Published: 01 September 2024

RESEARCH-ARTICLE

Developing an OpenCV-Based App for Segmenting Cells in Tomato Fruit Tissue

JORGE DANIEL QUINTEROS, The College of Wooster, Wooster, OH, United States

LEI ZHANG, University of Georgia, Athens, GA, United States

ESTHER VAN DER KNAAP, University of Georgia, Athens, GA, United States

SOFIA VISA, The College of Wooster, Wooster, OH, United States

Journal of Computing Sciences in Colleges, Volume 40, Issue 4 (September 2024)
https://doi.org/10.5555/3715638.3715666
EISSN: 1937-4763

Developing an OpenCV-based App for Segmenting Cells in Tomato Fruit Tissue*

Jorge Daniel Quinteros[1], Lei Zhang[2],
Esther van der Knaap[2], Sofia Visa[1]
[1]Mathematical and Computational Sciences Department
College of Wooster
Wooster, OH 44691
`{jquinteros25,svisa}@wooster.edu`
[2]Department of Horticulture
University of Georgia
Athens, GA 30602, USA
`{Lei.Zhang,esthervanderknaap}@uga.edu`

Abstract

We present an OpenCV-based app for segmenting and measuring cells in tissue images, and we illustrate its application for generating hypotheses in genetics research. The app is used to process thousands of tissue images to identify genotypes with large cell size. These selected genotypes will be further studied to identify genes responsible for cell size.

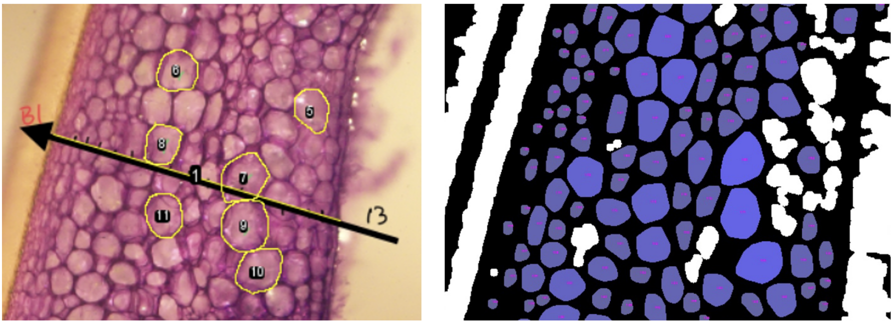
1 Introduction

Discovering what genes regulate size and shape in tomato fruit is important because it helps breeders to create new plant crosses for various consumer needs. For instance, small tomato fruit varieties like the cherry tomatoes are appreciated for salads and snacking, medium and meatier sized ones like the

---

Roma tomatoes are used for caning and sauces, while very large beefsteak tomatoes are used for sandwiches.

Recent advancements [4], [1], [8], [6] have identified several size and shape genes, however the developmental stage at which these genes are expressed is still unknown. Ongoing research [3] addresses this by generating microscopic images of tissue in the tomato fruit ovary at various developmental time points and by evaluating the cell size to identify how early these genes are expressed. Using OpenCV [2] and Python scripting, we have developed an app for automating this laborious cell tissue evaluation. Precisely, our tool detects cells, labels them, and computes their corresponding areas. While this tool is specifically developed for ovary and pericarp tissue in tomato fruit, it can be applied to other tissue types and other plants.



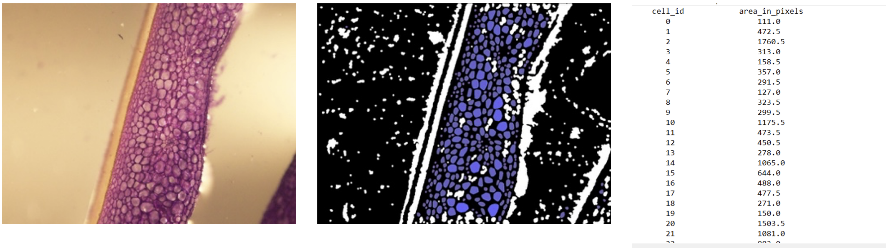
Manual (expert) segmentation

Labeled automated segmentation

Figure 1: Manual (expert) segmentation of top seven largest pericarp cells (left) and automated segmentation of pericarp cells (right).

