2950 Niles Road, St. Joseph, MI 49085-9659, USA
269.429.0300 fax 269.429.3852 hq@asabe.org www.asabe.org

*An ASABE Meeting Presentation*
***DOI: https://doi.org/10.13031/aim.201900779***
***Paper Number: 1900779***

# *Visual Control of Cotton-picking Rover and Manipulator using a ROS-independent Finite State Machine*

***Kadeghe Fue[1,4], Edward Barnes[2], Wesley Porter[3], Glen Rains[4]***
***[1]School of Electrical and Computer Engineering, University of Georgia, Athens, GA***
***[2]Cotton incorporated, Cary, NC***
***[3]Department of Crop and Soil Sciences, University of Georgia, Tifton, GA,***
***[4]Department of Entomology, University of Georgia, Tifton, GA***

**Written for presentation at the**
**2019 ASABE Annual International Meeting**
**Sponsored by ASABE**
**Boston, Massachusetts**
**July 7–10, 2019**

***ABSTRACT.*** **Small rovers are being developed to pick cotton as bolls open. The concept is to have several of these rovers move between rows of cotton, and when bolls are detected, use a manipulator to pick the bolls. To accomplish this goal, each cotton-picking robot needs to accomplish three movements; rover must move forward/backward, left/right and the manipulator must be able to move to harvest the detected cotton bolls. Control of these actions can have several states and transitions. Transitions from one state to another can be complex but using ROS-independent finite state machine (SMACH), adaptive and optimal control can be achieved. SMACH provides task level capability to deploy multiple tasks to the rover and manipulator. In this research, a cotton-picking robot using a stereo camera to locate end-effector and cotton bolls is developed. The robot harvests the bolls using a 2D manipulator that moves linearly horizontally and vertically. The boll 3-D position is determined by calculating stereo camera parameters, and the decision of the finite state machine guides the manipulator and the rover to the destination. PID control is deployed to control rover movement to the boll. We demonstrate preliminary results in a direct-sun simulated environment. The system achieved a picking performance of 17.3 seconds per boll. Also, it covered the task by navigating at a speed of 0.87 cm per second collecting 0.06 bolls per second. In each mission, the system was able to detect all the bolls but one.**

***Keywords. Cotton-picking, Cotton harvesting, SMACH, Robot***

# Introduction

The cotton industry holds an important position as a commercial crop worldwide, especially in the U.S, China, India, and Brazil. Cotton production in the United States has been growing and is now the 3rd largest agricultural industry in the US, employing more than 200,000 people with a value of $25-billion per year (USDA/NASS, 2017). The U.S is third in production of cotton in the world behind India and China. As a large industry, however, it has several challenges. Among the biggest is cotton harvesting. Timely harvesting of the quality cotton fiber is among the most pressing challenges in the cotton production industry. The current practice of mechanical harvesting after defoliation leads to huge loses in the industry since its inception in the 1950s (Fue et al., 2018). The open bolls can sit 40 days waiting to be picked since it is advised to pick the cotton after at least 60% to 75% of the cotton bolls are opened (UGA, 2018). This waiting time exposes the open bolls to harsh conditions that degrade their quality. Any solution that would reduce cotton losses and improve quality would be welcomed by the industry. In most cases, the mechanical combine machines are very big and expensive (the new 2019 John Deere CP 690 picker costs around $725,000). Unfortunately, the expensive cotton pickers are stored for more than 9 months a year. Also, the machines weigh more than 33 tons causing soil compaction which reduce land productivity. Maintenance of such machines is also expensive and complicated. Breakdowns in the field can take days, reducing operating efficiency and exposing bolls to further weather-related quality degradation. Also, the machines need defoliated plants to harvest, which adds expense to the farmer (UGA, 2018).

Hence, in this study, a 2DOF cartesian arm that moves vertically and horizontally is developed, and kinematics of the arm is analyzed. Kinematic control of the robot arm is developed by using an artificial neural network so that it becomes easier for farmers to automatically recalibrate the machine every time before using it. Also, the PID control is developed to enhance the robot moves along the crop rows.

# Materials and Methods

### System Setup

An autonomous 4-wheel articulated-steer rover with an attached single 2D manipulator is proposed in this study. The system (Figure 1) has a master controller which obtains images and depth maps from the stereo camera, rover controller to control the rover movement and arm controller to control the end-effector manipulator. With this configuration, the robotic system is able to do translational movements only. The research red rover (Figure 4) used in this study is a custom-built articulated vehicle (West Texas Lee Corp., Lubbock, Texas) that has been modified to be remotely controlled using embedded systems or autonomously controlled to navigate in an unstructured agricultural environment (Rains et al., 2015).

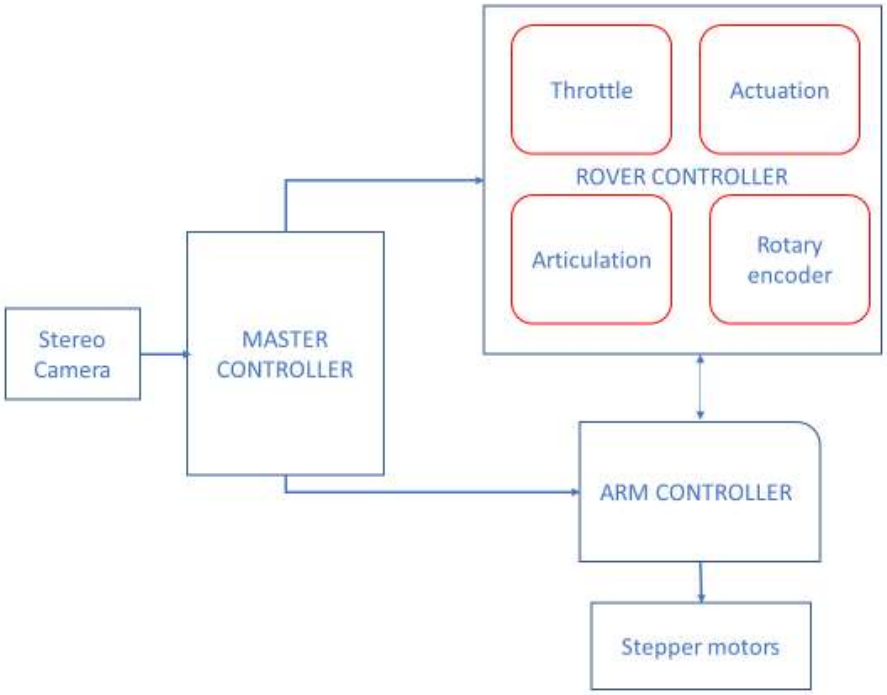


*Figure 1. A contextual block diagram of the robotic system hardware*

Research rover tires are controlled using a servo to the engine throttle and a servo to the variable-displacement pump swashplate servo. The front tires are connected to a rotary encoder (1024 P/R (Quadrature), Karlsson Robotics, Tequesta,

FL, USA) to provide feedback to the adaptive PID to control movements of the rover along the crop row (Figure 2). The rover controller which is Mega Arduino controller is used to collect data from the encoder and feed output to servos for PID position control of the rover movement. The throttle is provided by the onboard Kohler Command 20HP engine (CH20S, Kohler Co, Wisconsin, USA) with a maximum of 2500 RPM and controlled by an axial-piston variable rate pump (OilGear, Milwaukee, WI, USA) with a maximum displacement of 14.1 cc/rev. The OilGear pump is connected with a swashplate for directing the forward and rearward movement of the rover. The swashplate angle is controlled by the rover controller that determines the distance of the linear electric servo (Robotzone HDA6, Servocity, Winfield, KS) that can move approximately 20cm. Left/right articulation is controlled by using linear relays that push hydraulic actuators powered by a 0.45 cc/rev fixed displacement pump (Bucher Hydraulics, Italy) and a 4-port 3-way open-center solenoid valve. The rover can turn to a maximum of 45 degrees with a wheelbase of 190 cm. Also, the rover's height can be adjusted to a maximum clearance of 122cm and a maximum width of 234 cm.

