Digital Picture Processing for Avocado Segmentation and Recognition

Jhoao Alejandro Martinez[1,2], Pedro Luis Florez [1,2]

[1]Engineering Faculty, Universidad Tecnológica de Pereira
[2]Masters on Computer Science, Universidad Tecnológica de Pereira

Abstract

During the implementation of this project the main goal was to employ an unlabeled datset for the task of image segmentation and object identification on pictures outside of the training sample following similar characteristics. This implied the implementation of a labeling algorithm to generate the polygons corresponding to the avocados associated with the pictures of each dataset as well as the training of the YOLOv8-Seg model employing the generated polygon dataset.

Introduction

Image processing is a valuable tool for supporting automated processes regarding maturity assessment, classification and non-invasive disease identification in fruits. During the realization of this project, a tool was conceived to aid in the identification of avocados on images; this could be used in future projects as a module for detection, segmentation and classification; or, as well, might be used as a base for other classification tasks, which will represent a reduction in time compared to having to start an iteration from scratch.

Project Development : Segmentation

Data & Processing

This dataset was acquired from a classification project between three different avocado classes, which had a significant similairty with the expected properties of Haas avocado [1].

This dataset consists of 1195 pictures of three avocado maturity levels, to know:

- Mentah: Unripe
- Matang: Ripe
- Terlalu matang: Over-ripe

These pictures were already prepared through a process of data augmentation, and had predefined bounding boxes which were compatible with YOLO models for object detection.

However, for the purposes of this project, this dataset proved insufficient, and, as such, another dataset was generated using polygons which more accurately represent the space occupied by avocados in the picture. In order to perform this process, digital image processing techniques were employed and the corresponding dataset was stored and used for training the model.

Figure 1: *Random avocado image selected from the testing dataset.*

This process followed these steps, which will be illustrated along with the corresponding pictures to understand the process, starting with the original image (in this case it will be Figure 1) and all the way to the resulting polygon.

