Hungarian Association of Agricultural Informatics
European Federation for Information Technology in Agriculture, Food and the Environment

Journal of Agricultural Informatics. Vol. 10, No.2
journal.magisz.org

Computer vision in agriculture, application development using open source tools and systems

Mihály Tóth[1], Dániel Dér[2], Szilvia Borbásné Botos[3], Róbert Szilágyi[4]

I N F O
Received 19-12-2019
Accepted 22-01-2020
Available on-line 28-01-2020
Responsible Editor: M. Herdon

Keywords:
computer vision, agriculture, artificial intelligence, neural network

A B S T R A C T

Nowadays the optimization of agricultural production is a crucial task. The use of computer vision may serve in increasing efficiency as this technology has achieved significant results in many research and practical applications to date. In the following years, we will experience more and more use cases of the technology and its usability in the intensification of agricultural production to meet the demand of the growing population. Computer vision appears as a sub-domain, also in the new and popular concept of Industry 4.0, ensuring an integrated aspect of the technology. Our practical experiment was performed to examine the utility of the currently available open-source toolkits in commuter vision, utilizing OpenCV and Google TensorFlow libraries. In this experiment, the typical processes of computer vision were implemented using various algorithms for each step, including imaging, pre-processing, post-processing and finally, classification. For the experiment, pictures of apples have been used as training data, representing various conditions. The steps including processing, segmentation, and identification of the fruit, were presented. The most commonly used detection algorithms were tested to determine estimated size, shape, and texture properties. Using a convolutional neural network, the identification of the fruit was presented with a recognition accuracy greater than 93%.

1. Introduction

In the paper, the possibilities of open-source computer vision methods will be assessed, in agricultural production. Due to the changing nature of our environment as well as the increasing population, it has become a major challenge to reduce its environmental impact, increase food security and develop sustainable agricultural production (Agathokleous and Calabrese, 2019). Machine vision systems have already been developed to a level where it is possible to assess existing electronic systems, upgrade them, or replace them with more efficient ones, thus supporting clients by automating their routine tasks (Van der Stuyft et al., 1991). For the sensory analysis of agricultural and food products, automatic control systems based on cameras and IT solutions have recently been applied and tested for their effectiveness. The machine vision systems have been successful in objectively measuring various parameters of agricultural and food products, that influences a given aspect of the product. Machine vision involves capturing, processing, and analysing images, making it easier to evaluate the visual properties of a given object. Its potential has long been recognized in the food industry. Recent

[1] Mihály Tóth
University of Debrecen, Faculty of Economics and Business, Hungary
University of Debrecen, Károly Ihrig Doctoral School of Management and Business, Hungary
toth.mihaly@econ.unideb.hu
Mihály Tóth
[2] Dániel Dér
derdanis@gmail.com
[3] Szilvia Borbásné Botos
University of Debrecen, Faculty of Economics and Business, Hungary
botos.szilvia@econ.unideb.hu
[4] Róbert Szilágyi
University of Debrecen, Faculty of Economics and Business, Hungary
szilagyi.robert@econ.unideb.hu

developments in hardware and software have assisted at increasing the use of the system by providing low cost and efficient solutions, which have led to new studies that have contributed to the further development of machine vision. As a result, the methods of automated computer vision continue to grow significantly today in the food industry and other sectors as well due to cost-effectiveness, consistency, excellent speed, and accuracy (Brosnan and Sun, 2004).

1.1 Computer vision

Computer vision is part of the IoT (Internet of Things) concept, Cyber-Physical Systems (CPS) and, at the same time, artificial intelligence (AI), due to the fact that the former two is based on sensor-driven functions when the measured data (in this case, images in form of a matrix) are transmitted to computers for processing where artificial intelligence provides the framework for extracting an aggregated output, in form of a classification. After obtaining the data, the process is followed by a customized intervention that is performed according to the results.

Computer vision is the process by which a machine, usually a computer or embedded systems, automatically processes an image and sends a report of what is in the image considering an attribute (Snyder and Qi, 2017). The subject of machine vision is a crucial area of artificial intelligence. The term should not be understood solely as perceptual images (based on a colour imaging sensor) because it includes all sensors that provide spatially ordered visual information about the environment (IR camera etc.). Computer vision consists of two components, which can be divided further. The measurement must first be performed, thereafter the provided data by the measurement must be processed and interpreted. The complexity of this process varies, depending on the environment (real-life environment or laboratory, eliminating noises), the circumstances of the imaging (lighting parameter, including the intensity, temperature, and quality of the light) and finally, the expected results (determining the presence, quality parameters or damages). Based on the data, it is possible to recognize and follow objects, but also to draw new conclusions from indirect data. The digital image is produced by one or more image sensing sensors, which, in addition to the various imaging sensors, contain range sensors, tomography devices, radars or ultrasound cameras (Sinha, 2012).

