electronics

Article

Center-Articulated Hydrostatic Cotton Harvesting Rover Using Visual-Servoing Control and a Finite State Machine

Kadeghe Fue [1,2,*], Wesley Porter [3], Edward Barnes [4], Changying Li [1] and Glen Rains [2,*]

1 College of Engineering, University of Georgia, Athens, GA 30602, USA; cyli@uga.edu
2 Department of Entomology, University of Georgia, Tifton, GA 31793, USA
3 Department of Crop and Soil Sciences, University of Georgia, Tifton, GA 31793, USA; wporter@uga.edu
4 Cotton Incorporated, Cary, NC 27513, USA; ebarnes@cottoninc.com
* Correspondence: kadefue@uga.edu (K.F.); grains@uga.edu (G.R.); Tel.: +1-229-386-3520 (G.R.)

Received: 1 July 2020; Accepted: 28 July 2020; Published: 30 July 2020

Abstract: Multiple small rovers can repeatedly pick cotton as bolls begin to open until the end of the season. Several of these rovers can move between rows of cotton, and when bolls are detected, use a manipulator to pick the bolls. To develop such a multi-agent cotton-harvesting system, each cotton-harvesting rover would need to accomplish three motions: the rover must move forward/backward, turn left/right, and the robotic manipulator must move to harvest cotton bolls. Controlling these actions can involve several complex states and transitions. However, using the robot operating system (ROS)-independent finite state machine (SMACH), adaptive and optimal control can be achieved. SMACH provides task level capability for deploying multiple tasks to the rover and manipulator. In this study, a center-articulated hydrostatic cotton-harvesting rover, using a stereo camera to locate end-effector and pick cotton bolls, was developed. The robot harvested the bolls by using a 2D manipulator that moves linearly horizontally and vertically perpendicular to the direction of the rover's movement. We demonstrate preliminary results in an environment simulating direct sunlight, as well as in an actual cotton field. This study contributes to cotton engineering by presenting a robotic system that operates in the real field. The designed robot demonstrates that it is possible to use a Cartesian manipulator for the robotic harvesting of cotton; however, to reach commercial viability, the speed of harvest and successful removal of bolls (Action Success Ratio (ASR)) must be improved.

Keywords: robotics; machine vision; cotton harvesting; 3D position estimation; GNSS; cotton; ROS

1. Introduction

Cotton is an essential commercial crop worldwide. The cotton industry in the USA has been growing and is now the third largest agricultural industry in the USA, employing more than 200,000 people, with a value of $25 billion per year [1]. The USA is third in the production of cotton in the world, behind India and China. As a large industry, however, cotton production operations have faced multiple challenges, of which timely harvesting of quality cotton fiber is among the most pressing [2]. Since its introduction in the 1950s, the practice of mechanical harvesting after defoliation has provided fast harvesting speed, but also substantial losses in quantity and quality of cotton [2]. Open cotton bolls can wait up to 50 days to be picked when at least 60% to 75% of the cotton bolls are opened [3]. Moreover, harvesting is only recommended when it is done while moisture content in fiber is less than 12% after the application of defoliants [4]. The waiting time exposes the open bolls to harsh conditions that degrade quality. Any change that would reduce cotton losses, improve quality, and increase return on investment would be welcomed by the industry.

In most cases, the mechanical harvester is huge and expensive (a 2019 picker costs around $725,000 and sits in storage more than nine months a year, without being used). Cotton combines also weigh more than 33 tons, which can cause soil compaction that reduces land productivity. The maintenance of such machines is also expensive and complicated [5]. Repairing breakdowns in the field can take days, which can reduce operating efficiency and expose bolls to further weather-related quality degradation. In addition, most of the machines use proprietary software and hardware that prevent farmers from repairing their machines and, therefore, deny the use of the right-to-repair tools that they own [6]. Finally, cotton plants must be chemically defoliated to harvest, which can cause an additional expense to the farmer [3].

Furthermore, the labor shortage in agriculture is getting worse, while the cost and the wage of available labor are skyrocketing [7–9]. Moreover, the average age of 58 years old among the farmer population is threatening the development of agriculture, as they are too old to attend to most of the manual work required in farming [10]. The emergence of robotics in agriculture, particularly in specialty crops, has created an opportunity in the domain of row crops, such as cotton, which have received little attention until recently [11–15]. The most recent research done by Reference [15] discussed many kinds of robots and found that, currently, there are more robots developed in harvesting. Still, they did not mention any robotic system for cotton harvesting [15]. To the best of our knowledge, there have been no commercial cotton-harvest robots developed [14]. Most robotic harvesting investigations are examining how fruit can be picked individually to mimic conventional hand-harvesting [11,14].

Consequently, robotics may provide an efficient and cost-effective alternative for small farmers unable to buy and own large machines [11,12,16,17]. To achieve a robust robotic harvesting system, the robot designs must consider four aspects carefully: sensing, mobility, planning, and manipulation [14]. Locomotion, stability, and balancing in mobility is so challenging in agricultural robots because they move on uneven terrain, and oscillations might alter their position easily; hence, sensors like LIDAR and GPS can be utilized to help the machine to maintain its path efficiently [18–21]. Hence, a predefined navigation path and a Real-Time Kinematic–Global Navigation Satellite System RTK–GNSS can provide absolute path following for row crops, but the design of the manipulators must be considered carefully to match the harvesting speed and efficiency of harvest requirements to be economically viable for farmers. There have been several approaches proposed by scientists using high Degree-of-Freedom (DOF) manipulators. Nonetheless, the systems have proven to be slow because of the extensive matrix calculations required to determine joint manipulations that move the end-effector to the fruit for harvesting [22]. For row-crops with a high number of fruits to harvest, the best option would be a capable manipulator that harvests while moving. A potential solution is to use multiple harvesting manipulators with low degrees of freedom.

In this study, a four-wheel center-articulated hydrostatic rover with a 2DOF Cartesian manipulator that moves linearly, vertically, and horizontally was developed. The kinematics of the manipulator were analyzed to calculate the arm trajectories for picking cotton bolls. Image-based visual servoing was achieved by using a stereo camera to detect and localize cotton bolls, determine the position of the end-effector, and decide the movement and position of the rover. Additionally, a Proportional-Integral-Derivative Controller (PID) control was developed to enhance the robot's movement and control position along the cotton rows, using feedback control after obtaining visual information. A PID was used to control the hydrostatic transmission, which rotated to the robot's tires. The articulation angle was controlled by using a proportional controller to ensure that the vehicle maintained its path within the cotton rows. The robot used an extended Kalman filter to fuse the sensors to localize the rover's position while harvesting cotton bolls [23]. Therefore, the specific objectives of this study were as follows:

1. To design and develop a cotton harvesting robot, using a center-articulated hydrostatic rover and 2D Cartesian manipulator; and
2. To perform preliminary and field experiments of the cotton-harvesting rover.

2. Materials and Methods

2.1. System Setup

The rover (Figures 1 and 2) used in this study was a four-wheel hydrostatic vehicle (West Texas Lee Corp., Lubbock, Texas) that was customized to be controlled remotely by using an embedded system to navigate autonomously in an unstructured agricultural environment [24,25]. The rover was 340 cm long and with front and back parts being 145 cm and 195 cm long, respectively. The rover was 212 cm wide, with a tire width of 30 cm. Each of the four tires had a radius of 30.48 cm and a circumference of 191.51 cm. The rover's front-axle and rear-axle were 91 cm from the center of the vehicle, and the ground clearance was 91 cm. To make turns, the rover articulated (bent) in the middle. A 2DOF manipulator was attached to the front of the rover. The manipulator consisted of two parts: a horizontal axis, which was 70 cm, and a vertical axis, which was 195 cm. The manipulator was placed in the front of the rover with a 27 cm ground clearance. The rover had three electronic controllers: a master controller, navigation controller, and manipulation controller.

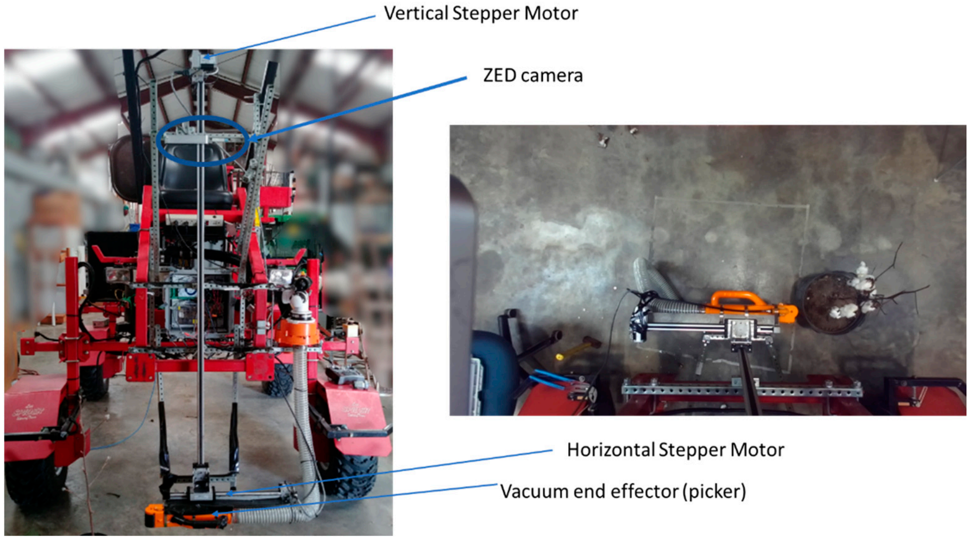


Figure 1. Left image: cotton-harvesting robot view from the front, which shows the ZED camera facing down. Right right: sample image obtained by the ZED camera that shows cotton manipulator and potted cotton plant with bolls.

