Simulation, Learning and Control Methods to Improve Robotic Vegetable Harvesting

Simon Birrell[1] and Fumiya Iida[2]

[1]sab233@cam.ac.uk, [2]fi224@cam.ac.uk, Bio-Inspired Robotics Lab, Dept. Engineering, Cambridge University, UK

Abstract. Agricultural robots are subject to a much harsher environment than those in the factory or lab and control strategies need to take this into account while maintaining a low cycle time. Three control strategies were tested on Vegebot, a lettuce-picking robot, in both simulation and on the real robot. Between a fast open loop that was vulnerable to environmental noise and a slow but robust visual servoing technique, a Learned Open Loop strategy was tested where the robot learned from successful picks to pick at an intermediate speed. This reduced the projected cycle time from 31s to 17.2s, a 45% reduction.

1 Introduction

Agriculture has historically been labour-intensive and subject to the vicissitudes of the weather, but automation has increased efficiency and had a downward effect on prices. However, automation has not been achieved for many crops and a series of challenges for agricultural robots have been exposed. Farmers' fields are not as sheltered as factories and robots will encounter vibrations, hard-to-navigate terrain and rough weather conditions, affecting every aspect of how they are designed and built. The present paper describes different approaches to tackling one particular problem for harvesting robots, that of reaching for and grasping the target vegetable under environmental noise.

This paper builds on the previously published work on the Vegebot [1], a lettuce-picking robot (Fig. 1). Lettuce harvesting is still a difficult manual task for farm labourers; the goal for Vegebot is to autonomously and accurately harvest lettuces without damaging them, in the shortest possible cycle time. The first iteration of Vegebot automated harvesting, achieving a cycle time of 31s (compared to a human cycle time of 6s), a harvest success rate of 88% and a damage rate of 38% [1]. All of these metrics need to be improved, but the most glaring failing is a cycle time that is 5 times slower than a human. Speed is critical to make the device economically viable, and new techniques are required for this.

Vegebot needs to swiftly move the end effector to the visually identified target lettuce. One problem that complicates this seemingly simple goal is the unpredictable and rough environment in which agritech robots find themselves. Robots with a less than perfect self-model and subject to environmental noise and wear and tear have difficulties in transforming a visual

Fig. 1. The Vegebot: (a) Vegebot deployed for field trials (b) Vegebot in the lab with a "virtual" lettuce.

cue into a precise location in robot task space. Early experiences in the field with Vegebot suggested that dealing with these conditions will be a central challenge for many different types of harvesting devices.

The purpose of this paper is threefold: (1) to develop a realistic simulation platform for Vegebot, (2) to test new harvesting algorithms, including visual servoing and learned behaviours and (3) to optimise the operation of Vegebot in the real world in terms of speed and accuracy under environmental perturbations. With a robust perception and reaching strategy, harvesting robots could produce a greater yield with a shorter cycle time.

Extensive detection and manipulation work has been performed on other individual crops such as broccoli [2], strawberries [3] and apples [4]. A good overall review with important metrics definitions can be found in [5]. Many of the crops that have been successfully automated such as tomatoes and sweet peppers have done so in semi-sheltered environments [6], where they are less subject to the environmental wear and tear tackled here.

Visual servoing has been extensively studied; see [7] for a review and [8] for an example application in harvesting. The use of multiple cameras for visual servoing along with authority switching algorithms has also been explored, for example in [9], which uses distance derived from an RGBD sensor to determine the change of authority. RGBD sensors are problematic outdoors, so Vegebot simply uses the presence or absence of the target in the end effector camera to switch authority.

2 Technical Approach

Vegebot has now been deployed in three environments: in physical field trials (see Fig. 1a), in the lab (see Fig. 1b) and in simulation (see Fig. 2). Field trials have been described in the previous paper [1]; for this paper experiments were performed in the lab and in simulation.

2.1 The real Vegebot

The Vegebot platform is two metres wide, 55cm high and 140cm long; the wheels straddle a standard sized lane of lettuces in the field. A Universal Robotics UR10 arm is mounted on top of the platform and carries a custom-designed end effector, which contains a soft gripper (for holding the lettuce) and a blade (for cutting the stalk) driven by pneumatic actuators via a circular belt. In its current form, the end effector weighs almost 10kg, at the limit of what the arm can support.

