Edge Detector) was performed. This function returned only the contours present in the binarized image, resulting in the image present in the Fig. 10.

This function consists of multiple parts, namely:

- Noise Reduction: Application of a 5x5 Gaussian filter;
- Encounter of the gradient of the edge (**G**) of the image: The Sobel Filter is applied in both directions of the image obtaining the magnitude, Eq. (7), and direction of the gradient, Eq. (8);

$$G = \sqrt{G_x^2 + G_y^2} \tag{7}$$

$$\theta = tg^{-1}\left(\frac{G_y}{G_x}\right) \tag{8}$$

Where G_x is the derivative in horizontal direction and G_y and the derivative in vertical direction.

- Non-Maximum Suppression: Removes pixels that are not considered edges, so it is tested for each pixel if it is a local maximum in its vicinity in the direction of the gradient. For example, point A in figure 9 is on the edge. The gradient is normal to the edge and has point B and C towards it. In this way, point A is checked if it is a local maximum along with B and C. If it is positive, the next step is executed, otherwise it is considered to be zero;

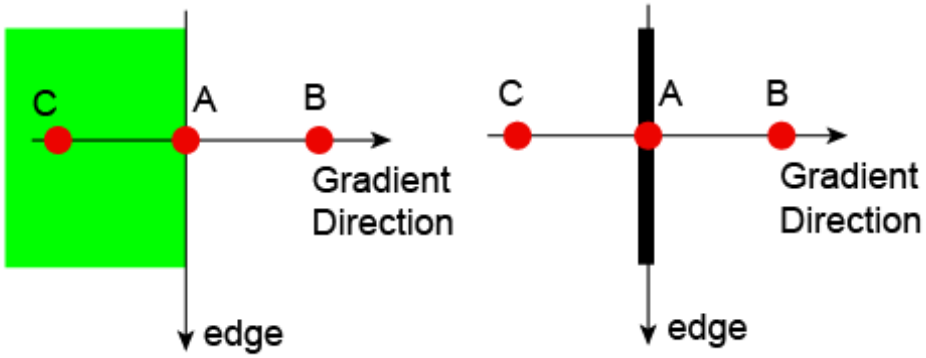

Figure 9. Detection of edge by gradient. Source: Soon *et al.* (2019)

- Hysteresis: It is decided in this step which of the edges are really edges and which are not. For this, the maximum and minimum values for the intensity of the gradient are established, for any edges that present values that are not between them are discarded as edges. Those that meet this requirement are only considered edges if they are connected to pixels that are definitely edge.

3.7 Convex Hull Operation

The function uses the Sklansky (1982) algorithm, that finds the convex hull of a 2D point set. Figure 11 represents the output of the function when executed on the image from the edge detection process.

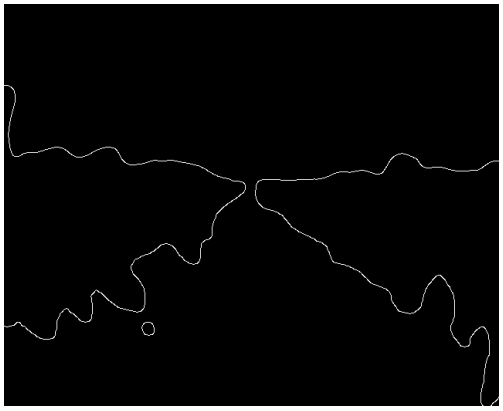

Figure 10. Image outlines after edge detection.

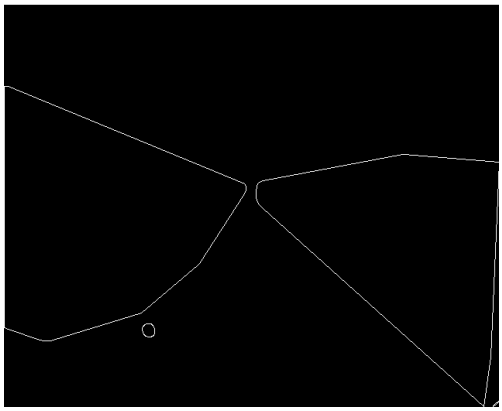

Figure 11. Convex Hull of the edge image.

3.8 Hough Transform

The Hough transform is responsible for identifying the lines present in an image. This one works by converting each point of the cartesian space (x, y) into a sinusoidal space $(\rho = xcos\theta + ysen\theta)$, corresponding to the Hough space (Fig. 12). A straight line in cartesian space is translated as an intersection of multiple sinusoids in Hough space, as explained in Chen *et al.* (2020). This is because for each point swept in the cartesian plane, all the lines passing through it are taken. An accumulator is responsible for determining locations where there are a large number of intersections. For the development of the method, the probabilistic Hough transform was used, which works by transforming random points of the limit image in the accumulator itself. Thus, when determining an infinite line, it is traversed until the limit pixel is found. This transform has the advantage of returning finite lines, which facilitates the line filtering steps, in addition to reducing execution time.