*Figure 2. Encoder installed on the front tire of the rover to give the feedback input pulses which indicates how far the rover has moved from one point to another*

**Master Controller Imaging System**

An embedded kit (NVIDIA Jetson TX2 development kit, Nvidia Corp., Santa Clara, CA, USA) was installed on the research rover and, together with machine vision software, used to extract features of cotton boll images and determine the 3D position of the boll relative to camera and ground (Fue et al., 2018). NVIDIA Jetson TX2 (NVIDIA Pascal 256 CUDA cores, Quad ARM and HMP Dual Denver CPU, 8GB 128-bit LPDDR4 RAM, 32GB eMMC SATA drive) with ZED SDK installed was used to provide high graphics computing resources for fast image analysis. An RGB stereo camera (ZED camera, Stereo labs Inc, San Francisco, CA, USA) was installed and used to acquire images. ZED is 175 x 30 x 33 mm and weighs 159g. ZED has a 4M pixel sensor per lens with large 2-micron pixels. The left and right sensors are 120 cm apart. ZED was chosen due to the nature of the tasks, such as needing to work outdoor and provide depth data in real-time. ZED camera provides a 3D rendering of the scene using the ZED software development kit (SDK) which is compatible with other platforms like Robot operating system (ROS), OpenCV library, MATLAB, and Unity. ROS was chosen since it provides all the services required for robot development like device drivers, visualizers, message-passing, package design, and

management and hardware abstraction (ROS, 2017 and Fue et al., 2018). ROS was initiated remotely by using a client machine and images were acquired using the ROS topics feature provided by the ZED wrapper. Images were parsed to the processing unit and analyzed using OpenCV (version 3.3.0) machine vision algorithms.

The ZED stereo camera system was mounted on a research rover (Rains et al., 2015) at 90° below the horizontal (means it was directly pointing downward) and took images at the rate of 60 frames per second at WVGA quality while the rover was stationary. The distance of the camera to the ground was 220 cm.

**Robotic Arm Controller**

Robot arm controller (Arduino Mega 2560, Arduino LLC) receive a 4-bytes digital signal from the Jetson TX 2. The signal provides the number of steps and direction of the arm (Up, Down, back and forth). Then, the controller sends the signal to micro-stepping drive which in turn sends to the appropriate stepper for action. Arm controller is connected to the Jetson using a USB 3.0 hub shared by the ZED camera. The micro-stepping drive that controls the motors were set to run a step pulse at 2MHz and 400 steps per revolution. This setting provides smooth motion for the arm.

In this study, a robotic arm was designed to work as a 2D cartesian system (Figure 3) and developed using two 2-phase stepper motors (MS048HT2 and MS200HT2, ISEL Germany AG, Eichenzell,Germany). MS048HT2 model was installed to run the horizontal linear axis (60 cm long), and MS200HT2 was installed to run the vertical linear axis (190 cm long). The connecting plates and mounting brackets use a toothed belt that is driven by the stepper motor to move back and forth (Figure 2). Two stepper drives (Surestep STP-DRV-6575 micro-stepping drive, Automation Direct, Cumming, Georgia) were installed so as to provide accurate position and speed control with a smooth motion. The DIP switch of the drive was set to 400 steps per revolution. The arms are connected as the vertical crossbench. The sliding toothed belt drive (Lez 1, ISEL Germany AG, Eichenzell,Germany) is used to move the end effector. The toothed belt has 3 mm intervals, width 9 mm attached to an aluminum miniature linear guide and can move at 150 cm per second. The error of the toothed belt is about +/- 0.2 mm. A wet/dry vacuum was installed on the red rover to help pick and transport the picked cotton bolls. The vacuum connects to the end-effector via a flexible 3" plastic tube. Cotton bolls are vacuumed into the end-effector which is placed close to the cotton bolls (Figure 3). Then, the cotton bolls move to a storage container that is connected to the vacuum.

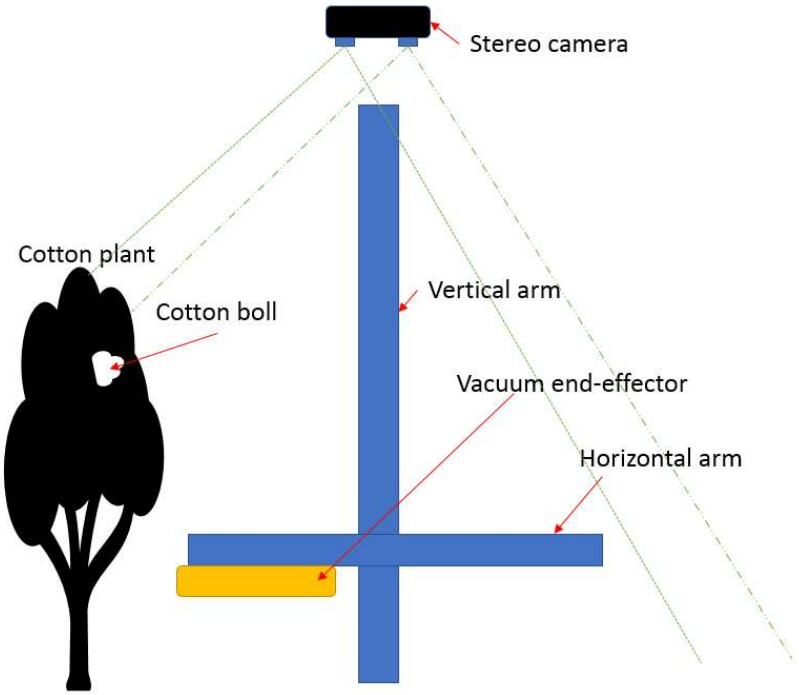


*Figure 3. Robotic cartesian arm contextual diagram*

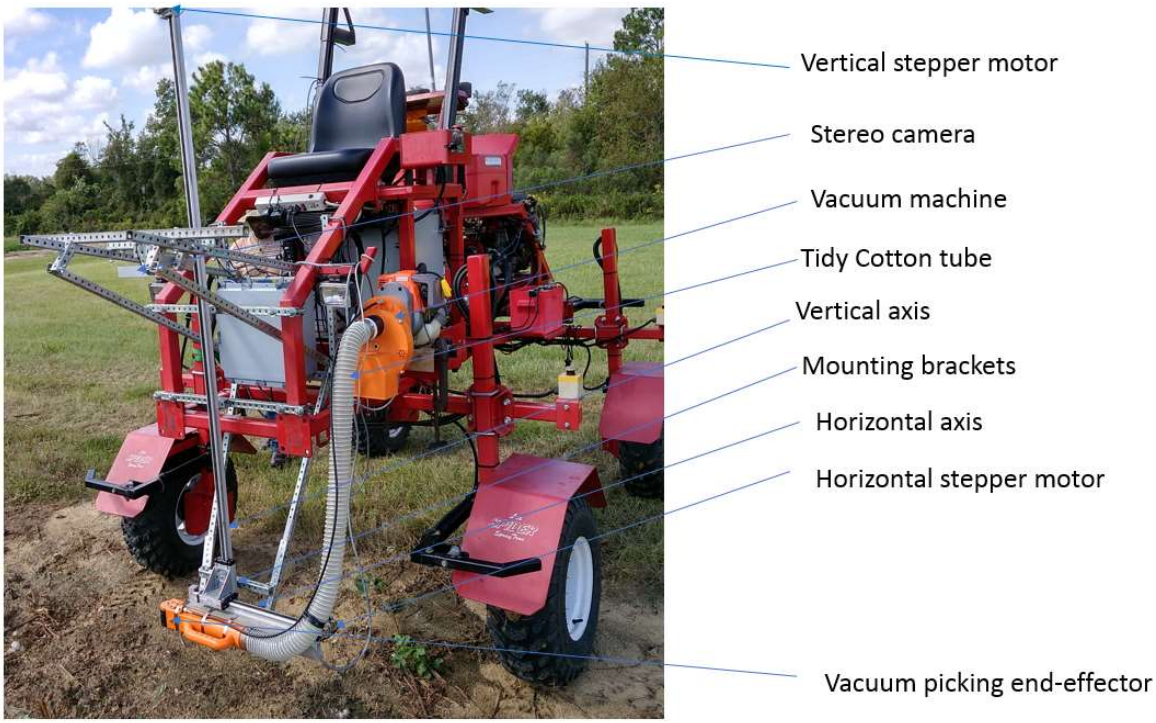


*Figure 4. The robotic arm mounted on a red research rover*

*Figure 5. The end-effector comes close to the cotton bolls and sucks them using vacuum suction*

**Data acquisition, Boll Image, and end-effector features Extraction**

Each image frame was acquired using ZED camera and analyzed by the algorithm developed using a 4-step machine vision algorithm (1. depth processing, 2. color segmentation, 3. feature extraction and 4.depth matching with the features). These steps are handled by the graphics optimized rugged development kit (NVIDIA Jetson TX2) to achieve improved performance as image calculations required massive graphics computing resources like NVIDIA CUDA cores. The ZED SDK acquired the images and processed them to get depth disparity and rectified images for both lenses. In this case, the ZED SDK was providing 30 fps for WVGA quality images and depth maps.