Vegebot has a two-camera vision system. One camera (640x480 pixels) is mounted on an overhead support and provides a broad view of the workspace. A second camera (320x240 pixels) is mounted inside the end effector and allows for fine adjustment in positioning as the end effector descends over the lettuce. Both cameras can localize lettuces, while the overhead camera can also estimate the relative pose of the platform and end effector from their attached Aruco boards (see Fig. 2). The robot arm will generally occlude much of the platform and end effector in the overhead camera's view, but the partially visible Aruco boards provide consistent readings nevertheless at an update rate of 6Hz.

A two-stage localisation / classification pipeline detects the lettuces visible in each camera. First, a YOLO network localises the lettuces, passing the cropped images of each one to the second stage. This latter stage classifies them as harvest-ready, immature or diseased. YOLO is a fast object detection neural network architecture [11]. The two stages are necessary to compensate for the unbalanced datasets available and could be collapsed into one on a full production system [1]. The output of the full pipeline is a set of lettuce hypotheses, updated at 5.5Hz on current hardware: each hypothesis with a bounding box in normalized u, v pixel coordinates and a class label, as shown in Fig. 2.

Finally, the robot contains an IMU (Adafruit IMU BNO055) for measuring the absolute orientation of the platform and a laptop with a gaming GPU for managing the whole system. The IMU is connected to an Arduino which streams quaternions to the laptop via a USB connection. The Vegebot software was built on ROS (Robot Operating System) in Python, PyTorch and JavaScript. Inverse kinematics and dynamics are calculated by the UR10's built-in controller.

2.2 The simulated Vegebot

The purpose of building a simulated Vegebot (see Fig. 2) was to provide a platform for experimentation of different control and learning strategies. CoppeliaSim 4.0 was selected, which provides a ROS-compatible environment with a ready-made model of the UR10 and pluggable physics engines. The Newton physics engine provided the best stability, and a model of the Vegebot platform was created using the real robot's dimensions and weights. The simulation ran at around 10 frames per second.

Virtual cameras were used for overhead and end effector cameras. Photographs of earth from the lettuce fields were mapped onto the ground textures. Wrapping photographs of lettuces onto spheres did not work with the

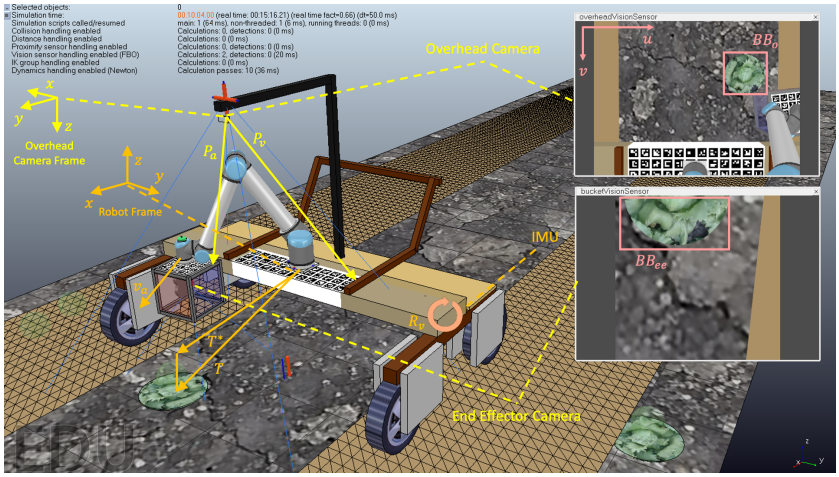

Fig. 2. The Vegebot simulated using CoppeliaSim. Virtual camera fields are inset and a "virtual" lettuce is shown.

real robot's vision system; presumably the features picked up by YOLO were distorted and no longer recognisable. Instead, an overhead photograph of a lettuce was mapped onto a flat circle, and this was topped with an almost transparent sphere, to give the simulated end effector something to grip onto (see Fig. 2. With that, the vision system worked from the virtual cameras in the same way as on the physical robot, providing bounding boxes (see camera insets on Fig. 2).

