On the Development of a Bespoke Computer Vision Pipeline for Autonomous Robotic Weeding

Jack Foster
25397813@students.lincoln.ac.uk

Abstract—**There is growing demand for autonomous weeding robotics systems. A significant problem within this task is accurately identifying weeds in real-time. This project leverages existing technologies, such as the YOLOv3 convolutional neural network architecture, to develop an effective and accurate computer vision pipeline to address this challenge. The solution proposed is integrated into the ROS stack to facilitate autonomous navigation and spraying of detected weeds.**

Index Terms—**Robotics, Computer Vision, YOLOv3, ROS, Python, Autonomous Weeding**

I. Introduction

Weeds in agricultural farms are aggressive plants which compete for nutrition and other resources, reducing crop production [1]. As such, commercial farms tackle them either through manual weeding, which is laborious; or herbicide band-spraying, which introduces chemicals that are harmful to both consumers and the environment. Both methods are inefficient [2]. There has been growing interest in the potential for autonomous robotics systems to undertake this task in an efficient, precise and reliable manner. To that end, this project integrates a cutting edge computer vision pipeline with a series of ROS packages to facilitate the autonomous detection and spraying of weeds across a field of crops. The system operates in the LCAS Thorvald in-silico simulation, and attempts to spray weeds across a series of 6 rows, comprised of 3 different crops and 3 different weed-types. The vision pipeline features traditional techniques such as colour-thresholding before leveraging the detection capabillity of YOLOv3 [3], a convolutional neural network (CNN), to identify the presence of plants and discriminate between weeds and crop.

II. Related Work

A. Autonomous Weeding

Site-specific weed management (SSWM) is the process of handling and removing weeds from precise locations within a farm; this is desirable as it falls within the European Union's remit of reducing the use of harmful chemicals and pesticides on farms [4]. Studies have explored the efficacy of different imaging systems to identify the presence of weeds within crop patches - a necessary step to conduct SSWM [4]. Imaging may be accomplished through proximal, on-the-ground imaging systems which are preferable as they are cheaper, and more accurate than broad-imaging systems. Proximal SSWM can be employed using robotic systems and this is the methodology used in this project as it is effective at identifying weeds at all phenological stages and facilitates their removal in a precise and environmentally-friendly manner.

The identification of plants can be accomplished in several ways. In contrast to contemporary vision systems, recent work on weeding in vineyards has leveraged a sonar scanner for the identification of trunks and vines, to facilitate navigation and obstacle avoidance [5]. These methodologies can be preferable to vision systems as they may run faster, and unlike machine learning approaches they are verifiable, which may be important for safety concerns when operating on physical systems. However, the task of discriminating between plant types (i.e weed and crop) requires the ability to identify subtle perturbations, and accomplishing this with noisy, low-resolution sonar imaging is much more challenging than using rgb imaging.

B. Object Detection

1) Colour Thresholding: A very rudimentary approach to object detection is to simply use colours as a discriminating feature, as has been used in problems such as robot soccer [6]. However, such a naïve approach is fundamentally limited, and lacks the axioms to accurately separate crop from weed as their colour profiles are too similar.

2) Harr Cascades: Haar-cascades are an object detection algorithm comprised of a series of weak-feature extractors which consider the pixel intensity of neighbouring rectangular regions [7]. This is lightweight, fast and has been used successfully as a feature-extractor in agriculture, such as for the detection of tobacco leaves [8], which found that Haar-cascades could detect tobacco leaves even against similar objects such as other plants. While this is much more effective than simple colour thresholding, it is inflexible as it primarily conducts single-class object detection.

3) YOLOv3: One of the most popular object detection architectures is YOLOv3 [3], a convolutional neural network. CNNs are well suited to image processing tasks [9], as they utilise spatial information within the input vector, unlike multi-layer perceptron neural networks. Furthermore, shared feature banks enable CNN architectures to be substantially deeper, without the exponential rise in connections [10], which facilitates the learning of more complex patterns. YOLOv3 is faster than other CNN architectures, being approximately 1000x faster than RCNN and 100x faster than Faster-RCNN [3]. YOLOv3 has seen widespread use in agricultural problems, such as in real-time leaf counting problems [11], which has very similar requirements to weed-identification, specifically the type of object being identified, and the need for real-time execution of the algorithm. YOLOv3 has also seen agricultural applications in problems such as apple lesion detection [12], or tomato diseases and pest detection [13].

III. System Architecture

A. Yolov3 Training

To train YOLOv3, $\sim$ 200 images per class were hand labelled, resulting in 600 samples - just 10% of the recommended number [3]. The dataset was split into 4 classes: crop, easy weed, medium weed, and hard weed. This strategy was selected as it provides extensibility to the project; different weeds may in practice require different management, this is achievable with a vision system that can identify what type of weed is present.