Currently, our geneticist collaborators hand-circle cells with the help of a general-purpose image analysis software like ImageJ [5]. Figure 1 illustrates seven cells manually annotated by a human expert. This is tedious, time consuming, and error-prone work as many times the human expert has a difficult time deciding the next largest cell among many cells of similar size. In this case, the expert simply chooses one, and moves along with this tedious work. Our tool, exemplified in Figures 1 and 2, speeds up this stage of researching genes regulating size and shape, by allowing to rapidly compare hundreds or thousands of tissues to identify the most likely time-point at which these size and shape genes are expressed.

There are several image processing tools available like the macro in ImageJ, for instance, but this macro requires high contrast images and therefore does not work well for our images which are a bit on the faded spectrum. Since working on this project, another similar software was published, namely Cell-



Input: Original pericarp image

Output: cells detected

Output: text file with cell areas

```
cell_id          area_in_pixels
   0                 111.0
   1                 472.5
   2                 1760.5
   3                 313.0
   4                 158.5
   5                 357.0
   6                 291.5
   7                 127.0
   8                 323.5
   9                 299.5
   10                1175.5
   11                473.5
   12                450.5
   13                278.0
   14                1065.0
   15                644.0
   16                488.0
   17                477.5
   18                271.0
   19                150.0
   20                1503.5
   21                1081.0
```

Figure 2: The app outputs an image with labeled cells (middle picture) and a text file listing the cell areas (right picture).

pose [7]. This tool uses deep learning, and it is trained on 540 images. At a first glimpse, Cellpose works well on a variety of tissues including ours, but our software processes images faster than Cellpose.

2 Development of the OpenCV app for cell segmentation

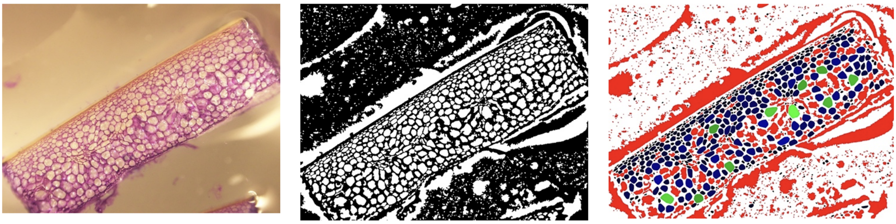
As illustrated in Figure 2, the app reads a pericarp tissue image of a mature green fruit and generates two outputs, namely an image with labeled segmented cells and a text file listing the cells and their corresponding areas. If the researcher is interested in processing only the top 10% largest cells, they will sort the cells in decreasing order of their size and select only the top 10% part of the text file. Alternatively, one can run the app requesting the segmentation of a specific number of largest cells. For instance, Figure 3 shows an output of 10 largest cells in a pericarp tissue.

Our cell detection and measurement app is written in Python and uses extensively OpenCV functionality. The app accepts .tif images but also works with other image types recognized by OpenCV, a free and open-source library of functions for real-time computer vision [2]. The main five steps of the app are included below and illustrated in Figure 3.

1. **Convert input image to RGB, and extract grayscale Color Channel.**
2. **Extract local maxima to capture cells using a Gaussian Threshold Noise-resistant Peak.** Here we apply cv.GaussianBlur(), a Gaussian filter to blur out sharp edges. Then we select those pixels from the original grayscale image with a value lower than the blurred image. In other words, we select many more core white pixels situated in the mid-

dle of the cell and fewer pixels situated at the edge of the cell. This step produces an OpenCV mask of the image.