The classification is one of the important aspects of computer vision. It can be based on K-Nearest Neighbour (KNN), Support Vector Machine (SVM) or Artificial Neural Network (ANN) (Naik and Patel, 2017), mentioning the well-known alternatives. KNN is one of the simpler methods, working in an unsupervised manner. On the basics, it assign data to the most represented category (Naik and Patel, 2017). as its name suggests, it represents each k number of clusters based on the weighted average of the data, included in the clusters (Chinchuluun et al., 2010). The number of clusters has to be chosen manually, resulting in varying result (Kim et al., 2012). SVM classification, on the other hand, uses the concept of decision hyperplanes. The linear hyperplanes are determined using the training dataset, which is being used to split the dataset. It is capable of linear two-class separation, but it can be extended to one or more class separation, non-linear separation, and non-linear regression problems as well. The support vectors are the samples, that are located close to the hyperplanes (Shastry et al., 2017). ANN is the most commonly used method nowadays, which is a network of interconnected nodes, associated with weights that determine the strength of a particular connection (Patrício and Rieder, 2018). We can distinguish between supervised, semi-supervised and unsupervised neural learning methods (Khan et al., 2018), but in this case, supervised methods are dominant. The structure of the network can be either convolutional (CNN) or recurrent (RNN) (Andreas Kamilaris and Prenafeta-Boldú, 2018) to mention the main structures, however as in computer vision we are working with matrices, convolutional neural networks are typical. A deep learning network is different in a sense, that they are built using multiple, interconnected layers. The disadvantage of ANN is that it needs larger dataset and proper labelling in order to obtain an applicable training set (A. Kamilaris and Prenafeta-Boldú, 2018).

1.2 Practical applications

Computer vision can be used throughout the food chain regardless of the process. Based on researches, categories, including weed identification, land cover classification, plant recognition, fruit counting and corn type classification were determined (Andreas Kamilaris and Prenafeta-Boldú, 2018),

however, other use cases can be examined as well, including grading by colour or external quality parameters, ripeness inspection or blemish detection (Bhargava and Bansal, 2018).

During the production process the most frequent method is to segment the specific object considered the required resolution (an object, in case of fruits of animals or an aggregated view of a field in case of arable crop production). In the case of detection, apple detection was performed on the tree, based on decision tree, KNN and SVM (Marzoa Tanco et al., 2018). In an experiment, various fruits could be detected and classified using deep CNN (convolutional neural network) with accuracy more than 82% (Sa et al., 2016). In the case of disease detection, research performed an experiment, where diseases of papaya could be determined using SVM classification with more than 90% accuracy (Habib et al., 2018). Other than classification, we can see examples of prediction, capable of predicting the yield based on colour, representing nutrient content (Shidnal et al., 2019). During food processing, sorting is an important task for quality management. The detection of defects in apples was determined using C4.5 classification with an accuracy rate between 73%-93% based on a research (Sofu et al., 2016). The defects and ripeness of tomato can also be determined based on ANN, with 1005 and 96,47% accuracy accordingly (Arakeri and Lakshmana, 2016). An experiment about dried fig grading explains that classification, based on various indexes can be performed with accuracy between 90% and 100% (Baigvand et al., 2015). Other research describes that soybeans quality evaluation (including the identification of stems, pods and defects) could be performed with accuracy between 75% and 100% (Momin et al., 2017). An experiment with K-means clustering for segmentation and multi-class SVM for classification shows that apple diseases could be determined with more than 93% accuracy (Dubey and Jalal, 2013). It is also possible to build a 3D model to determine length and width, thickness, surface area and volume (Su et al., 2017).

Table 1. Accuracy of the methods

Description	**Processing**	**Classification**	**Accuracy (%)**	
Detection and classification of apple disease	GLCM algorithm	SVM	93	(Dinesh et al., 2017)
Mango recognition	KNN	SVM	92	(Kumari et al., 2019)
Grape leaf disease recognition	KNN	SVM	88,89	(Padol and Yadav, 2016)
Forest recognition	N.a.	SVM	87,9	(Pal, 2005)
Detection of olive infection	Wilks' lambda	PLSDA, SVM	80,85	(Beyaz et al., 2019)
Tree recognition	GLCM algorithm	MLP	72-60	(Tou et al., 2007)
Apple damage detection	I-RELIEF algorithm	RVM	95,63	(Zhang et al., 2015)

In traceability, computer vision and QR codes also serve an important role, because it is necessary to track data starting from raw products, through food processing, transport, warehousing, to retailing and reaching the end consumer (Tarjan et al., 2014). As a consumer, we can also encounter computer vision in food recognition (Kawano and Yanai, 2014) and even food recommendation with balanced nutrition in mind to maintain a healthy diet (Resende Silva and Cui, 2017).