There remained the problem of reproducing the arm's movements from inverse kinematics in simulation. Previous work found the onboard UR10 controller to be more reliable than the ROS kinematics package. Fortunately, Universal Robotics provide a simulation of the controller called URSim which accepts position or velocity commands and streams the position of a virtual UR10 arm. This stream is translated into ROS messages and fed to the arm model in CoppeliaSim which then tracks the controller's virtual arm model. The software arrangement can be seen in Fig. 3. The bulk of the Vegebot software can be used unaltered in the two configurations.

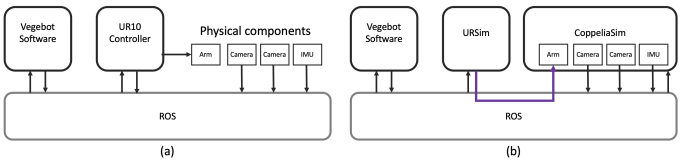

Fig. 3. The Vegebot Software Architecture: (a) on the real Vegebot (b) in simulation, with simulated cameras and IMU, plus a simulated arm tracking the output of the URSim simulated controller.

2.3 The Three Control Approaches

Once the vision system has detected the bounding box of a target lettuce, the end effector must be moved to a position just above the target and then brought down to the ground to envelope, grip and cut it. Three new control approaches were tested, first in simulation and then on the real Vegebot. Each method has two sequential stages (Fig 4) and begins with the **Overhead Camera** feeding a stream of 640x480 images to the **Lettuce Detector**, which generates one or more bounding boxes BB_o.

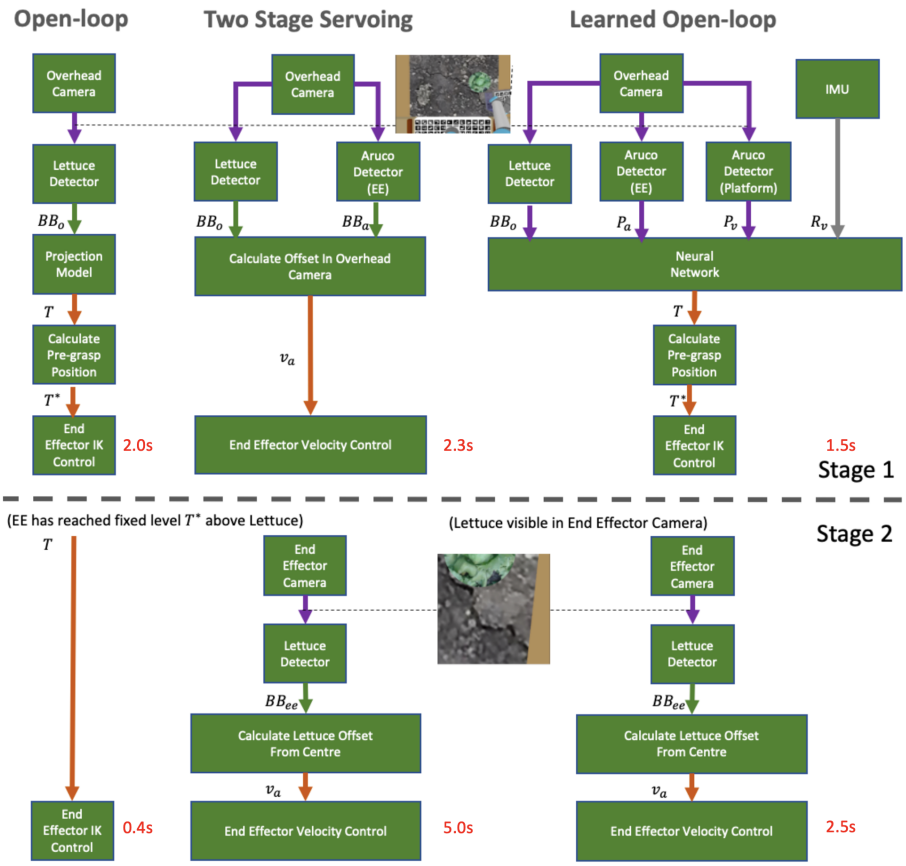

Fig. 4. Control System Information Flow for the three harvesting methods. The sources of delay are indicated in red.