The images acquired (Figure 4a) were first analyzed for arm movements. Since the arm is orange in color, the threshold color was determined to segment the image to obtain only the arm (Figure 4b). The cotton boll and end-effector segmentation task involved four steps (Gong and Sakauchi,1995):

1. Grab an image
2. Using RGB color threshold, separate each RGB component of the image. For cotton bolls, the white components of the image can be masked (all pixels with the value of Red, Blue, and Green greater than 220).

And for the end-effector, the orange can be masked (Red from 200 to 255, Green from 0 to 255 and Blue from 0 to 50).

3. Subtract the image background from the original image.
4. Remove all the region where the contours are less than value M. Value M can be determined by estimating the number of pixels defining the smallest boll.

Then, feature extraction is done by finding contours of the continuous points which have the same intensity and are clustered. Color masking of the grey image was performed, then boundary curves were applied to detect and distinguish all white pixels of the image (Fue et al., 2018). The cotton boll then can be obtained after segmenting the contour of the arm (Figure 4c and 4d).

After, obtaining the contours for each of the objects (arm and bolls), the segmented image is matched with depth disparity. All the depths are calculated for bolls and arm.

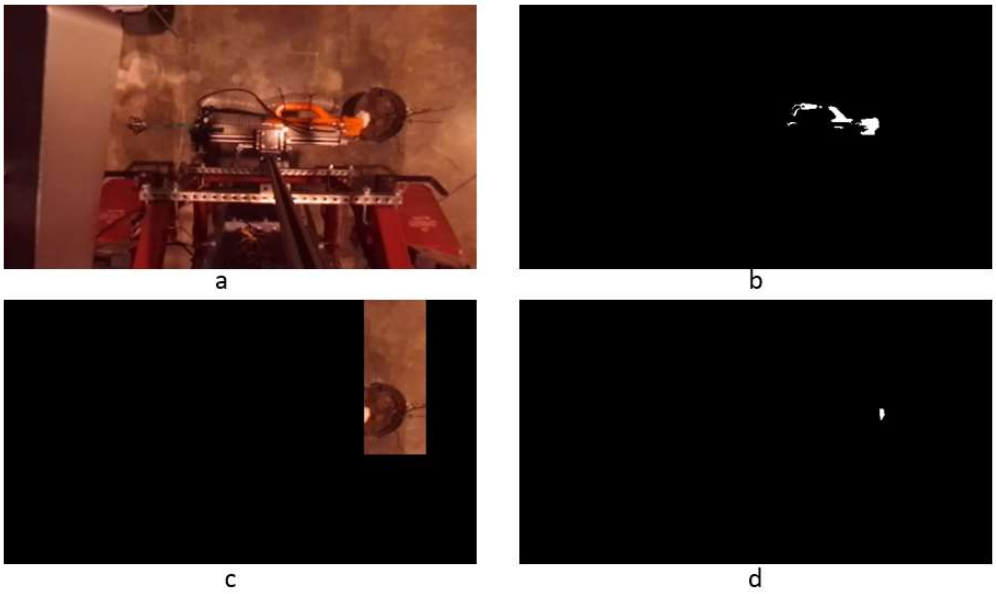


*Figure 6. Color segmentation to get the arm and cotton boll*

**Depth and coordinates of cotton bolls and arm determination**

After matching the depths and contours of the arm and bolls then each reading of the arm and boll positions is logged. Then, by using the tip of the arm (Figure 4c), the system gets the image coordinates of the front part of the arm. Then, using centroids of each boll, the system takes a reading of the most variable depths for each contour which may represent separate bolls (Figure 4d). The system calculates the real-world coordinates (W) from the image coordinates obtained (I) by using image geometry. Since the camera was calibrated using the ZED Calibration tool of the ZED SDK, the camera matrix (equation 1) values were obtained. The procedures to calibrate ZED camera can be obtained from their website. The camera matrix consists of **fx**, and **fy** (the focal length in pixels), C**x** and **Cy** (the optical center coordinates in pixels), and **k1** and **k2** (distortion parameters). This means the real-world coordinates of a cotton boll, Wx and Wy (Equation 3 and 4) can be obtained if we know the value of Ix and Iy which is the coordinate of the centroid of the front part of the arm. Alternatively, by finding the inverse of the camera matrix and multiply with Vector Image (I), the world coordinates can be obtained. Cx, fx, Cy and fy are found by the calibrated camera matrix while Wz can be found from the depth disparity map provided by the ZED SDK.

$$C = \begin{Bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{Bmatrix} \quad (1)$$

$$\begin{Bmatrix} I_x \\ I_y \\ 1 \end{Bmatrix} = \begin{Bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{Bmatrix} * \begin{Bmatrix} W_x \\ W_y \\ W_z \\ 1 \end{Bmatrix} \quad (2)$$

$$W_x = (I_x - Cx) * \left\{\frac{W_z}{f_x}\right\} \quad (3)$$

$$W_y = (I_y - Cy) * \left\{\frac{W_z}{f_y}\right\} \quad (4)$$

where

**fx**, and **fy** = the focal length in pixels,

C**x** and **Cy** = the optical center coordinates in pixels,

$\mathbf{k_1}$ and $\mathbf{k_2}$ = distortion parameters,

$\mathbf{W_x}$ and $\mathbf{W_y}$ = the real world coordinates, and

$\mathbf{I_x}$ and $\mathbf{I_y}$ = the coordinate of the centroid of the front part of the arm.

After obtaining such measurements, the system can execute other tasks like controlling the arm. For the machine to be able to execute each task separately but in connection to the other tasks, the finite state machine was developed.

**Finite State Machine (FSM) using SMACH**

FSM moves from one state to another after it is triggered by certain input. The states in the system are detecting the boll, moving the vehicle forward or back after obtaining the position of the vehicle relative to the cotton boll, moving the arm up or down or back and forth after obtaining the boll position and arm position. The FSM guides the system from harvesting the first boll to the last one. The robot starts by detecting the boll (detect_boll) and then decides to move the vehicle (get_vehicle_pos) to the boll to harvest it. Then, when in line with the boll, the arm moves up (move_up) or down (move_down) and match the vertical distance of the boll and the arm. If end-effector and boll are already vertically matched, then, the arm moves horizontally to harvest (harvest_boll) the boll (Figure 7). The arm always starts with the boll which is at the highest distance from the ground and harvest other lower located bolls after picking the upper bolls. This allows the system to see more bolls below the upper bolls. The system moves the end-effector back (move_back) to avoid obstruction of the camera view of the plants and bolls.