1.3 Material and methods

The experiment has been performed based on open-source libraries, available for Python programming language. OpenCV is an open-source library for machine vision and machine learning. OpenCV is built to provide a common infrastructure for computer vision-based applications. OpenCV

is an open-source library for computer vision and machine learning. OpenCV Python is a wrapper for the original C++ library that can be used with Python. OpenCV is built to provide a common infrastructure, with machine vision-based applications and the ability to increase the machine's sensing speed, as an important consideration. OpenCV supports some programming languages, namely Python, Java, C and C ++. It is true that Python-based programs are usually slower than their C++ alternatives, but they also take much less time to develop, since the codes used in Python can be three to five times shorter. It is open-source, unlike MATLAB, which specializes in data analysis, exploration, and visualization. Based on the mentioned advantages, it proved to be a better alternative in the case of a simple experiment.

The applied methods include pre-processing algorithms (exposure, contrast, gamma and white balance correction, scaling, sharpening, colour space conversion), as well as other methods, including Sobel filter, Gaussian blur, erosion and dilation. After pre-processing the training set using the mentioned methods, the following procedure was the segmentation. In this process, several methods were assessed, including adaptive thresholding, binary thresholding, K-means clustering, watershed algorithm and Canny edge detection. As an alternative solution, several feature extraction methods were assessed as well, including ORB detection (Oriented FAST and Rotated BRIEF), Harris corner detection and blob detection. Using the processed training set, size estimation (based on reference data) and classification were applied, using CNN (convolutional neural network), which can be applied for the whole area of the image as well as for extracted details.

2. Results and their evaluation

Figure 1. illustrates the practice test. The attributes and contact information of the captured images were loaded into a simple relational database. During the processing step, the obtained results were processed using various segmentation methods.

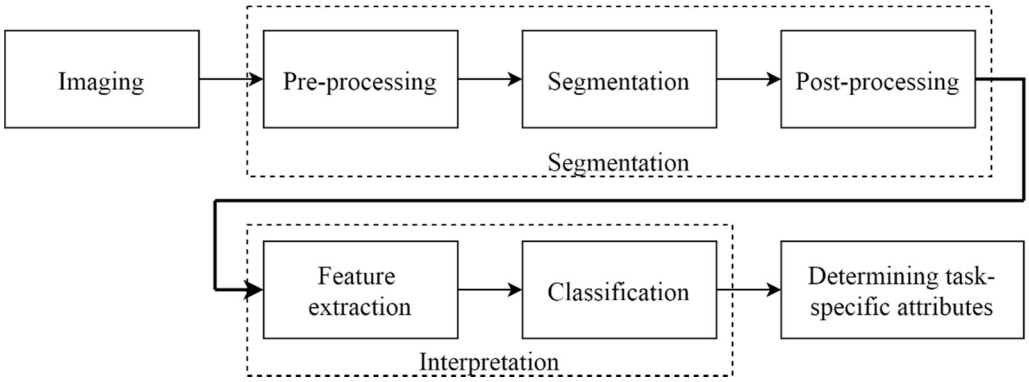

Figure 1. The applied Image Recognition Process (Own-work)

2.1 Imaging

The database is formed by 6 tables. In the database there were uploaded the metadata of 304 images, making it possible to load the file using ID and name, ensuring the usability in iterative functions. The pictures were taken using a mobile device and using a natural light source, pointed from above. The mobile has an aperture of f1.7, which greatly influences the image quality as it determines how much light the sensor can let in at given shutter speed. It also affects the depth of field, however, considering the size of the sensor, the effect is not noticeable. This value cannot be changed on the device being used, as it has a fixed aperture. The exposure and white balance are handled phone itself, which is a significant disadvantage since it results in inconsistent images. The ISO value was set to 125, high enough to work using the ambient light. The focal length is 4.20 mm, which is also a predefined value. The shutter speed is at 1/50s as average, which corresponds to daylight condition and indicates the time required for the exposure. The image metadata is stored in the database. The path to the image is stored here, which can be queried via code into a list, after which the application goes through the list. It also

writes the changes made to the image into the database. Each modified image will be given a new ID and may have a unique name.