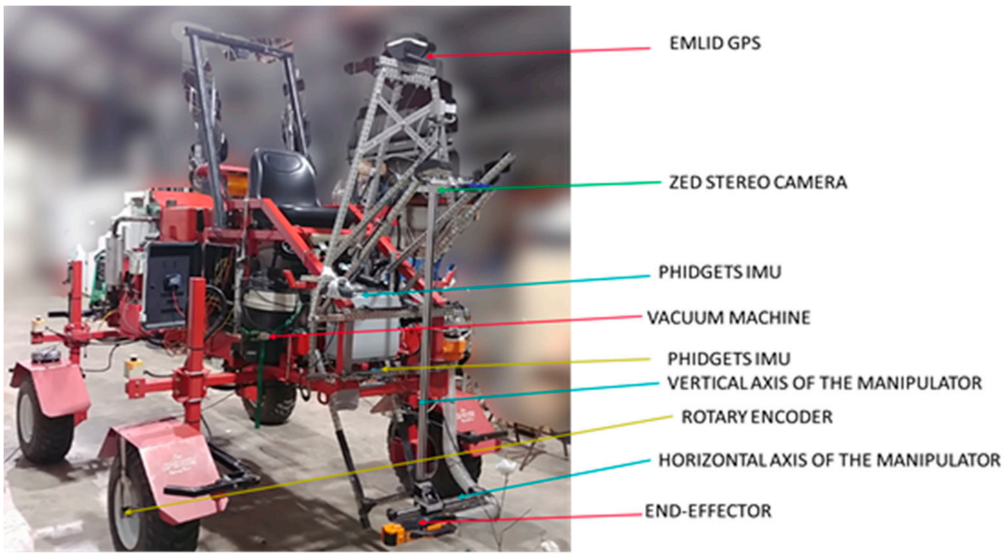


Figure 2. The robotic arm, vacuum, and sensors mounted on the red rover.

2.2. Master Controller System

The master controller (Figure 3) was installed with the embedded system (NVIDIA Jetson AGX Xavier development kit, Nvidia Corp., Santa Clara, CA, USA) connected to four sensors: two Inertial Measurement Units (IMUs), a stereo camera, and an RTK–GPS. IMUs were publishing data at 120 Hz, while the RTK–GPS was publishing data at 5 Hz. The two IMUs (Phidget Spatial Precision 3/3/3 High-Resolution model 1044_1B, Calgary, AB, Canada) were placed in front of the rover. The first IMU was placed 95 cm above the ground and 31 cm from the front of the vehicle (Figure 4). The second IMU was placed 132 cm above the ground and 46 cm from the front of the vehicle. The RTK–GNSS receiver (EMLID Reach R.S., Mountain View, CA, USA) with an integrated antenna was placed 246 cm above the ground and 30 cm from the front of the vehicle. The RTK correction signal was obtained by using NTRIP signal (eGPS Solutions, Norcross, GA, USA) with a mounting point within 2 miles of the test plot and downloaded to the Emlid through a Verizon modem (Inseego Jetpack MiFi 8800L, Verizon Wireless, New York, NY, USA) hotspot and 802.11 wireless signals.

An RGB stereo camera (a ZED camera, Stereo labs Inc., San Francisco, CA, USA) was installed and used to acquire images. The ZED was 175 mm × 30 mm × 33 mm and had 4M pixel sensor per lens with large 2-micron pixels. The left and right sensors were 120 cm apart. The sensor was placed 220 cm above the ground in front of the vehicle, facing downward, so it could image cotton bolls. The sensor took 720 p resolution images at the rate of 60 frames per second. The ZED camera was chosen because of the need to work outdoors and provide depth data in real time. The ZED camera provided a 3D rendering of the scene, using the ZED Software Development Kit (SDK), which was compatible with the robot operating system (ROS) and OpenCV library.

ROS provided all the services required for robot development, such as device drivers, visualizers, message-passing, package design, and management and hardware abstraction [26]. ROS was initiated remotely by using a client machine, and images were acquired directly from the ZED SDK, which took images and processed the 3D point clouds. Then, images were parsed to the processing unit and analyzed by using OpenCV (version 3.3.0) machine vision algorithms.

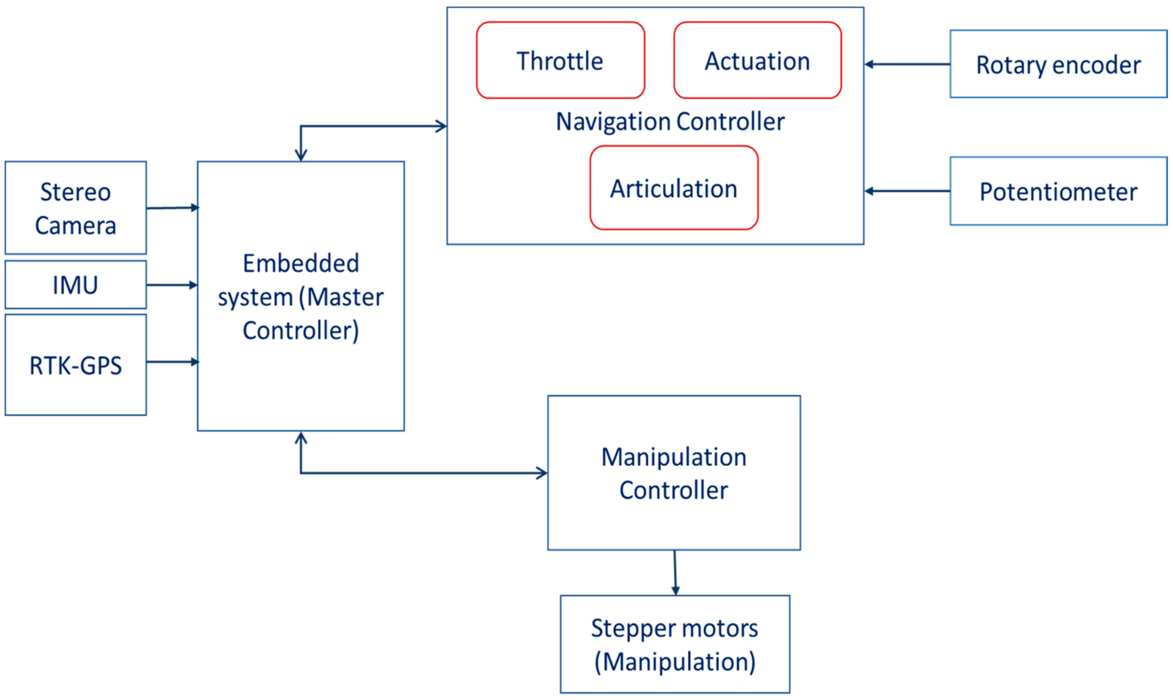


Figure 3. A contextual block diagram of the robotic system hardware. The Inertial Measurement Units (IMUs) were publishing at the rate of 120 Hz, Real-Time Kinematic (RTK)–GPS at 5 Hz, and stereo camera at 60 fps. The potentiometer and encoders where publishing changes only.

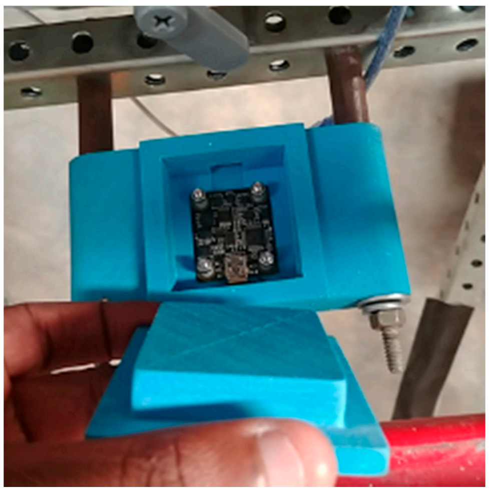


Figure 4. The first IMU Phidget Spatial Precision 3/3/3 High-Resolution model 1044_1B attached by using a 3D printed box to attach it to the rover.

2.3. Rover Navigation Controller Using Extended Kalman Filter for Robot Localization

The front two wheels of the rover were connected to rotary encoders (Koyo incremental (quadrature) TRDA-20R1N1024VD, Automationdirect.com, Atlanta, GA, USA). The encoders (Figure 5) were connected to the Rover navigation controller (Arduino Mega 2560, Arduino LLC, Somerville, MA, USA), using four wires (signal A, signal B, power, and ground) to register wheel rotation by detecting the rising edge of the generated square waves. A high-precision potentiometer (Vishay Spectral Single Turn, Malvern, PA, USA) reported the articulation angle of the vehicle by measuring the electric potential caused by the turn of the vehicle. The potentiometer was connected to the rover navigation controller.

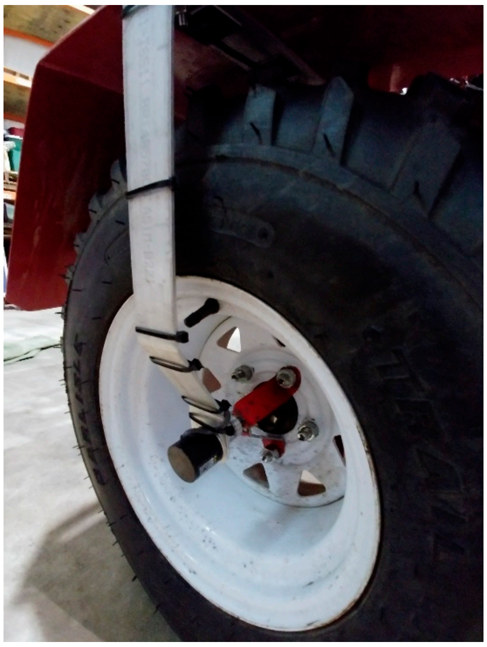


Figure 5. Encoder installed on the front tire of the rover to provide input pulses, which indicate how far the rover has moved.