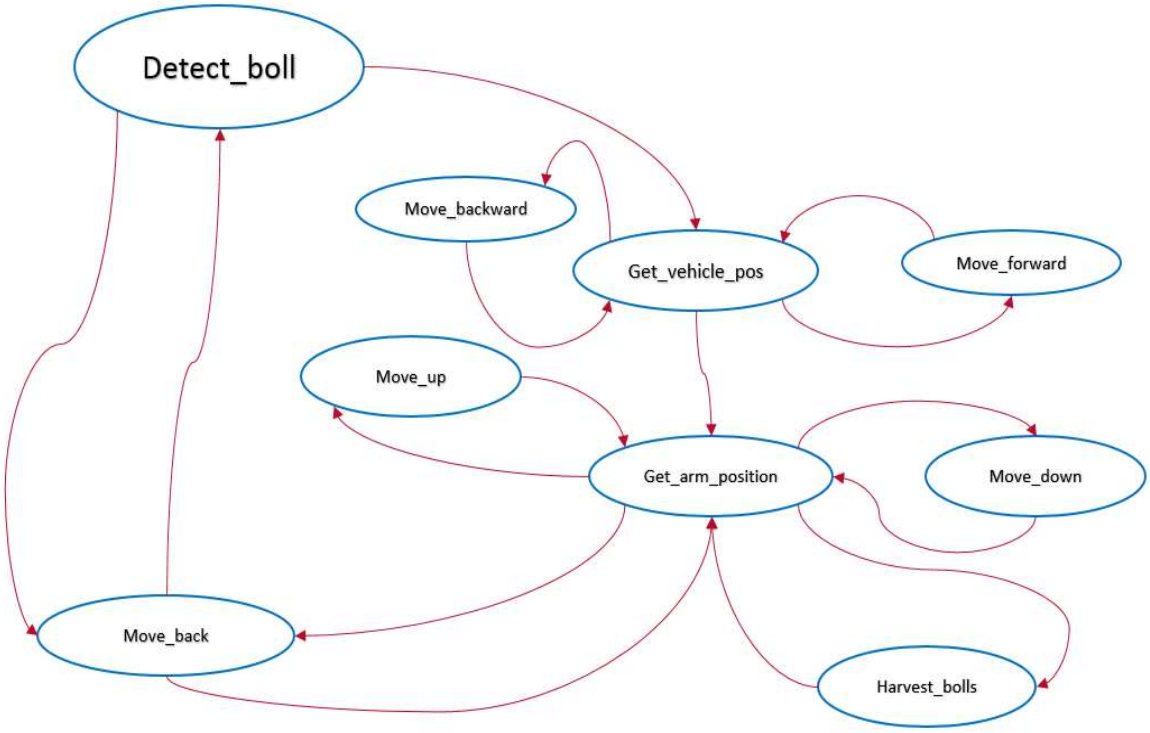


*Figure 7. Finite State Machine of the system*

In order to achieve a real state-of-the-art system, the robot operating system (ROS) was used to deploy the FSM. SMACH 'State MACHine' which is a ROS task-level architecture was deployed. SMACH provides a very good ROS independent architecture that can be deployed with simple programming rules. Each of the states and transitions can be published by the SMACH. Each mode of the machine can modularly be developed and incorporated to achieve state-of-the-art and robust behavior of the robot. Each state was programmed using python and then deployed to Jetson TX 2. If the decision is made, the robot arm moves according to the signal sent from Master controller to the arm controller which sends a number of steps to the micro-stepping drive and instructs the stepper motors to move to the target.

**Recalibration of the system using Artificial Neural Networks (ANN)**

The ANN was developed so that the machine can automatically be recalibrated before use. ANN consisted of the input

layer, 2 hidden layers, and the output layer. The input layer had 10 neurons, first hidden layer had 8 neurons, the second hidden layer had 5 neurons, and the output layer has only one (Figure 9). The ANN training involved two stages of training. When the results were estimated by equation (5) and when the results were projected using the ANN. The arm was moved randomly, and the new position was recorded together with the number of steps it has made. The data were shuffled to get testing and training data. The training data was separated by making 10% of data as testing data and 90% of data as the training data.

The system was calibrated on its' movements horizontally and vertically by doing the act repetitively for more than 1 hour. The linear inputs were the distance measured from the camera to the arm (Da), a distance of the camera to the target cotton boll (Db) and the differences in distance between the arm and target cotton boll (Dd). The output is the number of steps the machine should execute. The system steps were calculated as the modulus value of the equation of the steps against distance. The equation was obtained after taking a measurement of the distance of the arm from the camera when the arm is the furthest and change by each step until it is closest to the camera (Figure 8). The equation obtained by fitting the points in Figure 6.

$$\text{motor steps} = -410.7 * (\text{distance}) + 483.29 \quad (5)$$

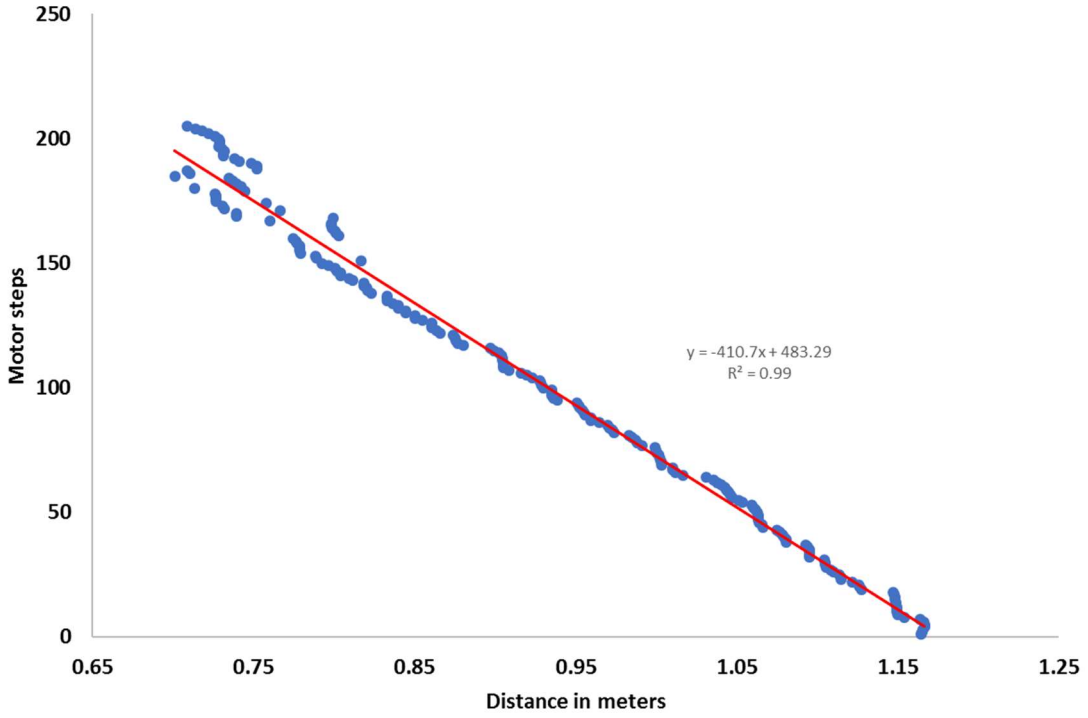


*Figure 8. Calibration of the distance against steps of the stepper motor*

Since the behavior of the motor changes, recalibrating the motor is necessary to achieve the best performance of the Cartesian arm. Hence, using these points, the ANN compared the true value obtained using equation 5 to start with and rerun the model so that it can adjust and predict arm position accurately.

The second stage, the values are adjusted accordingly after the true arm position (Db) is determined. Then, the system is retrained using the corrected Db for the second time with Dd being the difference of the corrected arm position and boll position. Boll positions are randomly assigned to give the system zigzag training data on all important movements of the arm. It should be noted that distance of the arm to move vertically was determined to be between 600mm to 1200mm from the camera while horizontally was determined to be from the center of the horizontal axis 125 mm to 570 mm. This is important so that the arm cannot attempt to move beyond the arms limits and destroy the toothed belt.

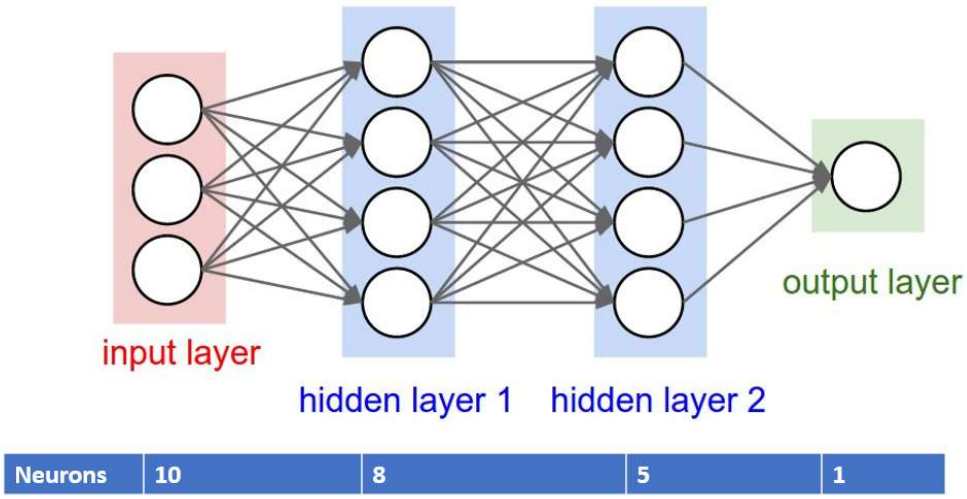


*Figure 9. Artificial Neural Network (ANN) for Arm Recalibration*

The system first moved horizontally for 30 minutes and then vertically, 30 minutes. Then, it moved again but with corrected values. The results of the experiment showed a great success.

The arm moved up and down for a few minutes, and the random movement was recorded (Figure 8). For each sample, the absolute error was recorded. The mean absolute error (MAE) for the samples was 10.9 steps, which indicates the system was making an error of 10.9 steps for each arm movement. After obtaining such results, the system used the steps predicted by ANN.