[1] https://universe.roboflow.com/ibnu-h3cda/avocado-g9fsl

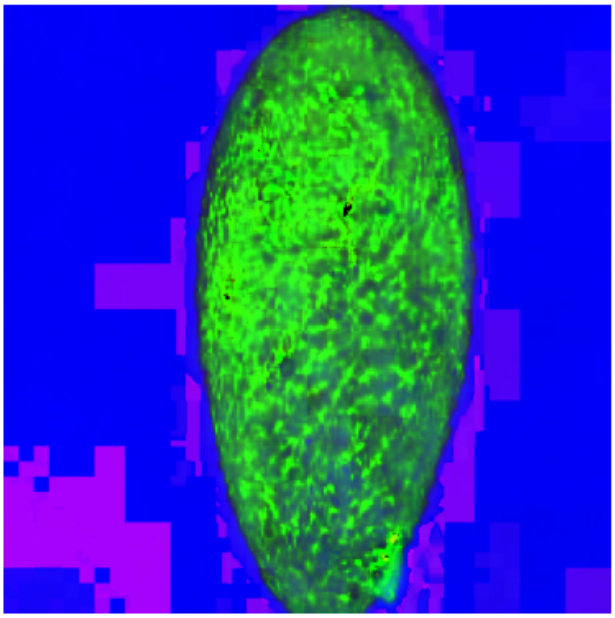

Figure 2: *Avocado sample image in HSV format.*

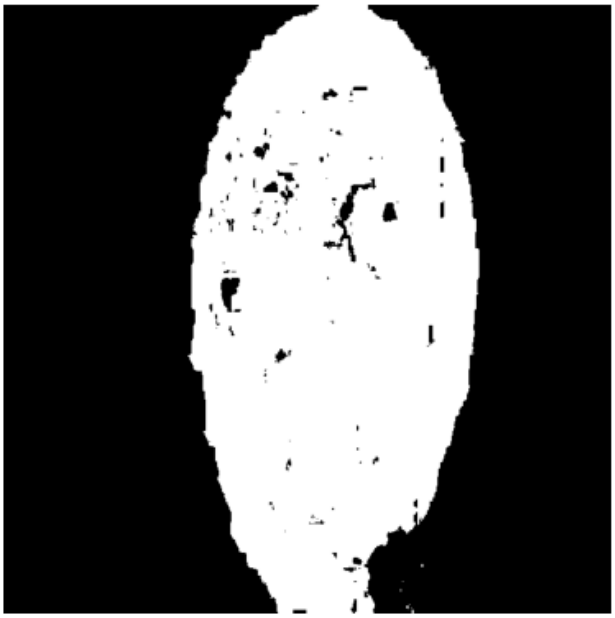

Figure 3: *Avocado sample image in HSV format after filtering.*

Convertion to HSV

Before identifying the avocado, it is important to first apply an HSV transformation to the image. HSV stands for Hue, Saturation, Value. It separates color (hue) from intensity (value).

- H (Hue): Type of color (angle on the color wheel, 0–179 in OpenCV). Which "pure" color we are examining. For example, all shadows and tones of the color "red" will have the same Hue.
- S (Saturation): Intensity or purity of the color (0–255). How "white" the color is. A fully saturated color would be "pure," as in "pure red." And a color with zero saturation would be pure white.
- V (Value): Brightness of the color (0–255). The Value allows to control the lightness of color. A Value of zero would indicate pure black, whereas increasing the value represents lighter colors.

According to the official OpenCV documentation [1]. This library was heavily relied on during the implementation of this project. An image that passes through this transformation shows a clear distinction between foreground and background, as seen in Figure 2, in the case of pictures with the conditions of the ones used during the development of this project.

Region isolation

After getting the picture in HSV format, the most extreme values, which would be outside of the expected range of what constitutes an avocado, are turned into black pixels, whilst the pixels inside the selected range will be colored white, as seen in Figure 3. This step is used to create a clear distinction between what is and isn't an avocado in the picture.

Afterwards, a Close filter is applied, which corresponds to Dilation followed by Erosion.

These operations being defined as follows:

Erosion The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). A kernel slides through the image (as in a 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

So what happends is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises, detach two connected objects etc.

Dilation The opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases.

Figure 4: *Avocado sample image in raw HSV format after applying a Close filter.*

Figure 5: *Avocado sample image in raw HSV format after applying an Open filter.*

It is useful in closing small holes inside the foreground objects, or small black points on the object [2]. This is made evident on the output result in Figure 4.

Afterwards, an Open filter is applied. Which, contrary to the Close filter, performs Erosion followed by Dilation. This process allows to remove the noise in the outer region of the identified silhouette, as can be seen in the lower area of Figure 5.

Once this separation has been performed, identifying the silhouette is easy, as it is only necessary to return the outer contour of the white block which corresponds to the avocado silhouette. This filtering is quite precise but not complete, as shown in Figure 6. It is however a practical example of how it is simple to estimate a close approximation to the borders of an image when the foreground and background are clearly differenciated, which is a characteristic of the chosen dataset.

Convex hull

Is clear that, after obtaining the polygon, there are regions which correspond to an avocado that aren't being treated as such. With the purpose of correcting this situation a convex hull is applied.

In geometry, the convex hull of a shape is the smallest convex set that contains it. The convex hull may be defined either as the intersection of all convex sets containing a given subset of a Euclidean space, or equivalently as the set of all convex combinations of points

Figure 6: *Resulting outer contour laid over the original image.*

Figure 7: *Convex Hull of the original output polygon.*

Figure 8: *Approximated polygon from Convex Hull.*

in the subset. For a bounded subset of the plane, the convex hull may be visualized as the shape enclosed by a rubber band stretched around the subset [3]. In the scope of the current project, it "fills out" the indent in the original polygon, which can be better appreciated in Figure 7.

Ramer–Douglas–Peucker algorithm

The Ramer–Douglas–Peucker algorithm, also known as the Douglas–Peucker algorithm and iterative end-point fit algorithm, is an algorithm that decimates a curve composed of line segments to a similar curve with fewer points. It was one of the earliest successful algorithms developed for cartographic generalization.

In this project it is used to simplify the resulting Convex Hull into a polygon which is less cost intesive for the training stage of the YOLOv8 model which will receive the coordinates of the points composing the polygon as labels in its training input. The result is similar whilst consisting on less edges in the final polygon.

