# Update Report

Reem Al-Halimi

Nov/13/2017

## I. Accomplishments so far

1. Implemented a method to cluster disconnected stem segments into single stem instances. (discussed in detail below).
2. Added ability to create annotated composites and masks for leaves and stems .
3. Added ability to detect instances of leaves and stems.
4. Added the SLIC method as another option for instance detection thus significantly improving instance detection of stems and leaves compared to the Watershed method (discussed more below).
5. Modified the code to create a clearer pipeline so it is easier to change image processing techniques, semantic segmentation networks (still in progress), and evaluation metrics.
6. Fixed bug in drawing the bounding box of instances consisting in more than one segment (see Figures 1 and 2 below.
7. Tweaked the acceptable area size: originally segments with large areas were removed from the ground truth annotations and from predictions. This caused many annotated leaves to be ignored.

*Figure 1: Bounding Box originally generated for a partially occluded tomato consisting of two segments. Note how the box fails to encompass the whole object.*

*Figure 2: Corrected bounding box: both segments of the tomato are completely enclosed within the bounding box.*

# How does the current code compare to Matt's documented results?

In this section I compare Matt's results as reported on Github to the performance of the same code when executed on my computer for semantic segmentation and instance segmentation.

## I. Semantic Segmentation

### Reported results

the results as reported by Matt can be found here: [https://github.com/mveres01/semantic-segmentation/blob/master/docs/semantic-seg-results.pdf](https://github.com/mveres01/semantic-segmentation/blob/master/docs/semantic-seg-results.pdf) some of which are shown below. The dataset used to generate these results is not included but the code that was used to create the dataset is part of the repositories.

Train / Valid: Raw output from the segmentation network+C2:K44C3C2:KC2:K40
CRF Train / Valid: Output from the network with CRF post-processing

**Mean IOU between training and validation splits**

| | Stem | Tomato | Tom. Border | Leaf | Bg. Tomato | Bg. Stem | Bg. Leaf | Background |
|---|---|---|---|---|---|---|---|---|
| Train | 0.534 | 0.793 | 0.472 | 0.464 | 0.351 | 0.213 | 0.310 | 0.449 |
| CRF Train | 0.646 | 0.798 | 0.538 | 0.428 | 0.525 | 0.325 | 0.197 | 0.613 |
| Valid | 0.497 | 0.738 | 0.417 | **0.439** | 0.352 | 0.170 | **0.271** | 0.430 |
| CRF Valid | **0.586** | **0.746** | **0.454** | 0.370 | **0.517** | **0.255** | 0.143 | **0.590** |

**Training Set (No Post-processing)**

| | | Actual Class | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Stem | Tomato | Tom. Border | Leaf | Bg. Tomato | Bg. Stem | Bg. Leaf | Background | Precision | Recall |
| Predicted Class | Stem | 2945120 | 432 | 55609 | 3002 | 11412 | 60709 | 14104 | 36480 | 0.553 | 0.942 |
| | Tomato | 2900 | 2353763 | 310242 | 1830 | 9783 | 353 | 558 | 1906 | 0.882 | 0.878 |
| | Tom. Border | 13425 | 181642 | 1259986 | 10630 | 13574 | 545 | 1142 | 3806 | 0.503 | 0.849 |
| | Leaf | 16101 | 6516 | 65997 | 6575380 | 12797 | 8386 | 997080 | 122254 | 0.542 | 0.843 |
| | Background Tomato | 4711 | 1408 | 6733 | 4277 | 1498139 | 7712 | 14383 | 33670 | 0.352 | 0.954 |
| | Background Stem | 53294 | 40 | 3389 | 519 | 13310 | 1075411 | 10672 | 42934 | 0.209 | 0.896 |
| | Background Leaf | 127769 | 3197 | 45629 | 2581831 | 187782 | 155000 | 7369729 | 831335 | 0.365 | 0.652 |
| | Background | 2160637 | 122499 | 759195 | 2959379 | 2508689 | 3833944 | 11810724 | 20359396 | 0.950 | 0.457 |

### Results from Matt's code on my computer

These results are generated by Matt's latest code, executed on my computer. The only difference from the code used for the reported results above is that I had to download and create the dataset (also using his unchanged code). The images in the dataset may or may not be identical.

**Mean IOU between training and validation splits**

| | Stem | Tomato | Tom. Border | Leaf | Bg. Tomato | Bg. Stem | Bg. Leaf | Background |
|---|---|---|---|---|---|---|---|---|
| **Train** | 0.531 | 0.767 | 0.415 | 0 497 | 0. 449 | 0.210 | 0.329 | 0.507 |
| **CRF Train** | 0.562 | 0.781 | 0.5121 | 0.354 | 0. 539 | 0.140 | 0.019 | 0.672 |
| **Valid** | 0.470 | 0.7476 | 0.390 | 0.457 | 0.444 | 0.176 | 0.279 | 0.511 |
| **CRF Valid** | 0.484 | 0.756 | 0.471 | 0.334 | 0.470 | 0.141 | 0.000 | 0.724 |