In the first of the new approaches, **Open Loop** control, a static **Projection Model** transforms the bounding box coordinates BB_o into an estimated 3D location in the robot arm space T. This model derives T by using knowledge of the camera's geometry to project a vector from the camera through the centre of the bounding box to where it intersects with an esti-

mated ground plane. The system then calculates the **Pre-Grasp Position** T^*, which is some 20cm vertically above T.

In Stage 1 of the movement, the end effector is moved using inverse kinematics to the **Pre-grasp position** T^*, and then in Stage 2 drops down to T so that the lettuce is left in the centre of the end effector cage. This **End Effector Inverse Kinematics Control** is executed by the UR10's controller (or URSim in simulation). The resulting movements are fast, but not robust to noise or to inaccuracies in the described projection model. It is a variant of the method used on the original Vegebot.

In the second approach, **Two Stage Servoing**, visual servoing is used to continually adjust the velocity v_{xyz} in robot arm space of the end effector to approach the visual target. The **Lettuce Detector** provides a bounding box BB_o for the lettuce as before, but the overhead camera image stream is also passed to an **Aruco Board Detector (EE)** which returns a Bounding Box BB_a for the top of the end effector.

An offset d_{uv} between the centres of the two bounding boxes BB_o and BB_a is calculated in normalized pixel coordinates, and this is used to derive the overall desired end effector velocity v_{xyz} in the module **Calculate Offset in Overhead Camera**:

$$v_{xy} = -\alpha_{cam} d_{uv} \tag{1}$$

$$v_z = -\beta_{cam}[(z - z_{target}) + \frac{\gamma_{cam}}{|d_{uv}| + 0.1}] \tag{2}$$

where z is the z position of the end effector in robot arm space, z_{target} is the estimated z position of the ground and α_{cam}, β_{ca} and γ_{cam} are constants that are defined separately for each camera. The resulting velocity commands v_{xyz} are passed to the UR10 Controller (or URSim) in the module **End Effector Velocity Control**. The end effector will accelerate towards the lettuce in the XY plane while slowly descending.

The overhead camera retains authority until the lettuce becomes visible in the end effector camera, at which point Stage 2 commences. Authority is switched to the **End Effector Camera** the output of which is used by the **Lettuce Detector** to derive a new, close-up bounding box BB_{ee}. The offset d_{uv} is now the offset of the centre of BB_{ee} from the centre of the visual field and the module **Calculate Lettuce Offset from Centre** derives the desired velocity v_{xyz} using the same equations (1) and (2) but with different constants for α_{cam}, β_{cam} and γ_{cam}. The end effector continues to approach the centre of the lettuce while now descending faster; given that the lettuce is close in the XY place, there is less chance of missing the target by moving down too quickly.

The Two Stage Servoing approach is limited in velocity by the speed of updates from the perceptual system (around 5Hz on the current hardware) but is more robust to environmental noise and changes in body shape: as the Vegebot receives blows, the trajectory self-corrects (within limits).

The third approach, **Learned Open-Loop**, attempts to combine the speed advantages of the first approach with the robustness of the second.

Var.	Description	Units	Frame of reference	Dim.
P_a	Pose of end effector Aruco board	Position (x, y, z) Orientation (x, y, z, w)	Overhead camera	7
P_v	Pose of platform Aruco board	Position (x, y, z) Orientation (x, y, z, w)	Overhead camera	7
R_v	IMU orientation	Orientation (x, y, z, w)	World	4
BB_o	Bounding box of lettuce in overhead camera	Centre, width, height (u,v,w,h) in normalized pixel coordinates	Overhead camera image	4
T	Pose of end effector that places it over the lettuce	Position (x, y, z) Orientation (x, y, z, w)	Robot arm	7

Table 1. Input and output variables to the Learned Open Loop neural network.

As well as the **Lettuce Detector**, the **Aruco Detector (EE)** now generates an estimated pose P_a for the end effector, and an additional **Aruco Detector (Platform)** derives an estimated pose P_v for the platform using the second Aruco Board, both poses being estimated in overhead camera space. The IMU measures the absolute orientation R_v of the platform as a quaternion in world space.