3. **Fill holes in Cell Mask with morphological operations.** Here we use cv.floodFill(), a floodfill technique that removes holes inside of a cell. In brief, we start with one black background pixel and set to white all the reachable background pixels. In this new mask, the only remaining black pixels are the holes in the cells. Next, we do an OR operation between the mask from step 2 and the negative of the floodfilled mask to obtain a new mask that includes all the cells without any hole.
4. **Extract individual cells using the OpenCV cv.findContours() routine.**
5. **Filter cells with OpenCV contour properties** – in this step we remove cells that are intersecting the margin of the image, and we use the solidity measure from OpenCV to remove concave objects and long line objects, as they usually are margins of the tissue.
6. **Output the annotated image and write into a text file the cell area.**



Original image

The mask after filling holes in the cell (mask after step 3)

The red cells are filtered out; the green cells are the top 10 largest cells (mask after step 5)

Figure 3: Various steps in our OpenCV app for detecting tissue cells and computing their areas.

The OpenCV app described above is run in command line and accepts three parameters:

1. The folder with all the images to be segmented.
2. A scale if using one; this is given as a number and the program uses it defined as the number of pixels per unit.
3. The number of largest cells to highlight/detect in the image; if this parameter is missing, the app detects all cells.

Currently, we are working on creating a user-friendly tkinter graphical user interface for the app.

3 Comparing segmentations obtained with the OpenCV app and the human expert

With an automation tool in place, next, we focus on evaluating if its output is comparable to the human expert one. For this, we compare the areas of the eight largest cells computed via the two methods. As shown in Figures 4, 5, 6, we use three different sections from the same fruit, namely images 4360, 4361, and 4363. In these figures, for each of the eight cells shown on the x-axis the three bars correspond to areas in square millimeters computed (1) by the human expert; (2) by the app; and (3) by the app corrected by a 1.7 multiplicative factor, respectively.

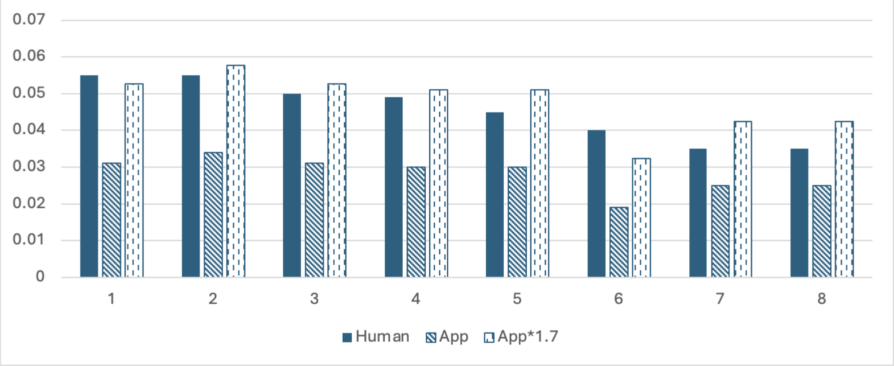


Figure 4: Cell area comparison for the eight largest cells in the first section of a single fruit (image 4360). For each cell, the three bars correspond to area computed by the human expert, by the app, and by the app*1.7, respectively.

Our app consistently underestimates the cell areas because it creates thicker intercellular walls visible in output images such as the ones shown in Figures 2 and 3. However, the areas computed by the app follow the same trend as the ones computed by the human expert. Further, by multiplying the app calculations with a corrective factor of 1.7, we observe that these new estimates approach the human expert evaluations – see Figures 4, 5, 6. From a genetics expert point of view, we have two remarks about this result:

1. **The human experts are not an absolute truth**, they too have biases in selecting by eye the largest cells, and at times, could not decide the largest of multiple cells of seemingly similar size. In addition, there are various biases between two or more human experts in segmenting these tissues, consistency of evaluation being one of them.

