Mobile Manipulation Final Project

Gerry Chen and L. Carter Price

April 2020

Contents

1 Introduction

1.1 Problem description

Robotics can find many applications in agriculture due to the ability of robotics to perform monitoring and maintenance at a faster frequency, greater consistency, and lower cost than is possible by humans. The application we are interested in is tracking the growth of hydroponic lettuce by taking thousands of photos of a lettuce plant over its growth cycle and performing 4D spatio-temporal reconstruction to extract the plant's geometry. The current difficulty with our system is in positioning our camera to take high quality images of the lettuces. The proposed robot system consists of a 4-DoF arm mounted on the end effector of a planar cable robot which is placed in front of a wall of lettuce plants. Although the plants are arranged in a grid, each plant is subject to deviate from its expected location by several inches and may grow several more inches further in one direction than the other. The result is that positioning a camera to capture in-focus images of the entire lettuce (ie centered in frame without leaves cut off) is impossible without human tuning per plant or feedback control. We seek to use visual servoing to solve the problem of positioning a camera to consistently take properly framed and focused images of lettuce plants from multiple viewpoints. We will work in simulation. In our problem, we will consider the arm–cable system to be functionally equivalent to an arm on an omnidirectional mobile base.

1.2 Related Work

In the domain of agricultural monitoring, current quantitative measures for tracking plant growth are often destructive (i.e. harvest the plant to take measurements). Although non-destructive methods, also known as indirect methods, exist such as Leaf Area Index (LAI), they fail to achieve good accuracy due to numerous assumptions and parameters which vary species-to-species and region-to-region [11, 7]. Such methods perform especially poorly under leaf-leaf occlusion, which lettuce exhibits extremely strongly [5]. Therefore, high quality 4D reconstructions will be extremely helpful in non-destructively extracting the structure and visual properties of lettuce as proxies for data such as plant mass, health, nutrient uptake, photosynthetic efficiency, and harvest time. Within the realm of robotics, the topics covered in class regarding kinematics of mobile manipulator provides significant background. Reference material includes [6]. Visual servoing was also introduced in class, as well as in [2, 3]. Visual servoing specifically for photography has also been studied in drone controls and gimbals [4, 9].

1.3 Project contribution

In this project, we contribute an algorithm to help automate the collection high quality image sequences of plants as they grow for use in creating 4D reconstructions. For reference, previous data collection required a skilled 2-person team on average 12 minutes per plant, equating to 3 hours per day 3 days per week. Our contribution will allow the image collection to be completely automated while also allowing us to collect image sets of many more plants.

By using our approach of visual servoing to properly frame each image given an approximate guess of where the camera should be, we gain the ability to collect high quality image sequences without the need for a human to manually adjust the camera poses when the lettuces are not in the exact right location, when the lettuces grow (requiring the camera to move further away), and when the lettuces are irregular shapes. The system will be able to automatically correct for minor deviations in lettuce framing expectations for an overall more robust image capturing system.

In developing our algorithm, we make a number of assumptions about our environment and task. First, we assume consistent, even illumination across lettuces over time. This is a valid assumption because the intended use case is an indoor hydroponic lab where strong lighting is supplied from many directions with consistent power levels. We also assume that we have a good estimate of where the lettuce should be, and thus a good estimate of where the camera should begin. More formally, we assume that the lettuce will always begin inside the image frame. This is also a reasonable assumption because the lettuce is planted in a grid. Finally, we assume for purposes of simulation that kinematic motion commands are executed faithfully. We believe the ability of the system to correct for framing error will also translate to an ability to compensate for slight kinematic non-idealities.

When taking the image, we make further assumptions about what an ideal image is. We make the strong assumptions that the ideal image has the lettuce centered in the frame and also taking up a fixed width/height in the image. These assumptions are actually not completely realistic since, as the lettuce grows very large, it becomes impossible to capture high quality images that capture the whole lettuce and also capture detailed texture of the leaves. This texture is necessary for good structure-from-motion reconstructions. Therefore, it may be necessary to take multiple zoomed-in images of small-scale textures but each image not containing the entire lettuce. We believe our assumption does not detract from the utility of our algorithm because we can use our algorithm to first position the camera centered and perpendicular to the lettuce, then move towards the lettuce before taking the actual photo used in structure-from-motion.

