Lincoln Agri-Robotics — AgriFoRwArdS

Poinsettia Challenge

https://lar.lincoln.ac.uk/poinsettia-challenge/

Entry due 17th December 2020 @ 9am GMT

(version 2.0: corrected formulae for computing red and green pixel percentages; corrected usage of the term "bract" and introduced the term "cyathia")

1 Overview

The Lincoln Agri-Robotics **Poinsettia Challenge** is a holiday-themed machine learning and computer vision competition to devise intelligent ways to help identify features that contribute to rating the "best" poinsettia. We will be looking at the height of the plant, colour of the bracts (leaves) and "bushiness", based on the number of clusters of bracts in an image.

→ The competition rules are released on Monday 14th December 2020 @ 10am GMT.
→ The competition entry due date is Thursday 17th December 2020 @ 9am GMT.
→ Winners will be announced at the Closing Ceremony on Friday 18th December at 12pm GMT.

An online meeting space for the competition is here:
`https://teams.microsoft.com/l/meetup-join/19%3ameeting_ZjUxMjk5NTgtOWVhMC00MWZkLWI3MzMtYzhkZWI5ZjdkNzI2%40thread.v2/0?context=%7b%22Tid%22%3a%22952e47c4-05de-4282-83af-5f4b46b1628f%22%2c%22Oid%22%3a%2269d27df4-fc23-4ee0-98d1-a562cc09e453%22%7d`
This Teams Meeting virtual space will remain available from Monday 14th December 10am through the closing ceremony on Friday 18th December. The virtual meeting space will be monitored during that time window, up to the entry due date, in case participants have questions about their entries or want to communicate with other participants.

There are **four** challenges outlined below. Participants can enter as few or as many challenges as they like. However, participants can only win one challenge. Prizes will be distributed (virtually) at the Closing Ceremony (on Friday 18th December at 12pm GMT, in the same Teams Meeting virtual space, link above).

Entries must include:

- a README file containing the name of the person or team entering and an explanation of the files submitted and how to run your code; and
- *for each challenge entered:*
  - Python code to process the images (***.py** files, NOT notebooks); and
  - a results file on the challenge-specific test data set (explained for each challenge).

All files must be compressed into a **zip** archive. No image files should be submitted.

Entries that do not adhere to the submission format instructions will be disqualified.

See Section 9 for details on submitting your entry.

Any questions? During the competition window, you can contact the challenge organisers through the Teams Meeting virtual space listed above.

2 Obtain the challenge data

The data for the Poinsettia Challenge will be provided during the Opening Ceremony, using the Teams Meeting link listed above. The data is divided into three data sets:

(a) side view (b) top view (c) sea of plants

Figure 1: Data sets

The **side view** data set (Figure 1a) is to be used for the "Find the pot" (Section 5) and "Find the top of the plant" (Section 6) challenges. This data set contains 12 images and a `csv` file with the ground truth data for these two challenges.

The **top view** data set (Figure 1b) is to be used for the "Find the bracts" (Section 8 challenge. The data is divided into **train** and **test** sets. The data set includes a `README.txt` file and a file with sample results (`results_sample.json`).

The **sea of plants** data set (Figure 1c) is to be used for the "Find the colours" (Section 7) challenge. The data is divided into **images** and one *example*. The example shows one original image and images highlighting the red and green pixels, as well as a `csv` file containing the ground truth values computed for the example image.

3 Coding

You will be submitting Python code for each challenge you enter. Python 3 is preferred, but Python 2.7 will be accepted.

Make sure that you submit a stand-alone Python file (***.py**) for each challenge you enter. It is fine to use standard packages such as numpy, scipy, scikit-learn, opencv (cv2), matplotlib, pytorch, colab, etc.

A standard Anaconda installation (version 3) is assumed (`https://www.anaconda.com/`). If you use other packages, then be sure to include documention in your README file about which packages you have used and where you installed them from. If we cannot run your code during the evaluation of submissions period, then you will be disqualified.