Next, a **Neural Network** that accepts BB_o, P_a, P_v and R_v as inputs and generates position T as an output, estimating the lettuce pose in robot arm space using the available sensory data (equation 3). T^* is derived in the usual way, begin 20cm above T. These variables are listed in Table 1 and are shown in Fig. 2.

$$T = f(P_a, P_v, R_v, BB_o) \tag{3}$$

The input variables are mapped to T using 4 fully-connected layers, tanh activation functions, a mean squared error loss function and the Adam optimizer. The input and output layer size is determined by the dimensions of the variables in Table 1. The size and number of the hidden layers was arrived at by iterative experimentation, trading off training speed against accuracy. The neural network was trained for 2000 iterations (Fig. 5), using data taken from successful Two-Stage Servoing picks. Separate datasets were gathered in simulation and reality: in simulation 2000 samples were gathered, while on the real robot a smaller dataset of 50 successful picks was duplicated to form 2000 samples and found to work quite acceptably.

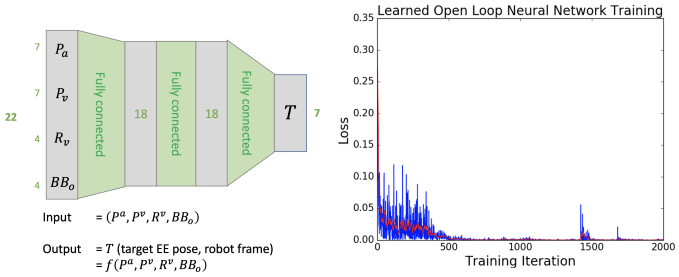

Fig. 5. (a) The inputs and outputs to the Learned Open Loop neural network. (b) Training the neural network over 2000 iterations.

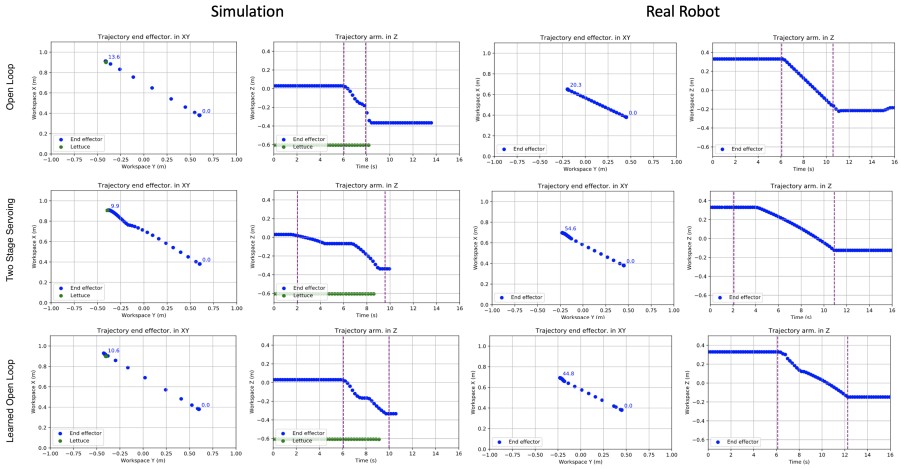

Fig. 6. Experiment One: End effector trajectories under the different control systems, (a) in simulation and (b) on the real robot. The graphs on the left show the end effector's trajectory in the X-Y plane with the start and end points labelled with time t. The graphs on the right show the end effector's trajectory along the Z axis, with time t along the bottom. The vertical purple lines indicate the start and end of the picking sequence.

Stage 1 of the movement is open loop and fast, with the end effector moving swiftly to T^*. Stage 2 is identical to stage 2 of Two-Stage Servoing, with the end effector descending over the lettuce, self-correcting as it goes. As a result, Learned Open Loop is faster than Two Stage Servoing, while retaining much of its robustness.

3 Experiments and Insights

3.1 Experiment One: Three different control systems

Experiment: The three different control methods were tested for speed in simulation and then on the real Vegebot robot in the lab environment. A lettuce (either a paper photograph for the lab environment or a sphere and texture mapped disc in simulation) was placed on the ground and each control method was tested 5 times.