Using a small dataset reduces the variance, and can lead to overfitting and a lack of generalisation. To alleviate this, the 600 images were flipped horizontally and vertically, providing a further 1200 images. This ensures that the network will not simply remember the location of the boxes for each of the training samples, but instead must learn the underlying structure of the objects - improving generalisation to unseen data.

The dataset was split with a ratio of 2:8 into a test and train set, whereby the test set would never impact the weights of the network, and thus can be used as an indication of the network's generalisation ability. The dataset was split randomly, but on a per-class basis, to ensure the class distribution remained uniform across both sets.

With the dataset prepared, the network was trained on the cloud (using Google Colab) in 12 hours for the recommended 6000 epochs. The network was unchanged from the original[3], and the sum squared error loss was used, as recommended. The intersection-over-union metric was used to calculate the loss, and is defined as such:

$$IOU = \frac{P \cap L}{P \cup L}$$

where P is the bounding box prediction, and L is the labelled bounding box.

B. Vision Pipeline

1) Colour Thresholding: As the later stages of the pipeline are computationally expensive, it is desirable to limit their usage to when there is a high-probability of a weed being present. Therefore, the first stage was to conduct green-channel colour thresholding, whereby the later stages of the pipeline are only called in the event that there is sufficient green in the image. While colour alone cannot discriminate between crop and weed, it does allow for separation between scenes with plants, and scenes without. To determine the threshold, a histogram of the mean-green value in each training image was created (fig 1). This clearly shows that provided the pipeline is called on any images with a mean green value of more than 37, then the system should not be missing any crop. Only $\sim$ 25 images of the 600 fell below a 38 threshold, and so it may be argued that missing that small percentage of weeds is worth the large increase in down-time for the system; however the system speed is not so problematic that this trade-off is required.

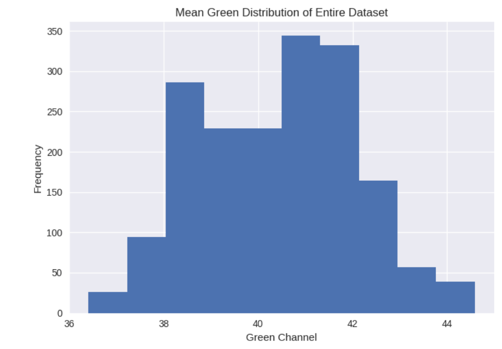

Fig. 1: Distribution of mean-green values for the entire dataset

2) Image Cropping: If the camera image stream publishes at a high enough frequency, it is possible for the network to receive multiple sequential images where the same weed is present. This can lead to repeated sprays of the same weed which is inefficient and in a real-world situation would cause increased operation cost and environmental damage through increase use of pesticide. This problem is alleviated by taking a centre crop of the image stream before passing this as input to YOLOv3. An additional advantage of this strategy is that, due to the implementation discussed in Frame Transforms (III-B4), by taking a more central region of the image the accuracy of the spraying is actually improved due to weed locations being closer to the camera's global coordinates.

3) Yolov3: YOLOv3 was selected because it offered the highest accuracy of methods reviewed, while still being fast enough to operate in real-time. YOLOv3 was initially run on a system without GPU capabilities, resulting in a 0.167 frame-per-second run speed (or 1 frame every 6 seconds). This rendered the solution infeasible and as such a GPU-enabled system was setup, and the necessary CUDA-functionality was implemented. Once the system had been optimised, YOLOv3 ran at 30fps, $\sim$ 180 times faster than previous.

Once YOLOv3 was trained, the ROS implementation was used to facilitate integration into the ROS stack [14]. Upon receiving an image, the ROS nodes responsible for the network publish a series of topics, the most notable being the list of bounding boxes found. A custom rose node (*vision_handler*) subscribes to this topic, and upon callback checks whether any of the boxes contain a weed. Once a weed is found, the current camera pose is stored in a list.

The process from the *vision_handler* node is rate limited to 15 calls-per-second, to ensure that the system is reliable and not creating a backlog of callbacks that cannot be processed in time. This also helps prevent the same weed from being sprayed multiple times, as even with centre-cropping 30 frames-per-second is still enough to occasionally process the same weed more than once. It is rate limited within this node instead of the YOLOv3 node for extensibility purposes, as other nodes may require the full 30 frames-per-second.