4 Scoring Metrics

4.1 Intersection over Union (IoU)

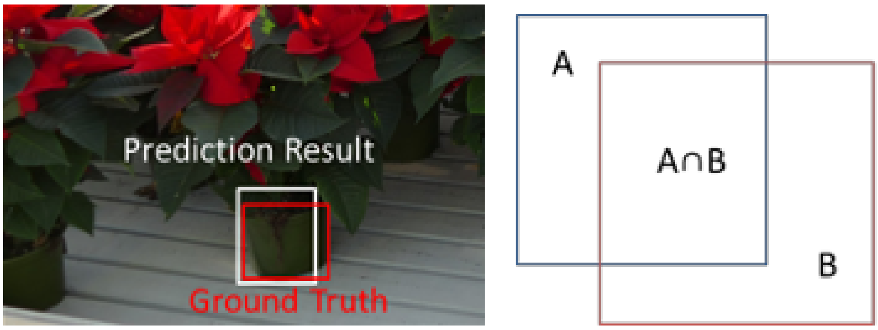
For scoring the accuracy of bounding box prediction, entries will be compared with ground truth, as shown in Figure 2a. The *Intersection over Union* or *IoU* metric is computed using the area of each bounding box (A and B, as illustrated in Figure 2b):

$$IoU = (A \cap B)/(A \cup B)$$

where

$$A = |(A.x\_top\_left - A.x\_top\_right)| * |(A.y\_top\_left - A.y\_bottom\_left)|$$

and similarly for bounding box B.



(a) example bounding boxes (b) computation of IoU

Figure 2: Intersection over Union (IoU)

4.2 Height vector score

For scoring the accuracy of a plant height vector prediction, entries will be compared with ground truth as the difference between the predicted and ground truth vectors. This is illustrated in Figure 3.

The vector score is computed as the sum of the normalised magnitude error (M_error) and normalised angle error (A_error):

$$vector\_score = M\_error + A\_error$$

To compute the normalised magnitude error, first the magnitude of each vector is computed. For the prediction ($\vec{Vp}$), this is:

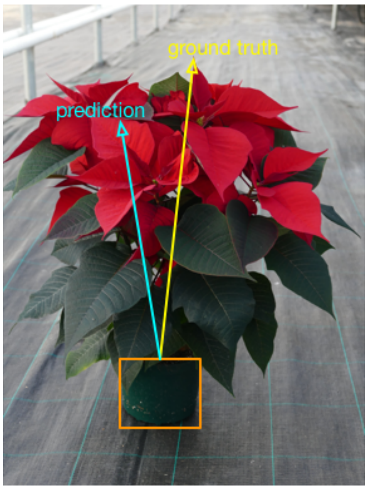
$$|\vec{Vp}| = \sqrt{(Vp1.x - Vp0.x)^2 + (Vp1.y - Vp0.y)^2}$$

where the vector extends from $(Vp0.x, Vp0.y)$ to $(Vp1.x, Vp1.y)$, and similarly for the ground truth vector ($\vec{Vg}$):

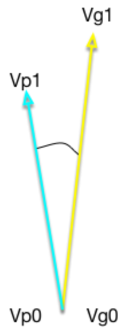
$$|\vec{Vg}| = \sqrt{(Vg1.x - Vg0.x)^2 + (Vg1.y - Vg0.y)^2}$$

The normalised magnitude error is:

$$M\_error = ||\vec{Vp}| - |\vec{Vg}||/|\vec{Vp}|$$



(a) example vectors



(b) computation of vector score

Figure 3: Vector score, where Vp is the predicted vector and Vg is the ground truth vector.

To compute the angle error, the dot product between the two vectors is calculated:

$$dp = (Vg0.x - Vg1.x) * (Vp0.x - Vp1.x) + (Vg0.y - Vg1.y) * (Vp0.y - Vp1.y)$$

and then normalised:

$$A\_error = dp/|\vec{Vp}| * |\vec{Vg}|$$

4.3 Colours score

For scoring the accuracy of a plant colour predictions, entries will be compared with ground truth using the difference between predicted and ground truth colour counts. Colour counts are stored in a 3-element vector containing: the percentage of red pixels in the image:

$$pct\_red = red\_pixel\_count/(red\_pixel\_count + green\_pixel\_count)$$

the percentage of green pixels in the image:

$$pct\_green = green\_pixel\_count/(red\_pixel\_count + green\_pixel\_count)$$

and the ratio of red to green pixels:

