FIELD TESTING OF THE AUTONOMOUS COTTON HARVESTING ROVERIN UNDEFOLIATED COTTON FIELD

K. G. Fue
College of Engineering, University of Georgia
Tifton, GA
W. M. Porter
Crop and Soil Sciences, University of Georgia
Tifton, GA
E. M. Barnes
Cotton Incorporated
Cary, NC
G. C. Rains
Entomology, University of Georgia
Tifton, GA

Abstract

This study proposes the use of an autonomous rover to harvest cotton bolls before defoliation and as the bolls open. This would expand the harvest window to up to 50 days and make cotton production more profitable for farmers by picking cotton before the quality is at risk. We developed a cotton harvesting rover that is a center-articulated vehicle with an x-y picking manipulator and a combination vacuum and rotating tines end-effector to pull bolls off the plant. The rover uses a stereo camera to see rows, RTK-GPS to localize itself, fisheye camera for heading, and stereo camera to locate the cotton bolls. The SMACH library is a ROS-independent task-level architecture used to build state machines for the rapid implementation of the robot behavior. First, the GPS waypoints are obtained, and then, the rover passes over the rows while picking the cotton bolls. The navigation is controlled by a modified pure-pursuit technique together with a PID controller. Two parallel programs organize the entire rover regarding when to pick and when to navigate. While navigating, the rover looks for harvestable bolls, and when bolls are discovered, the robot will stop and pick. It will do this repetitive work until it finishes all the rows. The rover navigation had an absolute error mean of 0.189 m, a median of 0.172 m, a standard deviation of 0.137 m, and a maximum of 0.986 m. The largest errors occurred during turning around at the end of rows and were caused by wet conditions and tire slippage. The rover picked cotton bolls at the average Action Success Ratio (ASR) of 78.5% and was able to reach 95% of the bolls. Most bolls that were not picked could not be pulled into the vacuum using the rotating tines on the end-effector.

Introduction

The cotton industry is a very important sector for Georgia and the United States. Harvesting of the cotton field has been done using big and expensive machines for years. These heavy machines contribute to soil compaction and are expensive and time-consuming to repair. Breakdowns during the season may lead cotton to be exposed to environmental conditions that may diminish the quality of the fiber. Farmers who cultivate small acreage can not afford the capital to buy the machine or attend its' maintenance (Fue et al., 2018).

However, the harvesters can only work after cotton fields are defoliated using chemical defoliants. These chemicals can degrade the land, but also, defoliation occurs around 50 days from the opening of the first bolls. Cotton lint quality is highly affected due to external weather and other farm elements that may contaminate and diminish the quality (Hayes, 2017 and Fue et al., 2018).

Any solution that could be developed to solve the problems associated with cotton harvesting would be well received by the cotton industry. However, large cotton harvesters are mechanical machines, and for them to be efficient enough, they need to be designed as they are now. They should be big enough to utilize the fuel well and harvest large acreage within a short time. Any solution to more economical harvesting should not be another large mechanical combine (Hayes, 2017 and Fue et al., 2018).

Currently, there is a booming industry in robotics and machine learning technologies. Robotics have been developed to solve some of the pertinent issues in agriculture. The advancement of robotics provides the aging farming society with the best solution to handle almost all farming activities (Fue et al., 2018; Hayes, 2017 and Rains et al., 2014). However, agricultural robotics has been researched more for horticultural crop production. In this study, we study a cotton harvesting robot that can work in undefoliated cotton. The robot software used open-source tools to promote further collaboration from interested developers.We present the performance of the robot in picking and navigation. Autonomous means the robot can navigate by itself and harvest the cotton bolls without the intervention of human subjects.
Therefore, in this study, we present three objectives;

i. Development and testing of the navigation systems of the robot
ii. Development and testing of the cotton manipulation systems for cotton picking

iii. Field testing of the autonomous cotton harvesting robot

Materials and Methods

Robot components and System Setup

The rover (Figure 1) was a custom-built four-wheel center-articulated robot (West Texas Lee Corp., Lubbock, Texas). The rover was 340 cm long and with front and back parts being 145 cm and 195 cm long, respectively. The rover tires are 91 cm from the center of the vehicle. The rover was 212 cm wide, with a tire width of 30 cm. The four tires have a radius of 30.48cm and a circumference of 191.51cm. The rover had a ground clearance of 91 cm. The manipulator consists of two parts; a horizontal axis, which is 70 cm, and a vertical axis, which is 195 cm. The manipulator was placed in the front of the rover with a 27 cm ground clearance. The rover used six sensors; two IMUs, two-wheel encoders, stereo camera, and RTK-GPS. The front tires were connected to a rotary encoder (Koyo incremental (quadrature) TRDA-20R1N1024VD, Automationdirect.com, Atlanta, GA, USA). The two IMUs (Phidget Spatial Precision 3/3/3 High-Resolution model 1044_1B, Calgary, Alberta, Canada) was placed in front of the rover. First, IMU was placed 95 cm above the ground and 31 cm from the front of the vehicle. The second IMU was 132 cm above the ground and 46 cm from the front of the vehicle. The RTK-GPS receiver (EMLID Reach RS, Mountain View, California) was placed 246 cm above the ground and 30 cm from the front of the vehicle. The RTK correction signal was obtained using NTRIP signal (eGPS Solutions, Norcross, GA) with a mounting point within 2 miles of the test plot and downloaded to the Emlid through a Verizon Hotspot and wireless signal. An RGB stereo camera (ZED camera, Stereo labs Inc, San Francisco, CA, USA) was installed and used to acquire images. ZED has a 4M pixel sensor per lens with large 2-micron pixels. The left and right sensors were 120 cm apart. The sensor was placed 183 above the ground in front of the vehicle facing down so that it can see cotton bolls. An embedded computer (NVIDIA Jetson AGX Xavier development kit, Nvidia Corp., Santa Clara, CA, USA) was installed and controlled rover navigation and manipulation controllers.