The main focus of our project is to apply the concepts we learned in class about the kinematics of mobile robots and visual servoing to a real-world problem. We seek to investigate and assess the applicability of theoretical concepts of visual servoing in practical scenarios with regards to robustness to poor initial estimates, control stability, and imperfect kinematic/robot models.

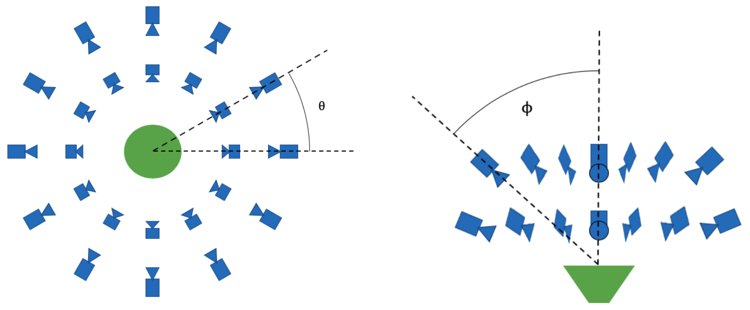

Figure 1: Top (left) and side (right) views of image capture locations - multiple "rings" of images are desired where each ring represents a fixed elevation defined by ϕ and changing azimuth defined by θ.

2 Approach/Research

2.1 Methodology

The mobile manipulator was moved to a known location in close proximity to the lettuce of interest. There were a series of pre-defined orientations that robotic arm would move to forcing a different point of view. From the specified location, visual servoing is performed to move the end-effector, with a camera attached, to a position where the lettuce was centered and the maximum width and height are roughly 80 percent of the pixel space in their respective dimensions. This process is repeated for each pre-defined location. The pre-defined locations are specified in a circular pattern with different angles from normal to the plane where the lettuce is attached, as shown in Figure 1. This allows for a variety of image angles which improves the results for structure-from-motion (SfM) algorithms. Each image may be at a different translation than predicted, but should be at the specified orientation given by θ and ϕ. The general programmatic flow is depicted in Figure 2.

2.1.1 Inverse Kinematics

The control of the 4 DoF arm on an $x/y/\theta$ mobile base was done using an analytical inverse kinematics solution and visual feedback for the estimated lettuce location.

The inverse kinematics of the arm on a mobile base can be computed by separating the arm and mobile base. The configuration of the arm can be seen in Figure 3 and consists of a 2 DoF shoulder joint, followed by two revolute joints (elbow and wrist) with parallel screw axes. The robot arm's end-effector can move with 3 translational degrees of freedom but only one angular degree of freedom: the yaw and roll of the end-effector are constrained given its translation, but the pitch can be independently controlled. However, we can al-

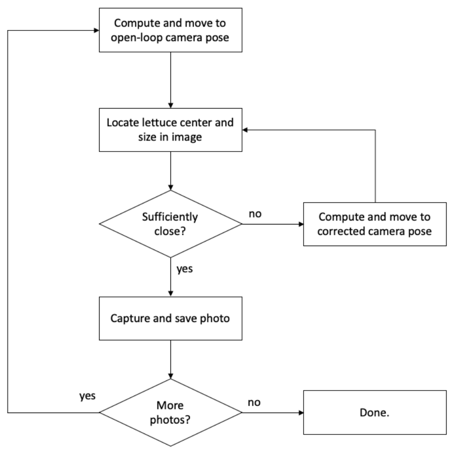
Compute and move to open-loop camera pose
Locate lettuce center and size in image
Sufficiently close?
no
Compute and move to corrected camera pose
yes
Capture and save photo
yes
More photos?
no
Done.

Figure 2: Flowchart of image collection algorithm

ternately interpret the end-effector as having 2 translational degrees of freedom in the plane defined by the yaw axis, and 2 rotational degrees of freedom in the yaw and pitch axes. This is an easier approach since the yaw axis directly corresponds with the yaw component of the shoulder joint. Thus, we have the following variables for the task and configuration spaces:

$$IK(\mathbf{x_{arm}} = [r, z, \theta, \phi]^T) = \mathbf{q_{arm}} = [q_1, q_2, q_3, q_4]^T \tag{1}$$

with (r, z) representing the position of the camera in the yaw plane. We also have the yaw angle of the shoulder joint:

$$q_1 = \theta \tag{2}$$

Proceeding according to the diagram in Figure 4, we seek to compute the inverse kinematics of the remaining 3 joints in the elbow-down configuration (this configuration minimizes collisions with the lettuce).