The rover navigation controller received a signal from the embedded computer to control the rover articulation, actuation, and throttling. The sensors (RTK–GNSS, IMU, and encoders) were fused by using a ROS software package "robot_localization" [23]. The package "robot_localization" provided continuous nonlinear state estimation of the mobile vehicle in a 3D space by using signals obtained from the RTK–GPS, two IMUs, and wheel encoders. The fusion of the data was done by using state estimation node "ekf_localization_node" and "navstat_transform_node" [23]. The vehicle was configured using the procedure described in http://docs.ros.org. The IMUs published two ROS topics (imu_link1/data and imu_link2/data), the encoders published wheel odometry (/wheel_odom), and RTK–GNSS published /gps/fix signal. The topics were both sent to ekf_localization to get vehicle state estimation simultaneously [23]. The system used two IMUs, RTK–GNSS, and odometry of the encoders, because Table 1 shows that this configuration provided excellent results [23]. Unlike the GNSS configuration shown in Table 1, our GNSS was a low-cost RTK–GNSS system that provided centimeter-level accuracy, so configuration (odometry + two IMUs + one GNSS) was a proper configuration to adopt.

Table 1. Errors for five difference sensor configurations [23].

Sensor Set	Loop Closure Error x,y (m)	Estimate Std. Dev. x,y (m)
Odometry (dead reckoning)	*69.65,160.33*	*593.09,359.08*
Odometry + one IMU	*10.23,47.09*	*5.25,5.25*
−Odometry + two IMUs	*12.90,40.72*	*5.23,5.24*
Odometry + two IMUs + one GNSS	*1.21,0.26*	*0.64,0.40*
Odometry + two IMUs + two GNSSs	*0.79,0.58*	*0.54,0.34*

The hydraulic motors mounted to the rover wheels were controlled by using a servo to the engine throttle and a servo to the variable-displacement pump swashplate arm. Each front tire was connected to a rotary encoder to provide feedback on the movement of the rover along the crop rows. The throttle was connected to the onboard Kohler Command 20 HP engine (CH20S, Kohler Company, Sheboygan, WI, 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 displacement was controlled by a swashplate for directing the forward and rearward movement of the rover. The swashplate angle was controlled by the rover controller, which determined the placement of the linear electric servo (Robotzone HDA4, Servocity, Winfield, KS, USA). Left and right articulations were controlled by using linear hydraulic actuators 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, Klettgau-Griessen, Germany) in tandem with the Oilgear pump. The rover could turn a maximum of 45 degrees with a wheelbase of 190 cm.

2.4. Manipulation Controller

The robot manipulation controller received a 4-byte digital signal from the Jetson Xavier embedded computer. 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, GA, USA), which in turn sent it to the appropriate stepper signal for action. The manipulator controller was connected to the Jetson, using a USB 3.0 hub shared by the ZED camera.

The robotic manipulator was designed to work within a two-dimensional Cartesian coordinate system (Figure 6). Each arm of the manipulator was moved by using two 2-phase stepper motors (MS048HT2 and MS200HT2, ISEL Germany AG, Eichenzell, Germany). The MS048HT2 model stepper motor was installed to drive the horizontal linear axis arm (60 cm long), and the MS200HT2 to drive the vertical linear axis arm (190 cm long). The connecting plates and mounting brackets used a toothed belt that was driven by the stepper motor. Two micro-stepping drives (Surestep STP-DRV-6575 micro-stepping drive, Automation Direct, Cumming, GA, USA) were installed to provide accurate position and speed control with a smooth motion. The micro-stepping drive DIP switch that controlled the motors was set to run a step pulse at 2 MHz and 400 steps per revolution. This setting provided smooth motion for the manipulator. The step angle was 1.8°. The arms of the manipulator were connected in a vertical crossbench Cartesian configuration (Figures 2 and 6). 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 and was 9 mm wide. The error of the toothed belt was ±0.2 mm per 3 mm interval.

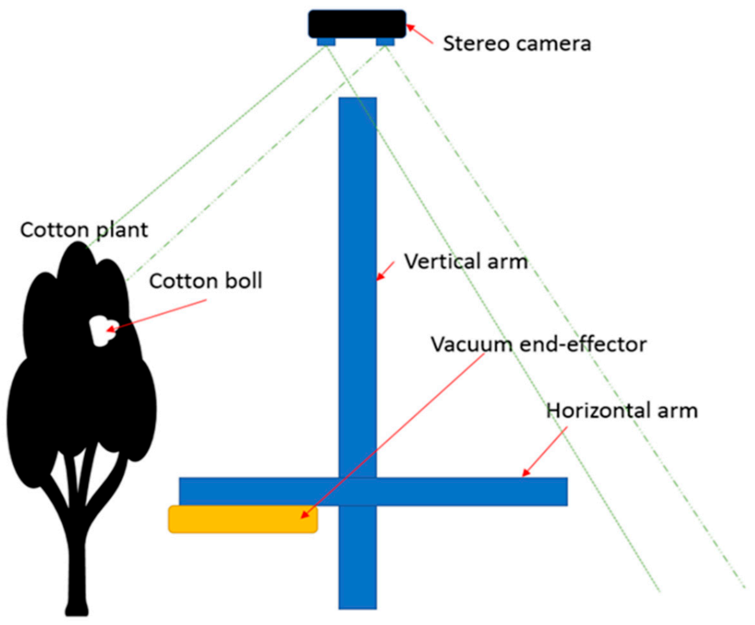
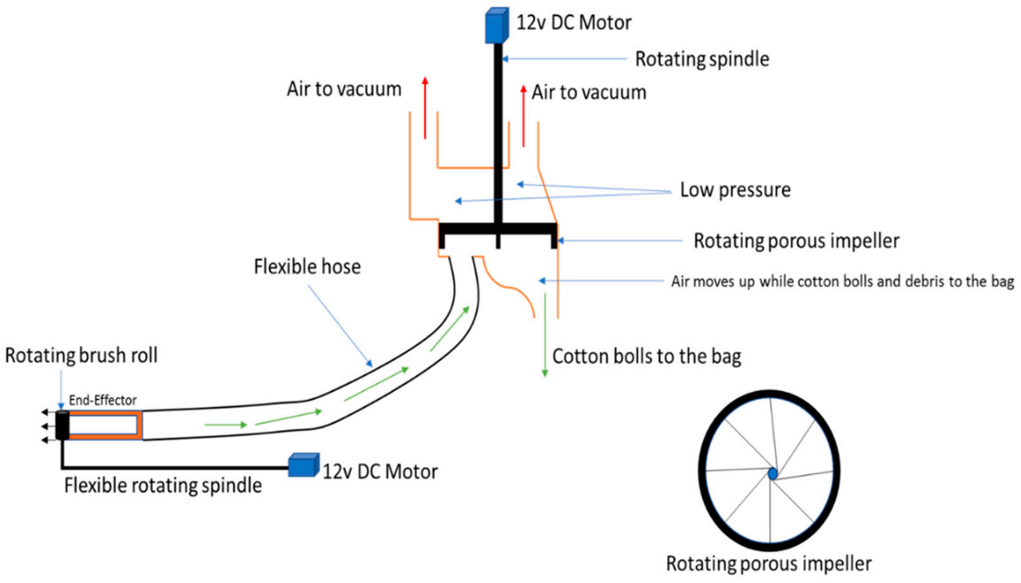


Figure 6. Robotic Cartesian arm contextual diagram.

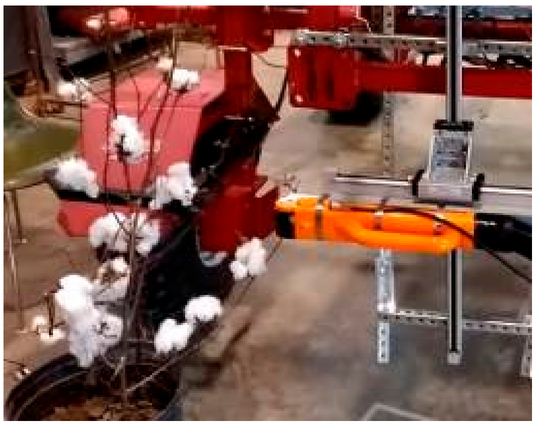
A 2.5 hp wet/dry vacuum was installed on the rover, to help pick and transport the picked cotton bolls—the vacuum connected to the end-effector via a 90-cm flexible plastic hose. Cotton bolls were vacuumed into the hose, which was placed close to the cotton bolls (Figure 7). The end-effector had a rotating brush roller that grabbed and pulled cotton bolls, using a combination of vibration, rotation, and sweeping motions. The brush roll was powered by a 12 VDC motor (Figure 7). The cotton bolls were grabbed and passed through a flexible hose to a porous impeller mounted with the suction opening from the vacuum. The porous impeller was rotated by using the 12 VDC motor, and the cotton bolls were dropped into a bag.

2.5. Boll Detection Algorithm Improvements Using Tiny YOLOv3

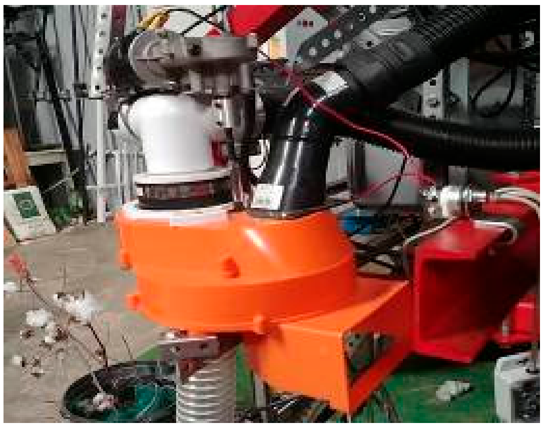
Cotton boll images were used to train a tiny YOLOv3 deep neural network model [27]. The tiny YOLOv3 is a simplified version of YOLOv3, which has seven convolutional layers [27]. Tiny YOLOv3 is optimized for speed with reduced accuracy compared to YOLOv3 [27]. The images were augmented 26 times, using various techniques, namely 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, salt and pepper, sharpen, shift channels, shearing, and translation. Augmentation was accomplished by using the CLoDSA library. CLoDSA is an open-source image augmentation library for object classification, localization, detection, semantic segmentation, and instance segmentation [28]. We collected and labeled 2085 images from the Internet and camera images. The new augmented and labeled image dataset consisted of 56,295 images.