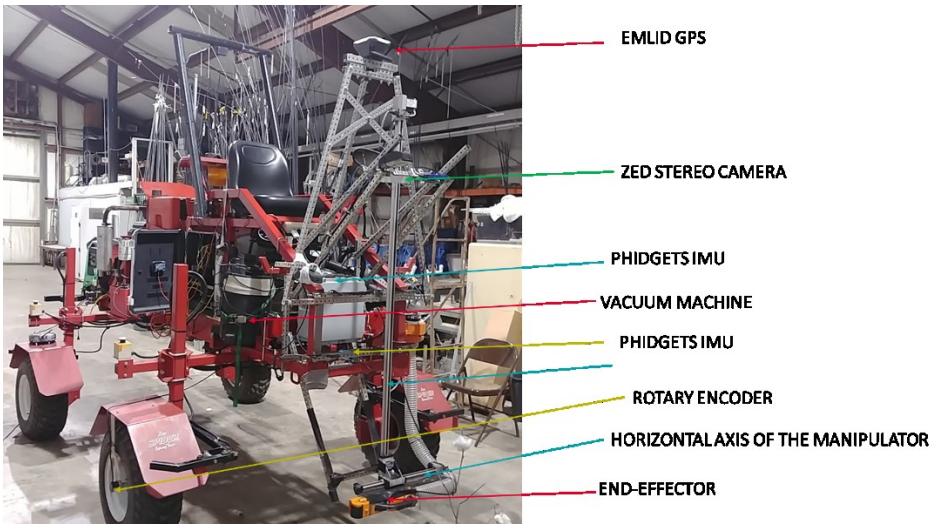


Figure 1. The research red rover with manipulator and sensors attached in front of the rover implemented for this study

All the sensors except rotary encoders were connected to it via a universal serial bus (USB)connection. The encoder was connected to the Rover navigation controller (Arduino Mega 2560, Arduino LLC) using 4 wires; signal A and B, power and ground so that it can register rotation of the tires by detecting rising of the square waves. Also, the robot manipulation controller (Arduino Mega 2560, Arduino LLC) was installed to control the axes of the manipulator. ROS (Robot Operating System), which is robotics middleware used for robot software development, was implemented to act as the middleware to connect the central controller (Jetson Xavier) with robot manipulation controller and Rover navigation controller. The robot software is developed to communicate using ROS topics. ROS topics are "named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data subscribe to the relevant topic, nodes that generate data published to the relevant topic. There can be multiple publishers and subscribers to a topic." (http://wiki.ros.org/topics).

Rover navigation controller received a signal from the embedded computer to control the rover articulation, actuation, and throttling. The rover tires were controlled using a servo to the engine throttle and a servo to the variable-displacement pump swashplate servo. The front tires were connected to a rotary encoder to provide feedback to the adaptive PID to control the movement of the rover along the crop rows. The throttle was connected to the onboard Kohler Command 20HP engine (CH20S, Kohler Co, Wisconsin, USA) with a maximum of 2500 RPM and driving an axial-piston variable rate pump (OilGear, Milwaukee, WI, USA) with a maximum displacement of 14.1 cc/rev. The OilGear pump was connected with a swashplate for directing the forward and rearward movement of the rover. The swashplate angle was

controlled by the rover controller that determines the placement of the linear electric servo (Robotzone HDA4, Servocity, Winfield, KS) that can move approximately 10.16cm (4inch). Left/right articulation was controlled by using linear relays connected to a 4-port 3-way open-center directional control valve (DCV) powered by a 0.45 cc/rev fixed displacement pump (Bucher Hydraulics, Italy) in tandem with the Oilgear pump. The DCV provided hydraulic fluid to hydraulic cylinders that controlled the rover's articulation. The rover could turn a maximum of 45 degrees with a wheelbase of 190 cm. Also, the rover's height and width could be adjusted to a maximum of 122cm and 234 cm, respectively.

The robot manipulation controller primarily received a 4-byte digital signal from the Jetson Xavier. The signal provided the number of steps and direction of the manipulator (Up, Down, back and forth). Then, the controller sent the signal to the micro-stepping drive (Surestep STP-DRV-6575 micro-stepping drive (AutomationDirect, Cumming, Georgia) , which in turn sends to the appropriate stepper signal for action. Manipulator controller was 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 provided smooth motion for the manipulator.

In this study, a robotic manipulator was designed to work as a 2D cartesian system and developed using two 2-phase stepper motors (MS048HT2 and MS200HT2, ISEL Germany AG, Eichenzell,Germany). The MS048HT2 model stepper motor was installed to run the horizontal linear axis arm (60 cm long), and the MS200HT2 to run the vertical linear axis arm(190 cm long). The connecting plates and mounting brackets used a toothed belt that is driven by the stepper motor to move back and forth. 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 of the manipulator were connected as a vertical crossbench. The sliding toothed belt drive (Lez 1, ISEL Germany AG, Eichenzell,Germany) was used to move the end effector. The toothed belt had 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 were vacuumed into the end-effector, which was placed close to the cotton bolls. The end-effector has a rotating brush roll which removes cotton bolls through a combination of vibration and sweeping. The brush roll is powered by a 12-V motor. The cotton bolls are grabbed and pass through a flexible hose to an impeller mounted with the suction opening from the vacuum machine. The cotton bolls are passed through it before being blown into a bag of cotton.

Robot Navigation Systems

The navigation system consists of the development kit and rover navigation controller. The rover used two algorithms to control navigation; modified pure pursuit and PID control. The system used a predefined path of the GPS signal to navigate over the rows. The path was obtained by recording the rover path as it drove through the cotton rows. The predefined path was then used by the rover to navigate while harvesting.

Since the rover used 5 sensors (two IMUs, two encoders, and RTK-GPS) to navigate (Figure 1), the Extended Kalman Filter was implemented for simultaneous localization and navigation. It was achieved by using the open-source library "Robot localization," which provides sensor fusion and nonlinear state estimation for IMUs, encoders, and GPS. "Robot Localization" is implemented in ROS. The IMUs publish two ROS topics (imu_link1/data and imu_link2/data), encoders publish wheel odometry (/wheel_odom), and RTK-GPS publish /gps/fix signal. The EKF localization (Figure 2) used the nav_sat_transform library to integrate fix data from the GPS. Basically, "navsat_transform_node" required three sources of information: the robot's current pose estimate in its world frame, an earth-referenced heading, and a geographic coordinate expressed as a latitude/longitude pair (with optional altitude) (http://docs.ros.org/).