Given the pitch, we constrain the pose of the last end-effector link to derive the position of the wrist joint, labeled x_{wrist} in Figure 4. Defining link lengths of the upper arm, lower arm, and hand as L_1, L_2, L_3 respectively, the position of the wrist joint in the yaw plane is given by

$$x_{wrist} = \begin{bmatrix} r \\ z \end{bmatrix} - R_{L_3}^{\mathcal{W}} \begin{bmatrix} L_3 \\ 0 \end{bmatrix}$$

where $R_{L3}^{\mathcal{W}}$ is the rotation matrix of link L_3 in the world frame defined by $R(\phi + \pi/2)$. Intermediate variables α and β depicted in Figure 4 can be calculated

Figure 3: PincherX 100 Robot Arm by Trossen Robotics, photo credit Trossen Robotics [8]

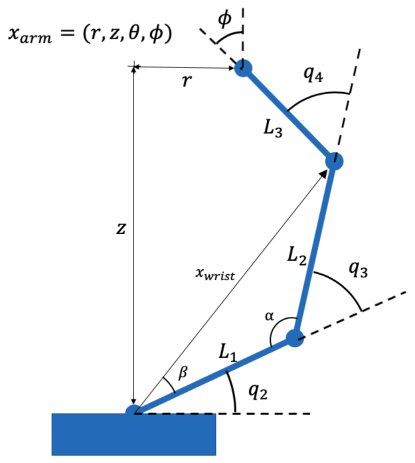

Figure 4: Diagram of the upper 3 degrees of freedom of the robot arm with variable labels

using the laws of cosines and sines, respectively:

$$\alpha = \arccos\left(\frac{L_1^2 + L_2^2 - ||x_{wrist}||^2}{2L_1L_2}\right)$$
$$\beta = \arcsin\left(\frac{L_2}{||x_{wrist}||}\sin\alpha\right)$$

Finally, denoting the convenience angle

$$\gamma \equiv \theta + q_2 = \texttt{atan2}(x_{wrist,z}, x_{wrist,r})$$

The upper 3 joint angles can then be represented:

$$q_2 = \gamma - \beta \tag{3}$$
$$q_3 = \pi - \alpha \tag{4}$$
$$q_4 = \phi + \alpha + \beta - \gamma - \pi/4 \tag{5}$$

Since we assume the mobile base has perfect control, the inverse kinematics of the mobile base is trivial:

$$IK(x, y, \theta) = [x_b, y_b, \theta_b]^T \tag{6}$$

Combining the inverse kinematics of the mobile base and arm, we notice that we actually do not have a full 6 DoF in the task space, lacking the roll of the camera. We thus choose to parameterize our task space coordinates as

$$\mathbf{x} = [x, y, z, \theta, \phi]^T \tag{7}$$

We use the optimization criteria of minimizing base motion since cable robot motions are less stiff and more energy intensive. Since the arm has full control of the end effector yaw, we can always set base angle to 0:

$$\theta_b = 0 \tag{8}$$

Additionally, since the arm has full control in the yaw plane, the base only needs to compensate for translation normal to the yaw plane:

$$\begin{bmatrix} x_b \\ y_b \\ 0 \end{bmatrix} = \mathbf{n}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \mathbf{n}\right) \tag{9}$$

Where $\mathbf{n} = [\sin\theta, -\cos\theta, 0]^T$ is a vector normal to the yaw plane.

Finally, we note that the arm is mounted on the base at position $[0, 0, b_0]^T$ where b_0 is the vertical offset of the arm from the base. Thus

$$\mathbf{x_{arm}} = \begin{bmatrix} ||(x, y) - (x_b, y_b)||_2 \\ z - b_0 \\ \theta - \theta_b \\ \phi \end{bmatrix} \tag{10}$$