Results in simulation: The **Open Loop** method is the fastest method (2.4s duration) as it can ignore any further sensory information once started (see Fig. 6). The end effector accelerates and moves rapidly in a straight line towards T^* as quickly as the hardware permits. On reaching T^* it accelerates downwards towards the ground. The changeover between the two stages can be seen as a kink in the time series graph of Fig. 6 (a) (Open loop, Simulation), at $t = 8.0s$. Note that the end point of the trajectory does not coincide perfectly with the lettuce centre, due to noise in the vision system and an imperfect static model.

The **Two Stage Servoing** method is the slowest (7.3s duration). While the arm could travel faster, the bounding box updates will then lag behind and the end effector may miss the lettuce. In the first stage the descent in Z is slower (between $t = 2.0 - 4.5s$) than the first method (Fig. 6a, centre right). Descending too rapidly runs the risk of the lettuce falling outside the field of view of the end effector camera, which then misses the target. In the second stage (between $t = 4.5 - 9,3s$) the descent is faster, but still limited by the need to arrive cleanly at the XY position of the lettuce before touching ground.

The **Learned Open Loop** is a compromise between the first two methods (4.0s duration). The first stage is open loop (between $t = 6.0 - 7.4s$) with rapid descent. The second stage uses the same algorithm and parameters as the Two Stage Servoing, using the bounding box BB_{ee} to drive the velocity control (between $t = 7.4 - 10.0s$). This second stage could potentially be further optimised for speed, sacrificing robustness to errors in the learned value of T. This would steepen the second stage curve (Fig. 6, bottom right) reducing overall travel time to close to the Open Loop value, but missing the lettuce in the face of extreme environmental noise.

Results on real robot: The trajectories on the real robot are slower, but follow the same pattern as in simulation. Open Loop is the fastest (4.8s duration), Two Stage Servoing is the slowest (8.9s duration) and Learned Open Loop is a compromise (6.2s duration).

The velocities of the different stages of the trajectories had to be adjusted downwards on the real robot for two main reasons. Firstly, more protective stops were triggered by the UR10 controller on the real robot than on the URSim software. This suggests that the URSim's simulation of the behaviour of the arm controller under real conditions is not exact. Secondly, the geometry of the real USB cameras differed from the simulated ones, meaning that the successful handover from overhead to end effector cameras was more sensitive to end effector velocity. The simulated cameras will be altered in future work.

Overall though, the three different control methods that were prototyped in simulation worked in reality, with minimal adjustments.

3.2 Experiment Two: Robustness to perturbations and distance

Experiment: The second experiment was designed to test the control methods' robustness to perturbations in the system and to the initial distance of the lettuce from the parked end effector.

Perturbations were added to the simulation by moving the UR10 arm off its usual alignment. In the simulator, a virtual ball joint was introduced under the arm and the yaw, pitch and roll modified. On the real robot, the platform itself was inclined while keeping the overhead camera support vertical. The magnitude of the perturbations is given by the IMU and measured in radians, referring to the yaw pitch and roll angles introduced.

On the real Vegebot platform, for practical reasons, the perturbations were modelled slightly differently. The end of the platform was jacked up to