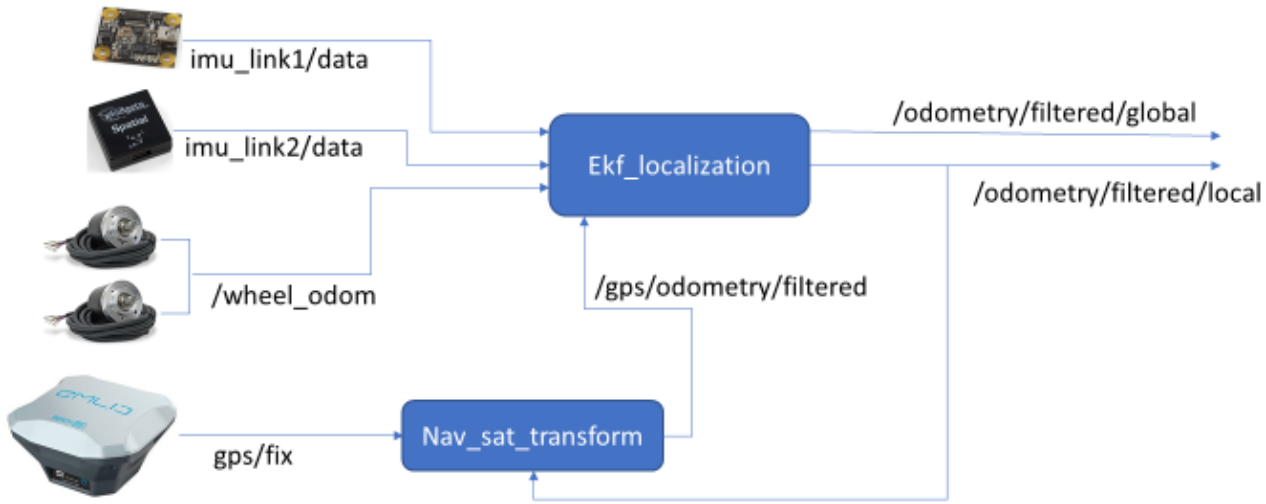


Figure 2. simultaneous localization and navigation of the rover using dual Extended Kalman Filter (dual EKF)

In order to get the correct readings from the IMU, Phidgets, manufacturer of the IMU, provided a software program for calibration. The IMU was calibrated by placing it on the mounting point on the rover and connect the IMU to the computer, which has the Phidget compass calibration program installed. The software required that the rover be driven in a circle to generate the calibrated compass parameters (https://www.phidgets.com/docs/1044_User_Guide).

EKF localization was implemented (Figure 2) as a dual EKF method that involved running two EKF's concurrently. The state and model could be estimated from the noisy observations from the IMU, wheel odometry, and GPS fix signal.

$$x_{k+1} = F(x_k, u_k, w) + v_k \quad (1)$$
$$y_k = H(x_k, w) + n_k$$

We consider the nonlinear problem in which system states ($\mathbf{x}_k$) and model parameters ($\mathbf{w}$) are simultaneously estimated from the observed noisy signals ($\mathbf{y}_k$). The processed signal ($\mathbf{v}_k$) determines the dynamical system, while the observed exogenous input noises ($\mathbf{n}_k$ and $\mathbf{u}_k$) are calculated (Eqn 1). In Figure 3, Using current model estimates $\mathbf{w}_k$, an EKF state filter calculates the new state in every step. Then, it calculates the new weights of the current state estimate $\mathbf{x}_k$. The model structure $\mathbf{F}$ and $\mathbf{H}$ represent multilayer neural networks, in which case $\mathbf{w}$ are the weights (Wan and Nelson, 2001). In this case, the model uses IMUs, and encoders to generate local localization estimates and then adds GPS to generate global localization estimates of the rover position.

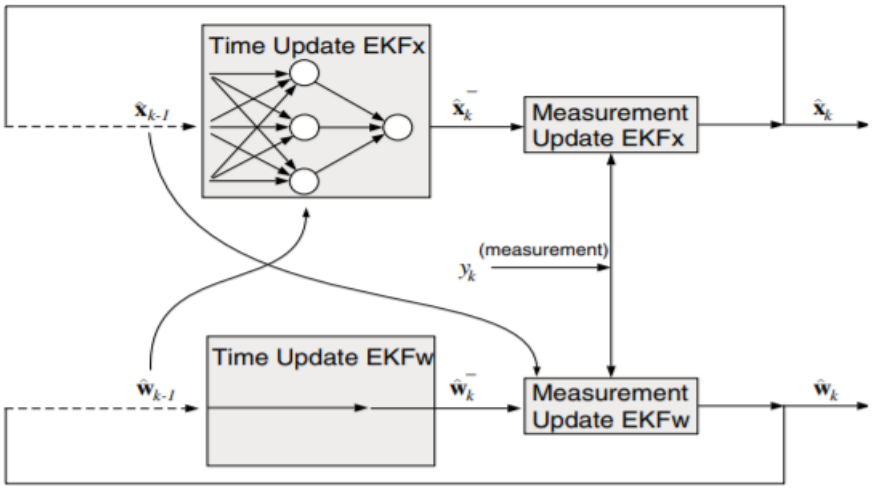


Figure 3. The dual extended Kalman filter. The algorithm consists of two EKFs that run concurrently. The top EKF generates state estimates and requires $\hat{w}_{k-1}$ for the time update. The bottom EKF generates weight estimates and requires $\hat{x}_{k-1}$ for the measurement updates (Wan and Nelson, 2001).

After getting the state estimates of the rover, then a modified pure pursuit and PID were used to control the rover's navigation.

Modified Pure Pursuit and PID Control

Pure pursuit (PP) is a tracking method that calculates the vehicle current position relative to a goal and then determines the curvature that would bring the vehicle back to the path. PP chooses the goal that is some distance in front of the vehicle. It looks ahead and determines the articulation of the tires to get into the path. This look-ahead distance changes depending on the curvature of the path and speed of the vehicle.

This rover achieved pure pursuit tracking by following six steps; determine the current location of the rover, find the path point closest to the rover, find the goal location, transform the goal location to rover coordinates, calculate the curvature and request the rover to set the steering to that curvature and then, update the vehicle's position (Coulter, 1992).

Figure 3, the center-articulated with all the sensors at the front, including GPS steering to pursue the red dot (goal point) at a distance L. The radius of turning is r while s, which is equal to x is the relative distance of the rover to the goal point.

$$\begin{aligned} x+d &= r \\ x &= s \\ x^2 + y^2 &= L^2 \end{aligned} \tag{2}$$

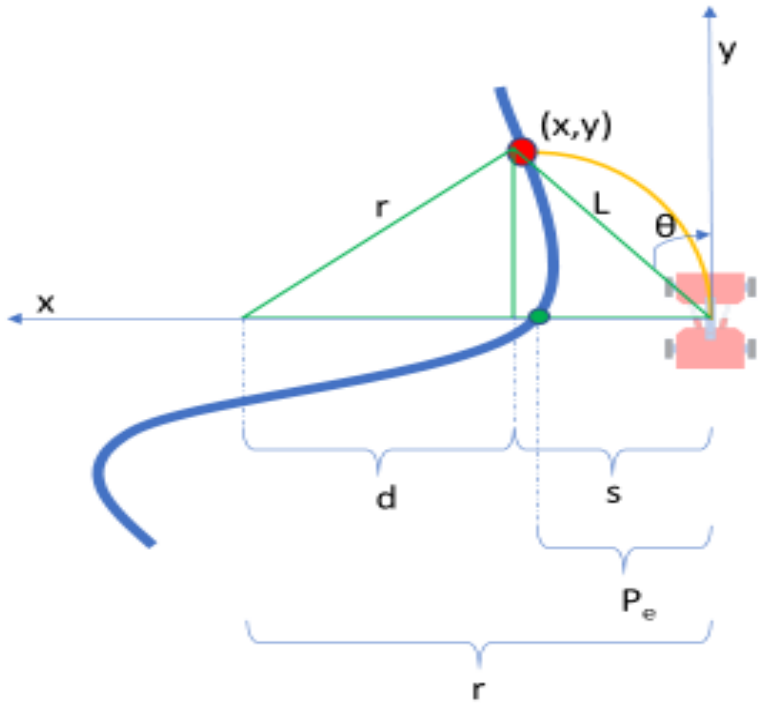


Figure 4. The geometry of the Pure Pursuit algorithm

Using the relationship of x, y , r, and L in figure 4, then we can derive the curvature. If x+d = r, and $d^2 + y^2 = r^2$, the relationship between r and the x, y coordinates can be solved to yield, $r = (x^2 + y^2) / (2x)$ (Rains et al., 2014). Then, the turning radius r becomes;