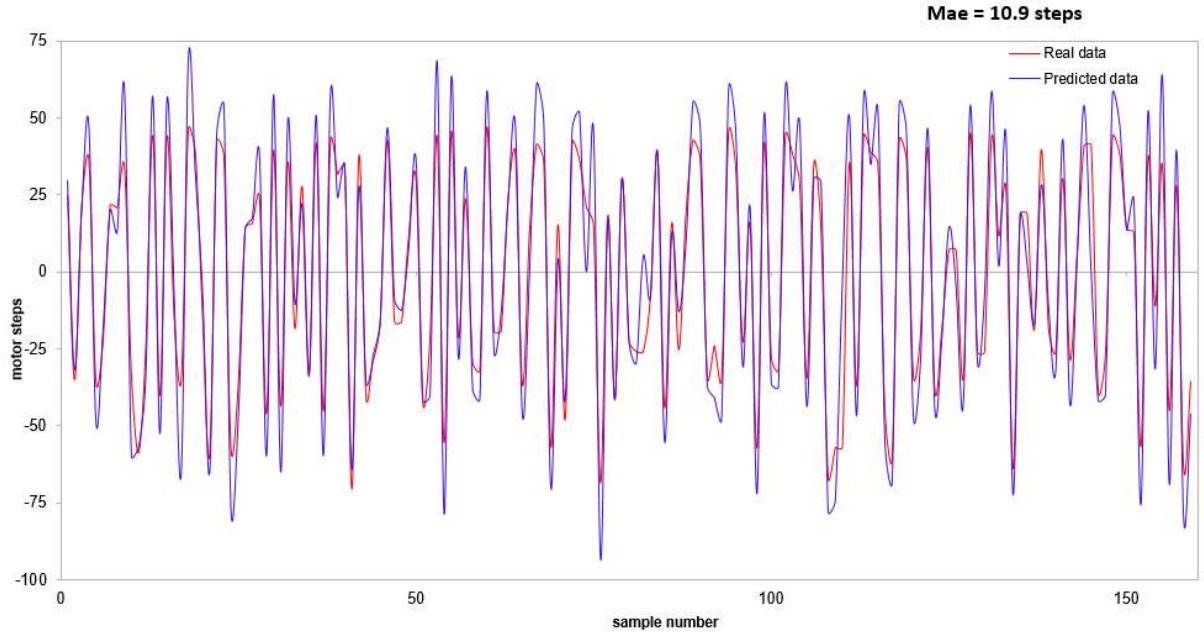


*Figure 10. Movement of an arm using equation 5 got MAE = 10.9 steps*

The system trained on the given data of equation 5 and true data obtained. The system used the data to predict the movement of the arm to reduce errors that are due to the camera and motor. The system used the prediction to learn the behavior of the true movement of the motor and recording camera. The data collected after training the ANN was tested and real movement compared to the predicted movement (Figure 10). The absolute error was recorded for each sample. The mean absolute error (MAE) was 7.0 steps which are lower compared to 10.9 steps recorded in the training data obtained using equation 5. Looking at Figure 11 and 12, it can be noted that the real data is presented by red lines while predicted data is presented by blue lines. By comparing the red and blue lines in Figure 11, the peaks of the blue line are either high or low all the times compared to red lines while in Figure 12, the peaks of the red line are closely alternating with blues lines and the prediction becomes good. This shows the ANN learns the behavior of the system and predicts well by compensating for the linearity in the equation which only prediction or real value to be large or small all the time and instead it is easily

equal, or any can be large or small. However, this is not the most important aspect of the use of ANN. The ANN achievement of lower MAE is considered more desirable for arm movements.

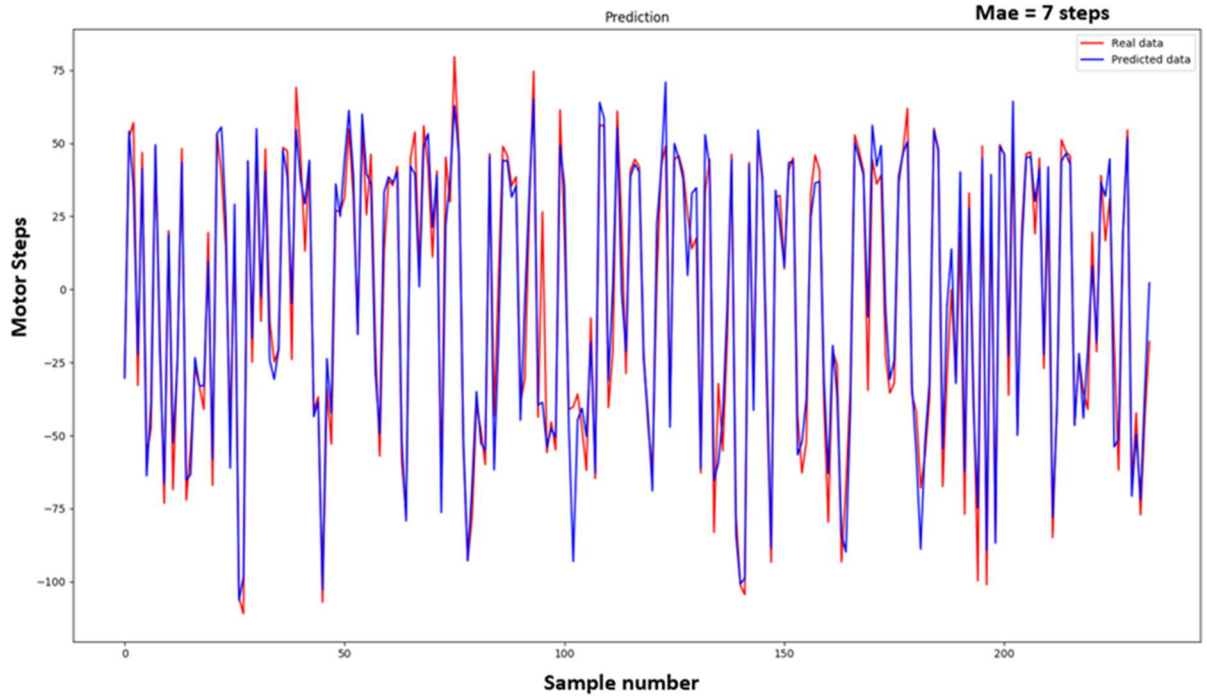


*Figure 11. The real verticle motor steps compared to predicted steps given by ANN. The MAE obtained is 7 steps*

This is an important component of the robotic arm control to obtain higher precision (step pulse at 2MHz and 400 steps per revolution) as the smooth movement of the robot arm mainly depends on the accuracy obtained. The motor steps are very smooth, and camera depth maps introduce error. However, the ZED camera manufacturer claims that the accuracy of the camera is 1 mm. Nevertheless, this claim might be true for the movement of the arm horizontally. The MAE error for horizontal arm movement using the ANN prediction was 0.37 steps (Figure 12). This might be due to short movements of the horizontal axis arm which is 60 cm long compared to vertical which is 190 cm long.

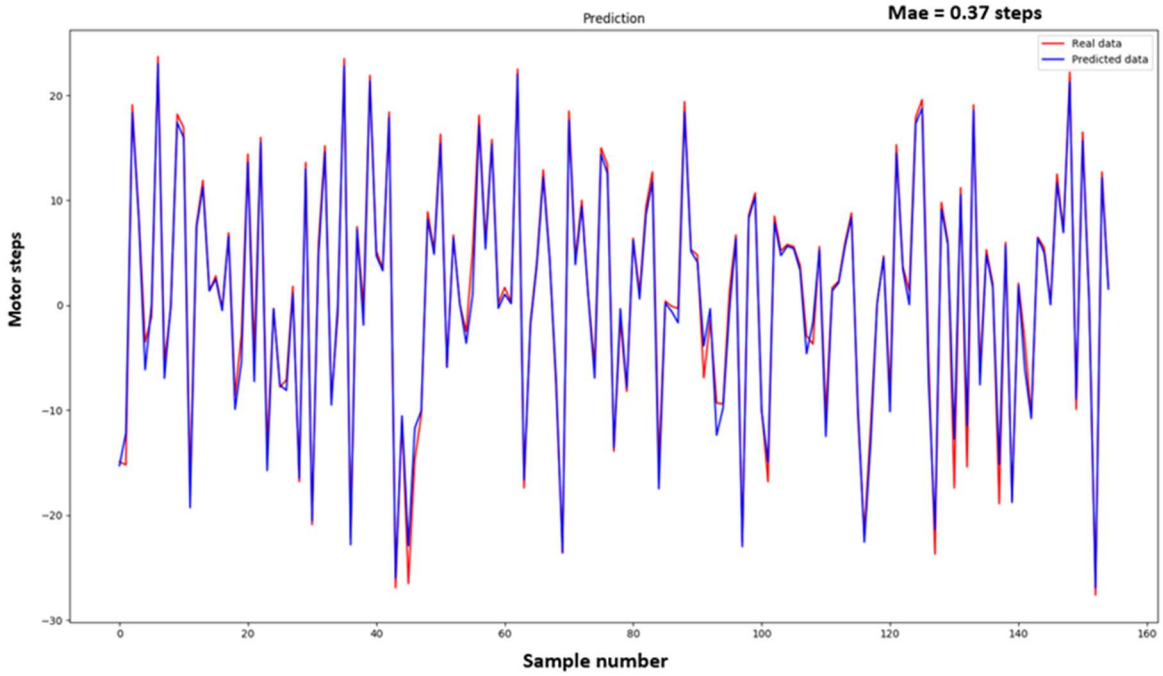


*Figure 12. Horizontal axis arm steps movement as predicted by the ANN. The MAE obtained was 0.37 steps*