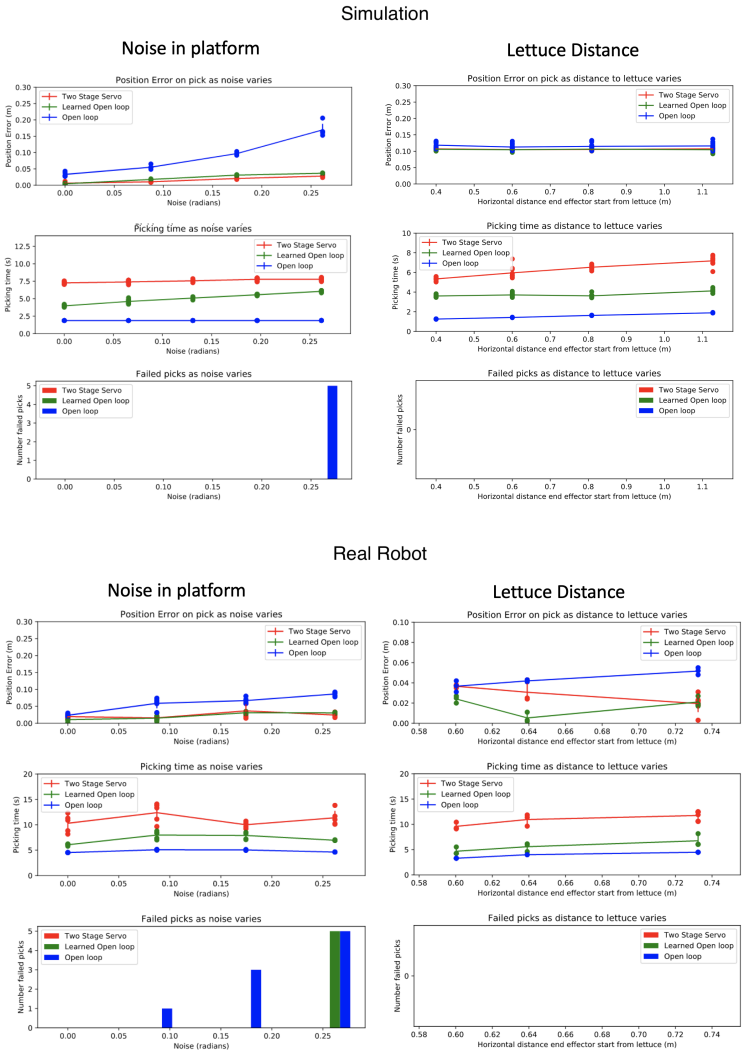

Fig. 7. Experiment Two: The three control systems in simulation and reality as perturbations are added (left) and the distance of the parked end effector to the lettuce is increased (right). The top graphs in each block show the final distance of the end effector centre from the lettuce. The middle graphs show the trajectory time. The lower graphs show failed picks from a total of 5 attempts.

incline it to a similar angle to the ball joint method. The overhead camera was then adjusted back to a vertical orientation. This is similar, but not geometrically identical to the method used in simulation. 5 sample picks were made for each combination of perturbation size and control method.

In separate tests, the distance of the lettuce from the parked en effector was varied to ensure that the methods worked over the range of the workspace. 5 sample picks were made for each combination of distance and control method.

Results in simulation: The results in simulation can be seen on the top half of Fig. 7). The position error of Open Loop degrades as the noise is increased, to the point where 100% of the picks fail when the noise reaches 0.25 radians. Learned Open Loop and Two Stage servoing are robust to the noise, as the visual servoing component brings the end effector back to the correct position. In terms of picking time though, Open Loop is fastest as once the initital lettuce localisation is performed, no further vision system updates are required. Two Stage Servoing is slowest, as it needs to move slow enough not to outpace the localisation update rate. In some ways, it is an over-cautious strategy: it would work even if the lettuce moved. Learned Open Loop, a combination of a fast open loop approach for stage 1 and slower servoing servoing in stage 2, resulting in a picking time between the two extremes.

The results for varying lettuce distance are as expected: there is no variation in picking error with distance from the parked end effector, suggesting that the projection model is accurate.

Results on real robot: Results on the real robot follow the trends seen in simulation. The increase in position error on Open Loop (from 3 to 9 cm) is not as extreme as in simulation: practical considerations prevented noise being applied to all three rotational exes. Learned Open Loop and Two Stage Servoing are robust to low noise, as in simulation, and the differences in picking time follow the same ordering. In general, it proved necessary to run the real robot at a lower velocity than the simulated version to prevent misses and protective stops.

Failures in Open Loop start at a lower noise level (0.10 radians), than in simulation (0.25 radians). In addition, Learned Open Loop fails as much as Open Loop does at the highest noise level (0.25 radians): the targeted positions are too far from the real ones to give the servoing time to compensate. Picking error distance increases with lettuce distance, suggesting inaccuracies in the projection model that grow towards the edge of the visual field.