Thus, the *HoughLinesP* function of OpenCV was used to identify the lines. This function takes some parameters in addition to the image, they are:

- *rho* : The resolution for the value of r;

- *theta*: The resolution of the value of θ;
- *threshold*: The minimum number of intersections for a thread to be detected;
- *minLineLength*: The minimum number of points for a line to be considered;
- *maxLineGap*: The maximum distance between two points so that they are considered to belong to the same line.

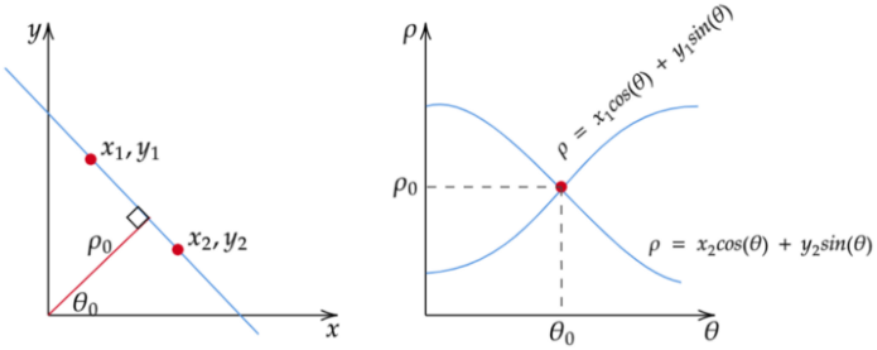

Figure 12. Representation of a line in Cartesian space and Hough space. Source: Lee (2020)

It is not desired a very small maxLineGap, as it would not be representing the guide correctly. The same is true for minLineLength, as a line without many points will probably not be representing the guide, or will be too small to be used for analysis. A very large threshold would reduce the number of lines found, which is positive because it reduces the effort in the analysis step, but could remove the lines that represent the tab. Finally, the resolution of r was kept at 1 pixel and that of θ at 1 degree.

Before applying the transform, a cutout of the area of interest was made, consisting of a smaller area, 36% of the original, in the center of the image. For the proper functioning of the function, it must receive those parameters and for that the use of sliders was made, observing which values provided the best lines. From this, the values shown in the Tab. 1 were obtained.

Table 1. Parameters of Hough Transform

Parameters	**Values**
rho	1
θ	1°
threshold	20
minLineLength	50
maxLineGap	70

The transform returns several lines (Fig. 13), however, not all of them are important for defining the robot's direction. In view of this, it is necessary to analyze these lines in order to use them efficiently.

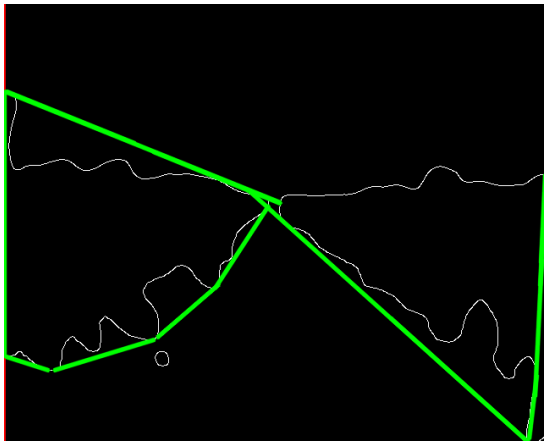

Figure 13. Result of applying the Hough transform.

3.9 Definition of Direction

Once the lines were identified, the binarized image was analyzed once more, this time with the lines and with inversion of tones, and then the contours, their areas and their centroids were taken using the *connectedComponentsWithStats* function from OpenCV.

This function has a more granular control of contour analysis and returns more than just contour when compared to what was used previously, at the expense of time. This one works with a region labeling algorithm using graph theory, where subsets of connected components are labeled exclusively based on a certain heuristic described in Grana *et al.* (2010). When executed, it returns several parameters, such as image height and width, in an array called *Stats* for each labeled region. A matrix with the centroids of each of the regions is also returned.