$$r = \frac{L^2}{2 * x} \tag{3}$$

Hence, the curvature **C** which is 1/r is given as;

$$C = \frac{2 * x}{L^2} \tag{4}$$

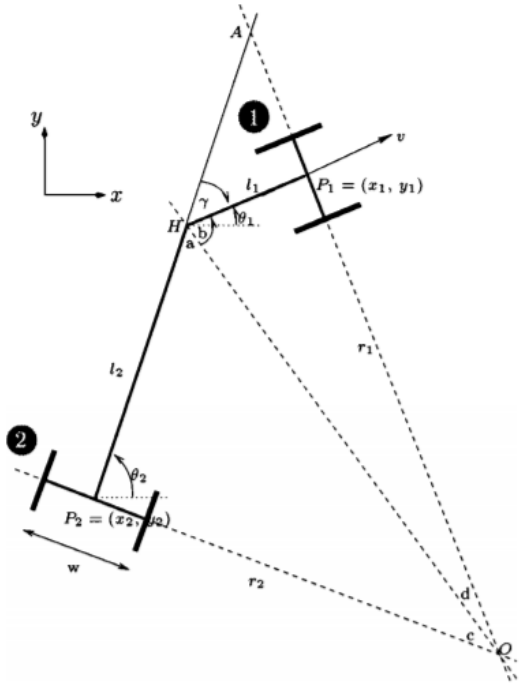


Figure 5. The geometry of the center-articulated rover while turning. $\mathbf{r}_1$ is the distance from the origin of the turning circle O (Corke & Ridley, 2001)

Consider Figure 5 that shows the geometry of the turning rover (Corke & Ridley, 2001).

At the point H, $a + b + \gamma = 180^0$. So, $b + d = 90^0$ and $a + c = 90^0$. Therefore we can substitute constants to make $\gamma = c + d$. γ is the articulation angle while θ is the heading angle.

Now, consider the right-triangle ΔOP_2A,

$$l_2 + \frac{l_1}{\cos\gamma} = r_2 \tan\gamma$$

$$(r_1 + l_1 \tan\gamma)\sin\gamma = l_2 + \frac{l_1}{\cos\gamma} \quad (5)$$

Then, we can simplify to get the turning radii, r_1 and r_2;

$$r_1 = \frac{l_1 \cos\gamma + l_2}{\sin\gamma}$$

$$r_2 = \frac{l_2 \cos\gamma + l_1}{\sin\gamma} \quad (6)$$

Since the rover is a symmetric vehicle, the distance from the center (L_f) to back tires or front tires is 91 cm. hence, we can simplify the radii equation above,

$$r = r_1 = r_2 = \frac{L_f \cos\gamma + L_f}{\sin\gamma} \quad (7)$$

Then, using trigonometry rules, *sin 2a = 2sin a cos a* and *cos 2a = = 2cos²a - 1*

$$\frac{r}{L_f} = \frac{\cos\gamma + 1}{\sin\gamma} = \frac{2\cos^2\frac{\gamma}{2}}{2\sin\frac{\gamma}{2}\cos\frac{\gamma}{2}}$$

Hence, by simplifying the trigonometry,

$$\tan\gamma = \frac{L_f}{r} \quad (8)$$

insert equation (3) which makes;

$$tan\gamma = 2 * L_f * \left(\frac{x}{L}\right)\frac{1}{L}$$

$$tan\gamma = 2 * L_f * (sin\,\theta)\frac{1}{L}$$

$$\gamma = 2 * tan^{-1}\left(\frac{2 * L_f * sin\,\theta}{L}\right) \quad (9)$$

So, equation (9) shows the relationship of articulation angle γ to look-ahead distance L, half-length of the rover L_f and heading angle θ. Consider, the vehicle at state (x_k, y_k) is articulating to the next goal (x_{k+1}, y_{k+1}). Next heading angle θ_{k+1} in relation to the current heading angle θ_k can be found by;

$$\theta_{k+1} = \theta_k - tan^{-1}\frac{x_{k+1} - x_k}{y_{k+1} - y_k} \quad (10)$$

Because, the rover has a slow turning action when moving, the horizontal distance of the vehicle from the goal should be increased by a certain factor "K"(See equation 11). The closest distance of the rover to the path is the path error. The next position of the path is moved further from the position by the factor K multiplied by Pe. This is the modified Pure Pursuit of the center-articulated rover. The value K is a random number that should be obtained by testing and experimentation. We started from 0.0 and increasing it in decimals to 4.0 while watching how fast the rover can act to approach and remain in the designated path.

$$x_{k+1} = x_{k+1} + K * P_e \quad (11)$$

Boll detection algorithm improvements using YOLOv3

Cotton boll images used for training the YOLOv3 deep neural network (DNN) model were augmented 27 times [Average blurring, Bilateral blurring, Blurring, Cropping, dropout, Elastic deformation, equalize histogram, flipping, gamma correction, Gaussian noise, gaussian blurring, median blurring, raise blue channel, raise green channel, raise hue, raise red channel, raise saturation, raise value, resizing, rotating, slat and pepper, sharpen, shift channels, shearing, and translation] using CLoDSA. CLoDSA is an open-source image augmentation library for object classification, localization, detection, semantic segmentation, and instance segmentation (*https://github.com/joheras/CLoDSA*). We collected and labeled 2085 images. We augmented the images to provide a new labeled dataset of 56,295 images.

The YOLOv3 model was used to train the dataset using Lambda Server (Intel Core i9-9960X (16 Cores, 3.10 GHz) with two GPUs RTX 2080 Ti Blowers with NVLink and Memory of 128 GB, Lambda Computers, San Francisco, CA 94107). 1000 iterations gave the optimal performance for YOLOv3. The training took only 4 hours to finish.