The criteria for choosing the edges of the resulting polygon follows an analysis purely based on geometry, which will be explained below.

Parting from an understanding that a Convex Hull is composed of an array of edges, or points as follows:

$$P_0, P_1, P_2, ..., P_n \tag{1}$$

With the last point and the first being the same. Then, the perimeter of the convex Hull is defined as:

$$L = \sum_{i=0}^{n-1} \sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2} \tag{2}$$

Then, with an ϵ, defined through experimentation as 0.003, this value L is multiplied and produces a threshold, such as, if a point has a perpendicular distance to the line between the neighboring previous and next adjacent points which is greater than the threshhold, this point will not be used to construct the final polygon. The end result of this operation can be appreciated more clearly in Figure 8.

This final polygon will be used as labels to train the segmentation-baseb object recognition model during the next stage.

Project Development : Model Training

The moel chosen for this project is YOLOv8-Seg, a model that extends the core YOLO detection framework by incorporating a dedicated segmentation head capable of predicting pixel-level masks in addition to bounding boxes. This capability is particularly advantageous in applications requiring precise object delineation, where shape, surface area, or defect localization must be determined with high spatial accuracy. The

Table 1: *Hyperparameters used for training the YOLO model.*

Epochs	120
Batch Size	16
Image size	640
Learning Rate	0.001
Patience	20
Optimizer	adam
Augment	true

decision to use the segmentation variant rather than a pure detection model was therefore motivated by the need to generate bounding polygons for each avocado instance—data instead of limiting to a 4 edged bounding box detection; which is a less specialized implementation and, as such, was not considered.

The training hyperparameters are as follow:

To train the model, a dataset consisting of labeled avocado images was prepared following the YOLO segmentation format. All training images were standardized to a resolution of 640 × 640 pixels to maintain input consistency and leverage the optimized performance profile of the YOLOv8 architecture at this scale. Importantly, although the training corpus uses fixed-size images, the final model is not restricted to this resolution. YOLOv8 employs fully convolutional operations, enabling inference on images of varying sizes without architectural modification. This flexibility is essential for real-world deployment, where image dimensions may differ substantially from those used during training.

Training was performed using the Ultralytics Python API, and configuration files define both dataset paths and hyperparameters, while the training loop initializes a pretrained YOLOv8-Seg "nano" model (yolov8n-seg.pt) and fine-tunes it on the avocado segmentation dataset sourced from Roboflow. This transfer-learning approach reduces training time and improves performance by starting from a model already optimized on large-scale segmentation datasets.

Results

The training process produced the performance curves shown in Figure 10, which summarizes the evolution of the YOLOv8-Seg model's loss functions and evaluation metrics over the course of 50 epochs. These curves provide insight into how effectively the model learned to localize and segment avocado instances, as well as the degree of generalization achieved on the validation set.

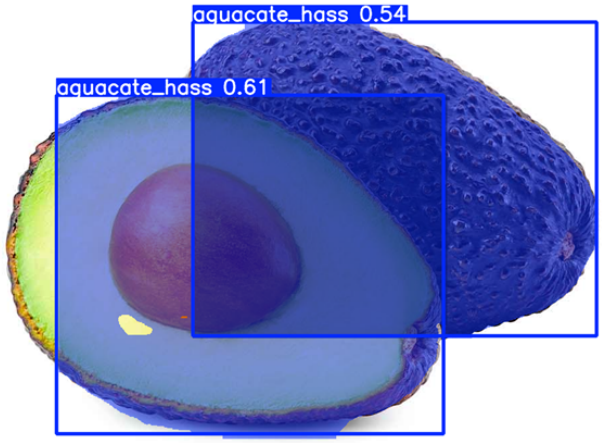

Figure 9: *Result of applying the model on an unrelated Haas avocado iamge outside of the scope of the project (open)*

The left section displays the loss curves for bounding box regression (box_loss), segmentation mask prediction (seg_loss), classification (cls_loss), and distribution-focal loss (dfl_loss). They generally exhibit a smooth, monotonic downward trend, indicating stable optimization behavior throughout training. The reduction in segmentation and bounding box losses is particularly important, as these directly influence the model's ability to produce precise contours for the avocados. The gradual decline of the DFL loss further suggests that the model becomes increasingly confident in predicting accurate bounding box distributions.