(**a**)



(**b**)



(**c**)

Figure 7. The end-effector was moved to the cotton bolls, using the Cartesian arm system, and then a rotating brush roller grabbed the bolls into the end-effector. An onboard vacuum transported the bolls to the rotating porous impeller, where the bolls were dropped into a collection bag. (**a**) design of the cotton picking end-effector using vacuum (**b**) End-effector is trying to pick the bolls by pointing directly to the boll (**c**) The vacuum chamber to separate cotton bolls from air to drop them to the bag.

The dataset was used to train the tiny YOLOv3 model, using a 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, USA). The learning rate was set to 0.001 and maximum iterations to 2000 [27]. The batch size was 32. The model was trained for 4 h and then frozen, a process of combining graph and trained weights, and transported to the rovers' embedded system.

The ZED Library, Zed-YOLO in Github [29], the free open-source library package provided by the manufacturer of the ZED camera (Stereolabs Labs), was used to connect ZED camera images to tiny YOLOv3 model to perform object detection. The library used image bilinear interpolation to convert ZED SDK images to OpenCV images so that it could perform detection tasks. The bilinear interpolation was compared with the nearest-neighbor and no interpolation, to evaluate object detection. The results (Figure 8) show that the model performed well when no interpolation was applied (Figure 8c). The images in Figure 8 show that the bilinear interpolation detected only three bolls, the nearest-neighbor interpolation detected five bolls, and with no interpolation, the system detected eight bolls. Therefore, the boll detection algorithm improved its accuracy when no interpolation was performed. The library code was modified to avoid image interpolation and implemented in this study. The algorithm was able to perform well with illuminated images of undefoliated cotton (Figure 9), where the right image in Figure 9 shows cotton boll detection in natural direct sunlight.

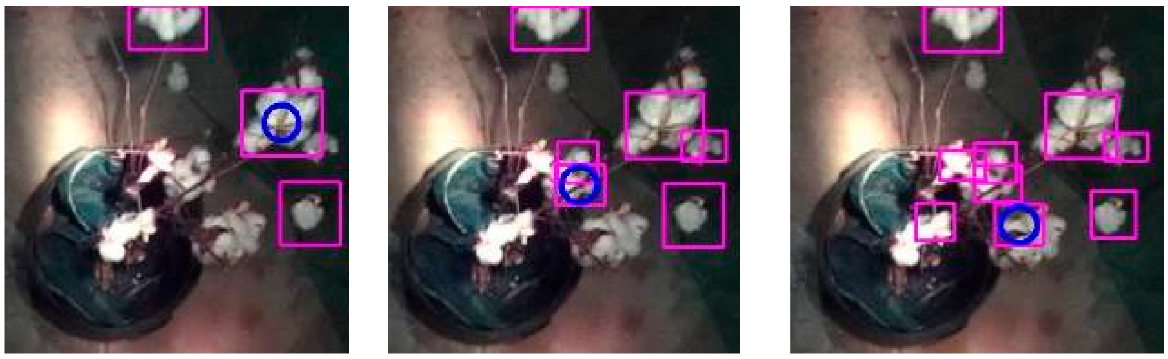


Figure 8. Detection of cotton bolls: (**a**) using bilinear transformation, 3 bolls were detected; nearest-neighbor interpolation (**b**) detected 5 bolls; and (**c**) without interpolation detected 8 bolls. The pink boxes are bounding boxes of the detected bolls, while blue represents the nearest boll to be picked.

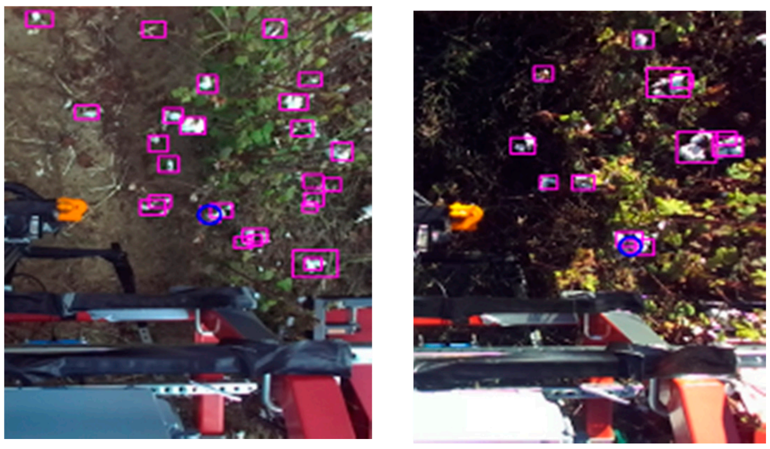


Figure 9. Boll detection using tiny YOLOv3 and ZED camera. (**a**): cloudy day. (**b**): sunny day.

2.6. End-Effector Detection Using Color Segmentation

Each image frame was acquired and analyzed to detect the orange end-effector, using a 4-step process: (1) depth processing, (2) color segmentation, (3) feature extraction, and (4) depth matching with features. These steps were handled by the graphics optimized rugged development kit (NVIDIA Jetson Xavier) to achieve improved performance, as image calculations require massive graphics computing resources like NVIDIA CUDA cores. The ZED SDK acquired and processed the images to get depth disparity and rectified images for both lenses. In this case, the ZED SDK was provided with 60 fps of 720 p quality images and 3D point clouds.

The images acquired (Figure 10a) were first analyzed for arm movement. Since the arm was orange in color, the threshold color was determined so that the arm could be segmented from the rest of the image (Figure 10b). The cotton boll and end-effector segmentation task involved the following four steps [30]:

1. Grab an image.
2. Using the RGB color threshold, separate each RGB component of the image. The end-effector, which is orange in color, can be masked. The threshold was (Red from 200 to 255, Green from 0 to 255, and Blue from 0 to 60).
3. Subtract the image background from the original image.
4. Remove all the regions where the contours were less than value M. Value M was determined by estimating the number of pixels defining the smallest boll.

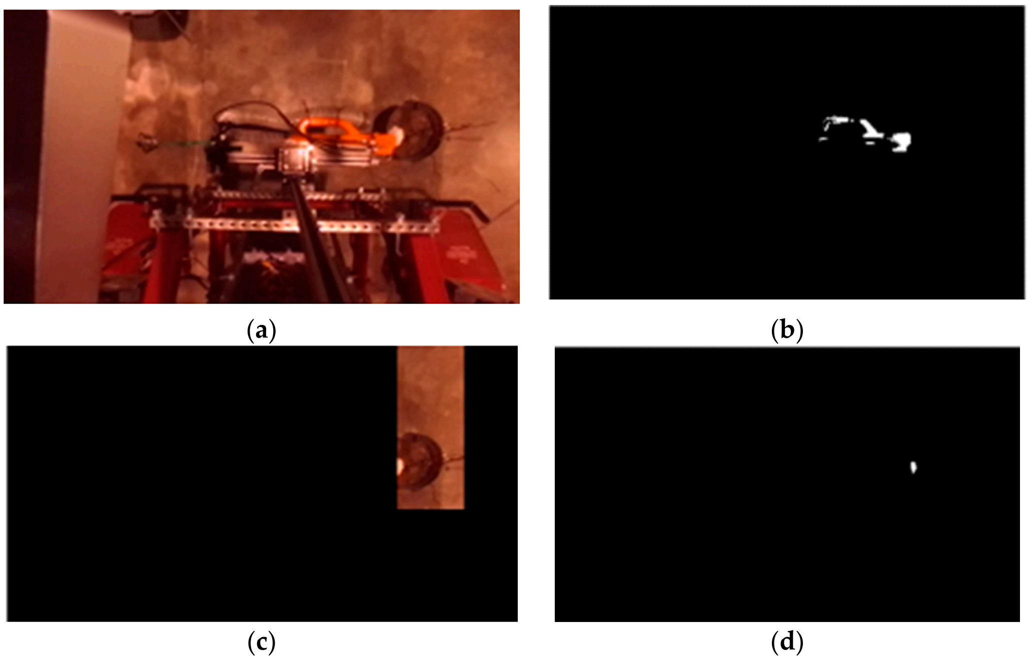


Figure 10. Color segmentation to localize the end-effector of the manipulator: (**a**) RGB image of the manipulator; (**b**) mask image of the end effector; (**c**) masking of all over image parts, leaving only an area that can be reached by the end effector, and, hence, it might contain bolls to pick; and (**d**) masking of the cotton bolls.

Feature extraction was performed by finding contours of the consecutive points which have the same intensity and were clustered. The cotton boll was then detected by using tiny YOLOv3 after segmenting the contour of the arm (Figure 10c,d).

After obtaining the contours for the end-effector, the centroid of the contour was calculated. All the depths were calculated for bolls by matching YOLOv3 detected bolls and end-effector coordinates with 3D point clouds or depth disparity maps, as described in Algorithm 1. The cotton bolls' coordinates (x,y,z) and robot manipulator coordinates (x_0,y_0,z_0) were obtained and used by the robot for picking decisions.

Algorithm 1 Algorithm describing the detection of cotton bolls.

```
Input: current video frame
Output: decision to move manipulator or rover (Ci)
1. Get end-effector position Xm, Ym and Zm;
2. Get prediction results of the YOLO model;
3. Get the list of centroids for each boll detected (Oj);
4. FOR EACH Oj in (Oj):
a. Boll_depth <- calculate the closest distance of the centroid using left lens point cloud;
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 from 0 cm to 37 cm from vertical arm;
14. Return the state decisions (Ci).
```

2.7. Inverse Kinematics of the Robot

The robot was designed to operate in 3D space and for a 2D Cartesian manipulator to pick a cotton boll as the end-effector reached the bolls. Figure 11 shows the inverse kinematics of the robot. The red cube represents the front of the rover, while the horizontal axis arm and vertical axis arm make up the rover's manipulator.