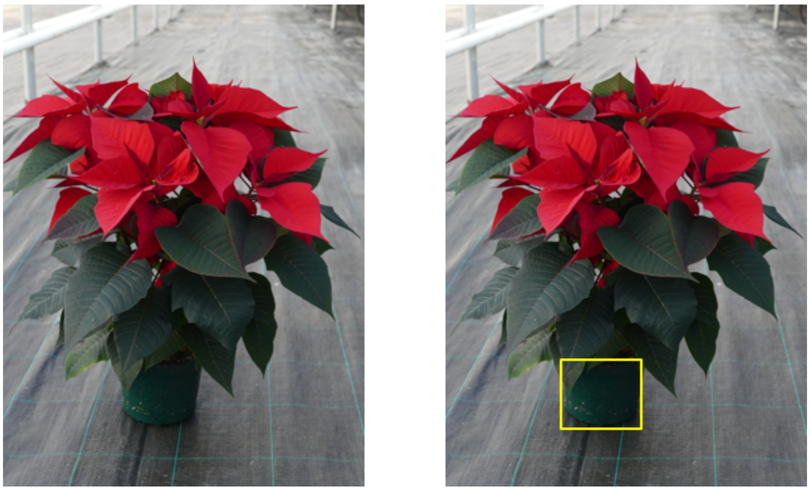
$$rg\_ratio = red\_pixel\_count/green\_pixel\_count$$

Then the difference between this 3-element colour count vector for the predicted (P) and ground truth (G) values is computed as the prediction error:

$$colours\_error = \sqrt{(P.pct\_red - G.pct\_red)^2 + (P.pct\_green - G.pct\_green)^2 + (P.rg\_ratio - G.rg\_ratio)^2}$$

5 “Find the pot” Challenge

The goal of the **Find the pot** challenge is to locate the pot(s) in which poinsettia are planted in each image. An example indicating the region to be found in a sample image is illustrated in Figure 4.



(a) side view (b) bounding box around pot (yellow)

Figure 4: “Find the pot” challenge

Data set:
Use the **side view data set** for this challenge.

Results format:
Entries should indicate the bounding box of the pot. Results should be provided in a **csv** file, with 5 columns:

```
image_file_name pot.x_top_left pot.y_top_left pot.x_bottom_right pot.y_bottom_right
```

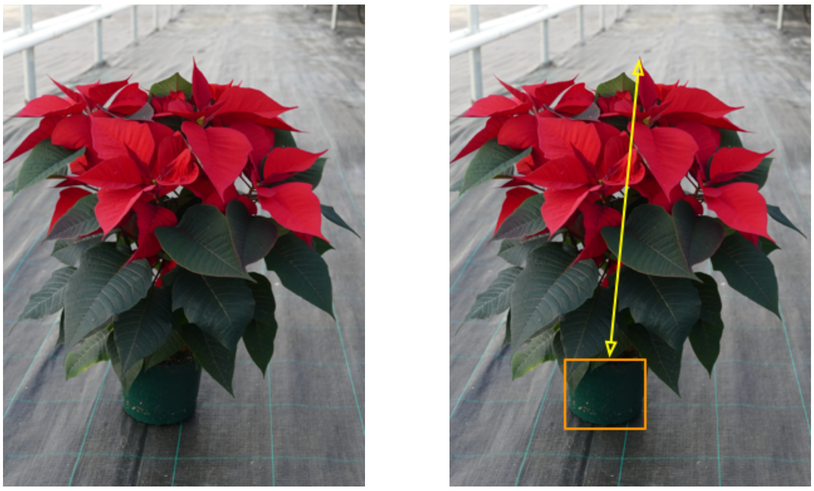
For each plant pot you have located in the image, list the name of the test image file followed by the (x, y) coordinates of the corners of the bonding box outlining the plant pot. Use the pixel coordinates in the image (where $(0, 0)$ is the upper left corner of the image). If you find multiple plant pots in an image, then insert multiple rows in the **csv** file, one per plant pot.

Scoring:
Entries will be scored using the **IoU** metric (see Section 4.1), comparing the overlap of the entry bounding boxes with our ground truth for each of the images in the evaluation data set. The highest average IoU score wins (averaged over all the images in the evaluation data set).

6 "Find the top of the plant" Challenge

The goal of the **Find the top of the plant** challenge is to locate the top of the plant, measured from the top of the pot in which a poinsettia is planted. This could be used to estimate plant height, one of the criteria for grading poinsettia plants used in nurseries.