2.2 Preprocessing

First of all, we need to standardize the images, and then make image corrections to work easier with them. Input images have a resolution of 4032x3024 pixels and contain hundreds of images in the dataset. The pictures have been resized in order to reduce the unnecessary details (as a possible source of noise) and to determine a standard resolution for all images. During scaling, the aspect ratios were kept despite the smaller resolution. By default, the images have red, green, and blue (RGB) colour channels (additive colour model). In the input image, a gamma correction was performed to optimize the contrast values. The next step was to convert the images to grayscale (12 bit), since the process of image segmentation does not handle colours. The colour images are stored in a separate variable in order to perform operations on the grayscale image so that the processing methods can be performed on the colour image as well, taking the modification into account. At the same time, a blur filter was applied to reduce further details. Of the median blur and Gaussian blur, the latter was our choice after testing both methods, since Gaussian uses a technique that assumes that our image is two-dimensional, so it puts the most weight on the middle pixels. The results of these are illustrated in Figure 2.

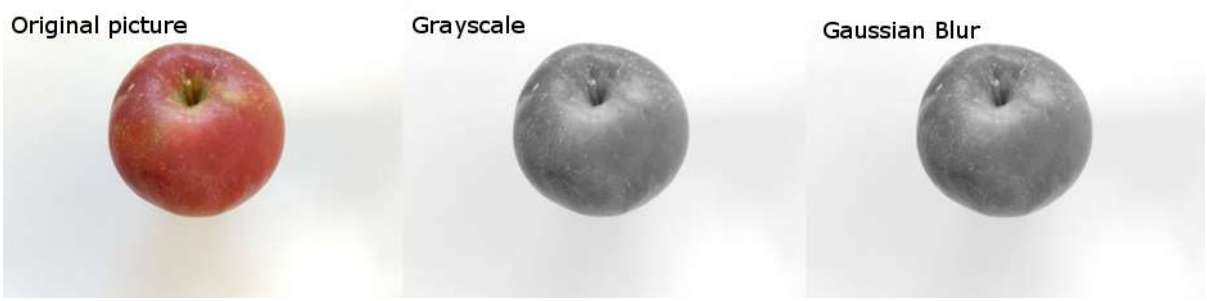

Figure 2. RGB picture grayscale version and Gaussian Blur (Own-work)

After the Gaussian Blur was performed, there was still a large amount of unnecessary information in the picture. This was removed by contour recognition, which identified the earliest next white pixel on each side. The image was cut while the background of the apple was 40 pixels larger than the background. This reduced the image to a uniform 300x300 pixel. Pre-processing can be performed using the Sobel operation, which allows us to recognize the edges of the image both vertically and horizontally. A similar operation used for edge detection, including Gabor filtering. The Gabor filter is mathematically structured to work with different image shapes, sizes, and applied filters. For example, if an image has diagonal edges, the Gabor filter set will only give a strong answer if its direction is the same as the edges Prateekvjoshi (2014).

Figure 3. shows the result for the Sobel operation in the top row, dl/dx detects the edges of the image horizontally and dl/dy detects vertically. The second line shows the Gabor filter, where the middle kernel image scans the image at a 45-degree angle to get the most detail.

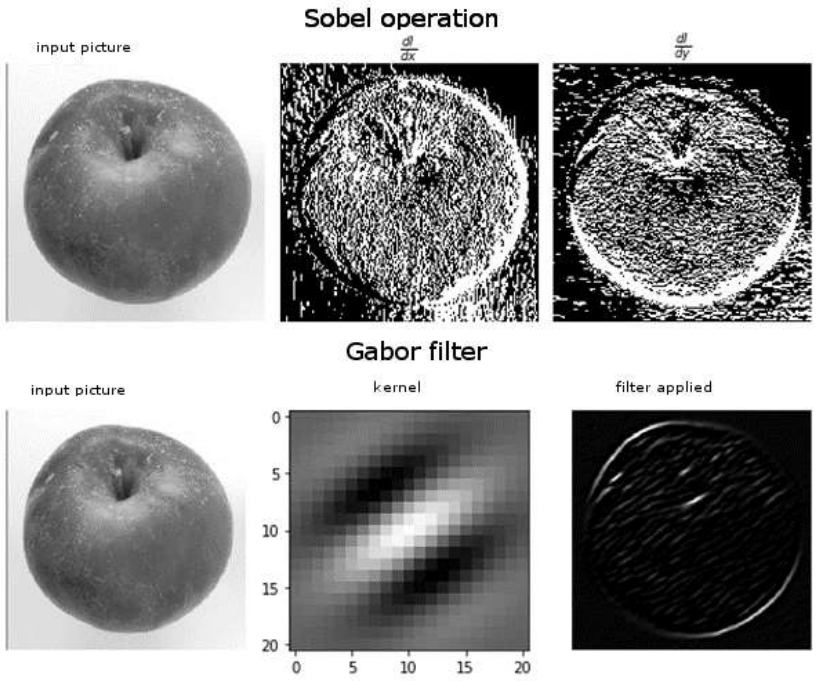

Figure 3. Sobel operation and Gabor filter comparison (Own-work)

The next image processing method, which already plays a role in the segmentation process, is thresholding. This will produce a grayscale image and convert it to a binary image, so it will have only 0 and 1 values. After defining the interval of the intensity range, running the code gives the result shown in Figure 4.