**Training Set (No Post-Processing)**

| **Predicted Class** | **Actual Class** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Stem | Tomato | Tom. Border | Leaf | Bg. Tom | Bg. Stem | Bg. Leaf | Bg | **Prec** | **Rec** |
| Stem | 2898871 | 1067 | 108746 | 1158 | 28935 | 76923 | 13949 | 57126 | 0.56 | 0.91 |
| Tom. | 915 | 2371155 | 353817 | 433 | 7018 | 151 | 360 | 1797 | 0.87 | 0.87 |
| Tom. Border | 2812 | 166513 | 1310391 | 2435 | 9996 | 416 | 456 | 2936 | 0.43 | 0.88 |
| Leaf | 18749 | 6724 | 87783 | 6033592 | 7725 | 16881 | 1506418 | 215393 | 0.65 | 0.76 |
| Bg. Tom | 6620 | 2496 | 9073 | 672 | 1520013 | 14085 | 4374 | 47965 | 0.45 | 0.95 |
| Bg. Stem | 66410 | 147 | 5085 | 263 | 7026 | 1048475 | 6714 | 67571 | 0.21 | 0.87 |
| Bg. Leaf | 136472 | 9574 | 82719 | 1578026 | 91370 | 228913 | 7604752 | 1540752 | 0.38 | 0.67 |
| Bg. | 2031690 | 177460 | 1090496 | 1583441 | 1671903 | 3681911 | 10625214 | 23431476 | 0.92 | 0.53 |
| Total actual stem | 5,162,539 | | | | | | | | | |

Comments and Observations:

1. Both results sets are with the same code.
2. The confusion matrices show that the datasets used for each run (the one reported by Matt and the one on my computer) were different. In my dataset the were 5,315,036 annotated stem pixels (5,162,539 in the training set and 152,497 in the validation set) as opposed to 5,768,252 stem pixels (5,313,957 for training and 454,295 in validation) in Matt's reported results.
3. Despite this difference in the dataset, the precision and recall values of the two runs are comparable. Also, the relative values between different categories in each run are similar. For example, the precision of the Tomato category is the highest in both runs, while that of the Background Stem is the lowest.

## II. Watershed Instance Segmentation

Again, we use Matt's unchanged code and evaluation metrics. To identify instances, bounding boxes are created around each ground-truth and detected instance, their centroid and the distance between those centroids are calculated. Matt's reported results are shown below, followed by the results obtained from running his code on my computer.

## Matt's Reported Results

Detection Results

| | TP | FP | FN | Precision | Recall | # Predictions Ignored | # Tomatoes with Multiple predictions | Pixel-Distance from prediction to nearest labeled tomato | Pixel-Distance from predicted tomato to correct predictions |
|---|---|---|---|---|---|---|---|---|---|
| **Hough** | 100 | 3 | 75 | 0.971 | 0.571 | 15 | 7 | 14.110 | 8.070 |
| **Watershed** | 137 | 16 | 25 | 0.895 | 0.846 | 44 | 21 | 16.340 | 11.500 |
| **Modified Hough** | 100 | 18 | 75 | 0.848 | 0.571 | | | | |
| **Modified Watershed** | 137 | 60 | 25 | 0.695 | 0.846 | | | | |

**TP:** Where a predicted tomato uniquely matches a labeled tomato
**FP:** Predicted a tomato, but no labeled tomato exists
**FN:** Did not predict a tomato, but a labeled tomato exists

**Hough / Watershed:** Remove all predictions where "multiple" tomatos were predicted for a single labeled tomato. This can occur when a tomato is occluded by (e.g. a stem), and there are two halfs of a tomato.
**Modified Hough / Watershed:** Place all "multiple" predictions onto the "False Positive" category

## Results from Matt's code on my computer

| | TP | FP | FN | Prec | Recall | # Predictions Ignored | # Tomatoes with Multiple predictions | Pixel Distance from Prediction to nearest labeled tomato | Pixel-Distance from predicted tomato to correct prediction |
|---|---|---|---|---|---|---|---|---|---|
| Watershed | 126 | 11 | 37 | 0.92 | 0.77 | 57 | 27 | 6.7 | 14.0 |
| Modified Watershed | 126 | 68 | 37 | 0.65 | 0.77 | | | | |

Comments and Observations:

1. True Positives are slightly less using Matt's code on my dataset (126 compared to 137). Remember, thought, that these were generated using different datasets.
2. False Positives we also less using the dataset on my computer (11 compared to 16) with more predictions ignored from my dataset than Matt's (predictions are ignored when there is more than a single prediction per tomato instance).
3. But the change in FP between Watershed and Modified Watershed is very similar in both runs with Matt's reported result going from 16 to 60 and Matt's code on my computer going from 11 to 68.
4. Precision and recall of both runs are similar (0.895 and 0.0.846 p and r reported, and 0.92 and 0.77 when run on my computer).

# Improving Our Instance Segmentation

In the rest of this document I describe changes made to the code to improve our instance segmentation performance. To improve those results, and to make the system able to detect leaves an stems in addition to tomatoes,  I made an incremental set of changes to our pipeline. In what follows I report the effect of each change on the effectiveness of instance segmentation.