4) Frame Transforms: As the camera and sprayer do not share the same pose, it is not feasible to simply spray upon detecting a weed. Instead, a transform listener obtains the pose of the camera and the sprayer within the map coordinate frame. As both poses are both with respect to the same, static coordinate frame (unlike the robot's base frame which is relative), these poses can be used to check if the sprayer is over a weed. Firstly, each time a weed is found, the global pose of the camera is stored within a fixed-length queue. Every time the sprayer's pose is updated, a function is called to check whether the sprayer is above any of the weeds. As the spray is not a single point but rather occupies a space in the coordinate system, and due to noise in the sensor and motion models, it was necessary to allow some leeway with the sprayer location.

C. Navigation

Robot navigation is accomplished with the movebase package in ROS [15]. This was selected as it offers a very robust local and global planner, achieving impressive navigation while also abstracting away complexities that are not the focus of this work. Movebase handles most of the navigation behind the scenes, but it does ultimately require a set of goals, towards which it will navigate. This can be accomplished through a movebase client, that iteratively and automatically publishes navigation goals to the movebase server. The goals are defined in a json file, goals.json, that the user can edit and update prior to runtime. The client first instantiates an ordered list of waypoints, and iteratively submits them to the movebase server. Once the server communicates the successful arrival of the robot, the client pops that waypoint from the queue and submits the next one. This system was selected as it is a simple and elegant solution to allowing a user to define an ordered set of goals, but without requiring their interaction during runtime.

IV. Evaluation

A. YOLOv3

While image flips were used effectively, there were many more augmentation strategies that could have been applied. Alongside structural transforms such as random crops, rotations, shears and affine transforms, perturbations in colour brightness, intensity and contrast could have been made. These would have enabled the vision system to become more robust by preventing colour from being a substantial classification feature. This is less important in simulation, but would be necessary should this solution be applied to real-world problems, as changes in weather, time of day or the presence of shadows from dynamic objects would lead to substantially impacted results.

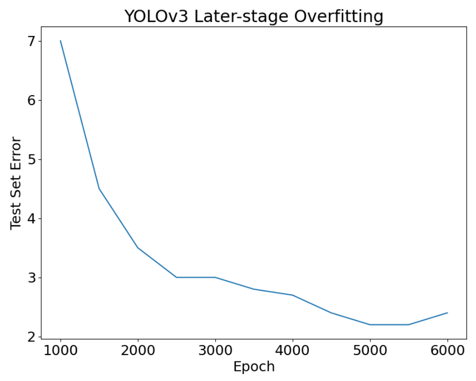

Fig. 2: Greater test-set loss indicates overfitting after the 5000th epoch

Figure 2 illustrates that after the 5000th epoch, the network began to overfit, and the test-set performance became worse. This is contradictory to the official documentation's guidelines of training for a minimum of 6000 epochs. This can potentially be attributed to the dataset size, as the lack of variance leads to overfitting and poor generalisation. Early stopping was applied and the network weights at epoch 5000 were subsequently used. In the original YOLOv3 paper [3], it is recommended to use a much larger dataset than the one used here (~ 4 times larger). Extending the dataset would likely help alleviate the overfitting problem. In addition, the extra data augmentation steps listed prior may also help reduce overfitting.

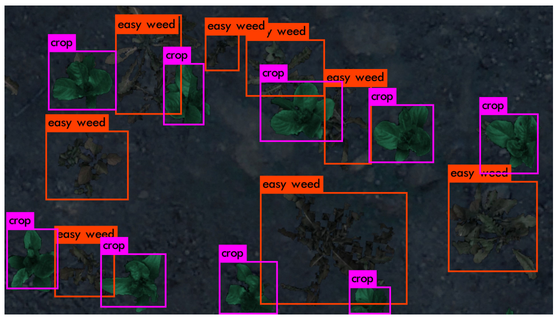

Fig. 3: Successful detection and classification of both crop and easy weeds

Despite this, YOLOv3 was able to detect all three weed types, as well as the crops (table I). YOLOv3 documentation indicates that the training may cease when the loss falls between 0.5 and 3.0, depending on the size of the dataset, number of classes, and complexity of the problem. The network achieved a loss of 2.1 on an 1800 image dataset, which falls well within the expected range and indicates successful training. This average can be considered successful considering the complexity of the problem and the number of classes. It could likely be improved, however, if the above changes were implemented.

Weed Type	Accuracy
Crop	0.76
Easy	0.91
Medium	0.73
Hard	0.64

TABLE I: YOLOv3 average accuracy across the test set

Ordinarily the mean average precision (MAP) metric is used for evaluating the accuracy of the network. However, due to hardware limitations as a result of CUDA requirements, the MAP functionality was not available. Instead, the network was compared to ground truth labels on an unseen dataset, and the mean percentage of correct labels was calculated. The limitations of this approach are three fold. Firstly, this has to be done by hand and so the test-set cannot be as large as an automated process. Secondly, it does not calculate a granular value for how precise each box was, it only calculates how many objects were correctly detected. Finally, the hardest crop row does not contain discrete weeds (fig 4) and so the accuracy is not as robust as it would ideally be.