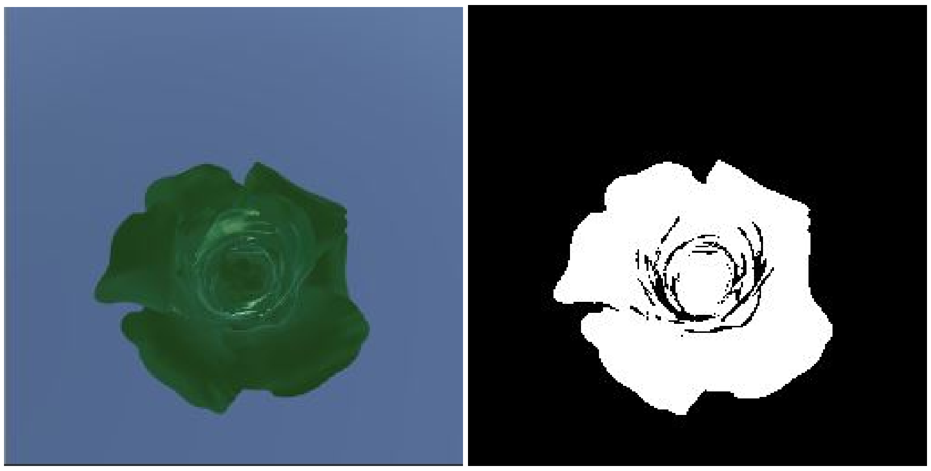

Figure 5: Original input image(left). Mask of the Lettuce after color threshold (right).

In summary, Equations 2–5 define the inverse kinematics equations for the arm and Equations 8 and 9 define the inverse kinematics equations for the base according to the configuration and task variables given by Equations 1 and 6 for the arm and base respectively. Equation 10 relates the task pose for the arm to the original task pose and the base configuration.

2.1.2 Computer Vision Algorithm

The image processing algorithm utilized the openCV python library [1]. The algorithm first converts the image into the Hue Saturation Value (HSV) color space from Blue Red Green (BRG). Then it performs a color threshold operation to capture a mask of all of the green values as seen in 5. The threshold values used to mask out the green pixels:

$$\text{Green Lower HSV} = [50, 50, 50]$$
$$\text{Green Upper HSV} = [70, 255, 255]$$

Subsequently, the contours of the mask image are found, and the center of the contour is calculated using the contour moments.

$$cX = m_{10}/m_{00} \tag{11}$$

$$cY = m_{01}/m_{00} \tag{12}$$

Finally, a bounding rectangle is calculated around the contour. The resulting image can be seeing in Figure 6.

The algorithm outputs the pixel location for the center of the lettuce as well as the height and width of the bounding rectangle which are used as to determine if the camera is an adequate distance from the plant.

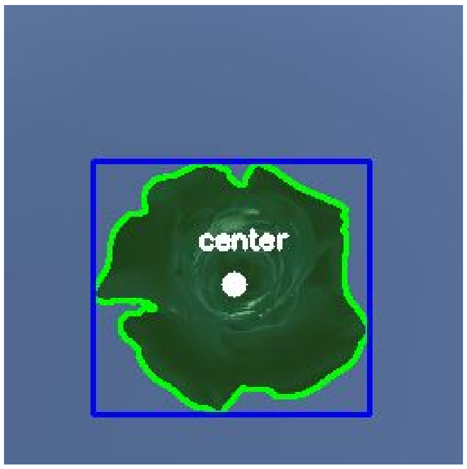

Figure 6: Resulting image annotated

2.1.3 Feedback Control Law

Given the coordinates and dimensions of the lettuce in the image, it is necessary to derive corresponding joint control signals to bring the lettuce to the center of the image and a fixed distance from the camera. We do this with the following 3-part control loop:

1. Back-project the estimated center location of the lettuce from the image to the camera frame
2. Transform the lettuce position to the world frame
3. Move towards the correct camera pose defined by a certain distance and orientation from the lettuce

To back-project the lettuce center location, we estimate depth from the dimensions of the lettuce. This is a reasonable assumption because the correct depth from the lettuce is the depth such that the lettuce is cropped in the frame properly. Alternative measures that will be investigated in real-life experimentation in the future include estimating depth from image focus, by stereo disparity from successive images, and by external sensor such as IR RGB-D, LIDAR, or ultrasonic rangefinder. Given a rectified image, we can calculate the back-projected coordinates of the lettuce in the camera/end-effector frame,

$t_l^{EE} = [x_l^{EE}, y_l^{EE}, z_l^{EE}]^T$, as

$$z_l^{EE} = D \cdot \min\left(\frac{f_x}{w}, \frac{f_y}{h}\right) \tag{13}$$

$$x_l^{EE} = (cX - c_x)\frac{z}{f_x} \tag{14}$$

$$y_l^{EE} = (cY - c_y)\frac{z}{f_y} \tag{15}$$

where cX, cY represent the center of the lettuce in the image in pixels, w, h represent the size of the lettuce in the image in pixels, D represents the assumed diameter of the lettuce in meters, and f_x, f_y, c_x, c_y are from the camera intrinsic matrix. We can then calculate the error, e^{EE},

$$e^{EE} = t_l^{EE} - \begin{pmatrix} 0 \\ 0 \\ d^* \end{pmatrix} \tag{16}$$

where d^* represents the desired distance from the lettuce to the camera.