## I. Increasing the area threshold from 200 to 300

When a dataset is created, the script create_dataset.py defines a minimum acceptable size of an annotated object. This value is called “area threshold”. The default value used in the script is 200 pixels. This value was used in creating the first version of the dataset which we will call V1. V1 was used to train and validate the segmentation network and to detect instances in the previous sections. However, I suspected that this threshold caused valid annotated (ground truth) tomatoes to be excluded. Smaller stem segments can also end up being removed from the annotated image. So I increased this threshold to 300 pixels, retrained the network, then repeated the instance segmentation experiments. The results are as shown below:

| Area Threshold | Approach | **TP** | **FP** | **FN** | **Prec** | **Recall** | **# Predictions Ignored** | **# Tomatoes with Multiple predictions** | **Pixel Distance from Prediction to nearest labeled tomato** | **Pixel-Distance from predicted tomato to correct prediction** |
|---|---|---|---|---|---|---|---|---|---|---|
| Default (200) | Watershed | 126 | 11 | 37 | 0.92 | 0.77 | 57 | 27 | 6.7 | 14.0 |
| | Modified Watershed | 126 | 68 | 37 | 0.65 | 0.77 | | | | |
| 300 | Watershed | 121 | 5 | 51 | 0.96 | 0.7 | 36 | 17 | 5.14 | 9.4 |
| | Modified Watershed | 121 | 41 | 51 | 0.75 | 0.7 | | | | |

As can be seen from the table above, increasing the area threshold improved both the precision and recall of tomato instance segmentation. The pixel distance from the prediction to the nearest labeled tomato also improved, and there were fewer multiple predictions per instance.

## II. Detecting Leaves and Stems alongside Tomatoes

We have only been detecting tomatoes up to this point. In what follows we look at detecting leaves and stems as well. But first, the dataset set must be modified to include stem and leaf borders in addition to

tomatoes. The semantic segmentation network is then retrained on the new dataset, then instance detection is performed.

## A. Semantic Segmentation of images that include stem and leaf borders

As per Matt's original code, image annotation masks only included stems, tomatoes, leaves, tomato borders, background tomatoes, background stems, background leaves, and background. In order to find instances of leaves and stems, we need to add stem borders and leaf borders to the annotated masks. To achieve this I modified the dataset creation script to find those borders in a manner similar to that used for tomato boarders. I then performed semantic segmentation using the same network as before (UNet).

**Mean IOU between training and validation splits**

| | **Stem** | **Stem B.** | **Tom.** | **Tom. B.** | **Leaf** | **Leaf B.** | **Bg. Tomato** | **Bg. Stem** | **Bg. Leaf** | **Background** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Train** | 0.523 | 0.341 | 0.741 | 0.384 | 0.465 | 0.182 | 0.153 | 0.147 | 0.246 | 0.410 |
| **CRF Train** | 0.561 | 0.417 | 0.773 | 0.473 | 0.359 | 0.119 | 0.230 | 0.209 | 0.001 | 0.648 |
| **Valid** | 0.456 | 0.295 | 0.720 | 0.353 | 0.452 | 0.177 | 0.137 | 0.123 | 0.225 | 0.420 |
| **CRF Valid** | 0.490 | 0.368 | 0.736 | 0.369 | 0.430 | 0.096 | 0.229 | 0.218 | 0.0 | 0.688 |

**Mean IOU between training and validation splits prior to adding leaf and stem borders**

| | **Stem** | **Tomato** | **Tom. Border** | **Leaf** | **Bg. Tomato** | **Bg. Stem** | **Bg. Leaf** | **Background** |
|---|---|---|---|---|---|---|---|---|
| **Train** | 0.531 | 0.767 | 0.415 | 0 497 | 0. 449 | 0.210 | 0.329 | 0.507 |
| **CRF Train** | 0.562 | 0.781 | 0.5121 | 0.354 | 0. 539 | 0.140 | 0.019 | 0.672 |
| **Valid** | 0.470 | 0.7476 | 0.390 | 0.457 | 0.444 | 0.176 | 0.279 | 0.511 |
| **CRF Valid** | 0.484 | 0.756 | 0.471 | 0.334 | 0.470 | 0.141 | 0.000 | 0.724 |

Adding stem and tomato borders slightly reduced the mean IOU of the other classes. However, as we will see below, this reduction has minimal impact on instance detection of tomatoes, leaves, and stems.

Once the semantic segmentation network has been trained, we can use it to detect instances of tomatoes, stems, and leaves as shown below.

## B. Instance Segmentation of Leaves, Stems, and Tomatoes using the Watershed approach

### Tomato Detection

We use the Watershed superpixel method to detect tomatoes.  The table below shows the detection results for tomatoes.

| | TP | FP | FN | Prec | Recall | # Predictions Ignored | # Leaves with Multiple predictions | Pixel Distance from Prediction to nearest labeled tomato | Pixel-Distance from predicted tomato to correct prediction |
|---|---|---|---|---|---|---|---|---|---|
| Watershed | 111 | 13 | 53 | 0.89 | 0.68 | 0.51 | 24 | 9.9 | 9.2 |
| Modified Watershed | 111 | 64 | 53 | 0.63 | 0.67 | | | | |

The tables above show that Watershed segmentation generated highly precise tomato instances: 89% of predicted instances are actually full tomatoes. But a  recall value of 68% indicates that quite a few of tomatoes in the image are missed.