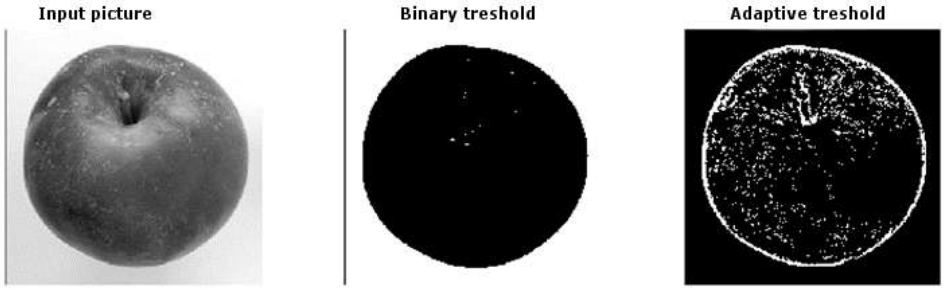

Figure 4. Binary and Adaptive treshold comparison (Own-work)

OpenCV provides various threshold methods. These methods fall into two groups: global, where each pixel is placed at the same threshold, and adaptive, where the threshold depends on the pixels, so the algorithm assigns its own threshold to each pixel.

2.3 Postprocessing

For the first test of the segmentation process, we used the k-means cluster algorithm. At times, the colour of pixels can help define semantically close areas. The algorithm only needs to know how many clusters there are in an image, in other words how many clusters we want our image to appear (Spizhevoy and Rybnikov, 2018). By defining this information, the algorithm will automatically find the best clusters shown in Figure 5.

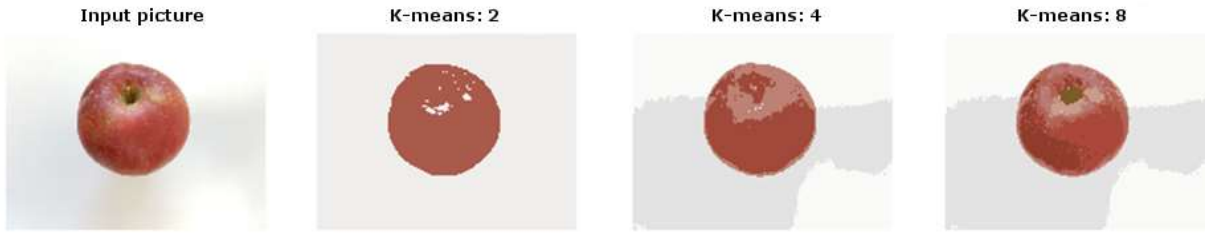

Figure 5. K-means algorithm examples based on different clusters (Own-work)

The figure shows more and more details appear in the images as we progressively increase the number of clusters. While in a two-cluster display the image is completely homogeneous, displaying a total of three colours, the eight-cluster, image is completely reproduced. This allows us to analyse the clusters separately and to classify the separately analysed parts. In segmentation processes, we often hear the expression Watershed. We can use the Watershed algorithm when we have initial segmented points and want to fill automatically the surrounding areas in the same segmentation class. These initial segmented points, called cores, need to be set manually, but in some cases it is possible to automate them (Spizhevoy and Rybnikov, 2018). Figure 6 shows the result.

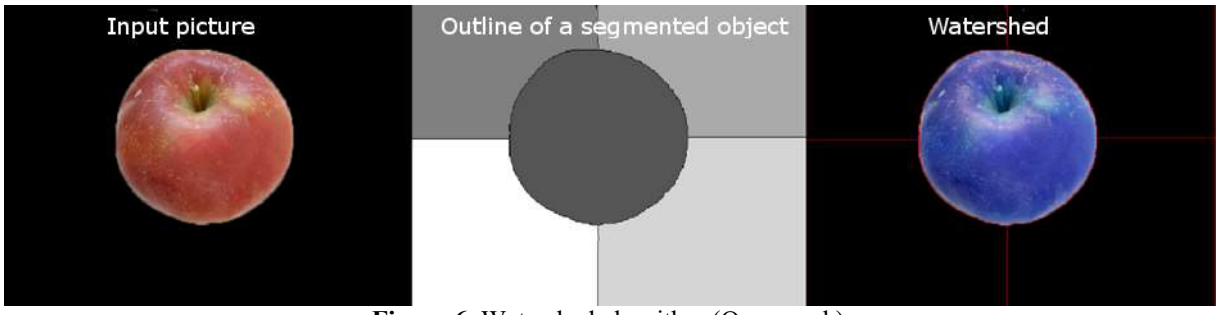

Figure 6. Watershed algorithm (Own-work)

The use of the Watershed algorithm became applicable after several attempts. Variable results were generated with input images with no background removed. The middle picture shows the outline of the segmented apple in 8-bit form, for which we used erosion and dilation, then it was given a specific threshold so that the fruit could be separated from the background. After the background has been successfully separated by the algorithm, it removes the modifications from the segmented object so that we finally get our input image highlighted by its contours (Joe Minichino, 2015).