To transform to the world frame, we can construct the end effector transformation matrix, $T_{EE}^{\mathcal{W}}$,

$$\begin{aligned} R_{EE}^{\mathcal{W}} &= R_z(\theta)R_y(\pi/2 - \phi) \\ T_{EE}^{\mathcal{W}} &= \begin{pmatrix} & & & x \\ & R_{EE}^{\mathcal{W}} & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{17}$$

We then compute the error between actual end-effector position and desired end-effector position, $-e^{\mathcal{W}}$, in world coordinates as

$$e^{\mathcal{W}} = T_{EE}^{\mathcal{W}} e^{EE} \tag{18}$$

and finally compute the new IK goal pose (variables defined in Eq. 7) as

$$\mathbf{x}_{k+1} = \begin{pmatrix} x_k + K_p e_x^{\mathcal{W}} \\ y_k + K_p e_y^{\mathcal{W}} \\ z_k + K_p e_z^{\mathcal{W}} \\ \theta \\ \phi \end{pmatrix} \tag{19}$$

where K_p is a gain term to limit excessively aggressive large movements.

We note that in place of the interaction matrix, L, popularly used in image-based visual servo literature, we directly compute the error using camera back-projection because we assume perfect position control of our mobile robot. Rather than controlling the robot via Jacobian-following to elicit a specified twist of the end-effector, we have a closed-form inverse kinematic solution for the mobile robot and therefore we can directly command the robot to move to the correct position. Alternatively stated, rather than relating the lettuce image error change and the camera twist, we instead directly compute the new camera pose because we have an analytical inverse kinematic expression. Nonetheless, Equations 13–15, bear a very clear resemblance to the terms in the interaction matrix for image-based visual servo from [2].

Also note that angular velocity of the end effector is constrained to zero since each image must be taken at a predefined angle in the world frame (not the lettuce frame). Thus, we never apply an angular velocity to the end effector and those terms in the interaction matrix are moot. Using the variable convention in [2], the angular velocity components of ξ are set to 0.

2.2 Implementation

The above methodology was implemented in simulation using the Unity Gaming Engine [10]. This method of simulation was selected for its advanced lighting and rendering capabilities which offer a photo-realistic results that would be of greater value for simulating structure-from-motion down the road when compared with other simulator options.

2.2.1 Simulation

The Unity gaming engine allows models to be defined graphically using primitive shapes like cylinders and rectangles. The px100 robot was modeled using these primitive objects mounted on a circular mobile base. The axes of rotation were defined using nested XYZ axes. Each axis is defined relative to a parent axis which allows for a straight forward implementation of forward kinematics by simply defining the joint angles. Also, a virtual camera was placed at the end-effector of the robot. In addition to the robot construction, a simple world with walls and ground were created. A model of a plant was also imported and used.
The primary implementation code was all written using the python programming language. Unfortunately, Unity was created to be used with C programming language. Therefore, a TCPIP inter process connection was created to communicate between the python scripts and the Unity simulation. The server side was created in python while the client was written inside the Unity framework in C#. The communication simply sends the desired position of the mobile base and joint angles as well as a flag to indicate when to capture an image from the end-effector camera. The decision for position control was made because that is how the robot which is being used for the physical application is controlled.

In effect, only the kinematics of the robot were being controlled in contrast to many physics simulators which utilize some form of "joint effort." However, we feel this is a justified method as the resulting task for structure-from-motion is largely independent of dynamics.

When the images are virtually captured in the Unity game engine simulation, they are saved to a local file. The python script then reads and immediately

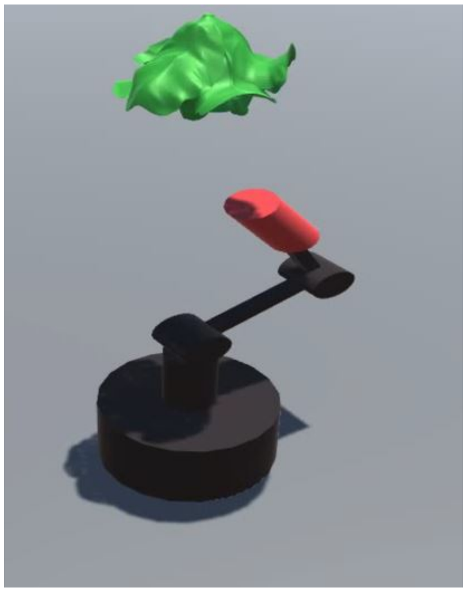

Figure 7: The Unity game engine simulation environment. Showing constructed robot and lettuce.

deletes that existing file. While this is not an ideal way to transfer the images between the applications, it was an effective method for this first iteration.