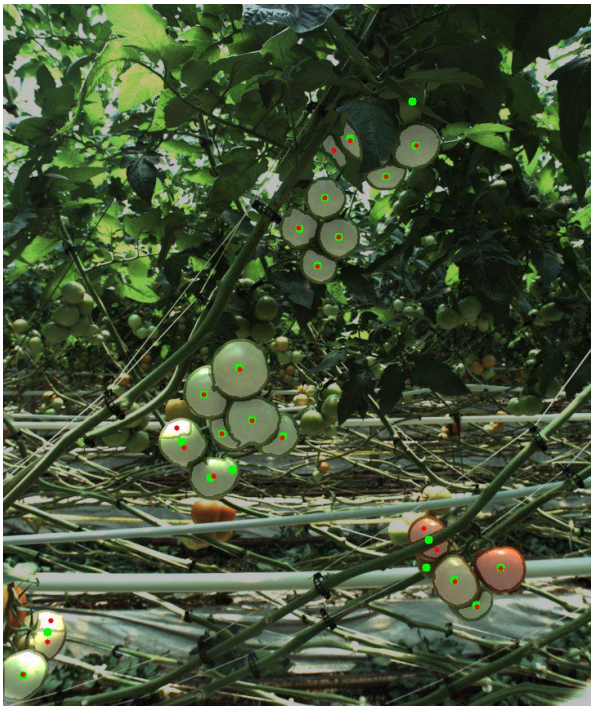

*Figure 3: Watershed tomato instance prediction. The red centroids mark the centroid of the predicted (highlighted) tomato instances. Green cetroids are the centroid of ground-truth tomatoes.*

Figure 3 below is one of the validation set images. Most of the tomatoes in the image were correctly predicted except for partially occluded tomatoes that consist of more than one image, and “shiny” tomatoes that Watershed segmented into multiple instances.

## Leaf Detection

Again, we use the larger area threshold of 300 pixels and Watershed superpixel method to detect leaves. At this point we are still using the same approach to evaluate the method as when detecting tomatoes. The table below shows the results.

| | TP | FP | FN | Prec | Recall | # Predictions Ignored | # Leaves with Multiple predictions | Pixel Distance from Prediction to nearest labeled leaf | Pixel-Distance from predicted leaf to correct prediction |
|---|---|---|---|---|---|---|---|---|---|
| Watershed | 25 | 198 | 80 | 0.11 | 0.24 | 183 | 8517 | 233.8 | 245.3 |
| Modified Watershed | 25 | 381 | 80 | 0.06 | 0.24 | | | | |

As the table shows, the Watershed approach is ineffective for identifying leaves. Figure 4 below shows an example of the predicted segments. Watershed often identifies sections of a leaf as an independent leaf instance.

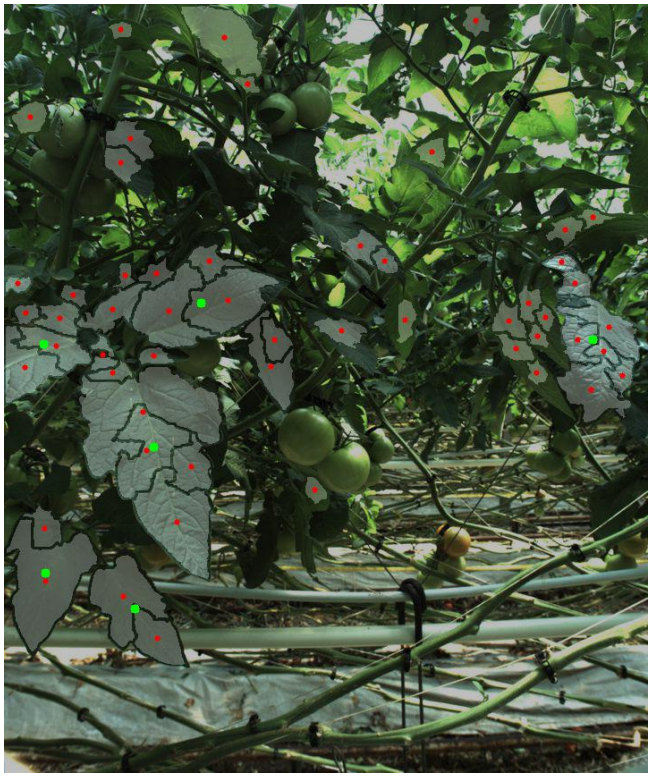

*Figure 4: Watershed leaf instance predictions for Image 1281.*

## Stem Detection

Stem instance segmentation results using the Watershed approach are shown in Table 8 below.

| | TP | FP | FN | Prec | Recall | # Predictions Ignored | # Leaves with Multiple predictions | Pixel Distance from Prediction to nearest labeled stem segment | Pixel-Distance from predicted stem to correct prediction |
|---|---|---|---|---|---|---|---|---|---|
| Watershed | 7 | 170 | 26 | 0.04 | 0.2 | 64 | 14 | 437.4 | 440.26 |
| Modified Watershed | 7 | 234 | 26 | 0.03 | 0.2 | | | | |