The contours with the two largest areas represent the sky and the corridor (Fig. 14), as only the corridor is of interest, a selection was made using its centroid. The centroid that has the highest value for the y coordinate represents the corridor, as the value of y grows from the top edge to the bottom. Once the contour is selected, the lowest value point for the y coordinate present in the contour is used, together with the middle point of the x coordinate of the lower edge of the zone of interest, to build the direction vector. Once the position vector was defined, a simple moving average was applied, to smooth the movement of this vector (Fig. 15).

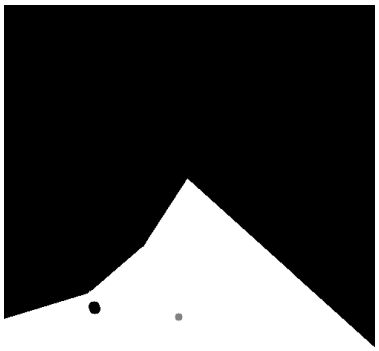

Figure 14. Biggest contour found in white, centroid in gray.

Figure 15. Direction defined by the method in green, expected result in red.

4. RESULTS

From the code described in the previous step, two sets of data were analyzed, one containing corn in its early stages of growth (Fig. 15) and the other with a more advanced stage of growth, which may have dry leaves on the ground (Fig. 16), which presents a greater difficulty for the vision code. This analysis of the data obtained by the code was done using the Matlab software.

Figure 16. Direction resulting from the second dataset used in green, expected result in red.

The evaluation of the result was made by comparing the expected location, manually positioned, with the results obtained by the method. For comparison purposes, the error was calculated, following the Eq. (9):

$$error = calculated\ angle - expected\ angle \tag{9}$$

Analyzing the results for the first dataset, the angle graph (Fig. 17) provides us with a visualization that the code is working properly and the correct position was determined, but with some difficulty. Three significant peaks were observed, around 200 frames, around 375 frames and around 530 frames, resulting from the presence of very sudden camera movement, causing heavy blur. By the end, the robot is close to the end of the crops rows, when it starts to lose reference. It is more evident in the error graph, where the error starts to oscillate more by the end. The standard deviation of the error is 1.2035 degrees.

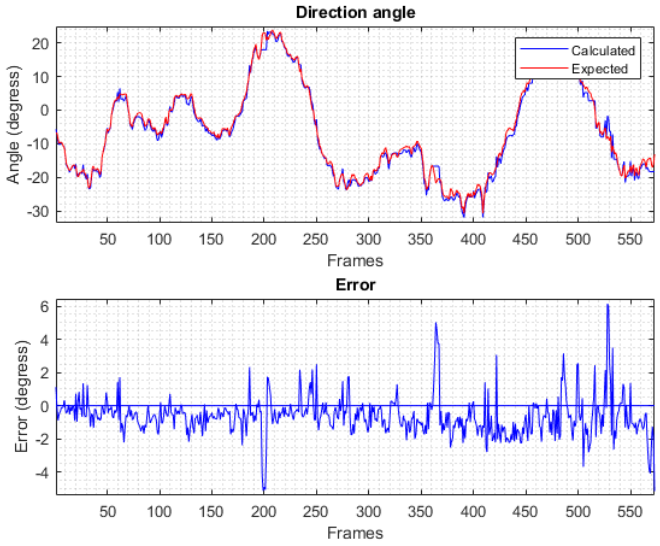

Figure 17. Results from execution of first dataset and error.

Analyzing the results for the second dataset, the angle graph (Fig. 18) provides us with a visualization similar to the first one, where the correct position was determined, but not perfectly. Two significant peaks are observed, around 230 frames and around 350 frames, resulting also from the presence of very sudden camera movement, causing a blur. On the other hand, the error graph shows that plantations in the most developed state have greater difficulty in determining the correct direction. It can be confirmed when calculated the standard deviation of the error that is 2.5295 degrees.

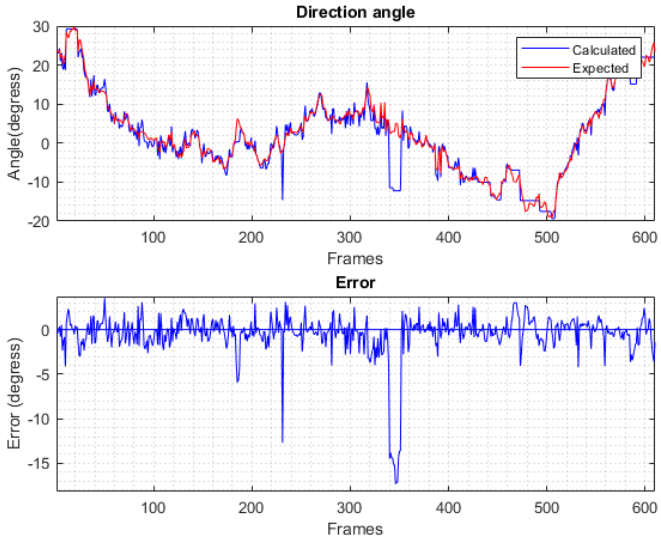

Figure 18. Results from execution of second dataset and error.