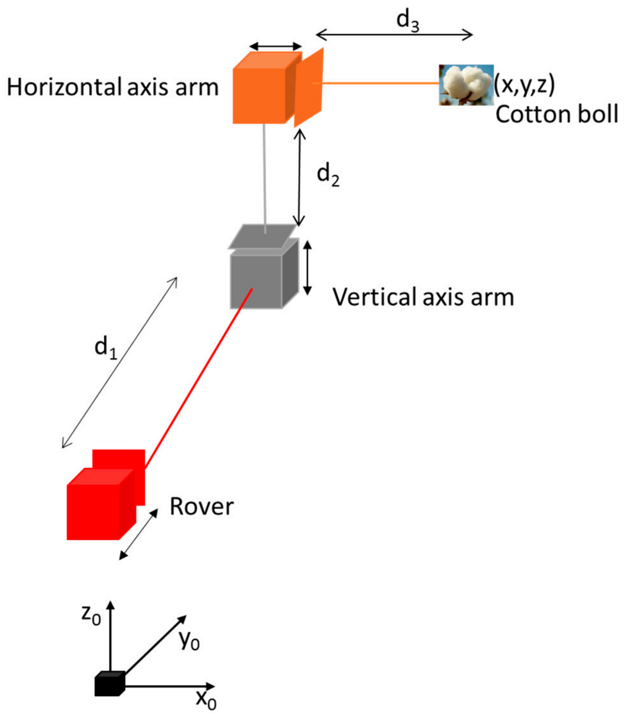
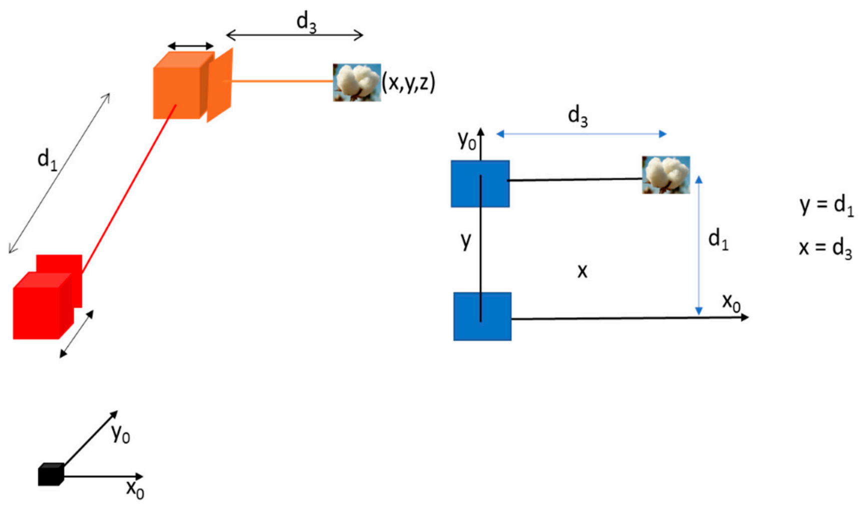
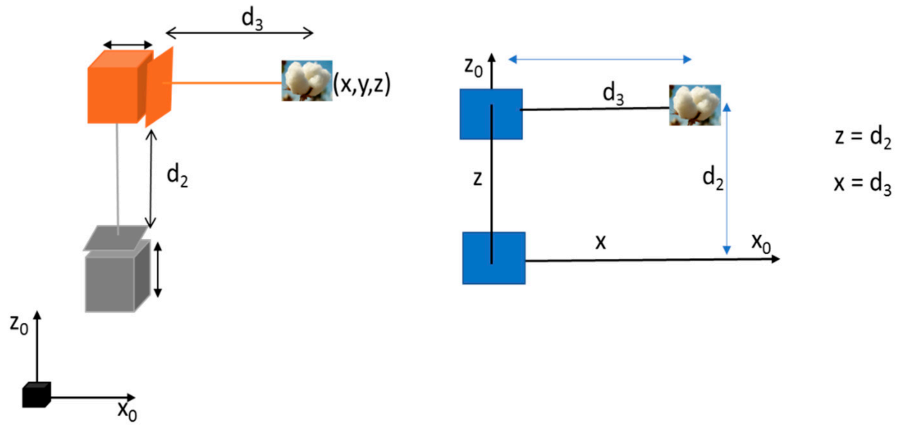


Figure 11. The inverse kinematics of the cotton-harvesting robot.

The inverse kinematics (Figures 11 and 12) was obtained by getting the values of the point (x,y,z), which is boll position from the origin of the rover (x_0,y_0,z_0). The robot could move distances d_1, d_2, and d_3 to pick the cotton boll at the point (x,y,z).



(**a**) Top view



(**b**) Side view

Figure 12. The inverse kinematics of the rover in detail (it is given by finding the distances d_1, d_2, and d_3 of the rover, using depth and coordinates of the boll (x,y,z) found by the ZED stereo camera).

2.8. Depth and Coordinates of Cotton Bolls and Arm

After matching the depths and contours of the Cartesian arm and cotton bolls, each reading of the arm and boll position was logged. Then, by using the tip of the end-effector (Figure 10c), the system obtained the image coordinates of its front part. Then, using the centroids of each boll, the system calculated the real-world coordinates (W) of the bolls from the image coordinates obtained (I) by using image geometry. The stereo camera was calibrated by using the ZED Calibration tool of the ZED SDK to obtain the camera transformation matrix (Equation (2)) parameters. The procedures to calibrate the ZED camera were learned from their website [31]. The camera matrix consists of f_x and f_y (the focal length in pixels); C_x and C_y (the optical center coordinates in pixels); and k_1 and k_2 (distortion parameters). The real-world coordinates of a cotton boll, W_x and W_y (Equations (4) and (5)), can be obtained if the algorithm is provided with the value of I_x and I_y, which was the coordinate of the centroid of the front part of the end-effector. Alternatively, by finding the inverse of the camera matrix and multiplying the vector image (I_x and I_y), the world coordinates (W_x and W_y) could be obtained. C_x, f_x, C_y, and f_y were found by calibrating the camera, while W_z was determined from the 3D point cloud provided by the ZED SDK. The parameter values of the calibrated camera used were as follows:

$$\begin{aligned} C_x &= 685.286 \\ C_y &= 361.248 \\ f_x &= 699.936 \\ f_y &= 699.936 \end{aligned} \tag{1}$$

$$C = \begin{bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} I_x \\ I_y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & C_x & 0 \\ 0 & f_y & C_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} W_x \\ W_y \\ W_z \\ 1 \end{bmatrix} \tag{3}$$

$$W_x = (I_x + C_x) \times \left(\frac{W_z}{f_x}\right) \tag{4}$$

$$W_y = \left(I_y + C_y\right) \times \left(\frac{W_z}{f_y}\right) \tag{5}$$

where f_x and f_y equal the focal length in pixels; C_x and C_y equal the optical center coordinates in pixels; W_x and W_y equal the real-world coordinates; and I_x and I_y equal the coordinate of the centroid of the front part of the horizontal arm.

Object depth in an image (W_z) was obtained from the 3D point clouds. For each boll and end-effector location, the distance points (d_x, d_y, d_z) were provided by the ZED SDK. It was recommended to use the 3D point cloud instead of the depth map when measuring depth distance. The Euclidean distance (Equation (6)) is the calculated distance of an object (end-effector or bolls) relative to the left lens of the camera.

$$W_z = \sqrt{{d_x}^2 + {d_y}^2 + {d_z}^2} \tag{6}$$

Depth distance (W_z) should be greater than zero and less than the distance of the camera to the lowest position of the end-effector, to avoid the rover attempting to harvest unreachable bolls. Moreover, the horizontal distance (W_x) of the boll from the center of the camera should not exceed the length of the horizontal axis arm. After obtaining such measurements, the system executed other tasks, like controlling the arm or moving the rover. For the machine to be able to execute each task independently and in coordination with the other tasks, the finite state machine was developed to manage task-based requests.

2.9. Finite State Machine

Robot tasks and actions were categorized as states, and state "transitions" were modeled in a task-level architecture to create the rover actions required to harvest cotton bolls. This approach provided a maintainable and modular code. Using an open-source ROS library known as SMACH, we smoothly implemented the tasks, to build complex behavior (Figure 13) [32].

The state machine had six necessary states and seven transitions (Figures 13 and 14). After every state, the system reverted to state 0 (get the image) and searched for cotton bolls. If a cotton boll was found, the system would calculate the distance of the boll from the manipulator in three-dimensional space (X,Y,Z), as described in the inverse kinematics section. If the boll was lined up horizontally, then the system would get the manipulator to move up or down relative to the position of the manipulator to the boll. If the boll was at the same level, then the system would harvest it. If the boll was in the front or back, the system would send a signal for the rover to move forward or back, using the PID control. If the system failed to see any bolls, the rover proceeded to pass over the cotton rows. Algorithm 1 describes the detection of the bolls and actions taken by the state machine algorithm to accommodate the rover transition of the tasks.