However, ZED Library called Zed-yolo in Github (*https://github.com/stereolabs/zed-yolo*), which is the free library provided by the manufacturer of ZED (Stereolabs Labs) to connect interface ZED camera images to YOLO models to do 3D object detection. Zed-yolo code had a bug in line 231 and 232. The transformation (interpolation) of the image size from the ZED image format to the OpenCV image format used INTER_LINEAR to change the format. In Python, OpenCV store images in NumPy arrays. The ZED SDK uses its own sl.Mat class (an application programming interface (API) that interface ZED camera to Python or C++ code) to store image data. The ZED SDK provides a function get_data() to convert the sl.Mat matrix into a NumPy array. Using this Numpy Array image (image) in equation (12) to resize distorts the quality of the image. We tried INTER_NEAREST interpolation but also didn't give good results (Figure 6).

image = cv2.resize(image, (lib.network_width(net), lib.network_height(net)), interpolation=cv2.INTER_LINEAR) (12)

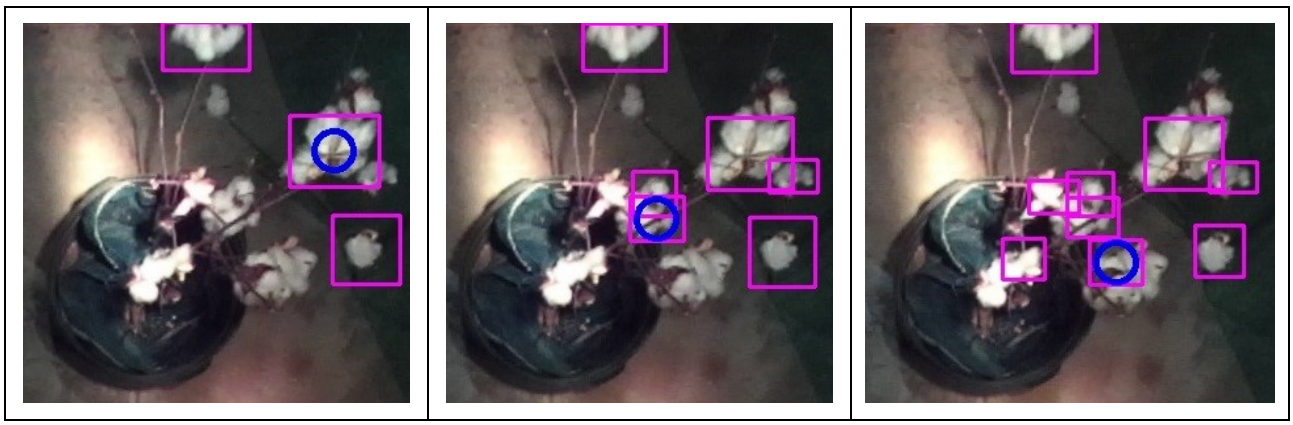


Figure 6. Detection of cotton bolls using Linear transformation (3 bolls), Nearest transformation (5 bolls), and the last with no transformation (all the bolls). The pink boxes are bounding boxes of the detected bolls, while blue represents the nearest boll to be picked.

Using the transformed image, the system was detecting images with less than 80% accuracy most of the time. We changed the code. Instead of transforming the image, we saved the ZED image and then used OpenCV to load the saved image to process the detection of the bolls using YOLOv3 (Figure 6 the right image). This improvement increased the detection of the cotton bolls to more than 95%. Figure 7 shows cotton detection using YOLOv3 and a modified code that doesn't use numpy array and instead stores the image using the ZED library and uses the saved image to detect the cotton bolls. You may watch boll detection using YOLOv3 (https://www.youtube.com/watch?v=hHqsFMcC-FE)

Figure 7. Boll detection using YOLOv3. Right when it was cloudy and left in direct sunlight.

Picking algorithm

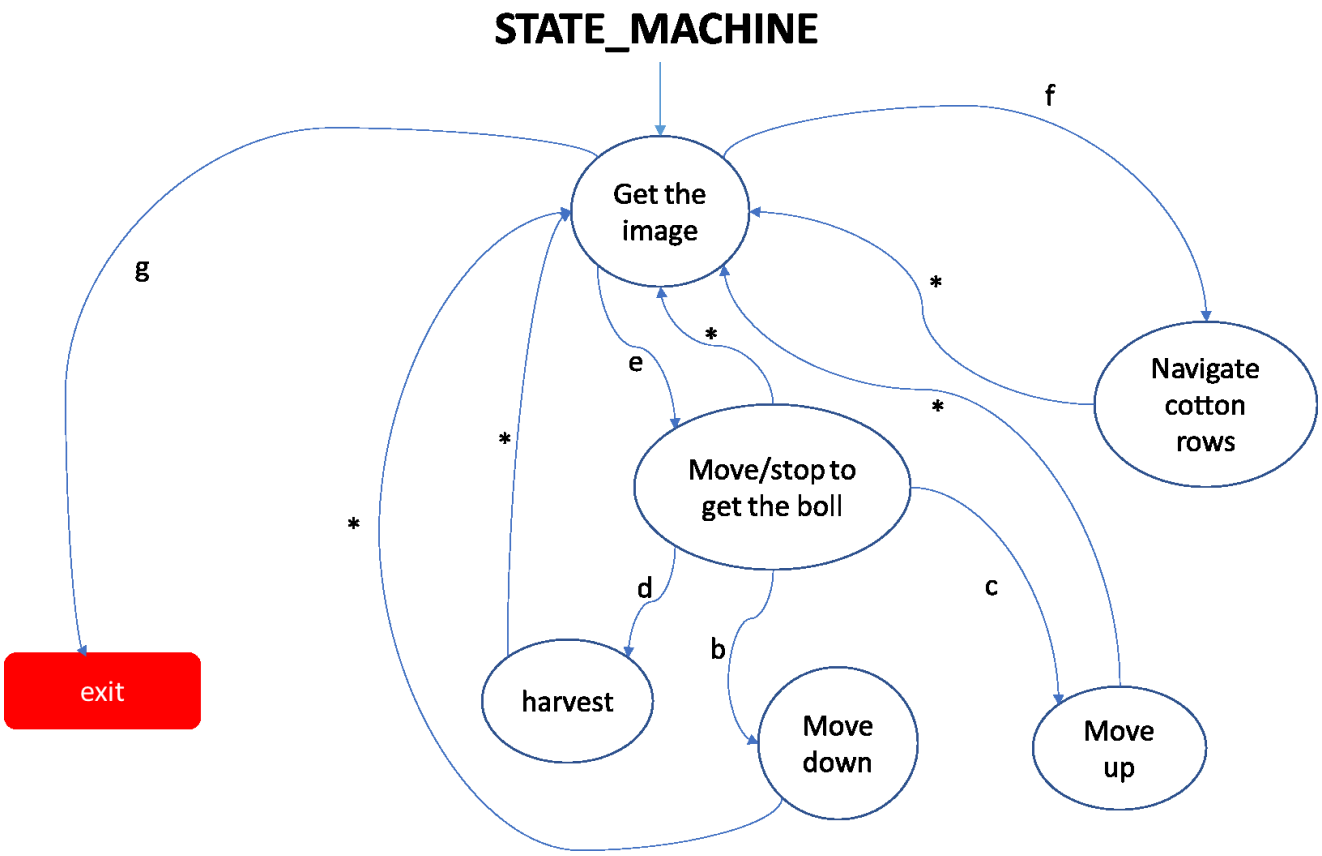
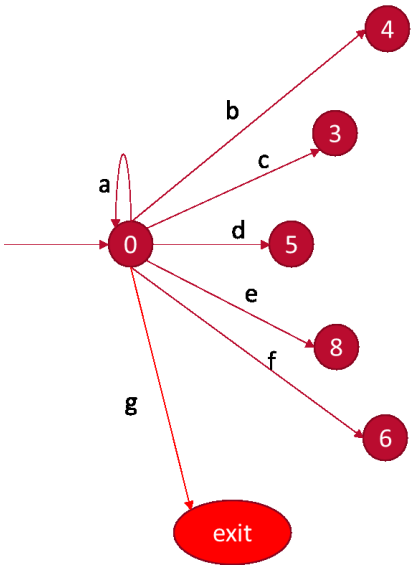