These values are misleading for several reasons:

1. Stems are annotated in segments. This means that each ground-truth segment consists of several stem segments.
2. To find centroids of annotated stems, a single bounding box enclosing all of a stem's segments belonging and its centroid is assigned tot he stem. But since stems can bend and have several branches, their shape can be irregular and a single centroid point is a poor representation of the whole stem.
3. The prediction methods used so far cannot detect instances consisting of multiple stem segmetns. Instead, each such segment is considered an independent instance. So even if the centroid of one of those segments is close to that of the ground-truth's it does not reflect the fit of the identified instance to the full ground-truth stem.

I plan to change this evaluation metric to something that is more suitable for irregular, disconnected shapes like stems. But for now let us visually inspect some of those predicted instance segments to gauge the effectiveness of the Watershed approach for stem instance segmentation.

Figures 5 and 6 highlight the predicted segments for two test images. Their centroids are shown in red. None of those predicted segments manages to cover a full stem segment. Also, many of the actual stem segments are missed altogether. Sometimes a single segment contains multiple predictions. These issues are prevalent in all the test images we examined. It seems that the Watershed approach fails to cluster a full stem segment as one single superpixel and misses a high percentage of each segment.

*Figure 5: Watershed stem detection for image 1281*

*Figure 6: Watershed stem detection for image 1790*

## C. Instance Segmentation of Leaves, Stems, and Tomatoes using SLIC

The previous section highlighted serious issues with the predictions generated by the Watershed method. First, Watershed instances for larger areas with many gray level variation, such as leaves, are fragmented. Most leaves consist of several Watershed superpixels making the Watershed approach a weak instance predictor. Second, for stems, the generated superpixels cover only a small fraction of a single stem segment missing large areas of the stem. My guess is that these issues may be a side effect of Watershed's reliance on gray levels in an image only. Another segmentation approach, SLIC, uses color in an image and is reported to be a more effective method for image segmentation. Thus, in what follows I replace Watershed with SLIC segmentation.

### Tomato Detection

We use the SLIC superpixel method to detect tomatoes. The table below shows the results.

| | **TP** | **FP** | **FN** | **Prec** | **Recall** | **# Predictions Ignored** | **# Tomatoes with Multiple predictions** | **Pixel Distance from Prediction to nearest labeled tomato** | **Pixel-Distance from predicted tomato to correct prediction** |
|---|---|---|---|---|---|---|---|---|---|
| SLIC | 112 | 7 | 71 | 0.94 | 0.61 | 10 | 5 | 6.6 | 7.8 |
| Modified SLIC | 112 | 17 | 71 | 0.87 | 0.61 | | | | |
| Watershed | 111 | 13 | 53 | 0.89 | 0.68 | 0.51 | 24 | 9.9 | 9.2 |
| Modified Watershed | 111 | 64 | 53 | 0.63 | 0.67 | | | | |

SLIC improved the precision of our instance detection from 89% using the Watershed method to 94%. When including multiple predictions, SLIC also improves prediction accuracy from 63% for Watershed to 87%. SLIC also generates predictions that are closer to the ground truth instances with an average pixel distance of 6.6 from the nearest ground truth tomato. This comes at the expense of recall, however, which goes down from Watershed's 68 and 67% for Watershed and Modified Watershed, respectively, to 61% for SLIC and Modified SLIC.

Figures 7 and 8 below show the different predictions of each method for image 1384. SLIC tends to generate more unified segments than Watershed (shown within the white circles). However, it also has a tendency to join two separate instances into one which affects its recall (shown in the yellow circle). In this example, the SLIC approach also successfully identifies one of the partially occluded tomatoes in the image (shown within the red circle).

*Figure 7: Watershed Instance Prediction for Tomatoes in Image 1384*

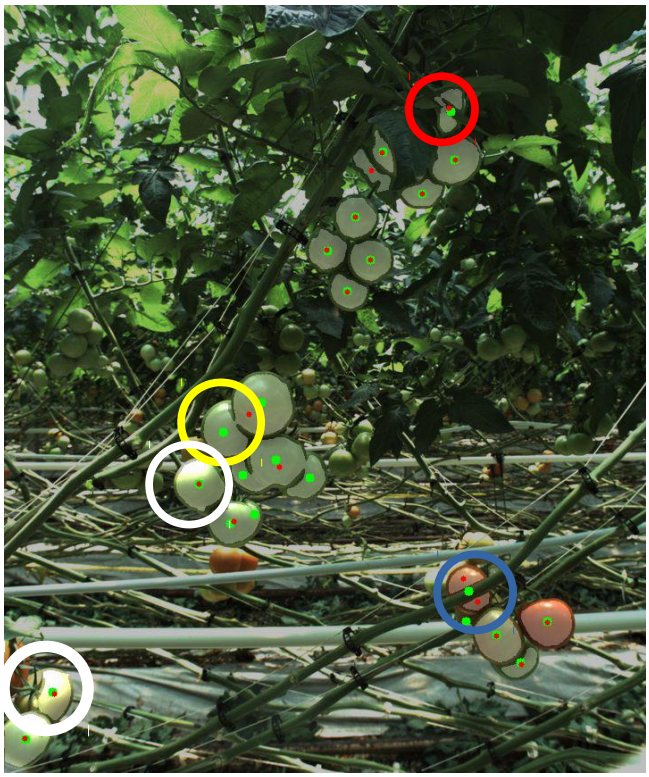

*Figure 8: SLIC Instance Prediction for Tomatoes in Image 1384*

## Leaf Detection

Again, we  use the SLIC segmetnation method to detect leaves and the same centroid approach to evaluate predictions.  The table below shows the results.