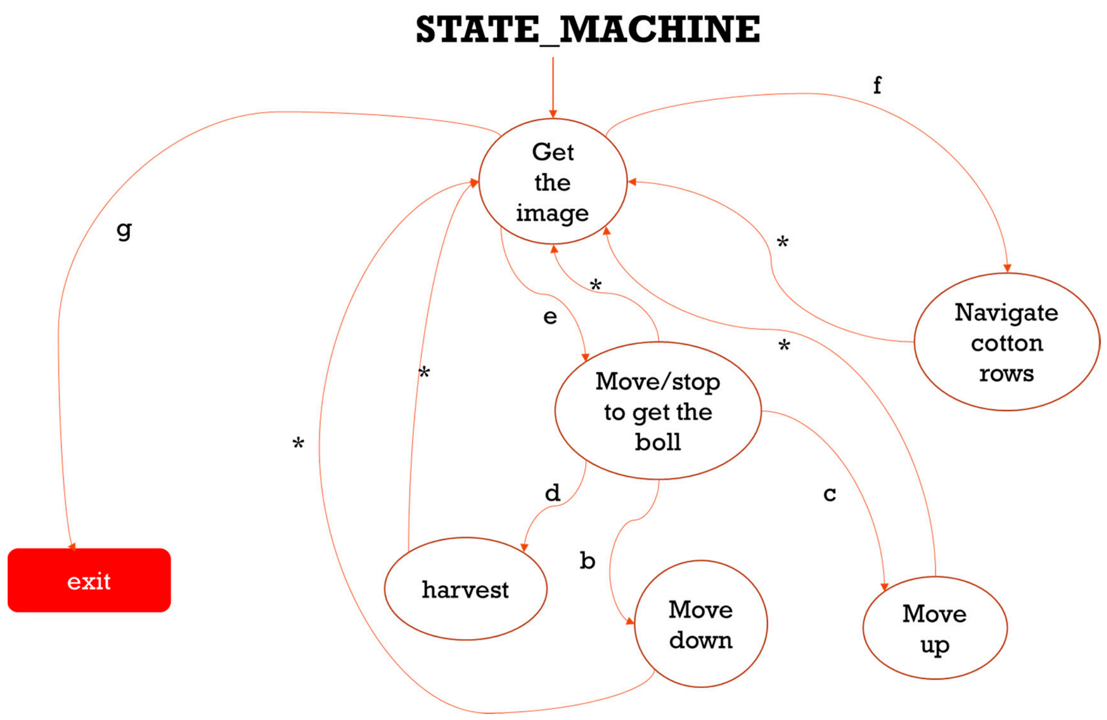
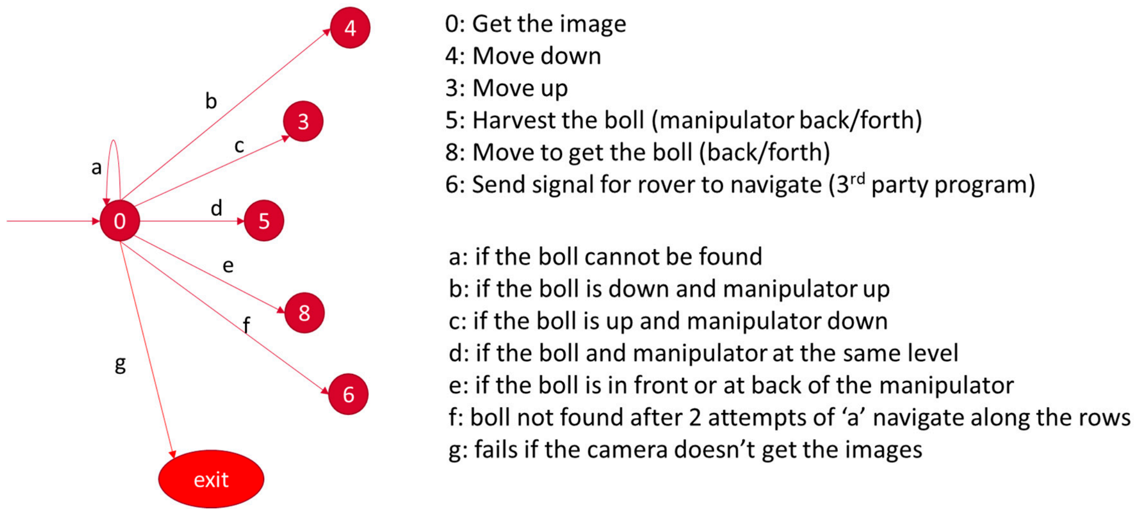


Figure 13. Finite state machine diagram of the rover states and transitions. * Means a return instruction to get a new image.



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 14. Linear transformation of the finite state machine diagram (Figure 13) of the rover states and transitions.

2.10. Calibration of the Manipulator

The manipulator was calibrated for its horizontal and vertical movements. The equation was obtained by first moving the arm to the furthest location away from the ZED camera. The distance of the arm from the camera was then recorded and was continually recorded as the arm was moved by each step of the stepper drive, until it was closest to the camera. The equation obtained ($R^2 = 0.99$) by fitting the points is as follows:

$$\text{motor steps} = -410.7 \times (\text{distance}) + 483.29 \tag{7}$$

2.11. Rover Movement Controller

The rover movement controller ran an adaptive PID to control the rover forward and rearward movement. The position of the rover and the target position were published from the master, and the Arduino clients subscribed to the topic accordingly. The throttle was set at a constant maximum RPM to maintain constant power to the hydraulic systems. A topic that subscribed to this message was developed in the rover microcontroller. Articulation was done after the rover controller received a topic that published the instruction for the rover to turn or go straight. For this study, the rover only moved straight. The master controller published an articulation message at the rate of 50 Hz to the rover controller, which had subscribed to the topic. The rover movement controller published the gains of the adaptive PID controller together with the position of the rover relative to the encoder pulses. The master controller subscribed to the messages, so as to provide an accurate position of the rover, which was used to pass the signal to the arm controller for boll picking.

Tuning was done by first identifying the deadband of the hydraulic swashplate arm for the variable displacement pump. Up to a particular movement angle of the swashplate arm, there was insufficient fluid flow and pressure to move the rover. For operation, the rover movement controller sent a servo signal to the linear actuator (Figure 15), to proportionally increase fluid flow to the hydraulic wheel motors. The linear actuator pushed the swashplate to a certain angle and was directly controlled by using a rover navigation controller. The linear actuator extends from 0 cm to 20 cm after receiving an analog servo signal. Retracting and extending the linear actuator changed the position of the swashplate arm, which in turn controlled the speed and direction of the hydraulic fluid, which provided the motion capability of the rover. The angle of the swashplate can be changed by sending a servo Pulse Width Modulation (PWM) signal that ranged from 65 to 120, using the Arduino servo library. When a 90 PWM signal was sent to the rover controller, the rover stoped, since the swashplate was positioned in its neutral position, and the fluid flow was zero. If the PWM signal was decreased slowly from 90 to 65, the rover moved in reverse motion, while if it was increased from 90 to 120, the rover moved forward. It meant the 65 PWM signal provided maximum speed in reverse motion, while 120 PWM signal provided maximum speed forward. The rotary encoder installed on the wheel of the rover sends back input pulses to the PID, which measures how far the rover has moved. However, the system had a large deadband from a PWM signal of 80 to 98, which means the angle change was not enough to make the rover move. The deadband was the signal angle sent from the rover Arduino microcontroller to the linear actuator servo that cannot make the rover move either forward or reverse. Consequently, when the rover missed the target, it became challenging to get close to the target, as the PID can either be a direct relationship or reverse relationship between output (actuator signal) and input (encoder pulses).

In order to remove the deadband, the system was redesigned such that it gave the output *u(x)* from −100 to 100 (Equation (8)). Then, the signal was mapped to the correct settings of the rover. The actual servo signal was set to move (extend) from 98 to 108 by mapping positive output values (0 to 100) of *u(x)*, while moving back (retract) from 70 to 80 for negatives output values (−100 to 0); zero was set to be a PWM signal of 90.

$$u(x) = \begin{cases} 10 \times (x - 98)\ if\ 98 \leq x \leq 108 \\ 10 \times (x - 80)\ if\ 70 \leq x \leq 80 \end{cases} \tag{8}$$

The PID controller was manually tuned. It was done by increasing K_p until the rover oscillated with neutral stability, while setting K_i and K_d values to zero. Then, K_i was increased until the rover oscillated around the setpoint. After that, K_d was increased until the system was settling at the given setpoint quickly. The PID gains obtained were $K_p = 0.5$, $K_i = 0.15$, and $K_d = 1$. Later, the navigation controller was recalibrated to determine if the performance could be improved. The PID gains that create a small overshoot response but aggressively moved the rover forward were obtained: $K_p = 0.7$, $K_i = 0.04$, and $K_d = 5.0$. The rqt_graph library [33], which is the ROS computation visualizing graph, was used to study the pulses. Each of the encoder pulses was equivalent to 1.8 mm in distance. The system performed the same way going forward or backward. The overshoot was approximately 18 cm.

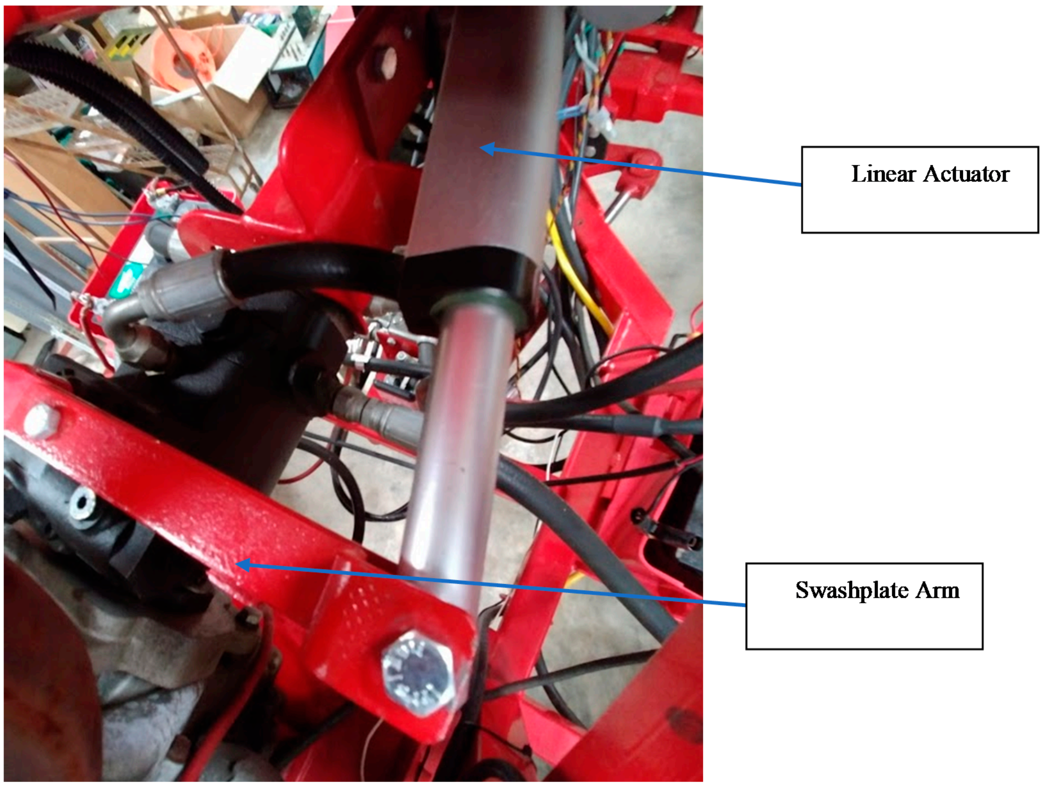


Figure 15. The linear actuator moving the swashplate arm to determine an angle for movement of the rover (forward when extending or rearward when retracting).