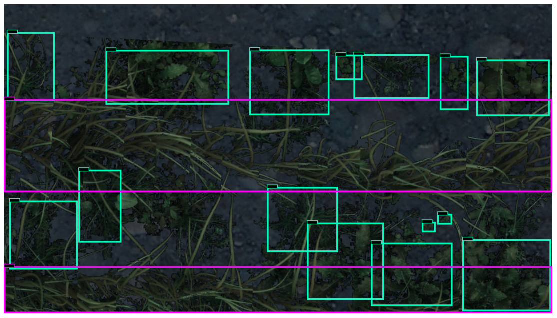

Fig. 4: Successful detection and classification of both crop and hard weeds

B. Colour Thresholding

While in theory the colour thresholding worked, in practice many scenes of just one or two weeds have a very low mean green threshold (between $37 - 40$). These values are not substantially greater than those for scenes without any plantation, and as such the colour thresholding commonly produces false positives and still calls the vision pipeline on an empty scene. While this is not ideal, it is not a major limitation as false positives are much less important than false negatives - which could lead to missed weeds. If the thresholding is allowing too many empty scenes through, however, then it may be worth removing all together as the thresholding itself requires compute time.

C. Frame Transforms

The frame transforms worked remarkably well, facilitating quick and reliable collision calculations to enable accurate spraying. The primary limitation of this entire system is the naïve belief that any crops viewed through the camera have the same global coordinates as the camera itself. For future work, image-to-world projection should be implemented to alleviate this issue and to acquire the precise location of each individual weed. In doing so, the process can become much more precise. This was excluded from the current solution because it very quickly saturated the fixed-length queue, and using a larger data structure lead to buffering that caused substantial spraying inaccuracy. Implementing a more precise spraying system in tandem with the image-to-world projection will lead to a fully fledged system that could feasibly be introduced to real-world environments.

V. Conclusion

This project leveraged the potential of existing software components to achieve the accurate identification of weeds within a simulated environment. The project successfully implemented a vision system pipeline, utilising a range of technologies including the detection capabilities of the YOLOv3 convolutional architecture. This complex vision system provided accurate, high-speed predictions of the presence and location of weeds and crops within a given scene, and these predictions enabled the autonomous spraying of the weeds in simulation through custom ROS nodes as well as packages such as transform listeners and the movebase navigation stack. The solution proposed is effective within the scope of the project and has been designed to be extensible, such that the proposed changes may be implemented in future work.

References

[1] S. P. Adhikari, H. Yang, and H. Kim, "Learning semantic graphics using convolutional encoder–decoder network for autonomous weeding in paddy," *Frontiers in Plant Science*, vol. 10, p. 1404, 2019.

[2] M. Nørremark and H. Griepentrog, "Analysis and definition of the close-to-crop area in relation to robotic weeding," 2004.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[4] F. LÓPEZ-GRANADOS, "Weed detection for site-specific weed management: mapping and real-time approaches," *Weed Research*, vol. 51, no. 1, pp. 1–11, 2011.

[5] D. Reiser, E.-S. Sehsah, O. Bumann, J. Morhard, and H. W. Griepentrog, "Development of an autonomous electric robot implement for intra-row weeding in vineyards," *Agriculture*, vol. 9, no. 1, p. 18, 2019.

[6] N. Lovell and V. Estivill-Castro, "Color classification and object recognition for robot soccer under variable illumination," in *Robotic Soccer*, Citeseer, 2007.

[7] S. Soo, "Object detection using haar-cascade classifier," *Institute of Computer Science, University of Tartu*, pp. 1–12, 2014.

[8] C. Marzan and N. Marcos, "Towards tobacco leaf detection using haar cascade classifier and image processing techniques," pp. 63–68, 10 2018.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] M. Buzzy, V. Thesma, M. Davoodi, and J. Mohammadpour Velni, "Real-time plant leaf counting using deep object detection networks," *Sensors*, vol. 20, no. 23, p. 6896, 2020.

[12] Y. Tian, G. Yang, Z. Wang, E. Li, and Z. Liang, "Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense," *Journal of Sensors*, vol. 2019, 2019.

[13] J. Liu and X. Wang, "Tomato diseases and pests detection based on improved yolo v3 convolutional neural network," *Frontiers in Plant Science*, vol. 11, p. 898, 2020.

[14] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS." https://github.com/leggedrobotics/darknet_ros, 2016–2018.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.