One of the apples had a noticeable injury showing cuts at right angles to each other. This gave us the opportunity to use Harris's corner detection method. Chris Harris and Mike Stephens developed the algorithm in 1988, and let us not only detect corners, but also detect edges. The algorithm is capable of recognizing edges and corners and cannot be used for texture recognition. Figure 7. illustrates the input image and the result.

Figure 7. Harris's corner detection (Own-work)

This algorithm assists in detecting the corners of an image by scanning the image and detecting locations with the greatest deviation. Once the algorithm recognizes the corners in the image which can be highlighted as shown in the figure (Gollapudi, 2019). The process began by converting the image to grayscale, then it is possible to apply Harris edge detection, based on the built-in methods of the library.

2.4 Classification

One way to extract properties is to determine and recognize the size of the object. In the following test, we tried to determine the size of an apple based on a reference point. The result of this process is illustrated in Figure 8.

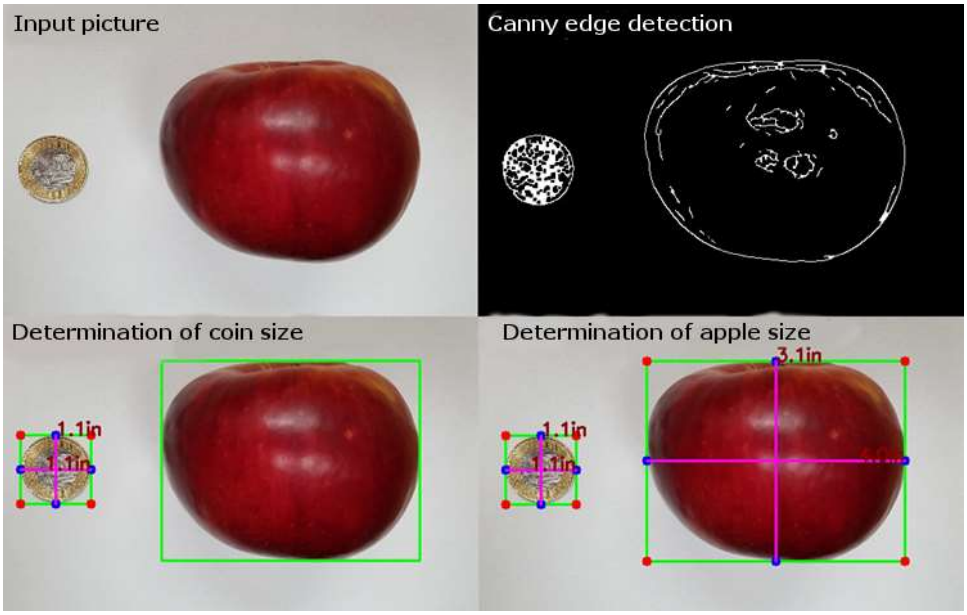

Figure 8. The process of apple size determination (Own-work)

The test requires two objects in one image. In this case, we used a 200 HUF coin with a diameter of 2.8cm or 1.1 inches. To use this method, we also need to know the resolution of the image, which is 600x450 pixels. After converting it to a grayscale image, the input image is subjected to a variety of operations such as erosion, dilation, and Canny edge detection. These play an important role in the mapping of contours, as further operations are based on them. Once the contours are created, you need to know the size of the coin and enter its value in order to determine the size of the apple in the image. Also, it is important that there must be enough space between the reference object and the other object (Rosebrock, 2016). The method has a various limitation, mainly due to the perspective way of imaging (we cannot be sure how far the object is to the reference point in Z-axis), which can be solved by utilizing an alpha channel (using a correspondent sensor and projector), describing the distance from the imaging sensor.

The next step was the classification, which should be done in multiple steps, beginning with the object itself (to determine if it's really is an apple and, in that case, what kind of apple is that), which can be followed by classifying features, like the extracted damages. However, due to the requirement of the dataset, it was only possible to perform the former analysis. The classification of the apples was performed using a basic convolutional neural network. It was important in this experiment to rely on basic, predefined settings, thus the form of the activation function and the depth of the model is not considered. After 30 epochs, the result was 93.38% based on the test dataset.

3. Conclusions

The main purpose of this article was to present the details of the process and activities of computer vision, from image acquisition to processing through an application-specific presentation. As a development opportunity, a system concept was compiled. To make our machine vision operations more professional and environment-friendly, we recommend using a Raspberry Pi-based system.