2.12. Proportional Control of the Articulation Angle

The rover turned to the target articulation angle, γ, by using proportional control. The current articulation angle, γ_k, and required target angle, γ_{k+1}, were used to find the error used to control the signal to turn the rover. The gain K_p was set to 1. The articulation angle was controlled by the hydraulic linear actuators, which were connected to two relays. The two relays were used to connect the left and right control signals from the navigation controller to a 12 V power source to move the hydraulic directional control valve spool. Hydraulic cylinders in series were used to push and pull the front and rear halves of the rover, to create a left or right turn. If the rover turned left, the left actuator retracted while the right actuator expanded, until a desired left articulation angle was achieved. If the rover turned right, the right actuator retracted while the left actuator expanded, until a desired right articulation angle was achieved. The angle was reported by a calibrated high-precision potentiometer.

2.13. Preliminary Experimental Setup

The navigation and manipulation of the system were tested as one unit of the robot. It is essential to determine the performance of the whole robot in navigating to cover 3D space and picking of the bolls.

Five experiments were set up at the University of Georgia (UGA) Tifton campus grounds (N Entomology Dr, Tifton, GA, 31793). The experiments were undertaken on 29th and 30th May 2019. Six cotton plants were placed 30 cm apart, three bolls per plant. The slope of the ground surface in the direction of the forwarding movement was 0.2513°. The rover was driven over the plants, and the manipulator moved into a position to pick the bolls from the plants (Figures 16 and 17). The camera was 220 m above the ground (l). The first boll detected by the system was distance "m" from the camera. The camera also checked the distance of the arm from the camera, and then the manipulator controller sent a signal to move the arm to harvest the first boll. Subsequent boll picking was repeatedly accomplished by getting the distance of the current end-effector position to the next boll position in the camera images.

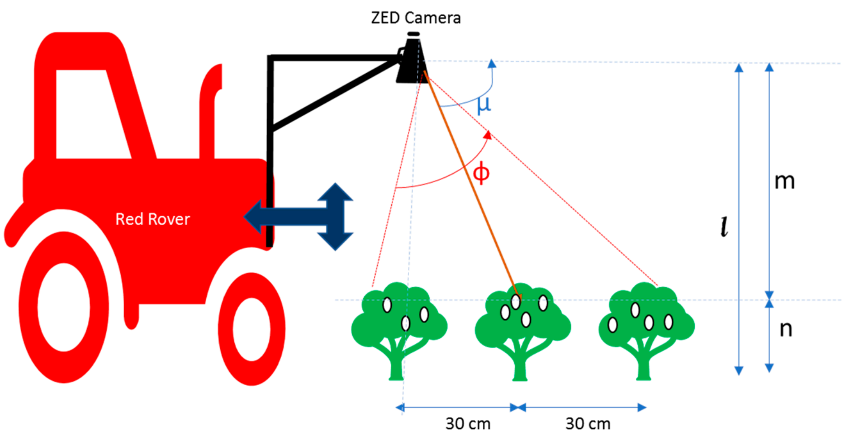


Figure 16. System testing was done by putting the six potted defoliated plants in front of the system, to collect preliminary performance data.

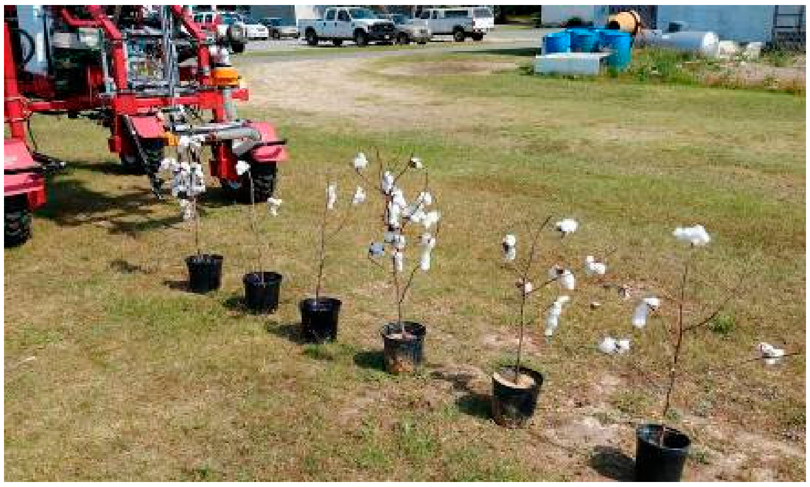


Figure 17. The experiment was set up at University of Georgia (UGA) grounds, to test the robot on picking the bolls in a simulated environment consisting of six potted cotton plants.

The results of the five experiments, using six plants with three bolls each, were collected and analyzed, using the robot performance metrics mentioned by Reference [34].

2.14. Field Experiment

The field experiment in undefoliated cotton was conducted at the Horticulture Hill Farm (31.472985 N, 83.531228 W), near Bunny Run Road in Tifton, Georgia, after establishing the calibrated parameters of the rover. The field (Figure 13) was planted on 19th June 2019, using a tractor (Massey-Fergurson MF2635 tractor, AGCO, Duluth, GA, USA) and a 2-row planter (Monosem planter, Monosem Inc, Edwardsville, KSN USA). The cotton seeds (Delta DP1851B3XF, Delta & Pine Land Company of Mississippi, Scott, MS, USA) were planted every two rows, and two rows skipped. The rows were 36 inches (91.44 cm) wide, and the seed spacing was 4 inches (10.16 cm). The field experiment was done after finishing the preliminary experiments. Two tests (5.3 m) for picking the cotton bolls were conducted on 22nd November 2019 and 2nd December 2019. The slope of the ground surface in the direction of the forwarding movement was 2.7668°. It is steeper than the preliminary experiment field. We measured the 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.

3. Results and Discussions

3.1. Simulated Harvesting of Potted and Defoliated Cotton

The results (Table 2) of the test with defoliated cotton in pots were obtained by counting the cotton bolls that the robot was able to pick and the time it took to collect the boll. Moreover, images taken by the camera were checked to determine if the system could detect cotton bolls by using only color segmentation. 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 in the simulated environment.

Cotton Bolls	Detec-ted	Reached	Picked	Time (s)	OV (cm s^{-1})	PR (bolls s^{-1})	CT (s)	ASR (%)	DP (%)	MRR (%)
18	17	16	15	214	1.12	0.07	14.27	83.33	94.44	88.89
18	18	17	16	343	0.70	0.05	21.43	88.89	100.00	94.44
18	18	17	15	219	1.10	0.07	14.60	83.33	100.00	94.44
18	17	17	17	323	0.74	0.05	19.00	94.44	94.44	94.44
18	18	18	17	285	0.84	0.06	16.76	94.44	100.00	100.00
Average	17.4	17	16	276.8	0.87	0.06	17.3	88.88	96.67	94.44

Notes: ASR = Action Success Ratio; CT = Cycle Time; DP = Detection Performance; MRR = Manipulator Reaching Ratio; OV = Operation Velocity; PR = Production Rate.

The system picked a boll at an average of 17.3 s, at a total distance of 2.4 m. The experiment showed that our rover design is a viable solution for cotton harvesting. However, the system was observed to attempt (MRR) twice or more to pick an average of 17 of the 18 bolls. The manipulator missed an average of two bolls because of the overshoot by the PID controller that reduced the production rate.

3.2. Field Picking of Cotton

Two tests (Table 3) to investigate the effectiveness of the rover to pick the bolls were successfully conducted in a field with undefoliated cotton. In the first test, the rover picked 67 bolls and left behind 17 bolls for an Action Success Ratio (ASR) of 80%. The rover was able to reach (Manipulator Reaching Ratio (MRR)) 94% of the bolls (79 bolls). The distance covered by the rover was 5.3 m. For the second test, the robot picked 89 bolls and left behind 26 bolls, for an ASR of 77%. The MRR was 96%, and the distance covered was 5.3 m. The average ASR was 78.5%. The average MRR was 95%, while CT and DP were 38.35 s and 0.03 bolls per second, respectively. The CT for field testing was over twice the cycle time in the defoliated and potted cotton plant test, while DP was about half the performance in defoliated and potted cotton plants. MRR was comparable in both field and simulated testing (95.3% and 94.4%, respectively). This performance indicated the system was having trouble picking bolls in the real field due to the unstructured topography of the field and the leaf and stem obstruction of the bolls. Moreover, in the simulated environment, the bolls were loosely attached to the plant; hence, it was easier to remove from the branch in a single attempt.

Table 3. The collected data for the experiments done to validate the performance of the system in the real field environment.