| | TP | FP | FN | Prec | Recall | # Predictions Ignored | # Leaves with Multiple predictions | Pixel Distance from Prediction to nearest labeled leaf | Pixel-Distance from predicted leaf to correct prediction |
|---|---|---|---|---|---|---|---|---|---|
| SLIC | 52 | 57 | 91 | 0.48 | 0.36 | 15 | 7 | 74.9 | 14.8 |
| Modified SLIC | 52 | 72 | 91 | 0.42 | 0.36 | | | | |
| Watershed | 25 | 198 | 80 | 0.11 | 0.24 | 183 | 8517 | 233.8 | 245.3 |
| Modified Watershed | 25 | 381 | 80 | 0.06 | 0.24 | | | | |

In the case of leaf detection SLIC is a significant improvement over Watershed. SLIC has more than twice as many true positives, less than one third the false positives, but a slightly higher number of false negatives (91 for SLIC and 80 for Watershed). SLIC's precision is more than 4 times that of Watershed

(48% and 42% for SLIC and Modified SLIC, respectively, and 24% for Watershed and Modified Watershed). SLIC also has only 7 predictions ignored as opposed to a whopping 8517 predicitions for Watershed. The distance from the nearest ground truth instance is also several folds less at 75 pixels for SLIC and 234 for Watershed.

A quick look at some predictions for the two approaches clarifies the source of that difference in performance. For example, Figures 9 and 10 show the instances predicted by Watershed and SLIC, respectively. Watershed's tendency to generate fragmented instance segments is very clear in this case, while SLIC mostly predicts single segments per ground truth instance.

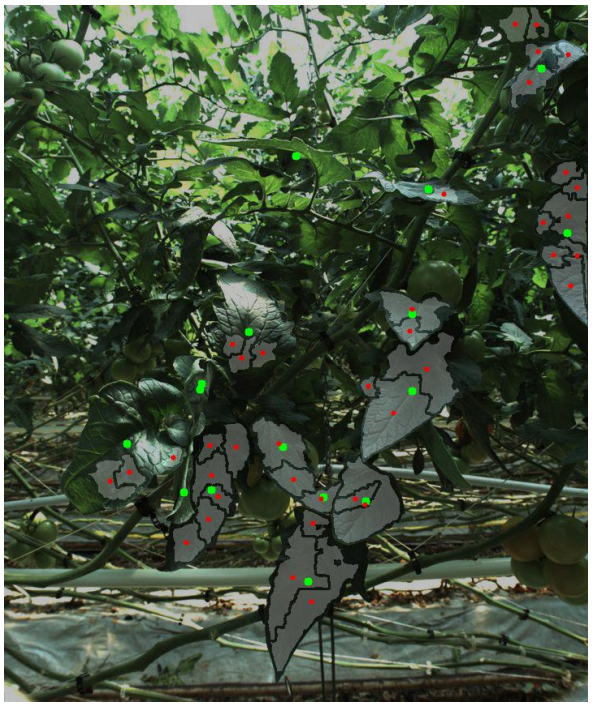

*Figure9: Watershed Leaf Instance Predictions for Image 1263*

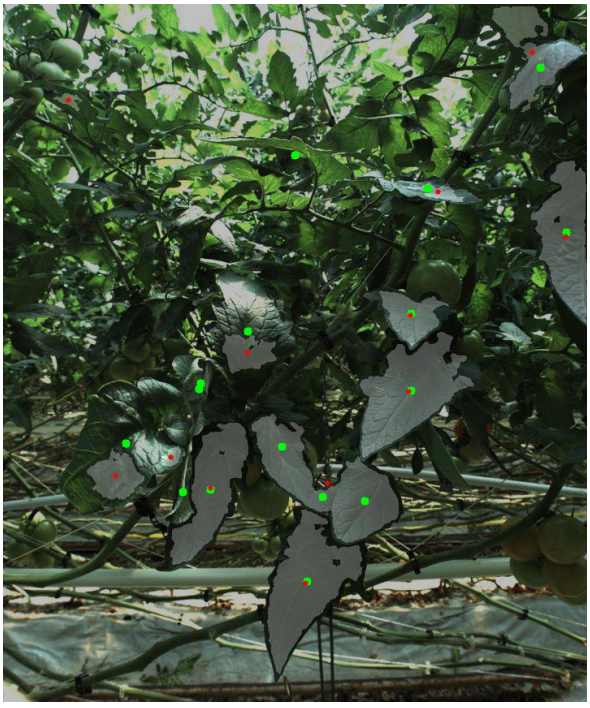

*Figure 10: SLIC Leaf Instance Prediction for Image 1263*

One issue I did notice while examining those images is a randomness in annotating leaves. Some images have similar leaves where some of which were annotated as foreground leaves, while others as background leaves even though they are all similar in size and proximity to the camera. The image in Illustration 11 below clearly illustrates this point. In this image the foreground plants are dense with leaves. However, only six of those leaves were annotated as foreground leaves (marked with green dots). The rest were marked as background leaves. The red dots in this image show SLIC's instance predictions. Most of these were considered false positive by our metric because they were improperly annotated.