The collective behavior of the curves shows that the YOLOv8-Seg model converged efficiently and exhibited excellent generalization capability. The combination of low, stable validation losses and near-maximum precision/recall scores (as shown in the right side of Figure 10) suggests that the model successfully learned a compact and discriminative feature representation for avocado segmentation. These results also validate the suitability of the chosen hyperparameters and confirm that the training dataset provides sufficient variability for the model to develop robust segmentation performance.

Furthermore, during an edge case evaluation displayed in Figure 9, with an image which displays strange and not introduced behavior (an open avocado), its shown the bounding box is able to estimate accurately the space which the avocado would occupy behind. Additionally, it is able to fill the space of the avocado shell with accuracy, although, as expected, it would struggle with the clearer boy of an open avo-

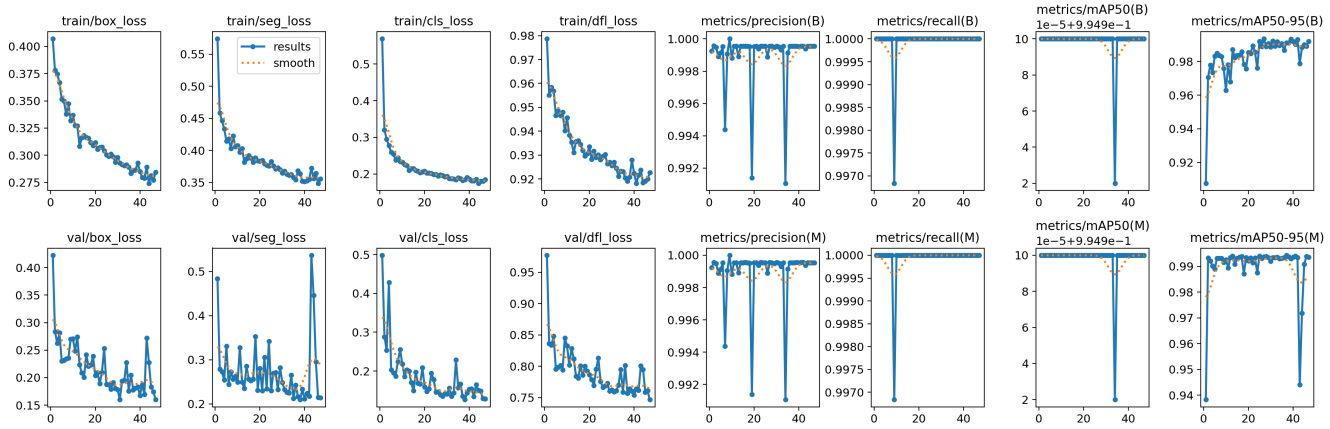

Figure 10: *Training results generated after final model epoch.*

cado.

Model Limitations

Although the trained YOLOv8-Seg model demonstrates high accuracy within the scope of this project, its applicability is subject to several constraints:

- **Limited to avocados**: The model is trained exclusively on images containing Hass avocados and therefore cannot reliably detect or segment other fruit types or objects.
- **Dependence on contrasting backgrounds**: Optimal performance is achieved when avocados appear against backgrounds with sufficient color contrast. Low-contrast scenes, cluttered environments, or backgrounds with similar hue may reduce segmentation quality.
- **Sensitivity to lighting conditions**: Extreme shadows, reflections, or strong highlights can distort color-based features, leading to inaccurate masks.
- **Scale and viewpoint constraints**: The model performs best when avocados occupy a reasonable portion of the frame. Very small objects, partially occluded avocados, or extreme viewing angles can negatively affect detection and segmentation.
- **Single-class limitation**: Since the model is trained on a single class, it cannot distinguish between ripeness stages or different avocado varieties without additional training.

References

[1] Moukthika. *Color spaces in opencv*. Apr. 2025. URL: https://opencv.org/blog/color-spaces-in-opencv/#h-hsv-hue-saturation-value-color-space.

[2] URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html.

[3] Ky Fan. "Convex sets and their applications. Lecture notes". In: *Applied Mathematics Division, Argonne National Laboratory, Lemont, IL* (1959).