The operating system would be a Debian based, standard Raspbian Linux distribution, with Python installed by default. The suggested packages that we need, such as Google Tensorflow and OpenCV, can be installed using commands specific to the console (using packet managers). In order to achieve correct imaging, diffuse illumination is required for homogeneous lighting. The advised method is to use light modifiers with a large surface area (softbox and tripod) to illuminate the object on all sides.

A further area of study could be the segmentation and classification of injuries and texture, which would also enable us to classify apples by quality. It is now possible to create a graphical interface using the PyQt graphical framework. PyQt is a cross-platform library for developing GUIs for Python-based applications. This allows us to create an entire development environment that contains the necessary libraries, packages, and frames. In practice, the user can configure the interface in a widget-like way.

Acknowledgments

The work/publication is supported by the EFOP-3.6.1-16-2016-00022 project. The project is co-financed by the European Union and the European Social Fund.

References

Agathokleous, E., Calabrese, E.J., 2019. Hormesis can enhance agricultural sustainability in a changing world. Glob. Food Sec. 20, 150–155. https://doi.org/10.1016/j.gfs.2019.02.005

Arakeri, M.P., Lakshmana, 2016. Computer Vision Based Fruit Grading System for Quality Evaluation of Tomato in Agriculture industry. Procedia Comput. Sci. 79, 426–433. https://doi.org/10.1016/j.procs.2016.03.055

Baigvand, M., Banakar, A., Minaei, S., Khodaei, J., Behroozi-Khazaei, N., 2015. Machine vision system for grading of dried figs. Comput. Electron. Agric. 119, 158–165. https://doi.org/10.1016/j.compag.2015.10.019

Beyaz, A., Martínez Gila, D.M., Gómez Ortega, J., Gámez García, J., 2019. Olive fly sting detection based on computer vision. Postharvest Biol. Technol. 150, 129–136. https://doi.org/10.1016/j.postharvbio.2019.01.003

Bhargava, A., Bansal, A., 2018. Fruits and vegetables quality evaluation using computer vision: A review. J. King Saud Univ. - Comput. Inf. Sci. https://doi.org/10.1016/j.jksuci.2018.06.002

Brosnan, T., Sun, D.-W., 2004. Improving quality inspection of food products by computer vision—a review. J. Food Eng. 61, 3–16. https://doi.org/10.1016/S0260-8774(03)00183-3

Chinchuluun, A., Xanthopoulos, P., Tomaino, V., Pardalos, P.M., 2010. Data Mining Techniques in Agricultural and Environmental Sciences. Int. J. Agric. Environ. Inf. Syst. 1, 26–40. https://doi.org/10.4018/jaeis.2010101302

Dinesh, S., Ramya, G., Menaga, S., 2017. A Computer Vision Based Diseases Detection and Classification in Apple Fruits. Int. J. Eng. Res. V6, 161–165. https://doi.org/10.17577/IJERTV6IS100073

Dubey, S.R., Jalal, A.S., 2013. Adapted approach for fruit disease identification using images. Image Process. Concepts, Methodol. Tools, Appl. 3–3, 1395–1409. https://doi.org/10.4018/978-1-4666-3994-2.ch069

Gollapudi, S., 2019. Learn computer vision using OpenCV : with deep learning CNNs and RNNs.

Habib, M.T., Majumder, A., Jakaria, A.Z.M., Akter, M., Uddin, M.S., Ahmed, F., 2018. Machine vision based papaya disease recognition. J. King Saud Univ. - Comput. Inf. Sci. 0–9. https://doi.org/10.1016/j.jksuci.2018.06.006

Joe Minichino, J.H., 2015. Learning OpenCV 3 Computer Visionwith Python Second Edition.

Kamilaris, Andreas, Prenafeta-Boldú, F.X., 2018. Deep learning in agriculture: A survey. Comput. Electron. Agric. 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

Kamilaris, A., Prenafeta-Boldú, F.X., 2018. A review of the use of convolutional neural networks in agriculture. J. Agric. Sci. 156, 312–322. https://doi.org/10.1017/S0021859618000436

Kawano, Y., Yanai, K., 2014. Food image recognition with deep convolutional features. UbiComp 2014 - Adjun. Proc. 2014 ACM Int. Jt. Conf. Pervasive Ubiquitous Comput. 589–593. https://doi.org/10.1145/2638728.2641339

Khan, S., Rahmani, H., Shah, S.A.A., Bennamoun, M., 2018. A Guide to Convolutional Neural Networks for Computer Vision, Synthesis Lectures on Computer Vision. https://doi.org/10.2200/S00822ED1V01Y201712COV015

Kim, J., Kim, B.-S., Savarese, S., 2012. Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines. Appl. Math. Electr. Comput. Eng. 133–138.