2.3 Experimental Setup

To test the robustness of the approach, an experiment was conducted where the plant was perturbed from the expected nominal starting position by some distance in the XY plane and then run through the visual servoing from a set of 12 different starting orientations of the manipulator arm. The 12 starting positions increased θ from 0 to 360 degrees by 30 while ϕ is held constant at 30 degrees, refer to figure 1. 8 different offset distances (distance from where the plant is expected to be) were tested. Figure 8 helps visualize the offset lettuce. For each offset distance, the number of successful completions are tracked. Where a completion is constituted by converging to the desired image from the starting orientation. This process is shown in figure 9.

3 Results and Discussion

3.1 Results

The test results shown in Table 2 give the XY location of the plant in the plane and the number of visual servo attempts successfully completed. The results show that the algorithm is relatively robust until the plant is more than 0.7

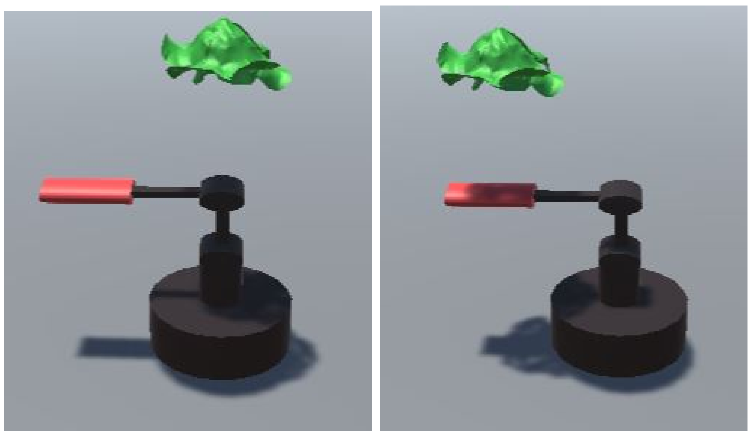

Figure 8: Shows the results of the position offset in the simulation. (Left) the lettuce is in the expected location. (Right) it is 0.8 offset in the Z direction.

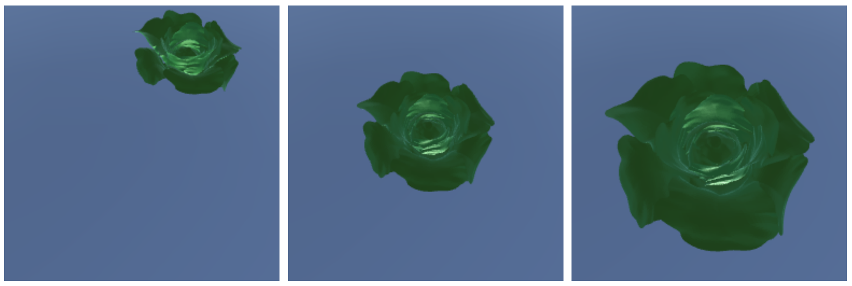

Figure 9: Shows sub-sampled sequence of images from starting orientation (left) to convergence (right) during the visual servo operation. On average, it took 28 iterations to converge on the goal image.

Table 1: Parameter values used in experiments.

L_1	L_2	L_3	b_0	D	$f_x = f_y$	$c_x = c_y$	d^*	K_p
3 m	3 m	3 m	1 m	1 m	221.7 px	128 px	1.2 m	0.9

Table 2: Test Results

trial	plant position	complete [max=12]
base	[0, 0]	12
1	[0, 0.2]	12
2	[0, 0.5]	12
3	[0.7, 0]	12
4	[0.8, 0]	10
5	[0.9, 0]	9
6	[1.0, 0]	6
7	[1.2, 0]	2
8	[1.5, 0]	2