4 Conclusion

All the tested methods show the expected trade-offs in simulation and in reality. **Open loop** harvesting is fast but vulnerable to pertubations and to model imperfections. **Two Stage Servoing** is more accurate and robust, but slower where the frequency of vision updates is constrained by the vision system hardware. The speed of the **Learned Open Loop** method lies between the other two, trading off some robustness to improve speed. There is an analogy here to human reaching, where an open-loop initial ballistic reach moves the wrist to the vicinity of the target object, followed by a more measured feedback-driven grasping process [10].

The Vegebot therefore has a number of picking strategies to choose from. Under normal circumstances, it should choose the fastest strategy that works. To the extent that it can measure its own picking error (or receive feedback from a human operator), it could adaptively switch between methods. This

suggests a combined strategy that would be resistant to slight perturbations and changes in body shape caused by wear and tear in the field.

Each time Vegebot successfully picks a lettuce, the final position could be added to a "successful pick buffer". Once this buffer reaches a certain size, the samples could be augmented through cloning and the neural network retrained. While 2000 training iterations were required to train the network (Fig. 6 (b)), only 50 successful pick samples were actually used on the real robot. This suggest that Vegebot could adapt quickly to wear and tear by falling back to a robust but slow method like Two Stage Servoing as soon as it starts detecting picking errors, gather a set of successful picks and then retrain itself. By using Learned Open Loop, the approach time of 20s can be reduced to 6.2s and so the overall cycle time decreased from 31s to 17.2s, a 45% improvement and closer to the target human value of 6s.

Acknowledgments: This project was possible thanks to EPSRC Grant EP/L015889/1, the Royal Society ERA Foundation Translation Award (TA160113), EPSRC Doctoral Training Program ICASE AwardRG84492 (cofunded by G's Growers), EPSRC Small Partnership AwardRG86264 (in collaboration with G's Growers), and the BBSRC Small Partnership GrantRG81275. Special thanks to G Growers, George Walker and Josie Hughes for their invaluable assistance.

References

1. Birrell, Simon and Hughes, Josie and Cai, Julia and Iida, Fumiya (2019) A field-tested robotic harvesting system for iceberg lettuce. Journal of Field Robotics. doi: 10.1002/rob.21888
2. Kusumam, Keerthy and Krajnik, Tomas and Pearson, Simon and Cielniak, Grzegorz and Duckett, Tom and others (2016) Can you pick a broccoli? 3D-vision based detection and localisation of broccoli heads in the field. IEEE
3. Hayashi, Shigehiko et al. (2010) Evaluation of a strawberry-harvesting robot in a field test. Biosystems engineering v105, 2. 160–171
4. Silwal, Abhisesh et al. (2017) Design, integration, and field evaluation of a robotic apple harvester. Journal of Field Robotics v34, 6. 1140–1159
5. Bac, C Wouter et al. (2014) Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. Journal of Field Robotics v31, 6: 888–911
6. Hua1, Yanbin and Zhang1, Nairu and Yuan1, Xin and Quan1, Lichun and Yang1, Jiangang and Nagasaka2, Ken and Zhou, Xin-Gen (2020) Recent Advances in Intelligent Automated Fruit Harvesting Robots. Open Agriculture Journal. doi: 10.2174/1874331501913010101
7. Chaumette, François and Hutchinson, Seth and Corke, Peter (2016) Visual Servoing. In: Springer Handbook of Robotics, 2nd Ed. Springer.
8. Mehta, Siddhartha S and MacKunis, William and Burks, Thomas F (2016) Robust visual servo control in the presence of fruit motion for robotic citrus harvesting. Computers and Electronics in Agriculture 123: 362–375
9. Cuevas-Velasquez, Hanz and Li, Nanbo and Tylecek, Radim and Saval-Calvo, Marcelo and Fisher, Robert B. (2018) Hybrid Multi-camera Visual Servoing to Moving Target. arXiv preprint arXiv:1803.02285
10. MacKenzie, Christine and Iberall, Thea. *The Grasping Hand* Elsevier, Amsterdam 1994.
11. Redmon, J., Farhadi, A. (2018). Yolov3: An incremental improvement. Retrieved from https://arxiv.org/abs/1804.02767