An example indicating the region to be found in a sample image is illustrated in Figure 5.
Hint: Solve the "Find the pot" challenge first (Section 5).



(a) side view

(b) bounding box around pot (orange) and line to the top (yellow)

Figure 5: "Find the top of the plant" challenge

Data set:
Use the **side view data set** for this challenge.

Results format:
Entries should indicate the vector from the midpoint along the top of the pot to the plant's highest point. Results should be provided in a **csv** file, with 5 columns:

```
image_file_name  pot.x_top_mid  pot.y_top_mid  plant.x_top  plant.y_top
```

For each plant pot you have located in the image, list the name of the test image file followed by the (x, y) coordinates of the midpoint along the top of the pot and the (x, y) coordinates of highest point of the plant that is grown in that pot. Use the pixel coordinates in the image (where $(0, 0)$ is the upper left corner of the image). If you find multiple plant pots in an image, then list multiple rows in the **csv** file, one per plant pot.

Scoring:
Entries will be scored using the **height vector score** metric (see Section 4.2), comparing the difference between the entry vector with our ground truth for each of the images in the evaluation data set. The smallest average vector score wins (averaged over all the images in the evaluation data set).

7 "Find the colours" Challenge

The goal of the **Find the colours** challenge is to find the red and green leaves in the plant and compute the ratio of red to green. This could be used to estimate plant maturity, one of the criteria for grading poinsettia plants used in nurseries. This is a challenge that could be completed using standard image processing methods for colour segmentation or machine learning methods.

(a) original (b) reds (c) greens

Figure 6: "Find the colours" challenge

Data set:
Use the **sea of plants data set** for this challenge.

Results format:
Entries should indicate the total number of pixels in the image, the number of pixels that are in red leaves and the number of pixels that are in green leaves. Results should be provided in a **csv** file, with 4 columns:

```
image_file_name total_pixel_count red_pixel_count green_pixel_count
```

Scoring:
Entries will be scored using the **colours score** metric (see Section 4.3), comparing the difference between the entry vector with our ground truth for each of the images in the evaluation data set. The smallest average colour vector score wins (averaged over all the images in the evaluation data set).

8 Find the bracts (leaves)

The goal of the **Find the bracts** challenge is to find, in an image, the centre of clusters of bracts (leaves)—these are called *cyathia*. This could be used to estimate plant density or "bushiness", which is another criteria for grading poinsettia plants used in nurseries.

Figure 7: "Find the bracts" challenge: the centres of clusters of bracts are labeled

Data set:
Use the **top view data set** for this challenge. Note that a `README.txt` file and a file with sample results (`results_sample.json`) is included in this data set. The data is divided into **train** and **test** sets.

Results format:
The results for this challenge will be submitted using the **COCO** format for specifying bounding box labels. For information, see the **Object Detection** section of the COCO data format documentation, here: `https://cocodataset.org/#format-data/`
We will use the following fields, as shown in example below:

```
[{"image_id": 0, "category_id": 1, "bbox": [338.2, 425.8, 27.7, 28.7], "score": 0.99}]
```

Scoring:
Entries will be scored against the ground truth using the Average Precision (AP) (AP at IoU=.50:.05:.95) metric. For more information about this metric, please see: `https://cocodataset.org/#detection-eval` The highest average precision score wins (averaged over all the images in the evaluation data set).

9 SUBMISSION INSTRUCTIONS

Generate ONE ZIP archive that contains your submission, which must consist of:

- A README (plain text) file that explains which challenge(s) you are entering, which source code file(s) answer which challenge(s) and a results file for each challenge
- The RESULTS file(s) for each challenge you are entering, in the format specified above for each challenge.
- All the Python (***.py**) files you used to generate your answers. *NOTE:* Your Python code MUST be saved as ***.py** files, not iPython Notebooks. **We will NOT accept notebooks. No *.ipynb files allowed! This will ensure we can score all entries quickly.**

Your entry could be disqualified if these submission instructions are not followed.

SUBMIT YOUR ZIP SUBMISSION ON THE POINSETTIA CHALLENGE PAGE:

`https://lar.lincoln.ac.uk/poinsettia-challenge/`

where it says **Submit Hackathon Entry**.

BE SURE TO SUBMIT BEFORE THE DEADLINE!!!
(17th December 2020 by 9am GMT)