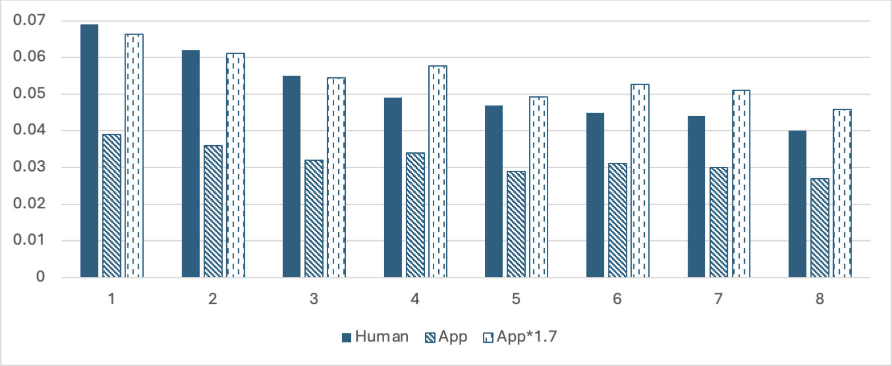


Figure 5: Cell area comparison for the eight largest cells in the second section of a single fruit (image 4361). For each cell, the three bars correspond to area computed by the human expert, by the app, and by the app*1.7, respectively.

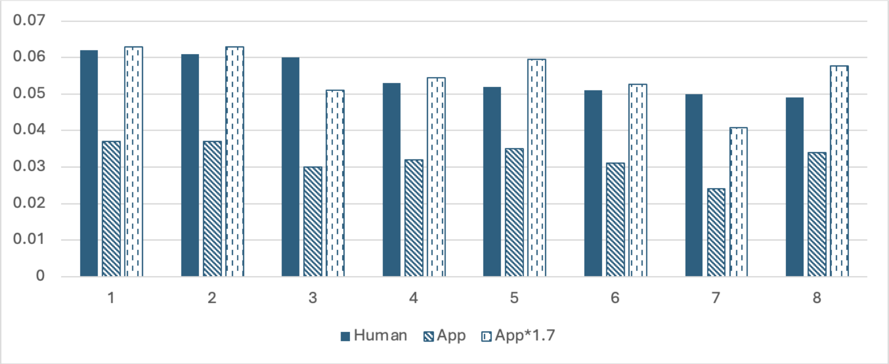


Figure 6: Cell area comparison for the eight largest cells in the third section of a single fruit (image 4363). For each cell, the three bars correspond to area computed by the human expert, by the app, and by the app*1.7, respectively.

2. **Both, the app and the human expert generate same hypothesis.** This automated model is used to generate hypotheses that can be further researched, so its main purpose is to analyze efficiently hundreds or thousands of tissues to select candidates for additional genetic studies. For instance, let's assume that the task is to compare tissues from varieties A and B to decide which one develops larger cells. If the human expert evaluates the average cell area of the top ten largest cells to 0.04 and 0.09, respectively, this will direct the researcher to study the variety B for identifying genes that regulate cell size. However, the automated model arrives to the same hypothesis as both average cell areas are smaller by a factor of 1.7, i.e. 0.024 for variety A and 0.053 for variety B.

4 Generating hypotheses with the OpenCV app

The next example illustrates one way in which our collaborators from Univ. of Georgia use this app to identify worthy hypotheses that are further investigated via more time-consuming lab experimentations. In this example, our plant geneticist collaborators utilized a tomato population 20S87 with plants mostly identical in their genome. Only a small region containing a few genes are different between those noted as genotypes A, B, and H. The genotype A plants carry larger fruits than genotypes B and H plants. The hypothesis is that the larger fruits from genotype A is caused by larger cell size. If such hypothesis is proven, we may further deduce that one of the genes in the differed genomic region regulate cell size.