**Inverse Kinematics of the Arm (Manipulator)**

Assume ($\alpha_x$, $\alpha_y$) is the number of steps predicted by the ANN to move the arm from ($x_0$, $y_0$) to (u, v). $\alpha_x$ is the number of

steps horizontally while $\alpha_y$ is the number of steps vertically. Arm location ($x_0$, $y_0$) is any point of the arm that is closest to the boll while (u, v) is the closest point of the cotton boll which is the closest to the arm.

Assume D is the ANN model matching the values by taking 3 inputs (3 x 2 matrix) and output one vector which is the recommended number of steps the arm should move to the target boll.

$$\begin{pmatrix} x_0 & u & x-u \\ y_0 & v & y-u \end{pmatrix} * D = \begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix}$$
$$\begin{pmatrix} \alpha_x \\ \alpha_y \end{pmatrix} * S = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad (5)$$
$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \xrightarrow{yields} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

The system tries to minimize the value of $\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$

This means, when the front part of the arm is touching the boll, the difference in x and y is 5 and 15 mm, respectively. The kinematics summary of the arm is described in Table 1 using pseudocode.

*Table 1. Pseudocode of the manipulator kinematics*

| | **Pseudocode of the manipulator kinematics** | |
|---|---|---|
| 1 | Vertical distance = $V_d$ = v – y | |
| 2 | Horizontal distance = $H_d$ = u – x | |
| 3 | If (Vd =<-15) move the arm up | Δy > 15 |
| 4 | If (Vd >= 15) move the arm down | Δy > 15 |
| 5 | If (-15 < Vd < 15) and (Hd > 5) move the arm to harvest | Δy < 15 |
| 6 | If (-15 < Vd < 15) and (Hd < 5) move the arm back to position for next boll (1) | Δy < 15 and Δx < 5 |

**Rover movement Controller**

Rover controller runs an adaptive PID (Figure 13) to control the rover forward and rearward movement. The position of the rover and the target position will be published from the master, and the Arduino clients subscribe to the topic accordingly. The throttle is provided at a constant maximum RPM to keep constant and enough power. A topic that subscribes to this message will be developed in the rover microcontroller. Articulation is done after the rover controller receives a topic that publishes the instruction to turn or go straight. For this study, the rover was only moving straight. Master controller publishes an articulation message to the rover controller, and it subscribes. Also, the rover controller publishes the gains of the adaptive PID controller together with the position of the rover relative to encoder pulses. Master subscribes to the messages so as to provide an accurate position of the rover which is used to pass the signal to the arm controller for boll picking.

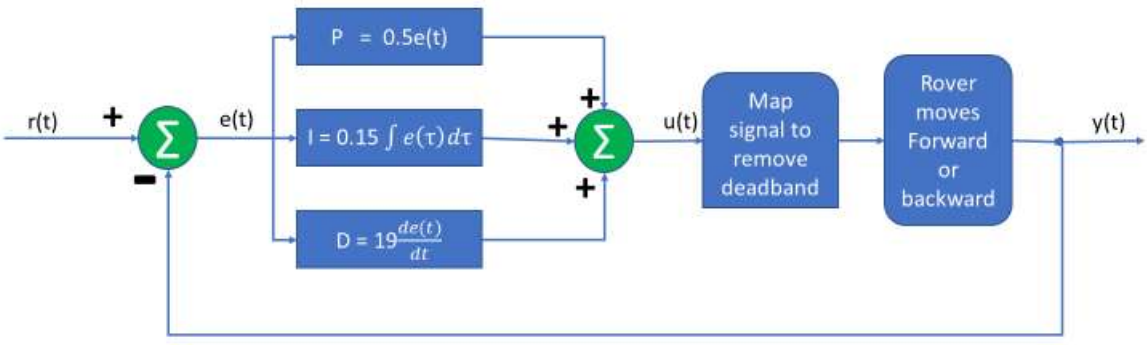


*Figure 13. The PID controller implemented in a rover controller to achieve accurate target position*

The control function (Equation 6) of the controller above is presented as;

$$u(t) = P + I + D$$

$$u(t) = 0.5e(t) + 0.15 \int e(\tau)d\tau + 19\frac{de(t)}{dt} \quad (6)$$

where,

constants $K_p$, $K_i$ and $K_d$, are given as 0.5, 0.15 and 19 respectively,

r(t) is a setpoint which is the number of pulses required to reach a certain distance, and

y(t) is the measured number of pulses read by the rotary encoder.

The system tries to minimize error e(t) which is given e(t) = r(t) – y(t). $K_p$, $K_i$, and $K_d$ denote the coefficients for the proportional, integral, and derivative gains respectively.

Tuning was done by first identifying the deadband of the linear servo that does not make the rover move. The rover controller sends a servo signal to the linear actuator (Figure 14) which moves the rover. The linear actuator pushes the swashplate to a certain angle is directly controlled using rover controller. The linear actuator extends from 0 to maximum 20 cm after receiving an analog signal. The servo signal sent from the rover ranges from 65 to 138 pps. When 90 pps is sent to the rover controller, the rover stops. If it is decreased slowly from 90 to 65 pps, the rover moves in reverse motion while if it is increased from 90 to 138 pps, then the rover moves forward. This means 65 pps provides maximum speed in reverse motion while 138 pps provides maximum speed forward. The rotary encoder installed on the wheel of the rover sends back input pulses to the PID control which measures how far the rover has moved. However, the system has a very large deadband from 80 to 98 pps since the encoder readings don't change which means the vehicle is not moving. The deadband (Figure 15) is the amount of signal sent from the rover Arduino microcontroller to the linear actuator servo that cannot make the rover move either forward or reverse. It means when you miss the target; it becomes very difficult to get close to the target point as the PID can either be a direct relationship or reverse relationship between output (actuator signal) and input (encoder pulses).