*Figure 11: SLIC predicted leaf instances for Image 1281. Note how there are very few ground truth leaves in this image with many valid foreground leaves marked as background leaves.*

## Stem Detection

As with all our previous evaluations, we use the SLIC segmentation method to detect stems and the same centroid approach to evaluate predictions. This evaluation measure is not at all ideal for stems and will be my next action item. For now we will look at sample test images and centroid distance evaluation metrics to gauge the difference in the performance of the two approaches. The table below shows the results.

| | **TP** | **FP** | **FN** | **Prec** | **Recall** | **# Predictions Ignored** | **# Leaves with Multiple predictions** | **Pixel Distance from Prediction to nearest labeled stem segment** | **Pixel-Distance from predicted stem to correct prediction** |
|---|---|---|---|---|---|---|---|---|---|
| SLIC | 8 | 88 | 29 | 0.08 | 0.2 | 28 | 10 | 364 | 351 |
| Modified SLIC | 8 | 116 | 29 | 0.06 | 0.2 | | | | |
| Watershed | 7 | 170 | 26 | 0.04 | 0.2 | 64 | 14 | 437.4 | 440.26 |
| Modified Watershed | 7 | 234 | 26 | 0.03 | 0.2 | | | | |

These values are a very poor reflection of the performance of either method as discussed before. However, Illustrations 12 to 16 clearly show SLIC's advantage over the Watershed approach. In each case SLIC identifies most of the area within each stem segment and very few segments are missed. Watershed, on the other hand, only find small swatches of some segments and misses a significant portion of most stems.

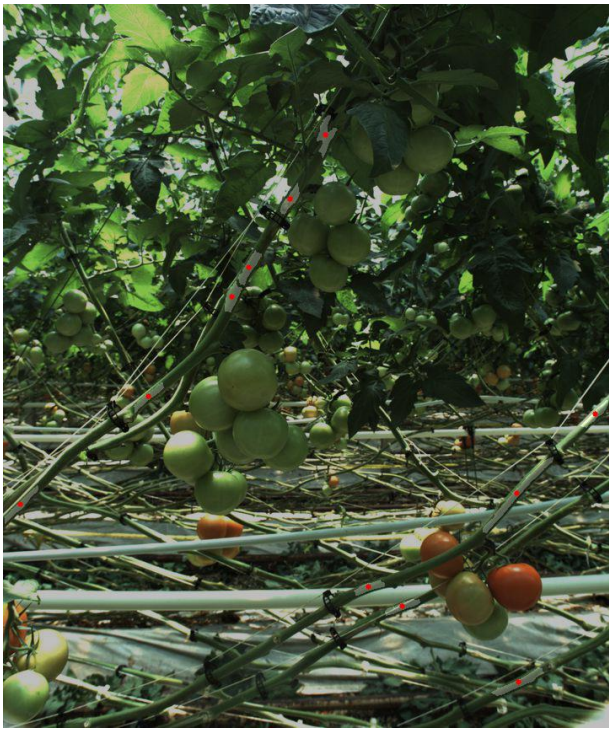

*Figure 12: Watershed stem predictions for Image 1384*

*Figure 13: SLIC stem predictions for Image 1384*

*Figure 14: Watershed stem predictions for Image 1691*

*Figure 15: SLIC stem predictions for Image 1691*

*Figure 16: Watershed  stem predictions for Image 1790*

*Figure 17: SLIC  stem predictions for Image 1790*

Despite this advantage of SLIC over Watershed, neither approach properly detects any full stems. SLIC does have a head start since it properly identifies most of the stem segments.

Starting from  SLIC's output our next step is to figure out how to connect relevant stem segments to form a stem instance. In the next section I present a method to cluster disconnected segments in an image into full stem instances and preliminary results.

## D. Forming proper stem instances from a collection of stem segments

The output of SLIC stem detection is a collection of independent stem segments. As they stand, the system considers each detected segment an independent stem instance. However, in the greenhouse setting, rarely will a stem consist of a single segment. Instead, most stems will be partially occluded by a variety of objects including clips, leaves, other stems, tomatoes, and string. Therefore, given the collection of stem segments as detected by SLIC, our next task is to sort out these segments and cluster them into stem instances.