Table 1: The 135-image data for population 20S87 are obtained from 3 genotypes x 5 plants x 3 fruit x 3 sections

Tissue name	Plant number "pl"	Fruit number "fruit"	Section number "sect"
A-20S87-pl-fruit-sect	17, 23, 29, 31, 33	1, 2, 3	1, 2, 3
B-20S87-pl-fruit-sect	20, 26, 27, 30, 66	1, 2, 3	1, 2, 3
H-20S87-pl-fruit-sect	07, 15, 16, 19, 35	1, 2, 3	1, 2, 3

To compare the cell size in tomato fruits, pericarp tissue (flesh tissue) was sectioned from each fruit and imaged under microscope. The geneticists provided us with 135 images (3 genotypes * 5 plants * 3 fruit * 3 sections). For each genotype, we collect fruits from five different plants denoted by "pl" in Table 1. From each plant we collect three different fruits, labeled as "fruit" in the table, and from each fruit we dissect three pericarp sections to view under microscope.

The research question we are interested in answering is: *which (if any) of these three genotypes carry a gene allele that produces large cells?* To answer this, we use the OpenCV app for identifying the 10 largest cells in every image and then compute their areas. Next, for each genotype, we compute with a Python script the per-plant minimum, average, and maximum areas (in pixels), as shown in Figure 7. On this figure we observe that genotype A-20S87 has the largest cells while genotype H-20S87 has the smallest cells; however, a statistical test will tell how relevant these differences are.

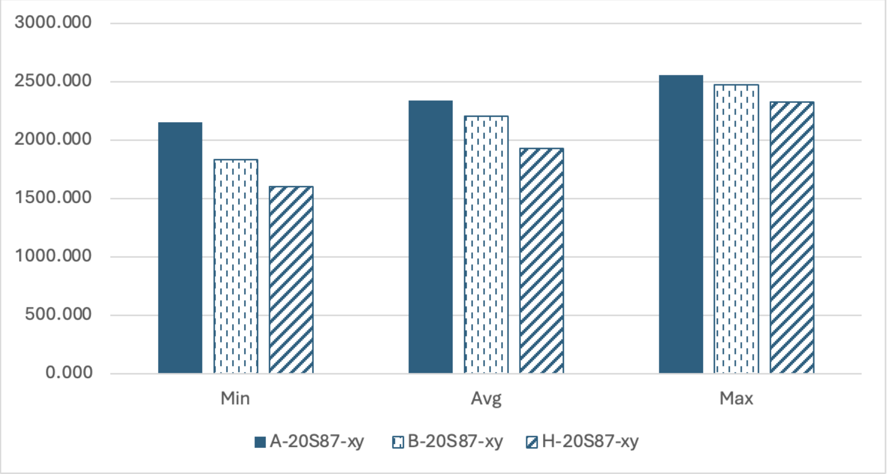


Figure 7: Per-genotype minimum, average, and maximum area; genotype A-20S87 has the largest top 10 cells.

Table 2: Pairwise ANOVA between A, B and H genotypes, analysis of the 10 largest cells; The A vs H experiment rejects the null hypothesis meaning there is a significant difference between the cell areas in these two genotypes, namely there are larger cells in genotype A than in H.

ANOVA statistics for the OpenCV app	A and B	**A and H**	B and H
F	0.993	**8.881**	2.670
p	0.348	**0.017**	0.140

Table 2 shows the pairwise results of an ANOVA analysis for areas of these three genotypes. Based on F and p values, the two genotypes to be further explored are A and H. Namely, ANOVA for these two entities rejects the null hypothesis, meaning there is a significant difference between the cell areas in genotypes A and H. This further implies that the dominant allele in A creates significantly larger cells than the allele in H, thus answering the question posed above. Therefore, the gene allele or mutation that will be researched further is the one in genotype A.

Table 3: Pairwise ANOVA between A, B and H genotypes, analysis of the 100 largest cells; Only the A vs H experiment rejects null hypothesis meaning there is a significant difference between the cell areas in these two genotypes, namely there are larger cells in genotype A than in H.