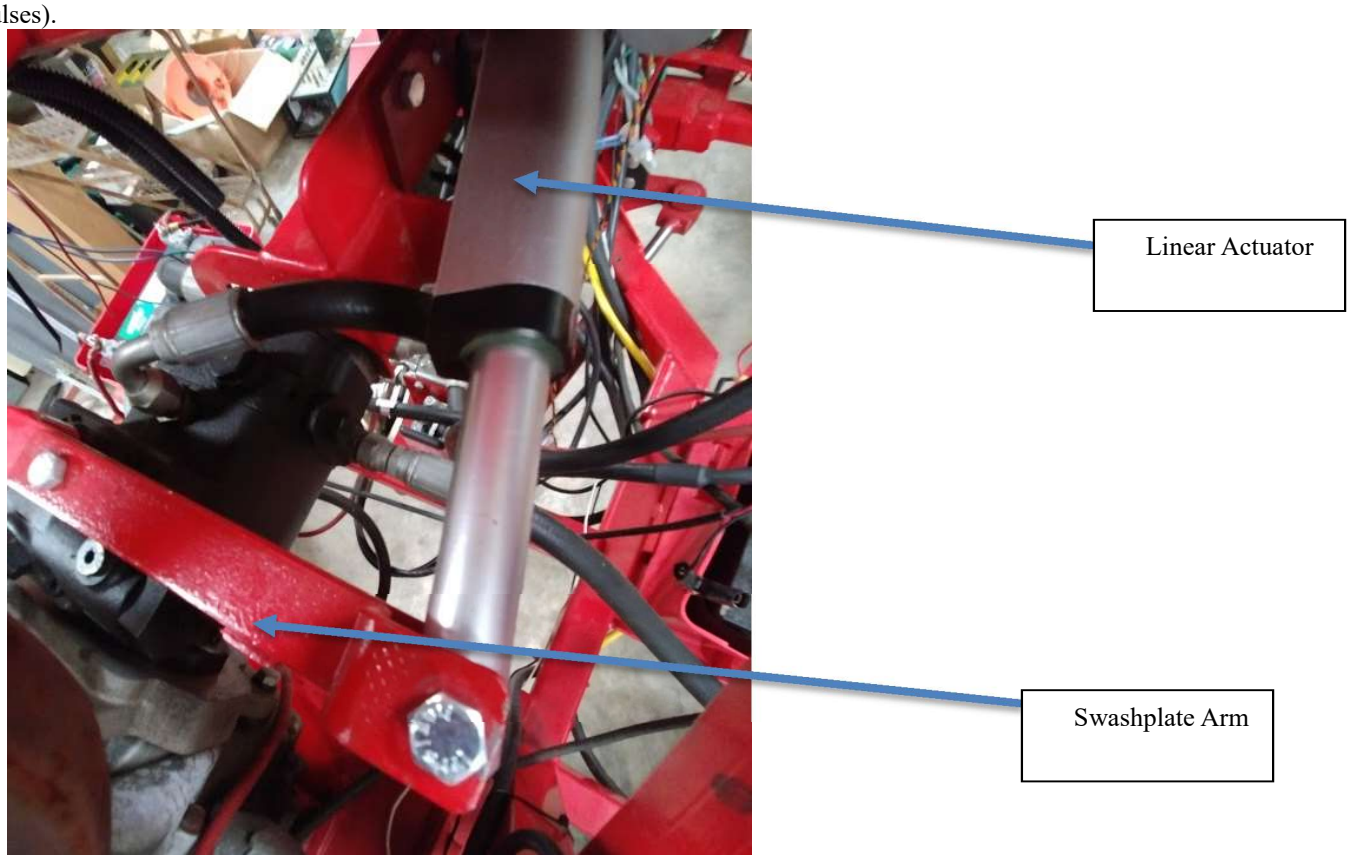


*Figure 14. The linear actuator moving the swashplate arm to determine an angle for movement of the rover (forward when extending or rearward when retracting)*

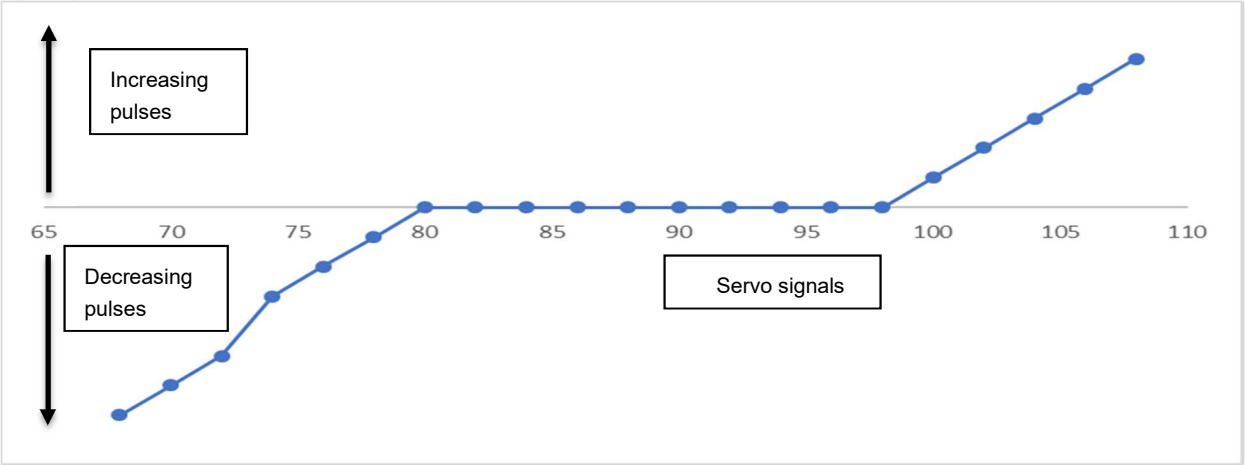


*Figure 15. Preliminary experimental data shows deadband between 80 to 98 of the actuator signal (actuator should move forward when the signal to the linear actuator is increased from 90 to 120 and move reverse if the signal is decreased from 90 to 65)*

In order to remove the deadband, the system was designed such that it gives the output u(t) from -100 to 100. Then, the signal was mapped to the true settings of the rover. The actual servo signal was set to move (extend) from 98 to 108 pps by mapping positive output values (0 to 100) of u(t) while moving back (retract) from 70 to 80 pps for negatives output values (-100 to 0) while zero was set to be 90.

Manual tuning was chosen after other methods like Ziegler-Nichols and Cohen-Coon methods didn't give optimal performance. But Ziegler-Nichols and Cohen-Coon methods were used to predict a small overshoot and no overshoot PID gains. Manual tuning was done by increasing $K_p$ until the rover oscillates while setting $K_i$ and $K_d$ values to zero. Then, $K_i$ was increased until the rover was oscillating around the setpoint. Now, the system is oscillating around the set point. After that, $K_d$ was increased until the system was settling at the given setpoint quickly. The PID gains were obtained; $K_p = 0.5$, $K_i = 0.15$ and $K_d = 19$. Also, later, we recalibrated the system to find out if we can get the best performance using different values. The PID gains that bring a small overshoot but aggressively were obtained; Kp = 0.7, Ki = 0.04 and Kd = 5.0. In Figure 16 obtained using rqt_graph, the master controller sends a servo signal to the rover controller to instruct the system to move from 500 positions to 2000 position and then going back to the initial position. The system uses only 25 of pulse unit time which is equivalent to 1250 ms to settle (Figure 16). The system performed the same way going forward from 500 to 2000 or backward from 2000 to 500. The overshoot is approximately 100 encoder pulses which is equivalent to 18 cm. Each of the encoder pulses is equivalent to 1.8 mm distance.

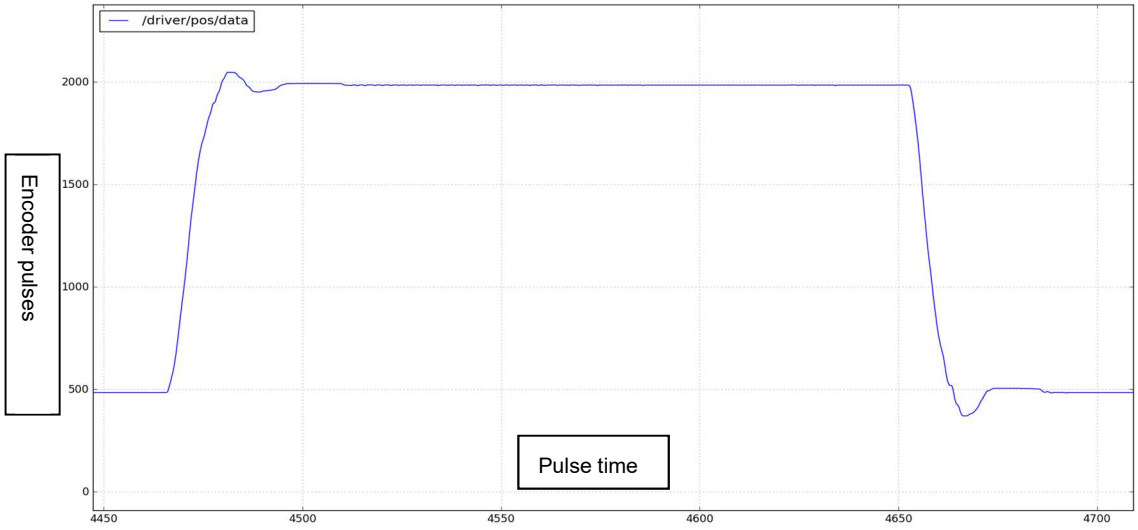


*Figure 16. The PID control graph of the rover. The signal is sent from the Master controller to the rover controller to move the vehicle from point 500 to 2000 and then back to 500. Each pulse time unit is equivalent to 50 ms. This graph was obtained using rqt_graph by subscribing to a published topic driver/pos which provide position feedback from the encoder. The rqt_graph provides a GUI plugin for visualizing the ROS computation graph.*

**Experiments Setup**

Five experiments were set up at the University of Georgia (UGA) Tifton campus grounds (N Entomology Dr, Tifton, GA, 31793) at (31˚28'N 83˚31'W). The experiments were undertaken on 29th and 30th May 2019. Six cotton plants were put to the test. Each plant contained 3 bolls. The defoliated plants were put 30 cm apart covering a total length of 240 cm including the pot diameter. The rover was set to move over the plants and move the arm to get the bolls out of the plants (Figure 16 and 17). The camera is 220 m above the ground (l). The first boll detected by the system is m distance from the camera. The camera also checks the distance of the arm from the camera and try to move the arm to get the first boll. The camera can see several bolls and get distance from the current boll to harvest to the next one. Other two bolls are x cm and (x+y) cm from the current boll to be picked. After picking the current boll, the system went on getting the boll which is x cm from it by invoking the rover controller which instructs the linear actuator which moves the rover by pushing the swashplate. By doing so, the system was able to pick the bolls from the plants, one after another. Time was recorded for all of the experiments of picking 18 bolls from the six plants.

