Milestone 3: Object Recognition and Localisation

(Last modified: 30 Aug 2022)

In this milestone (Week 6 and 7), you will program your robot to perform object recognition and localisation. Various fruits will be placed in the arena and your robot has to detect the fruits from the camera images. For this, you will implement a deep learning solution by collecting a dataset, training a deep neural network, and testing your trained detector. Then, using the results of the detection, you will need to estimate the poses (x-y location) of the fruits in the arena, which will then be validated against the ground truth poses.

Setting Up

Server (**Only do this part AFTER your M2 Demo**)

Launch PuTTY and run `git pull` to update the robot's code. Then, recalibrate your camera with the checkerboard by taking new images. Again, only do this AFTER you have demo-ed for M2, otherwise you will have code errors during the demo session.

Client (Your laptops)

Duplicate your M2 folder and rename it as M3. Delete operate.py in M3. Download the zip folder for M3 in Moodle and unzip them into the new folder. Edit the new operate.py file by importing your teleoperating codes (& any other part you may have modified).

Launch a terminal and install a few packages:

```
pip install h5py tqdm torch torchvision
pip install -U PyYAML
```

Verify the installation by running `python`, and then `import torch`
If there is no error, exit python by typing Ctrl+Z or exit().

There are a few large files to be generated in this milestone. As GitHub has an upload limit, you may want to create a `.gitignore` file (can be edited with notepad) to avoid uploading certain files to GitHub.

Activities

Part 1: Data Collection

1. With your robot, take plenty of pictures of the fruits (redapple, greenapple, orange, mango, capsicum) and the arena separately. For the fruits, use a simple background like the picture below. Be sure to take pictures of multiple orientations of the fruits and multiple angles for the arena.

2. Write a script to generate a sizable dataset using the pictures that you have taken. A suggested workflow is as follows:
   - Remove the background of the fruit pictures by setting the background pixel to 0. This is easier if your fruit pictures have a uniform background.
   - Superimpose the fruit image onto the arena pictures at random locations. With multiple fruit orientations, arena pictures, and random fruit placements, you should be able to easily generate thousands of images.
   - For every image, a label is required (make use of the cropped image). The label has the same size as the image, with all pixels as 0 (background) except the pixel locations of the fruits. Assign the value of the pixels as follows: *background: 0, redapple: 1, greenapple: 2, orange: 3, mango: 4, capsicum: 5*

3. Save your dataset into "network/scripts" according to the folder hierarchy as follows:

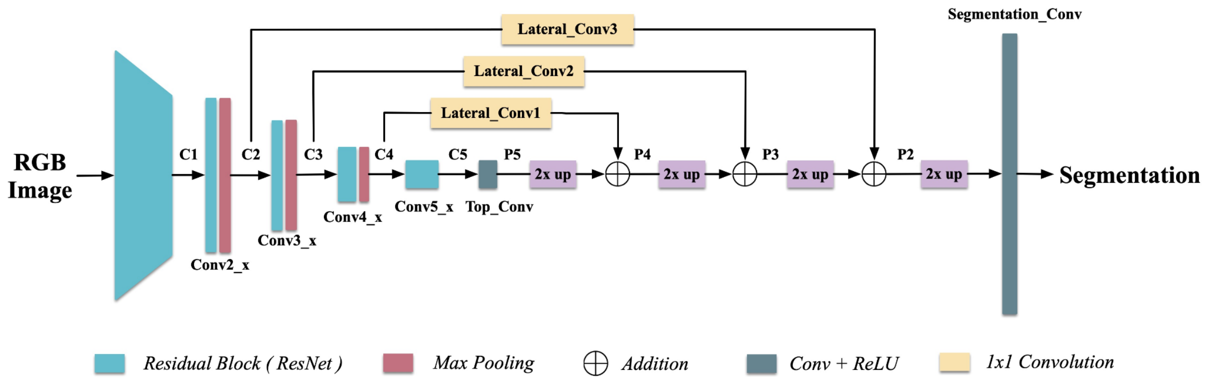
```
network/scripts/dataset/images/image_0.png
network/scripts/dataset/images/image_1.png
…
network/scripts/dataset/labels/image_0_label.png
network/scripts/dataset/labels/image_1_label.png
…
```