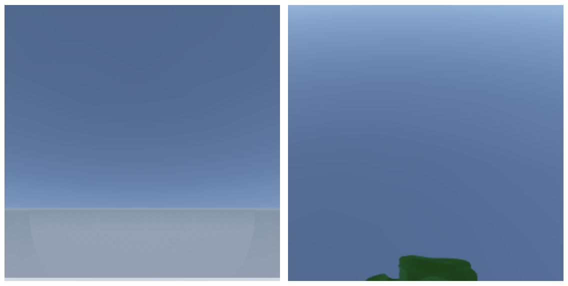
Figure 10: Failure instances: Mode 1 - the plant is not found in the image. Mode 2 - The plant is partial seen resulting in a high initial error and uncontrolled motion.

outside of its expected position. Beyond this point there are a few attempts where the starting orientation returns an initial image that results in a couple failure modes: 1. the plant is not found in the frame and 2 the plant is so far on the edge that the high pixel error results in an unstable movement. The unstable movement eventually leads to a pose and image where the plant is no longer found. Examples of of the failure modes can be found in Figure 11.

Tables 3 and 4 illustrate the contrast between open loop position command images and those using the visual servoing when the lettuce is offset by 0.5. The visual servo images are nicely centered an zoomed in to the appropriate scale, while the open loop method results in images where the lettuce is partially out of frame or too small.

Here is a link showing an abbreviated sample run:
http://www.youtube.com/watch?v=A6Lk_ifYphQ

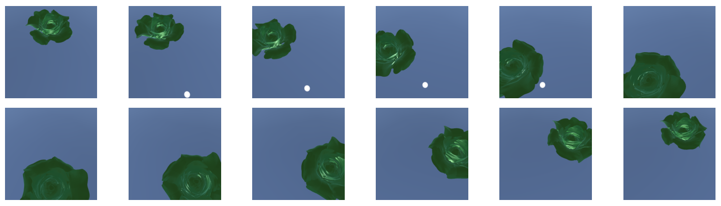

Table 3: Images open loop - Plant at [0.5, 0]

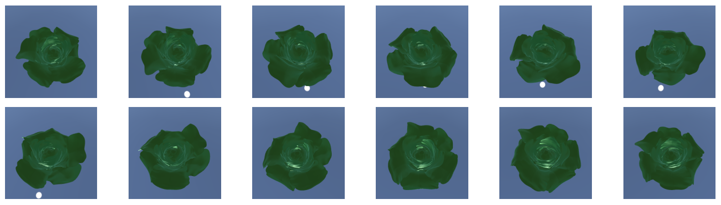

Table 4: Images using visual Servoing - Plant at [0.5,0]

3.2 Discussion

While a somewhat naive implementation, our algorithm shows promise for successfully adapting to the uncertainty in the position of the plants when collecting images for structure-from-motion.

In future work, we would like to look at improving the vision system so that it is more robust. The color threshold as well as the contour detection relies heavily on the clean images (consistent and contrasted background with the ideally colored green plant) that we collect in simulation. More robust algorithms, better outlier detection, or a machine learning based plant detector would likely help make the visual servoing robust. Additionally, we would like to improve the inverse kinematics to more accurately reflect the robot configuration, including link geometry, joint limits, and collision. Workspace determination poses a particularly interesting problem because, although the size, placement, and setpoint image poses in our experiments didn't cause us to approach the arm's joint limits / workspace boundaries, this is not necessarily a valid assumption as the lettuce grows very large and we may need to make greater use of the mobile base to reach around large plants.

In addition to upgrading the IK, another point of improvement would be to use previous lettuce pose estimates to inform future open-loop camera pose estimates. Currently, the open-loop camera poses are all pre-specified at the start of the program, but each time we perform a visual servoing update, we can store a measurement of the lettuce pose (ie in GTSAM) so that future open-loop camera poses will be closer to the correct pose and require less visual servo action.