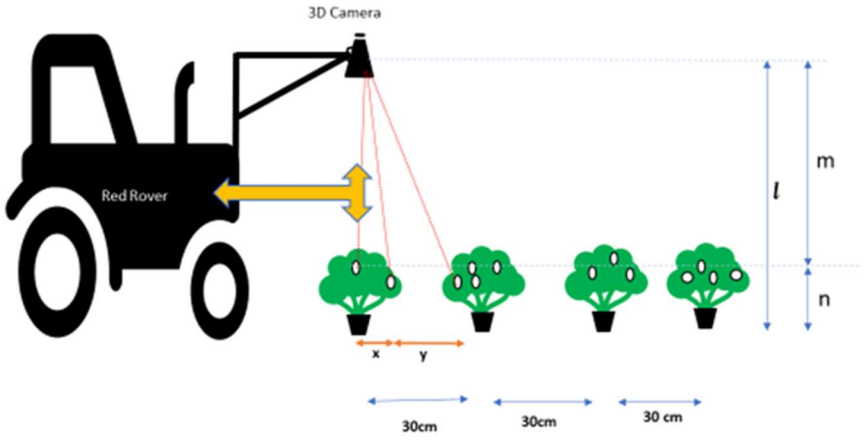


*Figure 17. System testing was done by putting the four potted defoliated plants in front of the system to collect preliminary data on its performance*

*Figure 18. The experiment was set up at UGA grounds to test the robot on picking the bolls on a simulated environment consisting of six potted cotton plants*

The system was first tested by putting only few plants. First, only two plants were kept with two bolls, then three, four, six bolls and eight bolls to test the system working capacity before fully testing the system with six plants with three bolls in each. The results of the five experiments are collected and analyzed using the robot performance metrics mentioned by Bechar and Vigneault (2017).

# Results and Discussions

The results were obtained by counting the cotton bolls that the robot was able to pick. Also, we checked the images taken by the camera to find out if the system was perceiving the bolls correctly by color segmentation. For the last two experiments, time was also recorded. The following parameters were determined; Operation Velocity (OV) under real-time conditions (cm $s^{-1}$), Production Rate (PR) (bolls $s^{-1}$), Cycle Time (CT) (s), Action Success Ratio (ASR) (%) and Detection Performance (DP) (%). OV is the average velocity measured during a mission under real-time. PR is the number of cotton bolls picked per time unit. CT is the average time required to complete one cotton-picking action. ASR is the ratio of success in the action of picking the bolls. DP is the ratio of the number of appropriate detections (True positives + True negatives) over the sum of all boll detection attempts made by the system.

*Table 2. The collected data for the experiments done to validate the performance of the system*

| Cotton bolls | Detected | Picked | Time (sec) | OV | PR | CT | ASR (%) | DP (%) |
|---|---|---|---|---|---|---|---|---|
| 18 | 17 | 15 | 214 | 1.12 | 0.07 | 14.27 | 83.33 | 94.44 |
| 18 | 18 | 16 | 343 | 0.70 | 0.05 | 21.43 | 88.89 | 100.00 |
| 18 | 18 | 15 | 219 | 1.10 | 0.07 | 14.60 | 83.33 | 100.00 |
| 18 | 17 | 17 | 323 | 0.74 | 0.05 | 19.00 | 94.44 | 94.44 |
| 18 | 18 | 17 | 285 | 0.84 | 0.06 | 16.76 | 94.44 | 100.00 |
| Average | 17.4 | 16 | 276.8 | 0.87 | 0.06 | 17.3 | 88.88 | 96.67 |

The system performed by picking a boll every 17.3 seconds. In this preliminary testing, the system was experiencing abnormal performance since most of the computer code is preliminary. The experiments showed that our robot design could be a viable solution for cotton picking if the end effector is optimized to pick the bolls fast. Most of the time the system was attempting twice or more just to get the boll out of the plants. Also, the system was struggling to get a boll if it is located behind the stem or branch. The system was missing some bolls because of the jerking caused by PID overshoot which takes the system 18 cm far from the point before settling. Sometimes it causes the system to leave behind some bolls. By retuning the PID and remove no overshoot will be done. However, if we change our mechanical rover to electric rover, then no overshoot can be achieved easily with electric systems. Some modifications of the system picking rate and positioning will be done in spring 2019 to improve the performance before field testing beginning in August 2019.

# Conclusion

In this research, the preliminary design of the cotton-picking robot is reported. The system is optimized to use visual controls to pick the bolls. The system uses SMACH which is a ROS-independent finite state machine library that provides task-level capabilities to design a robotic architecture that provides robust control of robotic behavior. The 2D robotic system with gasoline engine rover was optimally controlled to pick the bolls. The system achieved a picking performance of 17.3 seconds per boll. Also, it covered the task by navigating at a speed of 0.87 cm per second collecting 0.06 bolls per second. In each mission, the system was able to detect all the bolls but one. Also, we will improve the end-effector, so it does not have to reach out multiple times or back up. Also, we will improve by tuning the control system for navigation, so there is no overshoot, reduce the time it takes to find boll and make the decision to pick. In Fall 2019, we are going to undertake more research on un-defoliated cotton plants. Most of the early opening bolls tend to appear at the bottom of the plants. This will be a big challenge to the location of the camera relative to detection. The camera cannot be placed near the plants because it will be destroyed. All of this can be solved when we do tests on the fields.

**Acknowledgments**

The authors are very thankful for project support from Cotton Incorporated, Agency Assignment #17-038. We also thank Mr. Ricky Fletcher and Mr. Gary Burnham for their helpfulness and technical assistance in building the robotic platform and collection of data.

# Reference

Bechar, A., & Vigneault, C. (2017). Agricultural robots for field operations. Part 2: Operations and systems. *Biosystems engineering*, *153*, 110-128.

Fue, K., Porter, W., Barnes, E., and Rains, G. (2019). Visual Inverse Kinematics for Cotton Picking Robot. BWCC Paper No. 19379. Cordova, TN: NCC

Fue, K., Porter, W., Barnes, E., and Rains, G. (2019). Visual Row Detection Using Pixel-Based Algorithm and Stereo Camera for Cotton

Picking Robot. BWCC Paper No. 19376. Cordova, TN: NCC

Fue, K., Porter, W., and Rains, G. (2018). Real-Time 3D Measurement of Cotton Boll Positions Using Machine Vision Under Field Conditions. BWCC Paper No. 18385. Cordova, TN: NCC

Fue, K. G., Porter, W. M., & Rains, G. C. (2018). Deep Learning based Real-time GPU-accelerated Tracking and Counting of Cotton Bolls under Field Conditions using a Moving Camera. In 2018 ASABE Annual International Meeting (p. 1). American Society of Agricultural and Biological Engineers.

Rains, G., B. Bazemore, K. Ahlin, A. Hu, N. Sadegh, & McMurray, G. (2015). Steps towards an Autonomous Field Scout and Sampling System. ASABE Paper No. 15219077. St. Joseph, MI: ASABE.

UGA. (2018). Georgia cotton production guide. Cooperative Extension Service/The University of Georgia, College of Agricultural and Environmental Sciences Bulletin, Tifton. Retrieved from http://www.ugacotton.com/vault/file/2018-Georgia-Cotton-Production-Guide.pdf

USDA/NASS. (2018). 2017 State Agriculture Overview for Georgia. Washington, DC: USDA-NASS. Retrieved from https://www.nass.usda.gov/Quick_Stats/Ag_Overview/stateOverview.php?state=GEORGIA