ANOVA statistics for the OpenCV app	A and B	**A and H**	B and H
F	4.753	**10.831**	2.493
p	0.060	**0.011**	0.152

Table 4: Pairwise ANOVA between A, B and H genotypes, analysis of all cells; none of the three comparisons reject the null hypothesis.

ANOVA statistics for the OpenCV app	A and B	A and H	B and H
F	4.452	4.001	2.915
p	0.067	0.080	0.092

Table 3 summarizes the ANOVA analysis when looking at the largest 100 cells. In this experiment, again, we observe that the largest cells are in genotype A and the smallest ones in H. In fact, the same hypothesis is generated as in the 10 largest cells experiment, but with even better statistics: a larger F-value and a lower p-value. However, when repeating the experiment using all cells (see Table 4), this hypothesis is no longer evident. Namely, while the F-value of 4.001 might suggest some differences between the cell sizes from the two genotypes, the p-value of 0.080 is larger than the confidence threshold of 0.05 to be able to reject the null hypothesis. At a second look, this result is not surprising if considering the average per-plant cell count plotted in Figure 8. On average, genotype A has the larger number of cells (an average of 582 per plant), and genotype H has the fewest number of cells (463 cells per plant).

This suggests the following two hypotheses which are evident in the tissue samples from Figure 9:

1. Genotype A consists of many large cells and of many small cells ready to grow.
2. Genotype H has fewer cells in general, and these cells are of average size.

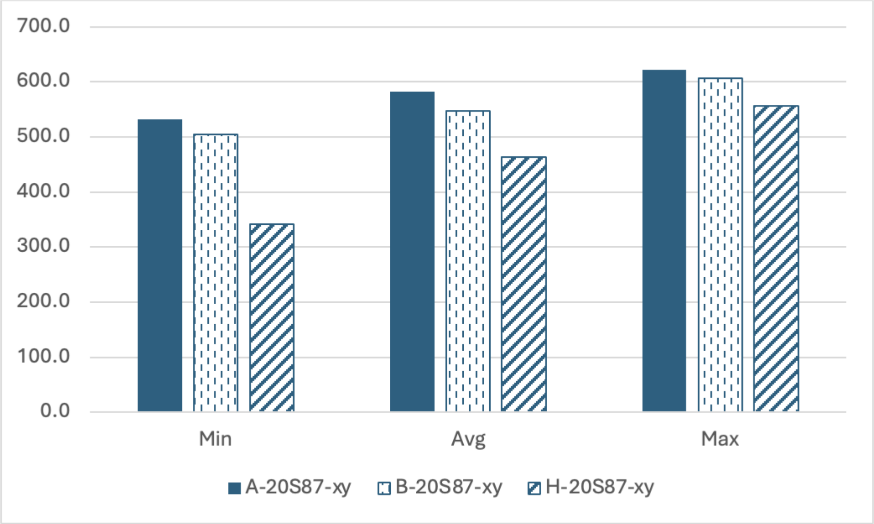


Figure 8: Per-plant (x-axis) minimum, average, and maximum number (y-axis) of cells; on average, genotype A-20S87 has the largest number of cells (582 per plant) and genotype H-20S87 has the least number of cells (463 cells per plant).

When we incorporate all cells in the ANOVA analysis, we work with overall similar total tissue areas in both genotypes A and H. Further, under the two hypotheses described above, the cell area average in the entire genotype A tissue can be similar to the one in genotype H. For instance, an average of 4 can be generated by cell areas of 7, 7, 2, 2, 2, an A-like genotype area distribution with large variance of cell sizes, some cells being large and others being small. However, an average of 4 can also be generated by cell areas of 4, 4, 4, 4, an H-like genotype area distribution consisting of less cells of medium size. Figure 9 shows actual tissue examples of the two genotypes. These samples are consistent with the above analysis and hypotheses. Also, as the cells get smaller, they are harder to segment from the background in the image. Therefore, including more of the cells past a certain point would decrease the effectiveness of the experiment – this is another deterrent for using all cells in the tissue for this type of tissue comparison.