Cotton Bolls	Detected	Reached	Picked	Time (s)	OV (cm s^{-1})	PR (bolls s^{-1})	CT (s)	ASR (%)	DP (%)	MRR (%)
84	83	79	67	2700	0.20	0.02	40.30	79.76	98.81	94.05
115	112	111	89	3240	0.16	0.03	36.40	77.39	97.39	96.52
Average	97.5	95	78	2970	0.18	0.03	38.35	78.58	98.10	95.28

4. Conclusions

The preliminary design and performance of a prototype cotton harvesting rover were reported. The system was optimized to use visual-based controls to pick cotton bolls in undefoliated cotton. The system used SMACH, an ROS-independent finite state machine library that provides task-level capabilities, to design a robotic architecture to control the rover's behavior. The cotton harvesting system consisted of a hydrostatic, center-articulated rover that provided mobility through the field, and a 2D manipulator with a vacuum and rotating mechanism for picking cotton bolls and transporting them to a collection bag. The system achieved a picking performance of 17.3 s per boll in simulated field conditions and 38 s per boll in a real cotton field. The increased time to pick each boll resulted from the cotton plants' overlapping leaves and branches obstructing the manipulator. Cotton boll detachment was much more difficult because the bolls were not placed artificially (as done in the potted plants), and the level of uphill and downhill movement resulted in poor control of the hydraulic navigation system.

The overall goal of the harvesting rover was to develop a system to harvest cotton that uses multiple small harvesting units per field. These units would be deployed throughout the harvest season, beginning right after the opening of the first cotton bolls. If this team of harvesting rovers is to reach commercial viability, the speed of harvest (CT) and successful removal of bolls (ASR) must be improved. To address these shortcomings, a modified end-effector and an extra upward-looking camera at the bottom of the manipulator in undefoliated cotton will be studied in future studies of the prototype. Furthermore, future research will be conducted to improve the rover's overall navigation, to improve the rover's and manipulator's alignment with pickable bolls.

Author Contributions: Conceptualization, K.F. and G.R.; methodology, K.F. and G.R.; formal analysis, K.F.; software, K.F.; writing—original draft preparation, K.F.; writing—review and editing, K.F., C.L., W.P., and G.R.; supervision, K.F., C.L., W.P., and G.R.; project administration, W.P., E.B., and G.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Cotton Incorporated, Agency Assignment Number 17-038.

Acknowledgments: The authors are very thankful for project support from Cotton Incorporated, Agency Assignment #17-038. We also thank Ricky Fletcher and Gary Burnham for their helpfulness and technical support in building the camera and robotic platform and collection of data. We also thank Logan Moran for field testing, William Hill, and Tearston Adams for labeling the cotton images, and Jesse Austin Stringfellow for 3D design and printing.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

RTK–GNSS	Real-Time Kinematic–Global Navigation Satellite System
DCV	directional control valve
SMACH	task-level architecture for rapidly creating complex robot behavior
VDC	Voltage of Direct Current
DC	Direct Current
OpenCV	Open Computer Vision Library
CUDA	Compute Unified Device Architecture
YOLO	You Only Look Once, an algorithm for object detection

DIP	Dual In-line Package Switch
USB	Universal Serial Bus
DOF	Degree of Freedom
PID	Proportional-Integral-Derivative Controller
IMU	Inertial Measurement Unit
ROS	robot operating system
SDK	Software Development Kit
3D	Three-Dimensional
2D	Two-Dimensional
Kp	Proportional Gain
Kd	Derivative Gain
Ki	Integral Gain
Hz	Heitz
OV	Operation Velocity under real-time conditions (cm s-1)
PR	Production Rate (bolls s-1),
CT	Cycle Time (s)
ASR	Action Success Ratio (%)
DP	Detection Performance (%)
Xm, Ym, Zm	3D position of the end effector end
Xb, Yb, Zb	3D position of the targeted boll to be pickedReferences

References

1. *2017 State Agriculture Overview for Georgia*; USDA/NASS: Washington, DC, USA, 2018.
2. Burnard, T. The American South and its global commodities. *Slavery Abolit.* **2017**, *38*, 215–217. [CrossRef]
3. UGA. Georgia cotton production guide. In *Ugacotton.org*; Team, U.E., Ed.; UGA Extension Team: Tifton, GA, USA, 2019.
4. Van Der Sluijs, M.H.J.; Roth, G.W. Comparing dryland cotton upland fibre quality from on-board spindle and stripper harvesting systems. *J. Text. Inst.* **2020**, 1–8. [CrossRef]
5. Parvin, D.W.; Martin, S.W. The Effect of Recent and Futuristic Changes in Cotton Production Technology on Direct and Fixed Costs Per Acre, Mississippi, 2004. In Proceedings of the Southern Agricultural Economics Association, 2005 Annual Meeting, Little Rock, AR, USA, 5–9 February 2005.
6. Waldman, P.; Mulvany, L. Farmers Fight John Deere Over Who Gets to Fix an $800,000 Tractor. In *Bloomberg*; bloomberg.com: New York, NY, USA, 2020.
7. Zahniser, S.; Taylor, J.E.; Hertz, T.; Charlton, D. Farm Labor Markets in the United States and Mexico Pose Challenges for U.S. Agriculture. *Econ. Res. Serv. Univ. Minn. Website* **2018**, 46. [CrossRef]
8. Richards, T.J. Immigration Reform and Farm Labor Markets. *Am. J. Agric. Econ.* **2018**, *100*, 1050–1071. [CrossRef]
9. Guthman, J. Paradoxes of the Border: Labor Shortages and Farmworker Minor Agency in Reworking California's Strawberry Fields. *Econ. Geogr.* **2016**, *93*, 24–43. [CrossRef]
10. Katchova, A.L.; Ahearn, M.C. Farm entry and exit from US agriculture. *Agric. Finance Rev.* **2017**, *77*, 50–63. [CrossRef]
11. Bergerman, M.; Billingsley, J.; Reid, J.; Van Henten, E.; Siciliano, B.; Khatib, O. Robotics in Agriculture and Forestry. In *Springer Handbook of Robotics*; Springer Science and Business Media LLC: Berlin/Heidelberg, Germany, 2016; pp. 1463–1492.
12. Comba, L.; Gay, P.; Piccarolo, P.; Ricauda Aimonino, D. Robotics and automation for crop management: Trends and perspective. In Proceedings of the International Conference Ragusa SHWA2010, Ragusa Ibla Campus, Ragusa, Italy, 16–18 September 2010; pp. 471–478.
13. Shamshiri, R.R.; Weltzien, C.; Hameed, I.A.; Yule, I.J.; Grift, T.E.; Balasundram, S.K.; Pitonakova, L.; Ahmad, D.; Chowdhary, G. Research and development in agricultural robotics: A perspective of digital farming. *Int. J. Agric. Boil. Eng.* **2018**, *11*, 1–11. [CrossRef]
14. Fue, K.; Porter, W.M.; Barnes, E.M.; Rains, G.C. An Extensive Review of Mobile Agricultural Robotics for Field Operations: Focus on Cotton Harvesting. *AgriEngineering* **2020**, *2*, 10. [CrossRef]

15. Fountas, S.; Mylonas, N.; Malounas, I.; Rodias, E.; Santos, C.H.; Pekkeriet, E. Agricultural Robotics for Field Operations. *Sensors* **2020**, *20*, 2672. [CrossRef]
16. Kondo, N.; Ting, K.C. Robotics for Plant Production. *Artif. Intell. Rev.* **1998**, *12*, 227–243. [CrossRef]
17. Lowenberg-Deboer, J.; Huang, I.Y.; Grigoriadis, V.; Blackmore, S. Economics of robots and automation in field crop production. *Precis. Agric.* **2019**, *21*, 278–299. [CrossRef]
18. Bietresato, M.; Carabin, G.; D'Auria, D.; Gallo, R.; Ristorto, G.; Mazzetto, F.; Vidoni, R.; Gasparetto, A.; Scalera, L. A tracked mobile robotic lab for monitoring the plants volume and health. In Proceedings of the 2016 12th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Auckland, New Zealand, 29–31 August 2016; pp. 1–6.
19. Iqbal, J.; Xu, R.; Sun, S.; Li, C. Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation. *Robotics* **2020**, *9*, 46. [CrossRef]
20. Mousazadeh, H. A technical review on navigation systems of agricultural autonomous off-road vehicles. *J. Terramech.* **2013**, *50*, 211–232. [CrossRef]
21. Higuti, V.A.H.; Velasquez, A.E.B.; Magalhaes, D.V.; Becker, M.; Chowdhary, G. Under canopy light detection and ranging-based autonomous navigation. *J. Field Robot.* **2018**, *36*, 547–567. [CrossRef]
22. Hohimer, C.J.; Wang, H.; Bhusal, S.; Miller, J.; Mo, C.; Karkee, M. Design and Field Evaluation of a Robotic Apple Harvesting System with a 3D-Printed Soft-Robotic End-Effector. *Trans. ASABE* **2019**, *62*, 405–414. [CrossRef]
23. Moore, T.; Stouch, D. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In *Intelligent Autonomous Systems*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 13, pp. 335–348.
24. Rains, G.C.; Faircloth, A.G.; Thai, C.; Raper, R.L. Evaluation of a simple pure pursuit path-following algorithm for an autonomous, articulated-steer vehicle. *Appl. Eng. Agric.* **2014**, *30*, 367–374.
25. Fue, K.G.; Porter, W.M.; Rains, G.C. Real-Time 3D Measurement of Cotton Boll Positions Using Machine Vision Under Field Conditions. In Proceedings of the 2018 BWCC, San Antonio, TX, USA, 3–5 January 2018; pp. 43–54.
26. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; p. 5.
27. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
28. CLODSA. Available online: https://github.com/joheras/CLoDSA (accessed on 31 January 2020).
29. ZEDYolo. Available online: https://github.com/stereolabs/zed-yolo (accessed on 31 January 2020).
30. Gong, Y.; Sakauchi, M. Detection of Regions Matching Specified Chromatic Features. *Comput. Vis. Image Underst.* **1995**, *61*, 263–269. [CrossRef]
31. ZED. Available online: https://www.stereolabs.com/zed/ (accessed on 31 January 2020).
32. SMACH. Available online: http://wiki.ros.org/smach (accessed on 31 January 2020).
33. Rtq_graph. Available online: http://wiki.ros.org/rqt_graph (accessed on 31 January 2020).
34. Bechar, A.; Vigneault, C. Agricultural robots for field operations. Part 2: Operations and systems. *Biosyst. Eng.* **2017**, *153*, 110–128. [CrossRef]