Figure 8. State Machine Diagram of the rover states and transitions



0: Get the image
4: Move down
3: Move up
5: Harvest the boll (manipulator back/forth)
8: Move to get the boll (back/forth)
6: Send signal for rover to navigate (3rd party program)

a: if the boll cannot be found
b: if the boll is down and manipulator up
c: if the boll is up and manipulator down
d: if the boll and manipulator at the same level
e: if the boll is in front or at back of the manipulator
f: boll not found after 2 attempts of 'a' navigate along the rows
g: fails if the camera doesn't get the images

Figure 9. Linear transformation of the State Machine Diagram (Fig 8) of the rover states and transitions

Robot tasks and actions were categorized as states, and state "transitions" were modeled in a task-level architecture to create complex robot behavior. This approach provides a maintainable and modular code. Using an open-source ROS library known as SMACH, the tasks can seemingly be implemented to build complex robot behavior and deliver state-of-the-art performance.

Fig 8 and 9; the system had 6 basic states and 7 transitions. After every state, the system goes back to state 0, which is getting the image and look for cotton bolls. If the boll is found, the system will calculate the distance of the boll from the manipulator in 3-dimensional space (X,Y, and Z). If the boll is lined up horizontally, then the system will get the manipulator to move up or down relative to the position of the manipulator to the boll. If the boll is at the same level, then

the system will pick it. If the boll is in front or back, the system will send a signal for the rover to move forward or back using PID control. If the system fails to see any bolls, the machine will continue to pass over the cotton rows using the modified pure pursuit algorithm. Table 1 describes the detection of the bolls and actions taken by the state machine algorithm to accommodate the robot transition of the tasks.

Table 1. Algorithm describing the detection of cotton bolls

Algorithm 1: Algorithm describing the detection of cotton bolls

Input: current video frame,

Output: Decision to move manipulator or rover Ci

1. Get manipulator position Xm, Ym and Zm
2. Get prediction results of the YOLO model
3. Get the centroid of each boll detected {Oj}
4. FOR EACH Oj in {Oj}
    a. boll <- calculate the closest distance of the centroid
5. END FOR
6. Get the closest boll position
7. Calculate the position of the boll Xb, Yb, and Zb
8. Find the difference between (Xm and Xb) , (Ym and Yb) and (Zm and Zb)
9. IF (Yb > Ym) transition e (move forward)
10. IF (Yb < Ym) transition e (move backward)
11. IF (Yb = Ym and Zm > Zb) transition b (move the arm up)
12. IF (Yb = Ym and Zm < Zb) transition c (move the arm down)
13. IF (Yb = Ym and Zm = Zb and Xm – Xb < 37 cm) transition d (pick the boll) ##the manipulator can only cover bolls close to it to about 37 cm
14. Return the state decision {Ci}

Field Testing

Navigation and Picking were tested in November and early December of 2020 at the UGA grounds. A preliminary experiment was done at the UGA grounds behind the engineering annex located at (31.475340N, 83.528968W) while field experiment was done at the Entomology farm (31.472985N, 83.531228W) near Bunny Run Rd. in Tifton Georgia (Figure 11?).

A preliminary experiment was done to study the navigation behavior of the rover when parameters like ROS update rates, look ahead, and path error were altered. It was done to calibrate the system to perform well in the cotton field.

The preliminary experiment involved 6 tests. When ROS rates were set to 10Hz and 3Hz, look ahead was set at 1m and 3m, and path error was set to 0 and 1.5m.

The field experiment was done after establishing the calibrated parameters. Three tests were conducted for navigation on 18.5m rows on 21st October 2019, and two tests (5.3 m) for picking the cotton bolls were conducted on 22nd November 2019 and 2nd December 2019. For picking cotton, we measure Action Success Ratio (ASR) which is the ratio of the number of the picked bolls to a number of all cotton bolls present and Manipulator Reaching ratio (MRR) which is the ratio of the bolls seen and attempted to be picked to the number of all bolls present.

Both experiments were conducted by setting the rover to follow the prescribed path. The prescribed path was obtained by moving the rover manually and collecting the GPS path positions. For the preliminary experiment, a path was obtained by driving the rover around randomly at the UGA grass grounds. For the field experiment, the path was obtained by driving the rover over two rows of cotton planted on 91cm centers.