Kumari, N., Bhatt, D.K., Dwivedi, R.K., Belwal, R., 2019. Performance Analysis of Support Vector Machine in Defective and Non Defective Mangoes Classification 1563–1572.

Marzoa Tanco, M., Tejera, G., Di Martino, M., 2018. Computer Vision based System for Apple Detection in Crops, in: Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. SCITEPRESS - Science and Technology Publications, pp. 239–249. https://doi.org/10.5220/0006535002390249

Momin, M.A., Yamamoto, K., Miyamoto, M., Kondo, N., Grift, T., 2017. Machine vision based soybean quality evaluation. Comput. Electron. Agric. 140, 452–460. https://doi.org/10.1016/j.compag.2017.06.023

Naik, S., Patel, B., 2017. Machine Vision based Fruit Classification and Grading - A Review. Int. J. Comput. Appl. 170, 22–34. https://doi.org/10.5120/ijca2017914937

Padol, P.B., Yadav, A.A., 2016. SVM classifier based grape leaf disease detection, in: Conference on Advances in Signal Processing, CASP 2016. pp. 175–179. https://doi.org/10.1109/CASP.2016.7746160

Pal, M., 2005. Random forest classifier for remote sensing classification. Int. J. Remote Sens. 26, 217–222. https://doi.org/10.1080/01431160412331269698

Patrício, D.I., Rieder, R., 2018. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. Comput. Electron. Agric. 153, 69–81. https://doi.org/10.1016/j.compag.2018.08.001

Prateekvjoshi, 2014. Understanding Gabor Filters [WWW Document].

Resende Silva, B.V., Cui, J., 2017. A Survey on Automated Food Monitoring and Dietary Management Systems. J. Heal. Med. Informatics 08. https://doi.org/10.4172/2157-7420.1000272

Rosebrock, A., 2016. Measuring size of objects in an image with OpenCV [WWW Document].

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C., 2016. Deepfruits: A fruit detection system using deep neural networks. Sensors (Switzerland) 16. https://doi.org/10.3390/s16081222

Shastry, K.A., Sanjay, H.A., Deexith, G., 2017. Quadratic-radial-basis-function-kernel for classifying multi-class agricultural datasets with continuous attributes. Appl. Soft Comput. J. 58, 65–74. https://doi.org/10.1016/j.asoc.2017.04.049

Shidnal, S., Latte, M. V., Kapoor, A., 2019. Crop yield prediction: two-tiered machine learning model approach. Int. J. Inf. Technol. https://doi.org/10.1007/s41870-019-00375-x

Sinha, P.K., 2012. Image Acquisition and Preprocessing for Machine Vision Systems, Image Acquisition and Preprocessing for Machine Vision Systems. SPIE. https://doi.org/10.1117/3.858360

Snyder, W.E., Qi, H., 2017. Fundamentals of Computer Vision. Cambridge University Press. https://doi.org/10.1017/9781316882641

Sofu, M.M., Er, O., Kayacan, M.C., Cetişli, B., 2016. Design of an automatic apple sorting system using machine vision. Comput. Electron. Agric. 127, 395–405. https://doi.org/10.1016/j.compag.2016.06.030

Spizhevoy, A., Rybnikov, A., 2018. OpenCV3: Computer Vision with Python Cookbook, Packt Publishing. Packt Publishing.

Su, Q., Kondo, N., Li, M., Sun, H., Al Riza, D.F., 2017. Potato feature prediction based on machine vision and 3D model rebuilding. Comput. Electron. Agric. 137, 41–51. https://doi.org/10.1016/j.compag.2017.03.020

Tarjan, L., Šenk, I., Tegeltija, S., Stankovski, S., Ostojic, G., 2014. A readability analysis for QR code application in a traceability system. Comput. Electron. Agric. 109, 1–11. https://doi.org/10.1016/j.compag.2014.08.015

Tou, J.Y., Lau, P.Y., Tay, Y.H., 2007. Computer Vision-based Wood Recognition System. Proc. Int'l Work. Adv. Image Technol.

Van der Stuyft, E., Schofield, C.P., Randall, J.M., Wambacq, P., Goedseels, V., 1991. Development and application of computer vision systems for use in livestock production. Comput. Electron. Agric. 6, 243–265. https://doi.org/10.1016/0168-1699(91)90006-U

Zhang, B., Huang, W., Gong, L., Li, J., Zhao, C., Liu, C., Huang, D., 2015. Computer vision detection of defective apples using automatic lightness correction and weighted RVM classifier. J. Food Eng. 146, 143–151. https://doi.org/10.1016/j.jfoodeng.2014.08.024