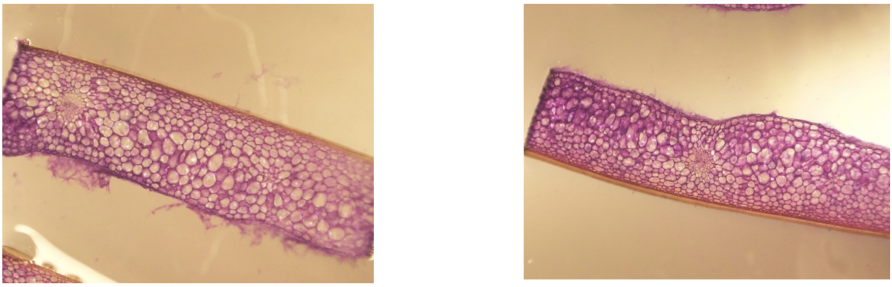


Figure 9: Tissue samples from genotypes A-20S87 (left) and H-20S87 (right).

The above analysis leads to an important observation. Namely, **to generate hypotheses about tissue cell-size, it is best to compare the top 10% or 20% largest cells in the tissues;** it is *not* recommended to compare all cells in the tissues.

Another important note is that, when this type of tissue comparison is done by the human researcher, the top 10 largest cells experiment is what the human would normally do, because segmenting the 100 largest cells would be too tedious. But with this app, the largest 100 cells analysis is completed within a few minutes, demonstrating the usefulness of this app.

5 Conclusions and future work

Here we present an efficient OpenCV app written in Python which is useful in segmenting cells in fruit tissues and in computing their area. In addition, we illustrate on a real example how this app generates hypotheses to be further researched by genetics experts. With recent advancement in deep learning for image analysis, similar tools were recently developed using this powerful technology. One such tool is Cellpose developed in 2021. Our next step is to compare our tool with Cellpose, in terms of recognition power and speed of processing. We will also look at how these tools fare against the human expert in many tissue images. An initial comparison reveals that Cellpose is more accurate in evaluating cell areas, but it is much slower than our OpenCV app.

Acknowledgement: This work is funded from the research grant NSF Award 2048425.

References

[1] J. Clevenger et al. "Network analyses reveal shifts in transcript profiles and metabolites that accompany the expression of SUN and an elongated tomato fruit." In: *Plant Physiology* 168 (2015), pp. 1164–1178. DOI: `http://dx.doi.org/10.1104/pp.15.00379`.

[2] Itseez. *Open Source Computer Vision Library*. `https://github.com/itseez/opencv`. 2015.

[3] E. van der Knaap, I. Goldman, and S. Visa. *NSF Grant, Robust organ patterning by OFPs, TRMs and SUNs in plant morphogenesis*. 2020.

[4] G.R. Rodriguez et al. "Distribution of SUN, OVATE, LC, and FAS in the Tomato Germplasm and the Relationship to Fruit Shape Diversity." In: *Plant Physiology* 156 (2011), pp. 275–285.

[5] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. "NIH Image to ImageJ: 25 years of image analysis." In: *Nature Methods* 9.7 (2012), pp. 671–675. DOI: `http://dx.doi.org/10.1038/nmeth.2089`.

[6] E. Sierra-Orozco et al. "Identification and characterization of GLOBE, a major gene controlling fruit shape and impacting fruit size and marketability in tomato." In: *Hort. Res.* 138 (2021).

[7] C. Stringer et al. "Cellpose: a generalist algorithm for cellular segmentation." In: *Nature methods* 18.1 (2021), pp. 100–106.

[8] S. Wu et al. "The control of tomato fruit elongation orchestrated by sun, ovate and fs8.1 in a wild relative of tomato." In: *Plant Science* (2015), pp. 99–104.