Results and Discussions

Preliminary Experiment

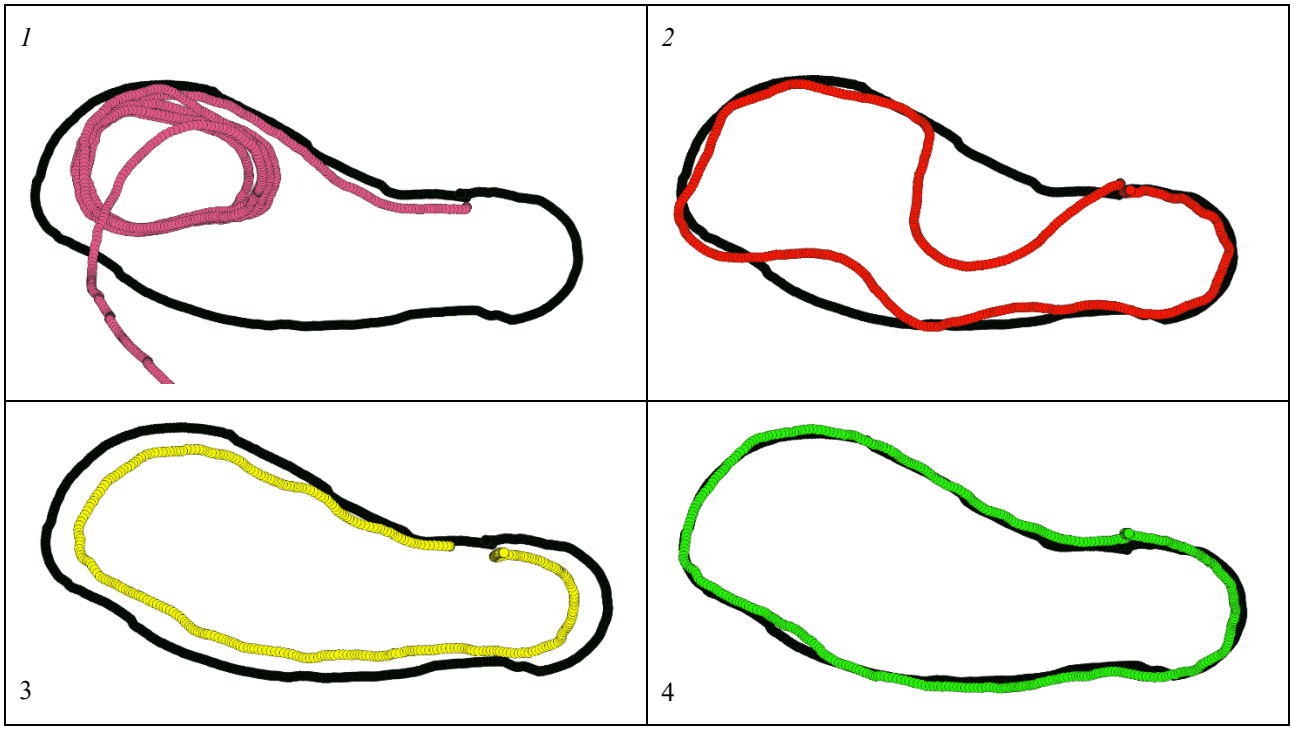


Figure 10. Performance of the rover when the (1)ROS updates are so high (10Hz), (2)Short look ahead distance (1m), (3) no path error correction is applied and (4)successful path tracking when look-ahead is 3m, path error is 1.5 times path error and ROS updates at 1Hz

The rover navigation tracking was negatively affected when ROS update rates were increased, or no path error correction was applied, and when very short look-ahead was applied (Figure 10). Pure pursuit algorithm is a technique whose goal is to make sure that the rover regains the designated path by articulation when it loses and maintains when it is on the designated path. Fast ROS update rates affected system performance since the mechanical response of the machine was slow to manage the updates that were provided by the controller. This short time between the new reading input reading of the system meant that the reaction time of the vehicle to the control signal was slow, and the vehicle reaction was always too far behind the control decision,causing the rover to lose tracking control. The ROS update rate is a very expensive parameter if it is not set right. It is the same with short look-ahead distances. With a small look ahead distance, the robot tries to move quickly to regain the path it has lost. However, this action causes the robot to overshoot the path and oscillates along the prescribed path.

With no path error corrections (when K is Zero), the rover can never converge to the path over time. Figure 10 (3) shows how the vehicle was not able to converge to the path, which means the error was consistently maintained. To avoid this behavior, modified pure pursuit increased the error of the system by 1.5 using equation (11) to force the system to converge to the path. It means if the error was big, then the system amplified the error forcing the rover to quickly act. When the error was small, the system acted slowly too because the amplification of the error became small too.

Field Experiment

The rover navigation was tested over the two rows with three repetitions (Figure 11).

Figure 11. The conditions of the farm at the time of the experiment

Fig 11 shows the navigation path traces as recorded by the GPS for the three tests. The rover performed well visually. The third test can clearly show that the rover was slipping compared to the first and second tests. The grass around the farm was wet and caused wheel sliding when turning at the end-of-row. But also, this may happen if the input readings were updated very fast for the machine to handle them or the system was late to take quick action when updated on articulation so the pure pursuit may lose the designated path.

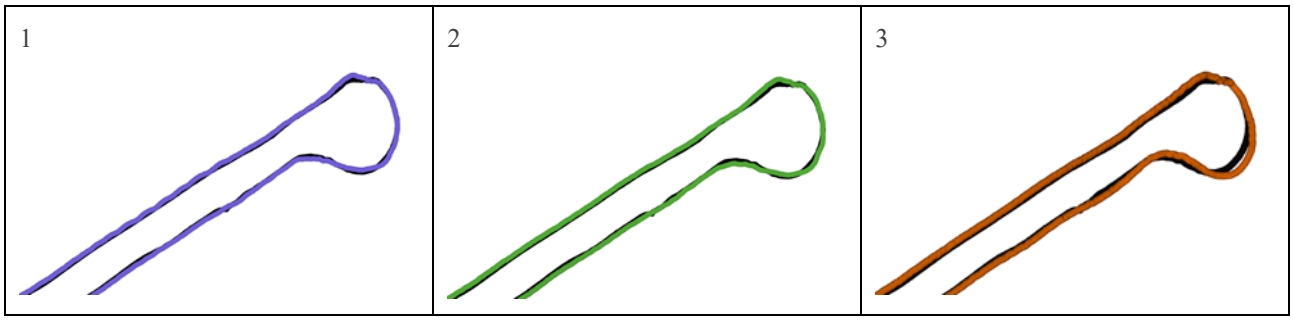


Figure 12. Path tracking of the prescribed path (black). The blue, green and orange path tracking is the GPS generated path of the rover when following the rows using prescribed path (black)

The rover was able to follow the path carefully and turn to come back to the next cotton row. The absolute error distribution was determined to characterize rover navigation behavior. The rover performed well, as figure 13 shows most of the path errors were less than 30 cm (0.30 m).