We would also like to utilize joint effort control to more accurately simulate a real world controls scenario. The sharing of data between the Unity game engine

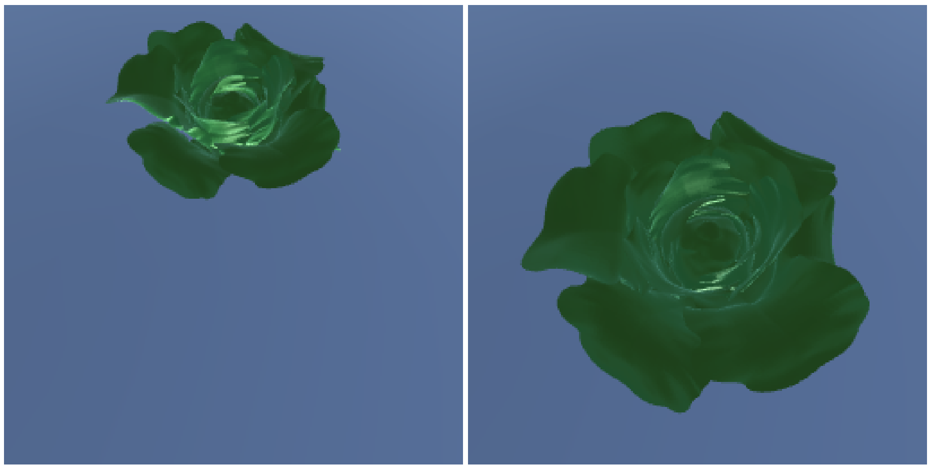

Figure 11: This figure compares the quality of the image captured using the open loop method(left) versus an image from implementing visual servoing (right). The IBVS method results in an image of the plant where the planted is centered and occupying a majority of the image frame. On the other hand, the open loop method captures an image where the plant is off-center and distant.

simulation and the python script could also be improved saving the images to file then reading them into python is not the fastest or most reliable method for data sharing; the images could be sent via a TCPIP connection on a different port for faster and more reliable data transfer. Lastly, future work will include an upgrade to the Unity game engine simulation to have higher quality rendering, and the models will be upgraded and adjusted to more accurately reflect the real world application.

3.3 Meta-learning

In working on this project to apply visual servoing to camera placement for lettuce structure from motion, we applied topics learned in class, including robot kinematics, coordinate transformations, and visual servoing, to a real-world application in a simulated environment.

In coding the inverse kinematics of the 4 DoF arm on a mobile base, we were able to see how redundancy affects the way we plan for robots with mobile bases which have infinite range and equations which are different than those of traditional revolute serial manipulators. In particular, we saw how we can exploit the additional degrees of freedom garnered by the mobile base to augment lacking degrees of freedom of the robot arm. We also saw how additional kinematic objectives can be fulfilled using the redundancy of our mobile robot. In our case, our secondary kinematic objective was to minimize base movement, whose motivation lies in the relative difficulty for base movement as compared to arm movement.

Implementing the image processing and visual servo feedback taught us how powerful, robust, and forgiving these techniques can be despite our relatively non-rigorous implementations. The visual servo feedback in particular was found to be remarkably successful at centering the lettuce in the frame even in the presence of noisy lettuce placement, robot geometry, and image processing. This bodes very well as we work towards testing on a real robot and dealing with the sim-to-real gap.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] F. Chaumette and S. Hutchinson. Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine*, 13(4):82–90, Dec 2006.

[3] F. Chaumette and S. Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics Automation Magazine*, 14(1):109–118, March 2007.

[4] Yi-Ling Chen, Wei-Tse Lee, Liwei Chan, Rong-Hao Liang, and Bing-Yu Chen. Direct view manipulation for drone photography. In *SIGGRAPH Asia 2015 Posters*, SA '15, New York, NY, USA, 2015. Association for Computing Machinery.

[5] Ronghai Hu, Guangjian Yan, Xihan Mu, and Jinghui Luo. Indirect measurement of leaf area index on the basis of path length distribution. *Remote Sensing of Environment*, 155:239 – 247, 2014.

[6] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[7] Antonio MartÃnez Ruiz, Irineo L. LÃ-Cruz, AgustÃn Ruiz-GarcÃa, Joel Pineda-Pineda, and J. VÃctor Prado-HernÃ¡ndez. HortSyst: A dynamic model to predict growth, nitrogen uptake, and transpiration of greenhouse tomatoes. *Chilean journal of agricultural research*, 79:89 – 102, 03 2019.

[8] Trossen Robotics. Pincherx 100 robot arm.

[9] Fadjar Triputra, Riyanto Bambang, Trio Adiono, and Rianto Sasongko. A nonlinear camera gimbal visual servoing using command filtered backstepping. *Journal of Unmanned System Technology*, 3:49–60, 11 2015.

[10] Unity Technologies. Unity.

[11] W. W. Wilhelm, K. Ruwe, and M. R. Schlemmer. Comparison of three leaf area index meters in a corn canopy. *Crop Science*, 40(4):1179–1183, 2000.