Finally, the execution time for each frame was measured, obtaining approximately 50 milliseconds. When compared to the research by Xue *et al.* (2012), the result is satisfactory, since under similar conditions, it had times of 300 milliseconds.

5. CONCLUSION

This study demonstrates the development process, as well as the results obtained, of an autonomous navigation method based on computer vision. Testing under two different conditions shows that the method achieves satisfactory performance. The smallest standard deviation of the error, 1.2035 degrees, occurred in the first dataset, which presents a cleaner trail and plants in the initial stage of development. The biggest standard deviation of the error, 2,5295 degrees, happened in the second dataset, which presents a trail with more fallen leaves, uneven terrain, and plants in a more advanced stage of growth. When compared to different researches, it has a lower execution time per frame, 50 milliseconds.

6. ACKNOWLEDGEMENTS

This paper was supported by grants no. 2020/11089-6 and no. 2018/10894-2, São Paulo Research Foundation (FAPESP).

7. REFERENCES

Blackmore, S., 2016. "Towards robotic agriculture". In J. Valasek and J.A. Thomasson, eds., *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping*. International Society for Optics and Photonics, SPIE, Vol. 9866, pp. 8 – 15. doi:10.1117/12.2234051. URL https://doi.org/10.1117/12.2234051.

Chen, J., Qiang, H., Wu, J., Xu, G., Wang, Z. and Liu, X., 2020. "Extracting the navigation path of a tomato-cucumber greenhouse robot based on a median point hough transform". *Computers and Electronics in Agriculture*, Vol. 174, p. 105472. ISSN 0168-1699. doi:https://doi.org/10.1016/j.compag.2020.105472. URL http://www.sciencedirect.com/science/article/pii/S0168169919324172.

EarthSense, 2020. "EarthSense". URL https://www.earthsense.co.

Grana, C., Borghesani, D. and Cucchiara, R., 2010. "Optimized block-based connected components labeling with decision trees". *IEEE Transactions on Image Processing*, Vol. 19, No. 6, pp. 1596–1609. doi:10.1109/TIP.2010.2044963.

Higuti, V.A.H., Velasquez, A.E.B., Magalhaes, D.V., Becker, M. and Chowdhary, G., 2018. "Under canopy light detection and ranging-based autonomous navigation". *Journal of Field Robotics*, Vol. 36, No. 3, pp. 547–567. doi:10.1002/rob.21852. URL https://doi.org/10.1002/rob.21852.

Jesus, E.O. and Costa Jr, R., 2015. "A utilização de filtros gaussianos na análise de imagens digitais". *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, Vol. 3, No. 1.

Kayacan, E., Zhang, Z. and Chowdhary, G., 2018. "Embedded high precision control and corn stand counting algorithms for an ultra-compact 3d printed field robot". In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation. doi:10.15607/rss.2018.xiv.036. URL https://doi.org/10.15607/rss.2018.xiv.036.

Lee, S., 2020. "Lines Detection with Hough Transform". URL https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549.

Raid, A., Khedr, W., El-dosuky, M. and Aoud, M., 2014. "Image restoration based on morphological operations". *International Journal of Computer Science, Engineering and Information Technology*, Vol. 4, pp. 9–21. doi:10.5121/ijcseit.2014.4302.

Sklansky, J., 1982. "Finding the convex hull of a simple polygon". *Pattern Recognition Letters*, Vol. 1, No. 2, pp. 79–83. ISSN 0167-8655. doi:https://doi.org/10.1016/0167-8655(82)90016-2. URL https://www.sciencedirect.com/science/article/pii/0167865582900162.

Soon, N.K., Abas, Z.A., Ahmad, A. and Rahmalan, H., 2019. "Improvement of the traditional canny edge detection algorithm by using combination of arithmetic mean filter, harmonic mean filter and geometric mean filter". *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*.

Xue, J., Zhang, L. and Grift, T.E., 2012. "Variable field-of-view machine vision based row guidance of an agricultural robot". *Computers and Electronics in Agriculture*, Vol. 84, pp. 85–91. ISSN 01681699. doi:10.1016/j.compag.2012.02.009. URL http://dx.doi.org/10.1016/j.compag.2012.02.009.

8. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.