Tips:

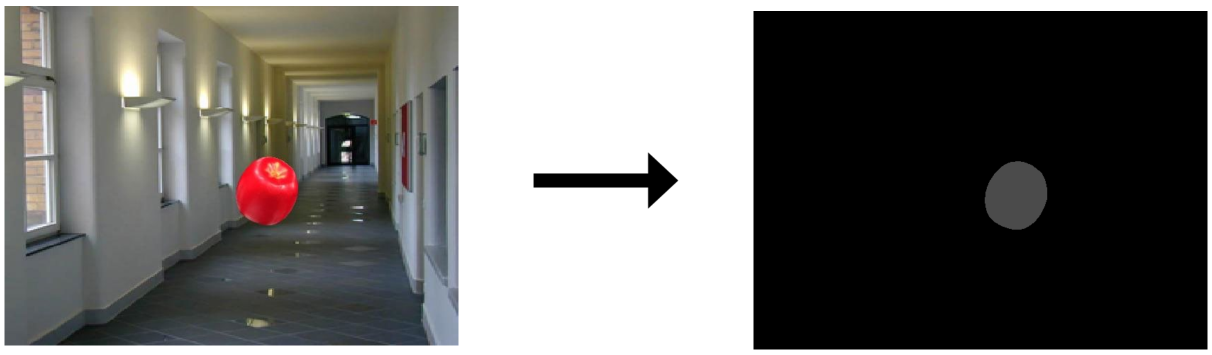
- You can use MATLAB or any other language for data generation. (Note that the model training for next activity is still be in python)
- For removing background, you can use any tools you like, such as an online tool, or write your own script.
- An image can contain more than 1 fruit.
- You can also obtain fruit pictures from the Internet, as well as other random background pictures (instead of the arena) to increase the size of your dataset and improve your trained detector’s robustness.

Part 2: Train your fruit detector neural network model

You are provided with ResNet-18 as the model, which is a deep convolutional neural network proposed for ImageNet classification.. The structure of the ResNet18 model is specified in `res18_skip.py`.



The code provided to you will train the network to accept an input image and generate label prediction for every pixel of the image. This is known as image segmentation.



1. After you have generated your data, run
   ```
   cd network/scripts
   python split_dataset.py --training_ratio 0.8
   ```
   This will separate the dataset randomly into a training set containing 80% of all images and an evaluation set containing 20% of all images, specified by "train.hdf5" and "eval.hdf5" generated under "dataset" folder, which will be used to train your neural network.

2. To train the neural network with the data you generated, run the commands below:
   ```
   cd network/scripts
   python main.py --dataset_dir dataset --model_dir model
   ```

3. You can change the default parameters by adding flags, such as `--lr 0.05`, when running main.py. Parameters that you can change, what they represent and their default values are specified in `args.py`.

4. You will see some information printed regarding how a model performs for training and evaluation during each epoch. Once training is done, a best performing model will be saved as "/network/scripts/model/model.best.pth". This model should be able to detect the different targets (fruits) and segment them from the background. If you want to train new models, delete old checkpoints (model.pth) before training.

5. If you would like to train your neural network on Google Colab, upload the ECE4078_2022_Lab_M3_Colab folder with your dataset included (the hdf5 files) to your Google Drive. Open main_colab.ipynb in Colab and then run its cells (you can change the default parameters in args.py). If you would like to use GPU with Colab, in the top menu, go to "Runtime" -> "Change runtime type" -> "Hardware accelerator" -> select "GPU" in the dropdown menu. After the training is done, you can download the generated best model "model.best.pth" to "network/scripts/model/".

6. To test the performance of your model, run operate.py, position the fruit(s) in the field of view of the robot, and then press "p" to run the detector. You may need to change the path to the best performing model by running operate.py with the --ckpt flag. Your network should segment the input image into different classes (background, redapple, greenapple, orange, mango and capsicum).

Tips:

- The materials provided to you are only a guideline to assist you to train a basic target detector that can provide sufficient performance for you to complete this milestone. Feel free to use your own codes and models, such as exploring other state-of-the-art object detection models like YOLOv5.
- If you decide to use other models, you can make your own file hierarchy structure but please make sure you follow the same naming convention and weight file (.pth) for evaluation purposes.