The approach I use relies on the fact that stems tend to grow in a relatively predictable direction. Stems do not suddenly twist and bend. Their rate of change is small. Therefore, two nearby segments are likely to be part of the same stem if they have similar growth patterns and reasonable proximity. One way to

measure that is by comparing each segment's slope to that of the centroids connecting them. For example, in Figure 18 red lines show the slope of each segment, while yellow lines show the slope between consecutive centroids. By using the similarity in these slopes, we can cluster segments into individual stem instances. The result of using this method on Image 1382 is shown in Figure 19 below. In this image, segments that were clustered in the same stem instance have the same color. The approach successfully identifies segment components of each of the three stems in this image.

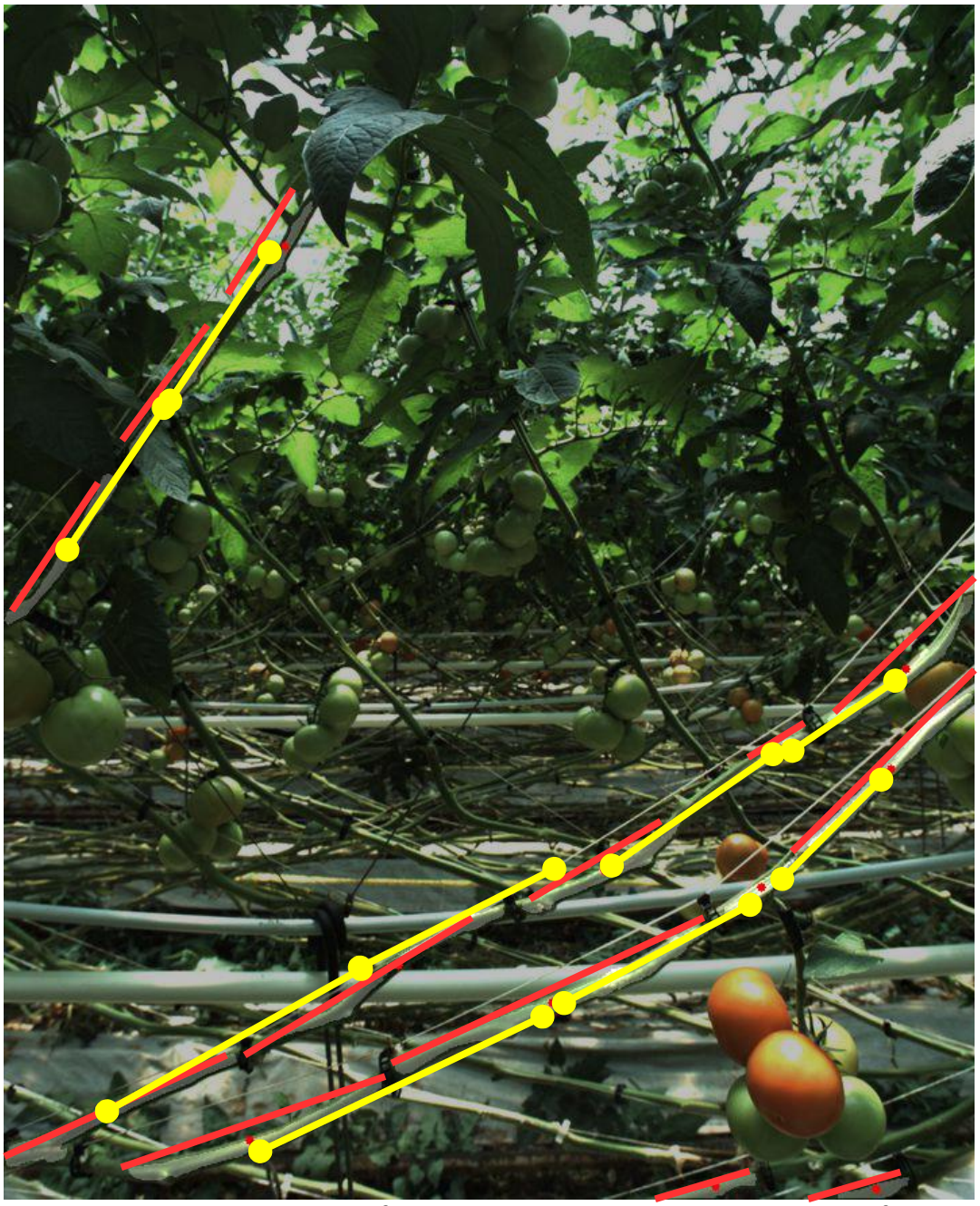

*Figure 18: SLIC stem segments for Image 1382. Red lines show the slope of each segment, while yellow lines show the slope between consecutive centroids. By using the similarity in these slopes, we can cluster segments into individual stem instances.*

*Figure 19: The result of using the slope similarity between segments and segment centroids to create stem instances on Image 1382 is shown in Illustration below. In this image, segments that were clustered in the same stem instance have the same color.*

This approach seems promising. It is simple and intuitive. But it still needs tweaking so it can accommodate segments with side branches and ones with a bit of a curve such as those in Figure 19 .

*Figure 20: An example of the effect of side branches on the result of our segment clustering approach. This is the output for Image 1672.*

My next action items:

1. Create an evaluation metric that is better suited to disjoint objects like stems and partially occluded tomatoes.
2. Tweak the stem segment clustering method to accommodate less straight segments and ones with side branches.