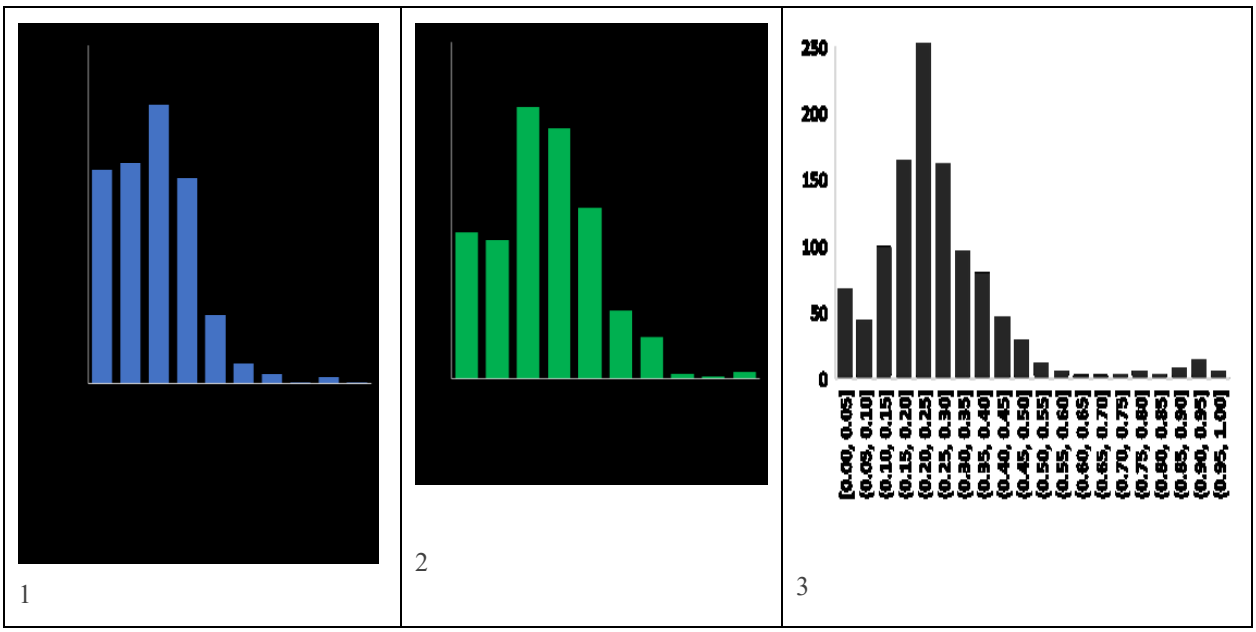


Figure 13. The distribution of absolute errors is clearly skewed to the left. It shows that the rovers were navigating along the rows safely with less than 30 cm absolute error.

Table 2. The absolute error mean, median, standard deviations, the minimum and maximum absolute error of the three tests to navigate along the cotton rows. The performance was measured overall, navigation over the first row, turning maneuver of the rover, and over the second row. The rover performs well for the first row and worst when turning. The worse outliers are obtained when the rover is turning. Overall the system performed adequately to pass over the cotton rows.

(metres)	Overall	1st Row	Turning	2nd Row
1st Experiment				
Mean	0.119	0.096	0.123	0.134
Median	0.118	0.090	0.094	0.137
Std Dev	0.073	0.057	0.104	0.048
Minimum	0.002	0.002	0.002	0.002
Maximum	0.456	0.241	0.456	0.257
2nd Experiment				
Mean	0.154	0.134	0.157	0.170
Median	0.150	0.145	0.125	0.163
Std Dev	0.084	0.068	0.119	0.056
Minimum	0.001	0.001	0.001	0.006
Maximum	0.473	0.284	0.474	0.327
3rd Experiment				
Mean	0.263	0.178	0.370	0.250
Median	0.231	0.172	0.361	0.238
Std Dev	0.165	0.092	0.255	0.059
Minimum	0.001	0.001	0.002	0.077
Maximum	0.986	0.419	0.986	0.451
Overall Performance				
Mean	0.189	0.141	0.233	0.194
Median	0.172	0.138	0.180	0.193
Std Dev	0.137	0.082	0.215	0.075
Minimum	0.001	0.001	0.002	0.002
Maximum	0.986	0.419	0.986	0.450

Figure 13 and Table 2 shows that the rover performance was adequate as it was safely navigating along the rows without driving over the plants. Most of the errors were less than 30 cm from the prescribed path. The rover was perfectly converging back to the prescribed path except for turning maneuver, which provided a big challenge to the rover. The first and second tests were good, but the third one had trouble turning in the muddy end-row that caused the wheels to slide.

Picking of the cotton bolls

Two tests to investigate the effectiveness of the rover to pick the bolls were successfully done, and results show that for the first test, the robot picked 67 and left behind 17 bolls, which means the Action Success Ratio (ASR) was 80%. The rover was able to reach (Manipulator Reaching ratio (MRR)) 94% of the bolls (79 bolls). The distance covered was around 5.3 meters. For the second test, the robot picked 89 and left behind 26 bolls, which means the Action Success Ratio (ASR) was 77%. The rover was able to reach (Manipulator Reaching ratio (MRR)) 96% of the bolls. The distance covered was around 5.3 meters. The average ASR was 78.5%. the average MRR was 95%.

Conclusion

The autonomous robot was developed, and preliminary testing has shown it can navigate safely along the cotton rows and pick the cotton (https://twitter.com/kadefue/status/1201640088322502656?s=20). The robot had an absolute error mean of 0.189 m, a median of 0.172 m, a standard deviation of 0.137 m, and a maximum of 0.986 m. Most of the path error occurred during turning when it is more challenging to maintain absolute path tracking, and the wheels were slipping in the wet grasses on the edge of the field. The challenge for the system to maintain path tracking while making sharp turns at the end-of-row is mainly attributed to the slow speed of the linear actuator controlling the swashplate angle and the deadband in the center of the swash plate angle that shows no hydraulic fluid rate change for approximately 2.5 cm of actuator movement. This will be addressed by changing the actuator and the lever arm length to provide more response for each cm of actuator movement. While navigating in the relatively straight rows of cotton, the results showed that the system was carefully navigating along the rows and had minimal damage to the plants. The rovers also picked the bolls at the average Action Success Ratio (ASR) of 78.5%, and it was able to reach 95% of the bolls. We will address improving ASR using a modified end-effector and an extra camera looking up into the canopy to see more of the bolls as they start to open early in the season.

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 support in building the camera platform and collection of data. Also, we also thank Mr. Logan Moran for Field Testing, Mr. William Hill, and Mrs. Tearston Adams for labeling the images, and Mr. Jesse Austin Stringfellow for 3D design and printing.

References

Coulter, R. C. (1992). Implementation of the pure pursuit path tracking algorithm (No. CMU-RI-TR-92-01). Carnegie-Mellon UNIV Pittsburgh PA Robotics INST.

Corke, P. I., & Ridley, P. (2001). Steering kinematics for a center-articulated mobile robot. IEEE Transactions on Robotics and Automation, 17(2), 215-218.

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. ASABE Paper No 1800831. St. Joseph, MI: ASABE. doi:10.13031/aim.201800831

Hayes, L. (2017). Those Cotton Picking Robots. Available at http://georgia.growingamerica.com/features/2017/08/those-cotton-picking-robots/

Rains, G. C., Faircloth, A. G., Thai, C., & Raper, R. L. (2014). Evaluation of a simple, pure pursuit path-following algorithm for an autonomous, articulated-steer vehicle. Applied engineering in agriculture, 30(3), 367-374.

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.

Wan, E. A., & Nelson, A. T. (2001). Dual extended Kalman filter methods. Kalman filtering and neural networks, 123.